ÉCOLE DOCTORALE DES SCIENCES EXACTES ET DE LEURS APPLICATIONS ED 211

THÈSE

pour l'obtention du

Doctorat de l'Université de Pau et des Pays de l'Adour

(spécialité informatique)

présentée par

The Nhan LUONG

Modélisation centrée utilisateur final appliquée à la conception d'applications interactives en géographie : une démarche basée sur les contenus et les usages

soutenue publiquement le 12 décembre 2012

Composition du jury

Président :	Philippe VIDAL	Professeur, Université de Toulouse 3
Rapporteurs :	Cécile ROISIN	Professeur, Université de Grenoble 2
	Pascal ESTRAILLIER	Professeur, Université de La Rochelle
Co-encadrants :	Christophe MARQUESUZAÀ	MdC, Université de Pau et des Pays de l'Adour
	Patrick ETCHEVERRY	MdC, Université de Pau et des Pays de l'Adour
Directeur :	Thierry NODENOT	Professeur, Université de Pau et des Pays de l'Adour

Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour — EA 3000

LIUPPA

Mis en page avec la classe thloria.

Remerciements

Je voudrais tout d'abord exprimer ma plus profonde gratitude à Thierry Nodenot, mon directeur de thèse, pour m'avoir accueilli au sein de l'équipe T2i du LIUPPA et pour m'avoir soutenu, aidé et conseillé, pour avoir cru en moi. C'est grâce à lui que j'avais des idées claires et une bonne direction de ma thèse. De plus, ses conseils m'ont permis de surmonter les moments de doute.

Je tiens à remercier vivement Christophe Marquesuzaà, qui a co-encadré ma thèse. C'est à lui que je dois non seulement un nombre de remarques sur mon travail, mais également quelques dîners sympathiques. Il a toujours su me motiver, m'encourager et me faire avancer. Merci également à sa famille : sa femme Chantal, leur petit Allande et leur petite Amaia.

J'adresse mes sincères remerciements à Patrick Etcheverry, qui a co-encadré ma thèse et était aussi mon tuteur de monitorat, pour l'intérêt qu'il a porté à mes travaux tout au long du déroulement de ma thèse et pour m'avoir aidé à faire mes premiers pas dans le monde de l'enseignement.

Je remercie particulièrement Philippe Lopistéguy, Sébastien Laborie et Pantxika Dagorret pour leurs relectures et commentaires et plus généralement pour leur gentillesse.

Je tiens à remercier pour leurs commentaires, leurs remarques et leurs conseils, mes deux rapporteurs, Cécile Roisin et Pascal Estraillier qui m'ont permis d'améliorer le manuscrit originel. Qu'ils soient ici remerciés d'avoir accepté de consacrer de leur temps et énergies à cette tâche.

Merci également à Philippe Vidal pour m'avoir fait l'honneur d'accepter de présider le jury de ma soutenance de thèse.

Je voudrais également remercier tous les autres chercheurs et doctorants que j'ai rencontrés et avec qui j'ai discuté. Merci en particulier à Albert Royer, Christian Sallaberry, Pierre Loustau, Damien Palacio, Jean-François Boullier avec lesquels j'ai collaboré au cours de ces années de thèse. Merci à Mike Deguilhem, Virginie Paillas et Pascal Nodenot pour leur contribution à nos expérimentations.

Mes remerciements vont également au Conseil Général des Pyrénées-Atlantiques, au Conseil Régional d'Aquitaine, à la société Géocime et à la Ligue de l'Enseignement dans le cadre des deux projets "Pyrénées : Itinéraires Educatifs" (2009-2012) et "Redécouvrir et valoriser nos documents patrimoniaux par le biais des nouvelles technologies : Concevoir à partir des usages et des contenus" (2007-2010) ayant permis une partie du financement ainsi que la mise en application de certains aspects de ma thèse.

Merci à Simone, Agnès, Francis, Eric, Yon, Cédric, Ghada, Keling et tous les autres : l'IUT de Bayonne et du Pays Basque offre vraiment une bonne ambiance de travail. Merci également au personnel de la cafétéria du parc Montaury pour leur bonne humeur quotidienne et leurs déjeuners pendant les années de ma thèse.

C'est avec émotion que je remercie aussi tous mes amis, en particulier Phương Mai, Hồng Nguyên et Sébastien, Vân Minh, anh Tuấn Dũng, chị Tuyết Mai, anh Phong và chị Hồng, Quốc Thắng. Merci pour le télé-soutien par téléphone et par mail, pour leurs encouragements et leurs belles pensées. Un immense merci à mes parents et mon grand frère pour avoir cru en moi pendant toutes ces années et parce que, sans eux, sans leurs conseils, et sans leur amour, rien de tout ceci n'aurait pu arriver. Et bien sûr, toutes mes pensées vont à *em iu* Quỳnh Anh, que je remercie tendrement pour sa patience et tout l'amour qu'elle me porte.

Merci beaucoup !
Thank you very much!
Cảm ơn rất nhiều!
Eskerrik hainitz!
...

Je dédie cette thèse à ma famille.

Table des matières

Chapitre 5
Bilan et perspectives

Table des figures

Chapitre 1

Introduction générale

Sommaire

L'information géographique fait l'objet de nombreuses études tant par les géographes que par les chercheurs en sciences cognitives ou les informaticiens spécialistes de géomatique. Par ailleurs, l'information géographique constitue un domaine qui est important dans les instructions officielles de l'Éducation Nationale et que les enseignants font donc manipuler à leurs élèves à travers les situations d'apprentissage qu'ils proposent. Les sections 1.1 et 1.2 de ce chapitre nous permettent de faire un état de l'art sur ces deux points de vue sur l'information géographique.

1.1 L'information géographique : un objet d'étude pour les géographes et les informaticiens

1.1.1 Information géographique et textes à références spatiales et temporelles

[DS04] définit l'information géographique comme un ensemble composé :

- D'une information relative à un objet ou phénomène du monde terrestre décrit par sa nature, son aspect et ses attributs (par exemple, un bâtiment décrit par sa hauteur, son nombre d'étages, sa fonction...). Cette description peut inclure des relations avec d'autres objets ou phénomènes (par exemple, tel bâtiment appartient à telle commune, telle rivière traverse tel département...).
- De sa localisation sur la surface terrestre, décrite dans un système de référence explicite (par exemple, un système de coordonnées de type longitude/latitude) ou implicite.
- De sa composante temporelle : on précise alors que la localisation de l'information ou l'expression du phénomène est valable à un moment (ou une période) donné(e).

L'information géographique présentée plus en détail dans la section 3.2.2.1 peut se définir comme un ensemble de trois facettes : spatiale, temporelle et thématique [Use96, Gai01] (Figure 1.1).

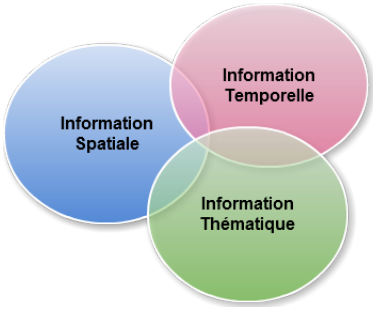


FIGURE 1.1 – Trois facettes de l'information géographique

De nombreuses situations de la vie courante nécessitent la compréhension de descriptions spatiales et temporelles. Ces descriptions sont utiles pour localiser et situer des objets ou des événements et plus généralement pour raisonner sur le monde qui nous entoure. L'information géographique apparaît sous sa forme complète (par exemple, "l'échec scolaire en France dans les années 90") ou sous des formes réduites (par exemple, "nous avons quitté Pau"), la composante spatiale étant la plus régulièrement énoncée. Dans ses travaux, [Lev96] a énoncé trois façons d'utiliser le langage naturel pour caractériser des relations spatiales (Figure 1.2) :

– un point de vue absolu pour lequel le repère est un repère cardinal indépendant de tous les objets considérés et de tous les acteurs observant la scène ;
– une approche relative (ou déictique) pour laquelle le repère n'est pas un des objets de référence mais un acteur qui observe la scène ;
– une approche intrinsèque qui consiste à utiliser un des objets comme référence du discours.

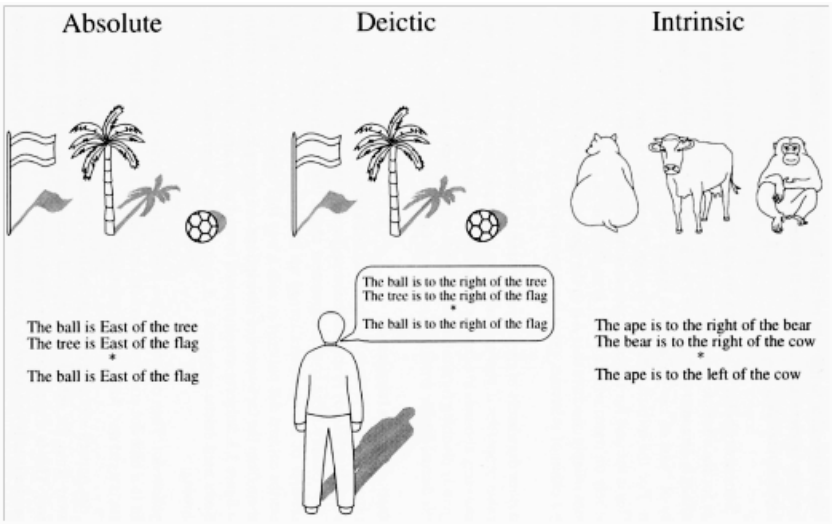


FIGURE 1.2 – Repères de référence identifiés par [Lev96]

1.1.2 Documents considérés : du récit de voyage au texte construit

[GP04] propose une typologie des documents incorporant de l'information géographique et utilisables avec des élèves :

1. Le premier type est le "document source". C'est l'outil à partir duquel travaillent les chercheurs en histoire, souvent un texte ou un document iconographique. Cette notion de document source n'est pas toujours évidente en histoire, car son statut et sa valeur peuvent être des sujets de débat. Elle l'est encore moins en géographie. Les chercheurs géographes utilisent assez peu d'outils qui sont qualifiés de "document source". Ils construisent plutôt par eux-mêmes leurs outils de recherche et leurs sources (enquêtes, interviews).
   En géographie, les cartes et inventaires topographiques, géologiques, de végétation, de risques (en fait toute carte thématique cherchant à rendre compte de manière exhaustive d'un espace sous un angle quelconque), les photos aériennes verticales, obliques, les images satellites, les données statistiques issues de recensements pourraient être considérées comme des "documents sources".

2. Le second type de document est le document "produit de la recherche". On y peut trouver tout document extrait de la publication d'un universitaire : texte, graphique, tableau statistique, croquis, schéma... Il pose souvent le même problème de compréhension par les élèves que la source géographique.
3. Le troisième type de document est le document sélectionné dans l'actualité. Il s'agit d'un type surtout propre à la géographie (ou à l'éducation civique) : sélection d'un article de journal, d'un document de communication voire de publicité.
4. Le dernier type enfin est le document construit. Il peut s'agir d'un document adapté pour rendre les deux premières catégories de documents accessibles à des utilisateurs de cette information. Ces documents sont souvent produits spécifiquement dans un but de transmission des connaissances. Il peut s'agir de cartes, croquis ou schémas, d'organigrammes, de tableaux statistiques, de graphiques qui constituent autant de représentations de la réalité.

Les informations géographiques que contiennent ces documents sont très variables. L'une des caractéristiques commune à ces divers types de documents est qu'ils mobilisent chez le lecteur des repères spatio-chronologiques [Guy01]. Ce sont ces types de documents sur lesquels nous allons focaliser notre attention et dans les prochains paragraphes, nous décrivons plus précisément les caractéristiques de deux types de documents : des topoguides décrivant un itinéraire à parcourir (guide de randonnées à pied...) et des récits de voyages relatant un déplacement réel ou fictif de façon chronologique.

1.1.2.1 Topoguides

[Etc99] a défini le topoguide en tant que "*document touristique et commercial qui présente les différents itinéraires de randonnée balisés d'un espace touristique. Chaque topoguide réalisé par la Fédération Française de la Randonnée Pédestre (150 titres environs) se compose par exemple* :

- *d'un plan général pour une visualisation de l'ensemble des itinéraires décrits ;*
- *du tracé des itinéraires reproduit sur un fond de carte topographique I.G.N. au 1/50 000 avec, en vis-à-vis, le descriptif détaillé, le temps de parcours moyen et les distances...*"

Les topoguides sont par essence riches en informations géographiques, mais ils diffèrent des textes descriptifs purement géographiques car les descriptions qu'ils contiennent ne sont pas uniquement faites dans le but de tracer un itinéraire sur une carte. Ils décrivent également des itinéraires à vivre et peuvent donc proposer des points de repères pertinents mais qui ne figurent pas nécessairement sur une carte. La figure 1.3 (extraite de `www.ffrandonnee.fr`) est un exemple de topoguide extrait du chemin de grande randonnée GR10.

FIGURE 1.3 – Exemple de topoguide : circuit de Lamotte

Il existe différents types de topoguides : orientés temps de randonnée, distance, dénivelé, et évidemment des topoguides adaptés à des enfants. Quels que soient leurs types, ils mêlent deux types de discours :

- le discours injonctif (ou prescriptif) donne les indications formelles pour réaliser le trajet. Ce mode a pour but de donner des ordres ou des conseils, il veut faire exécuter quelque chose à quelqu'un…
- le mode descriptif peut aller jusqu'à donner le point de vue subjectif de l'auteur.

Dans les topoguides, le lecteur devra interpréter ce qui lui est donné, il aura également à établir des liens entre la représentation visuelle d'un espace donné (carte, croquis) et la description textuelle d'un itinéraire.

1.1.2.2 Récits de voyages

C'est un genre littéraire à part-entière (voir www.cafe.umontreal.ca/genres/) dans lequel l'auteur raconte ce qu'il a vu dans un autre pays. Le rapport particulier que l'auteur entretient avec son objet le distingue d'emblée des autres types de récit. Contrairement au roman qui forme un univers clos, autonome, à l'abri des aléas du réel, le récit de voyage est ouvert sur le monde extérieur et soumis à ses règles ; le réel a priorité sur la fiction. "*La mise en évidence de la trame spatio-temporelle d'un discours suppose, d'une part, de pouvoir situer dans l'espace-temps les événements et les objets décrits et, d'autre part, de pouvoir détecter l'évolution spatio-temporelle de ces entités au fur et à mesure que le discours progresse.*" [Bra08].

Le récit de voyage s'élabore en deux temps. Il y a d'abord le voyage, où l'auteur du récit à venir entre en contact avec des réalités nouvelles, les découvre et les explore (le récit de voyage est lié à l'inconnu, à l'étranger, à l'inédit). Ensuite, il y a le récit, où l'auteur raconte les événements qui ont eu lieu durant son voyage, fait un compte rendu de ses explorations, rapporte ses découvertes, bref cherche à faire voir ce qu'il a vu.

Il existe une grande variété d'auteurs ayant écrit des récits de voyage (archéologues, ethnologues, voyageurs, peintres voyageurs, scientifiques voyageurs, écrivains flâneurs, journalistes...), ces récits oscillant entre : "narration mobile" et "narration post-mobile".

La Bibliothèque nationale de France (BnF) offre notamment un catalogue thématique sur son site Gallica (http://gallica.bnf.fr/dossiers/html/dossiers/VoyagesEnFrance/) (Figure 1.4). Ce catalogue est organisé selon une typologie qui mêle intentions, thématiques, et statuts de l'auteur. Notre équipe de recherche a aussi pu bénéficier des textes mis à disposition par la Médiathèque InterCommunale à Dimension Régionale (MIDR) de la Communauté d'Agglomération de Pau Pyrénées (CAPP).

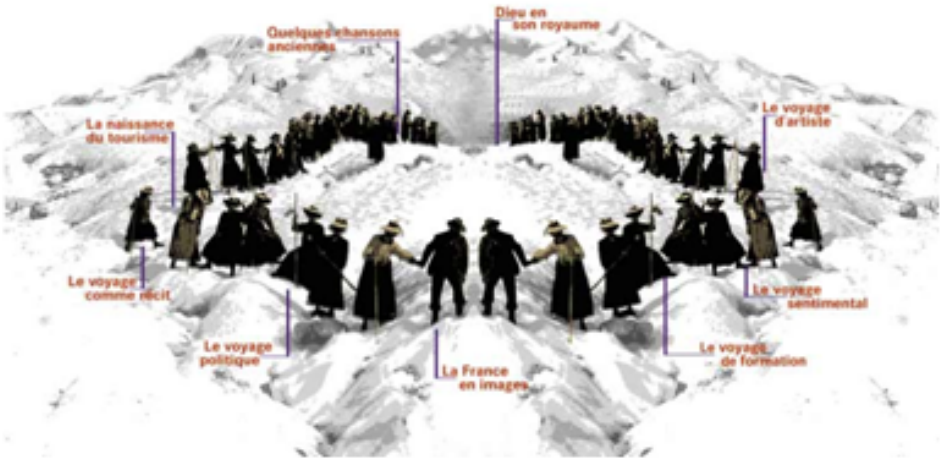


FIGURE 1.4 – Menu d'accueil thématique - Gallica

1.1.3 Représentation des informations géographiques

Dans le cadre de son stage de Master 2 Recherche en Sciences de l'Education, Virginie Paillas [Pai11] a particulièrement étudié les représentations multiples et complémentaires de l'information géographique.

1.1.3.1 Représentation de l'information spatiale

Représenter visuellement ou étudier l'espace nécessite d'effectuer des choix car différents types de représentations sont disponibles et chaque type aura des avantages et des inconvénients, et sera plus ou moins adapté à l'espace concerné.

Pour la représentation en deux dimensions, la photographie peut sembler la représentation prioritaire de l'espace réel mais l'observateur est soumis à la vision partielle du paysage que le photographe a choisi de prendre. Cependant, la photographie propose plusieurs niveaux de point de vue qui, en se complétant, permettent tour à tour la description physique du paysage et l'analyse de l'aménagement de l'espace.

A. Photographie

[Cla07] caractérise trois types de photographies utilisées en géographie :
- La "*photographie au sol*" propose une vision de proximité, proche de ce que l'œil est habitué à voir, mais par conséquent une vision limitée du paysage (Figure 1.5(a)).
- La "*photographie en oblique*" (photographie aérienne, du haut d'un promontoire...) propose un champ de vision plus vaste tout en conservant le rendu des trois dimensions (présence des ombres notamment).
- La "*photographie à la verticale*" (certaines vues aériennes, vues satellites) (Figure 1.5(b)) fait disparaître le relief. Sur ces représentations, la lecture du paysage devient plus difficile puisque le relief disparait mais ce type de prise de vue propose un rendu qui peut aider à réaliser la transition intellectuelle pour le passage à la carte : une transition entre la réalité du paysage vu de haut et la représentation abstraite, schématique, codifiée de la carte.

(a) Photographie d'éoliennes, Longeau Percey (52)

(b) Vue à la verticale, Google Maps, Longeau-Percey (52)

FIGURE 1.5 – Vues photographiques d'un espace

Les élèves peuvent également produire ou utiliser des représentations abstraites.

B. Représentations abstraites

Le croquis du paysage (Figure 1.6(a)) est un dessin simple et schématique qui permet une analyse du paysage. "*Un croquis cartographique a pour but de mettre en évidence les éléments essentiels d'une situation géographique par des moyens graphiques. Il permet de mettre en évidence les grandes lignes d'organisation et d'aménagement de l'espace étudié, ainsi que les principaux éléments du paysage et si possible leurs relations... En général, un croquis de paysage doit être orienté par rapport au nord. Ce n'est pas toujours facile à trouver. Le croquis comporte souvent une échelle numérique ou graphique, un titre et surtout une légende... L'objectif n'est pas de tout représenter, mais de faire figurer les grands ensembles et leurs relations...*" [Cla07].

(a) Croquis de la Place du Capitole - Toulouse. Odona, 4 ans

(b) Carte IGN, quartier du Capitole, Toulouse. Géoportail

(c) Plan local de stationnement, Toulouse. En rouge les rues à tarif normal et résidant 18h à 9h. En orange les rues à tarif normal et résidant 24h/24h

FIGURE 1.6 – Différentes représentations de la Place du Capitole, Toulouse

La carte (Figure 1.6(b) et Figure 1.6(c)) comme le croquis sont des représentations graphiques qui adoptent la vision à plat d'une photographie à la verticale. Cela permet de représenter une ville, une région, un pays... Le Comité Français de Géographie définit la carte comme étant une "*représentation conventionnelle, généralement plane, en positions relatives, de phénomènes concrets ou abstraits, localisables dans l'espace*". Fabien Guillot sur son site (`www.geographie-sociale.org`) étend cette définition en y ajoutant l'idée que la carte est une simplification de la réalité : ainsi pour lui la carte est "*une représentation géométrique, plane, simplifiée et conventionnelle de tout ou partie de la surface terrestre dans un rapport de similitude qui est l'échelle. Tout objet localisable dans l'espace (et tout phénomène affectable à un tel objet) est donc susceptible d'être représenté géographiquement sur une carte.*"

Il existe différents types de cartes liés à la diversité des phénomènes pouvant être cartographiés. Nous retiendrons les cartes topographiques et les cartes thématiques :

- La carte topographique décrit l'espace en donnant ses caractéristiques naturelles (indications de relief, cours d'eau, sommets...) ainsi que celles créées par l'homme (villes, frontières, voies de communication...). Pour ce faire, la carte comporte des symboles permettent de représenter et de visualiser ces informations.
- La carte thématique donne des informations géolocalisées : carte météorologique, carte historique, carte démographique, carte économique... Comme pour le croquis, la carte est une représentation abstraite et demande des "clés d'interprétation" : la lecture de la légende va de pair avec la lecture de carte. Trois principaux éléments permettent l'interprétation et sont indissociables de toute carte : l'échelle (indique la correspondance entre les distances sur la carte et les distances sur le terrain), l'orientation et la légende.

1.1.3.2 Représentation de l'information temporelle

La théorie aspectuelle considère trois notions dépendantes relatives au temps [DG10] :
- "*Un état (non permanent) est borné par deux événements : un événement qui fait entrer dans l'état ; un événement qui en fait sortir.*
- *Un événement établit une transition entre un état antérieur (avant) et un état postérieur (après) ; il comporte donc un début et une fin.*
- *Un processus (inaccompli) exprime une évolution saisie dans son développement ; il implique un premier instant mais pas la prise en compte d'un dernier instant de réalisation (qui est alors un terme d'accomplissement) ; lorsque le processus devient accompli (donc avec un dernier instant), il engendre un événement.*"

[Cos04] analyse la notion de temps et décrit cinq composants fondamentaux : la succession, la durée, l'irréversibilité, le cycle allié à la linéarité, l'horizon temporel. Pour connaître et se représenter les jours de la semaine ou les mois de l'année, il est nécessaire de prendre en considération un ordre de succession (la suite des jours ou des mois) et des durées (celles de chaque jour de la semaine entière et de chaque mois de l'année). L'ordre de succession des jours et des mois est fixe et les durées s'écoulent dans un sens (nous ne pouvons revenir en arrière) : c'est la notion d'irréversibilité. Cette irréversibilité donne à la progression des jours et des mois un caractère linéaire, mais cette linéarité s'allie à un aspect cyclique : celui du retour du début de la semaine et de l'année (après dimanche, c'est lundi qui recommence ; après décembre, c'est à nouveau janvier). Enfin, un jour de la semaine peut être le jour présent, ou un jour passé ou à venir : il s'agit de la notion d'horizon temporel.

Ainsi, les facteurs à prendre en compte sont multiples et parfois antagonistes : chronologie, continuité/rupture, instant/durée, temps linéaire/temps cyclique... Comme pour l'espace, il existe en fonction de ce que l'on souhaite observer ou figurer des représentations plus ou moins adéquates. Les représentations conventionnelles sont des représentations spatialisées du temps telles que la ligne, le cercle, parfois la spirale.

A. Représentations du temps les plus fréquentes

Les frises chronologiques (Figure 1.7) sont souvent utilisées. Ces représentations semblent par nature dédiées à représenter le passé, elles permettent de visualiser un temps long, et parfois lointain.

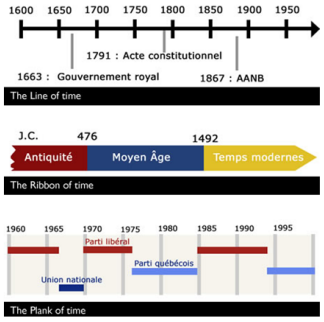


FIGURE 1.7 – Exemples de frises chronologiques

[BL00] décrit l'intérêt des frises par ce qu'elles permettent de montrer concrètement :
- "*la notion de durée, à condition que l'échelle du temps corresponde à une échelle de longueur cohérente et homogène tout au long de la frise (ex : 1 cm pour 1 an).*
- *la notion de chronologie, c'est-à-dire la place des événements les uns par rapport aux autres.*
- *la notion de diachronie, c'est-à-dire le déroulement dans le temps d'un phénomène, qui est perçue dans la lecture horizontale de la frise.*
- *la notion de synchronie, c'est-à-dire la simultanéité d'événements et de phénomènes au même moment, à la même époque. Elle apparaît dans la lecture verticale dans des registres différents.*

La frise est une construction progressive des repères temporels assimilés au fur et à mesure que le temps passe."

Les représentations cycliques ou parfois spiralaires (Figure 1.8) renvoient à l'aspect immuable de certains phénomènes.

B. Représentations tabulaires

Associées au calendrier, les représentations tabulaires (Figure 1.9) sont visibles au quotidien telles que l'agenda, l'emploi du temps... Ces représentations semblent plutôt dédiées à représenter le futur, avec pour ambition de permettre d'organiser des événements du futur. Cette visualisation est utile lorsque l'échelle du phénomène observé ne dépasse pas l'année.

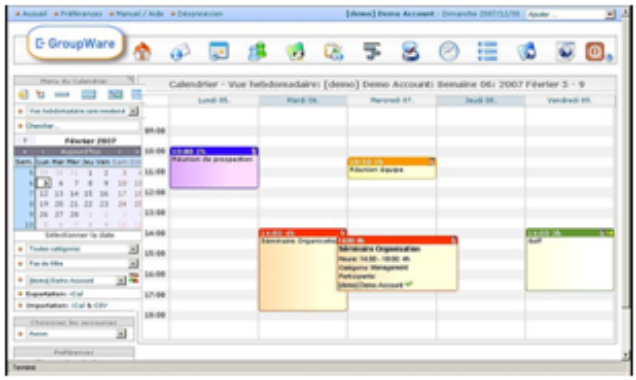
(a) Calendrier Maya, Musée National d'Anthropologie de Mexico

(b) Jeu "Les saisons de Ravensburger"

(c) 63 arômes de Cognac classés selon le cycle des saisons

FIGURE 1.8 – Différentes représentations calendaires

(a) Calendrier extrait du Rustican, de Pietro Crescenzi, MS 340, Musée Condé, Chantilly



(b) Vue hebdomadaire du calendrier

FIGURE 1.9 – Représentations temporelles tabulaires

Il est enfin à noter que les différentes représentations du temps, qu'elles soient conventionnelles ou plus personnelles renvoient à des manipulations intellectuelles proches de celles évoquées pour la compréhension des représentations de l'espace : notion de point de référence, d'échelle, de découpage... Face à sa propre perception du temps, à la lecture de ses représentations, il s'agira là d'être aussi capable d'interpréter les informations présentées.

1.2 L'information géographique : un objet d'apprentissage pour les pédagogues

L'équipe T2i (*Traitement des Informations Spatiales, Temporelles et Thématiques pour l'Adaptation de l'Interaction au Contexte et à l'Utilisateur*) est une équipe du laboratoire LIUPPA (*Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour*). T2i réalise ses recherches dans le domaine du traitement de l'information à références spatiales, temporelles et thématiques. Ces informations proviennent de documents numérisés ou du contexte d'utilisation d'objets communicants (flux de données et informations spatio-temporels issus et à destination de PDA, capteurs, téléphones, ou plus généralement de tout périphérique mobile).

L'objectif de l'équipe est de proposer des formalismes et des méthodes pour l'extraction d'informations et leur valorisation, pour la conception, la mise en œuvre et l'utilisation de systèmes interactifs porteurs de ces informations spatiales, temporelles et thématiques. Dans cette section nous considérons la valorisation de l'outillage mis au point dans le cadre des thèses de l'équipe [Les07], [Lou08]... (*cf.* Annexe A pour plus de détails). Partant des résultats de ces travaux, nous abordons la problématique de situations d'usage de l'information géographique avec une focalisation sur les situations d'apprentissage.

1.2.1 Référence aux savoirs enseignés

Pour [CP10], les questions des rapports au temps et à l'espace sont au cœur des façons de se comporter mais aussi de percevoir et de penser le monde et les relations humaines. [AM10] ajoute que passé-présent-futur et ici-ailleurs sont considérés comme des cadres pour penser le monde autant que se penser soi-même dans le monde et la société : savoir produire sur soi un récit raisonné, en dresser un croquis, pouvoir penser l'autre dans ce qu'il a de commun et de différent...

La structuration du temps et la construction progressive de l'espace sont des apprentissages qui sont travaillés dès la maternelle et poursuivis durant toute la scolarité primaire. Ces apprentissages longs et progressifs sur lesquels viennent s'appuyer d'autres apprentissages ont un rôle déterminant pour la suite de la scolarité et pour le citoyen de demain.

Progressivement, l'enfant va passer du temps vécu au temps perçu (montre, heure, histoire racontée) puis au temps conçu (tri, classement). Dans le même temps, lorsqu'il disposera d'une représentation orientée de son propre corps, il pourra commencer à s'en servir pour organiser l'espace qui l'entoure et verbaliser de manière plus assurée les relations spatiales en apprenant les marques de l'énonciation structurant l'espace à partir de celui qui parle.

Toutefois les notions de temps et d'espace sont des notions complexes qui se construisent progressivement. C'est surtout au cycle des approfondissements (cycle 3 : CE2, CM1 et CM2) et au collège que ces notions sont particulièrement travaillées. Les instructions officielles (*cf.* Annexe B) insistent sur la nécessaire acquisition des repères spatiaux et temporels, indispensables au développement cognitif et psychologique de l'enfant. Ce sont des concepts longs à appréhender et qui nécessitent une progressivité dans les apprentissages.

La lecture des différents textes officiels montre que l'acquisition d'une culture humaniste commune est liée à un enseignement articulant différentes disciplines et demandant de comprendre et d'utiliser différents types de documents. Les capacités évoquées dans le socle commun comme celle qui est de "lire et pratiquer différents langages", renvoient à la nécessité de confronter les élèves à différentes représentations visuelles (notamment des représentations de l'espace, des représentations du temps à associer à des textes que l'apprenant pourra annoter).

Les textes basés sur des itinéraires (récits de voyages, topoguides) ne figurent pas explicitement dans les recommandations des programmes, mais les récits de voyages (dans une acceptation plus grande pouvant intégrer les récits fictionnels) sont bien présents dans les listes d'œuvres. Les progressions de français mettent en avant des compétences liées à l'interprétation des textes basées sur des activités de repérage, de prélèvement d'informations explicites et implicites.

Le paragraphe suivant propose des éléments théoriques sur les pratiques d'annotation instrumentées.

1.2.2 Des scénarios d'apprentissage pour former de bons lecteurs

Dans cette section, nous abordons successivement le rôle particulier des annotations placées par un utilisateur/un système sur un texte, une carte ou une frise chronologique, puis nous proposons quelques exemples de scénarios ayant fait l'objet de travaux avec des enseignants de terrain.

1.2.2.1 Le rôle particulier des annotations

Dans sa thèse concernant l'annotation sémantique des documents pédagogiques, [Mil05] considère les pratiques d'annotation comme des activités et produits de la lecture active, et comme une trace de l'engagement du lecteur avec le texte. Le concept de lecture active a été introduit par [AVD72], pour distinguer l'ensemble des activités associées à la lecture de la simple consultation de mots sur une page [Rob01]. [SGP98] la perçoit comme une lecture enrichie : "*la lecture active n'implique pas seulement la lecture en soi mais aussi souligner, surligner et écrire des commentaires, que ce soit sur le texte lui-même ou sur un cahier à part*". Ainsi, cette lecture "*transforme le lecteur en acteur, qui enrichit le document*" [Rob01]. Selon [RPC07], "*la lecture active est*

une activité composée d'une lecture critique et d'une production par rapport à un document. Un lecteur se trouve en situation de lecture active dès lors qu'il effectue une lecture approfondie d'un document, pour s'en approprier le contenu, pour y rechercher des informations précises, ou même plus simplement pour en permettre une relecture simplifiée". [SGP98] considère que la lecture active est une partie fondamentale de l'éducation et du travail d'apprentissage, car elle combine la lecture avec des pensées critiques.

[Mil05] note qu'au niveau des pratiques pédagogiques au Québec, "*la lecture active est une discipline enseignée aux apprenants*". Par exemple, le site Web de José E. Igartua (www.igartua.ca), enseignant à l'Université du Québec à Montréal, contient des notes de son cours "*d'initiation au travail historique*" comprenant une partie sur la lecture active. Dans ce cadre, il cite une définition de la lecture active : "*annoter un texte de manière à obtenir une compréhension approfondie*". De même, la page du guide méthodologique du département d'histoire-géographie du Collège Ahuntsic de Montréal contient une partie sur la lecture active [1]. Pour les auteurs, "*la lecture active est une technique qui permet de rendre efficace et profitable la lecture d'un texte*". Elle est dite active car les apprenants doivent "*sans cesse annoter le texte en même temps qu'ils assimilent les connaissances contenues.*"

Cette dernière citation met en exergue le lien entre la pratique de la lecture active et les pratiques d'annotation. Des guides ou manuels donnent des conseils pour améliorer les compétences en lecture en pratiquant la lecture active ; ceux-ci mentionnent des activités d'annotation telles que le surlignage. D'autres mettent également en avant la schématisation comme support de compréhension : "*Le couple espace-temps est consubstantiel au roman et la clarification des repères en la matière est un élément souvent décisif de la compréhension de l'histoire, d'où l'opportunité de schématisations qui ont le mérite de visualiser les indications spatiotemporelles.*" [Bes96].

Les pratiques d'annotations sont variées et les définitions sont relatives à différents domaines de recherche. L'annotation est à la fois une activité et le résultat de cette activité. Pour ce travail, nous nous baserons sur la définition générique donnée par [Mil05] qui considère les annotations comme "*ce qui est ajouté à un document pendant la lecture, et qui est visible sur ce document*". [RPC07] a défini "*une annotation comme une inscription sur un document ou liée à celui-ci, décrivant, mettant en évidence ou ajoutant de l'information à une partie ou à la totalité d'un document. Les annotations dépendent donc principalement du support du document (numérique ou pas), de son type (texte, son, vidéo) et de leur ancrage dans le document*".

Nous ne retenons pas par contre la distinction qui est faite au sujet du support. Ainsi [Mil05] exclut dans son travail les ressources non textuelles (audios, vidéos, ni même des schémas, des dessins ou des graphes).

1. https://sites.google.com/site/latelierdhistoire/le-guide-methodologique/la-lecture-active

Concernant l'étude des formes d'annotation, [Mar97, Mil05] proposent un recensement des formes des annotations, qui sont réparties en quatre groupes :
- Texte surligné, souligné, entouré ou barré ;
- Symboles marginaux télégraphiques (astérisques), autres marques à l'intérieur du texte ;
- Notes marginales brèves, notes dans d'autres interstices textuels ;
- Notation appropriée en marge ou à proximité de figures ou d'équations.

Concernant l'étude des structures d'annotation :
- [Vé97] a proposé une structure composée de six propriétés : une forme, un objectif, un lieu, un auteur, une histoire et un support ;
- [Den00] a effectué un travail sur l'annotation dans le domaine du Web et a structuré une annotation en deux parties : une ancre exprimée dans le code HTML du document permet d'attacher l'annotation à une partie définie du document et un commentaire qui se compose des attributs (sujet, auteur, date de création...) ;
- [KK01] ont proposé Annotea qui est un outil d'annotation basé sur le Web où les annotations sont externes au document. Chaque annotation est une description RDF contenant plusieurs attributs tels que le type, l'auteur, le contexte, la date de création...

Concernant les pratiques électroniques d'annotation [Hua96, Mil05] proposent une classification des formes qui inclue les hyperliens, et trie les annotations "*selon leur finalité*" (Figure 1.10) : la mise en valeur, la création de liens ou bien les ajouts et modifications.

Mise en valeur	• Texte surligné, souligné, entouré, barré • Modification de la mise en forme typographie, la couleur, l'alignement
Lien	• Liens à partir du texte (référence ; le dictionnaire et les annexes de compulsion), • Liens vers le texte (marque page ; l'index et les annexes d'entrées), • Liens à double sens (références double ; sommaire et les annexes doubles)
Ajout et modification	• Ajout d'un objet (texte, graphique) en marge du texte – Objet annotatif : texte (commentaire, explication, traduction), graphique (courbe, dessin), des données (tableau, base de données, liste indexée), des formules, du son, de l'image, fixe ou animée – Remarque (ou note-it) : petit commentaire du genre "à relire" ou "à commenter" en marge du texte. Lorsque l'action est effectuée, le note-it est effacé. • Modification – Correction – Restructuration : suppression de passage, déplacement de passage, découpage d'un passage en 2, collage de deux passages consécutifs.

FIGURE 1.10 – Classification des formes d'annotation selon "leur finalité" [Hua96]

[Mil05] remarque dans cette classification la présence des remarques, qui sont des annotations actives. Ces remarques ne servent pas "*à donner une information qualitative ni un complément d'information. Elles sont là dans le but d'attirer l'attention du lecteur sur un passage, avec l'objectif sous-jacent de l'amener à effectuer une action*". Dans la mesure où ces remarques engagent une action à réaliser, [Mil05] parle d'annotation active : "*Avec l'informatique, ces remarques peuvent déclencher une action, et non plus seulement donner une instruction d'action au lecteur*". [Hua96] considère que la différence entre ces remarques et des commentaires est que les premiers sont provisoires : "*lorsque l'action est effectuée, la remarque est effacée*".

Dans le cadre de ce manuscrit de thèse, nous considérons l'annotation comme un objet de lecture active au sens de [Mil05] dans la mesure où l'annotation peut déclencher une action de la part des lecteurs. Nous retenons également le concept d'*effet* de l'annotation dans le travail de [Hua96] avec Note-it. En effet, selon [Hua96], il est possible qu'*une annotation commande une action du système comme par exemple "provoquer une sonnerie"*. Nous portons attention aux finalités et aux objectifs des annotations, en particulier parce qu'elles interviennent dans la compréhension d'un texte, la production de textes et de cartes et la confrontation de documents par la pratique d'annotations. Le travail mené avec de nombreux enseignants du primaire et du collège au cours des stages de Master 2 Recherche en Sciences de l'Education de Virginie Paillas [Pai11] et de Pascal Nodenot [Nod11] nous a amené à lister trois types d'usage qui peuvent contribuer à :

- enrichir un propos, c'est-à-dire apporter un complément d'information en établissant des liens avec des éléments externes au support (photos géolocalisées, éléments textuels sur frises chronologiques...) ;
- interpréter et commenter le document en cours de lecture ;
- corréler l'information, c'est-à-dire mettre en évidence des liens de compréhension en liant des documents (sélectionner une date dans un texte et la déposer pour la synchroniser sur une frise chronologique, sur un lieu remarquable...).

Au sens de [Hua96], il s'agit clairement d'annotation de types "*remarques*" qui sont à la fois des traces de l'activité cognitive de l'apprenant et des moyens de mise en oeuvre de cette activité. À la fin de l'activité, lorsque le but est atteint, ces traces méritent donc d'être conservées par l'enseignant même si l'apprenant estime ne plus en avoir besoin, d'où l'intérêt de systèmes permettant de conserver, éditer, rejouer ces traces posées par l'apprenant. Plusieurs équipes de recherche mènent des travaux spécifiques dans le domaine, tant au niveau national [BV07, Iks12, Car11] qu'international.

1.2.2.2 Quelques exemples de scénarios

Un travail de fond conduit avec des enseignants du primaire et du collège sur deux années (2008 et 2011) nous a permis d'identifier et de valider l'intérêt pédagogique de plusieurs scénarios de lecture active conduisant les apprenants à étudier conjointement et corréler des textes (récits de voyage, topoguides), des cartes géographiques et des frises chronologiques. Nous présentons ici certains de ces travaux sous la forme de deux

scénarios typiques qui constituent des exemples non exhaustifs de situations à pouvoir décrire et opérationnaliser.

Les deux sous-sections suivantes présentent deux scénarios pertinents présentés sous la forme d'une fiche de synthèse en treize points créée par des enseignants de terrain. Ces fiches ont été rédigées selon un modèle préconisé par l'Education Nationale pour faciliter la desciption et la recherche d'informations dans un contexte de diffusion/partage de ces fiches par des Professeurs des Ecoles.

A. Scénario 1 issu du stage de M2R de Nodenot [Nod11]

Niveau : Cycle 3 et Collège.

Objectifs visés : Percevoir les différents lieux d'un roman.

Contexte de mise en œuvre de l'activité : Activité qui s'insère dans un scénario plus général d'étude de ce roman.

Objectifs impliqués :
- Repérer les indices de lieux ;
- Repérer celui ou ceux qui situent l'ensemble du récit (par exemple : "à New York") ;
- Repérer ceux qui situent les événements les uns par rapport aux autres (par exemple : "dans Central Park", "sur la 5e avenue", "dans Chelsea"...) ;
- Repérer la valeur de l'indice de lieu (où on est, où on va, d'où on vient, par où on passe).

Compétences du socle : Repérer des informations dans un texte à partir de ses éléments explicites et des éléments implicites nécessaires (pour parvenir à une compréhension littérale du texte : rechercher, extraire, organiser des informations explicites pour saisir et construire le sens d'un texte).

Résumé de l'activité : L'enseignant a repéré dans le roman les informations à dimension spatiale. Il va les insérer dans le composant-Texte (en indiquant pour chaque extrait la page du livre correspondante). Les élèves vont positionner sur le composant-Carte des balises qui correspondent aux lieux repérés. Le choix de la balise dépend du type de lieu appréhendé (lieu où le héros s'est arrêté, est simplement passé, lieu d'où il vient...).

Plan du déroulement de l'activité :
- Par groupes de deux, les élèves surlignent l'information dans le texte et posent une balise sur le composant-Carte.
- Au Tableau Blanc Interactif (TBI), lecture collective des différentes productions, débat et validation.

– Mise à jour collective du composant-Carte après validation par la classe.

Note pédagogique : Cette activité est à mener au fur et à mesure de la lecture du roman. Elle oblige l'élève à s'intéresser aux paragraphes descriptifs qui sont parfois "sautés" par les lecteurs. On pourrait imaginer d'autres possibilités comme dans un premier temps, l'ajout des photos sur la carte ou encore dans un deuxième temps, la pose des balises en fonction des extraits du livre affichés dans le texte. La lecture collective des différentes propositions permettra de répondre à des questions sous-jacentes : pourquoi s'est-il arrêté, pourquoi si peu ou si longtemps, pourquoi n'a-t-il fait que passer...

Évaluation : En prenant appui sur le texte, faire construire une figure, un diagramme, un schéma... à partir des informations spatiales décrites. On pourra proposer de faire repérer dans un texte littéraire ou documentaire les indications de lieu et les actions d'acteurs en lien avec ces lieux.

Maquette de l'application imaginée pour cette activité (Figure 1.11) :

Une application que l'enseignant veut mettre à disposition des élèves comporte un composant-Texte et un composant-Carte avec des fonctionnalités adaptées à l'activité. La maquette se compose de deux composants : un composant textuel sur la partie gauche et un composant cartographique sur la partie droite. Chaque composant est doté d'outils spécifiques précisés ci-dessous.

FIGURE 1.11 – Maquette1 avec un texte et une carte (Scénario 1)

Outils associés aux composants :

Pour le composant textuel, un outil d'annotation doit permettre de surligner un segment du texte et un outil loupe permet d'afficher le texte en plein écran. Pour le composant cartographique, quatre balises doivent être positionnées sur la carte (par glisser-déposer). Il y a aussi une corbeille pour supprimer une balise (par glisser-déposer) et un outil loupe pour afficher la carte en plein écran. Les liens entre texte et carte ne peuvent s'effectuer que lorsque les deux composants sont à l'écran.

L'élève doit surligner un élément du texte puis poser une balise associée sur la carte. On ne peut créer de lien entre texte et carte que via le texte : seul le texte autorise à mettre la balise sur la carte. Les quatre balises associées au composant cartographique sont présentées dans la figure 1.12.

FIGURE 1.12 – Présentation des quatres balises

Comportement (lien texte-carte) : L'utilisateur annote le texte en surlignant les éléments qu'il souhaite voir symbolisés par une balise sur la carte. Il double-clique sur un élément annoté. Il a ensuite la possibilité de faire glisser la balise choisie sur le composant-Carte puis de valider. La balise n'est alors plus déplaçable. Le système crée alors le lien entre l'élément du texte surligné et la balise posée. Le clic sur un élément surligné du texte met la balise associée en surbrillance. Le clic sur l'icône loupe affiche le texte en plein écran (Figure 1.13) pour pouvoir surligner de manière plus aisée l'ensemble des éléments dont une balise sera posée sur la carte. Une fois le texte annoté, il y a possibilité de revenir sur l'interface à deux composants pour positionner les balises (double-clic sur un élément surligné du texte).

FIGURE 1.13 – Composant textuel en plein écran

Comportement (lien carte-texte) : Le clic sur une balise de la carte met en surbrillance l'élément du texte associé. Le double-clic sur une balise permet, par glisser-déposer, de déplacer la balise sur la carte ou bien de la faire glisser dans la corbeille, la supprimant de la carte et supprimant également la relation entre la balise et l'élément du texte associé. Le clic sur l'icône loupe affiche la carte en plein écran (Figure 1.14).

FIGURE 1.14 – Composant cartographique en plein écran

B. Scénario 2 issu du stage de M2R de Paillas [Pai11]

Niveau : Cycle 3 et Collège.

Objectifs visés : Vivre un topoguide de randonnée et en rendre compte.

Contexte de mise en œuvre de l'activité : Activité qui s'insère dans un cadre de sorties scolaires (à pied et/ou à VTT) ou de classes de découverte.

Objectifs impliqués :
- Rédiger un texte relatant un itinéraire vécu ;
- Repérer les indices spatiaux et temporels présents dans le texte pour pouvoir les exploiter sur la carte et la frise chronologique de sorte à pouvoir confronter ces repères à l'itinéraire vécu par les élèves.

Compétences du socle :
- Rédiger un texte en utilisant ses connaissances en vocabulaire et en grammaire ;
- Représenter un déplacement à partir de repères pris sur une carte ;
- Lire et utiliser des cartes et des croquis ;
- Restituer les événements selon un ordre chronologique ;
- Mettre en relation des événements vécus à une période donnée ;
- Associer des événements à une période, les ordonner et les replacer dans le contexte d'un déplacement.

Résumé de l'activité :
L'enseignant veut mettre à disposition des élèves une application comportant un composant-Texte, un composant-Carte et un composant-Frise chronologique avec des fonctionnalités adaptées à l'activité.

Pour ce scénario, on demande aux élèves de s'appuyer sur leurs souvenirs pour rédiger un texte relatant un itinéraire vécu et tracer sur une carte cet itinéraire vécu. Le tracé du trajet en repérant le sentier sur une carte est le cœur de l'activité. On allège la tâche des élèves du côté de la lecture de texte en leur proposant une annotation automatique Texte-Carte qui permet de placer des points remarquables sur la carte. On laisse les élèves en situation de se repérer dans le temps et/ou dans l'espace pour tracer le trajet (annotation de corrélation Carte - Frise). Les élèves vont annoter sur le composant-Texte des informations spatiales, temporelles et positionner (automatiquement ou manuellement) ces informations sur le composant correspondant (Carte ou Frise).

Plan du déroulement de l'activité :
- Par groupes de deux, les élèves surlignent les informations spatiales puis les informations temporelles dans le texte. Les balises sont automatiquement posées sur le

composant-Carte. Ils posent également des éléments de texte sur la frise chronologique.
- Les élèves complètent l'itinéraire sur la carte en traçant manuellement l'itinéraire vécu entre les points remarquables identifiés.
- Au Tableau Blanc Interactif (TBI), lecture collective des différentes productions, débat et validation.
- Mise à jour collective du composant-Carte après validation par la classe.

Note pédagogique : Cette activité peut permettre la lecture active pour les élèves. On pourra demander aux élèves de modifier le texte qui a servi de point de départ pour le rendre conforme à la carte de l'itinéraire vécu qui aura été validée par le groupe classe.

Évaluation : En prenant appui sur le texte, faire construire une carte contenant les lieux à partir des informations spatiales décrites ou une ligne de temps du trajet cité dans le texte.

Maquette de l'application imaginée pour cette activité (Figure 1.15) :

FIGURE 1.15 – Maquette2 avec texte, carte et frise chronologique (Scénario 2)

La marquette d'interface se compose de trois composants (Texte, Carte, Frise) et une barre d'outils permettant d'annoter. L'utilisateur sélectionne l'outil et intervient directement sur le composant pour l'annoter. En fonction du composant, l'outil a un comportement spécifique.

Déroulement d'une annotation simple : L'utilisateur se sert de l'outil "marqueur" pour annoter, c'est à dire il surligne des parties du texte pour identifier une information ; puis il peut placer des annotations sur la carte ou sur la frise chronologique. Le choix de la couleur de l'annotation est libre même si elle a pourtant un sens puisqu'il existe une légende.

Comportement (Interactions utilisateur) : Chaque composant a ses propres caractéristiques. Des outils permettent d'intervenir de façon globale sur les composants sous la forme d'activité d'annotations et d'ajout de contenus associés aux composants.

Composant (Interactions entre les composants) : La synchronisation entre les trois composants permet d'associer sur différents composants la même information issue des annotations (voir l'application créée depuis un récit de voyages de J.D. Forbes [Bou07] en Annexe A.2, Figure A.9).

Annotation de corrélation : A partir du moment où un élément est annoté, il est possible de le lier à un autre élément. Lorsque l'utilisateur clique une fois sur l'élément annoté, il est sélectionné, puis il est possible d'ajouter une annotation de corrélation par glisser-déposer (le curseur change de forme de flèche par exemple) et l'utilisateur fait glisser vers une zone cible (si l'utilisateur part du texte on peut annoter vers la carte ou la frise chronologique). Par exemple, l'utilisateur choisit de surligner "*Cauterets*" en bleu, le mot est surligné en bleu. Quand il clique sur le mot surligné et qu'il effectue un glisser-déposer sur la frise chronologique, un point bleu avec légende "*Cauterets*" apparait sur la frise.

Corrélation assistée (l'annotation de corrélation est automatisée entre le texte et la carte et permet de placer automatiquement des lieux sur le composant carte) : Lorsque l'utilisateur surligne un élément considéré en tant que lieu, il est repéré (affichage automatique) sur la carte. Ceci demande une préparation de la part de l'enseignant : pour le texte une liste de lieux est donc constituée et le système compare les lieux annotés et la liste pour procéder si nécessaire à la corrélation automatique. Par exemple, l'utilisateur annote dans le texte "*Cauterets*", le texte est annoté de la couleur choisie par l'utilisateur, un symbole de type lieu se place automatiquement sur la carte.

1.2.3 Bilan

Depuis 2008, l'équipe T2i a mené plusieurs études pour approfondir la notion d'information géographique en tant qu'objet d'apprentissage : conception d'applications éducatives, évaluation de prototypes avec des enfants (Figure 1.16), réflexion sur les pratiques pédagogiques dans les classes à propos de l'information géographique...

Les travaux de Master 2 Recherche en Sciences de l'Education de Paillas [Pai11] et de Nodenot [Nod11] conduits en parallèle à cette thèse ont permis de mieux formaliser le potentiel pédagogique d'applications valorisant des textes à connotations géographiques.

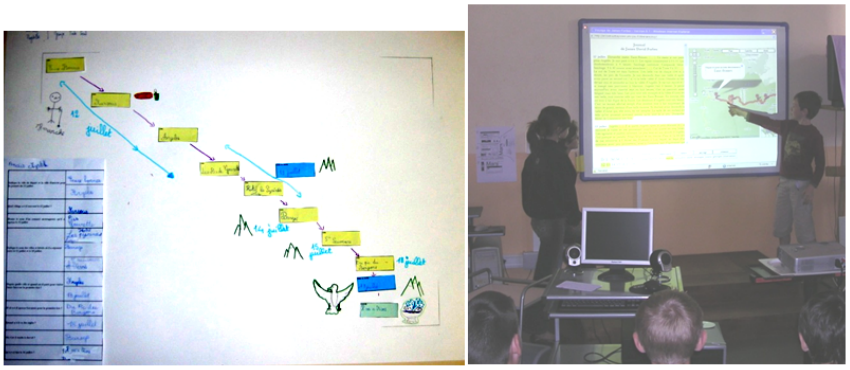


FIGURE 1.16 – Une production des élèves suite à une expérimentation en classe

Grâce aux collaborations mises en place avec les équipes d'enseignants, nous avons identifié des scénarios pédagogiques et des principes d'interaction au service des apprentissages identifiés dans les instructions officielles de l'Éducation Nationale (programmes disciplinaires et socle des compétences).

Le cadre ainsi défini nous amène dans cette dernière section du chapitre à définir les objectifs spécifiques de la recherche à conduire au service de la conception de ces applications éducatives afin qu'un enseignant de terrain soit en mesure de créer ces applications en toute autonomie.

1.3 Orientations scientifiques de la thèse

1.3.1 Constats de départ

Trois principaux constats sont à l'origine des travaux de thèse.

Premier constat : Au cours des dix dernières années, les travaux de la communauté internationale en EIAH dans le domaine de la spécification de scénarios ont permis des avancées dans de nombreux domaines :

- cycle de vie des scénarios, métaphores et langages de modélisation [IMS03, Laf04, VL07, EM10] ;
- langages et modèles pour la spécification d'interactions collaboratives entre apprenants [LVFAP+06, FMV07] ;
- opérationnalisation des scénarios pédagogiques par les techniques de transformation de modèles [VM03, Laf04, MVF+09, ATW08] ;
- environnements-auteurs pour scénariser des applications éducatives [Mia05, Dal06, PG07], environnements-auteurs pour la spécification du diagnostic et du tutorat associé à des activités d'apprentissage [AMSK09].

Ces travaux proposent des contributions nombreuses pour faciliter la conception d'EIAH, à la fois au niveau des approches de conception, des modèles, des langages de spécification et des environnements-auteurs. Toutefois, ces résultats ne peuvent être directement mis à la portée de non spécialistes en informatique qui doivent être assistés par des informaticiens ou des ingénieurs pédagogiques (possédant une double compétence dans les domaines de la pédagogie et de l'informatique). De plus, il n'existe pas de modèle permettant d'intégrer ces résultats et qui fournit un cadre de conception permettant à des non experts d'appréhender les différents aspects sus-cités relatifs à la conception d'un EIAH.

Deuxième constat : A part quelques travaux très ciblés notamment ceux menés à Pittsburgh au LearnLab (www.learnlab.org) par V. Aleven et son équipe pour concevoir des environnements-auteurs dédiés à la mise en œuvre de tuteurs cognitifs, il n'y a pas d'efforts de recherche pour aider des enseignants à décrire et évaluer le détail des interactions que doivent permettre les activités d'apprentissage instrumentées proposées aux élèves d'une classe. Les propositions de langages (*cf.* IMS-LD [2], E2ML [3] [Bot03], MOT+ [4] [PG07]...) et d'environnements-auteurs apportent des solutions :

- pour séquencer dans le temps différentes activités d'apprentissage (éventuellement collaboratives),
- pour assigner des activités aux apprenants en fonction d'objectifs pédagogiques préalablement spécifiés.

Ces propositions ne portent toutefois pas sur la conception d'objets d'apprentissage qui restent souvent prédéfinis (questions à choix multiples, questions à réponses ouvertes courtes...) ni sur les formes d'interactions proposées par ces objets (remplir un texte à trous, apparier des éléments de réponse...). Tout objet d'apprentissage sortant de ce cadre standard est considéré comme une boîte noire qui reste hors de portée du pédagogue non informaticien.

Troisième constat : Il est évident que, selon les domaines, il est plus ou moins facile de scénariser des activités s'intéressant au détail des interactions avec l'apprenant et donc au niveau cognitif. Pour le domaine mathématique, les travaux menés autour d'AplusX [Nic87], de Cabri Géomètre [LC94] ont montré que pour des domaines formels, il est possible de produire des outils-auteurs qui sont réellement utilisables par les enseignants.

Cependant, la scénarisation d'activités pédagogiques dans des domaines moins formels ("*ill-structured domains*") est peu abordée par la communauté scientifique. L'information géographique apparait comme un domaine relativement stable mais cependant moins bien formalisé que d'autres domaines tels l'algèbre [Nic87], la géométrie [LC94],

2. IMS - Learning Design, www.imsglobal.org/learningdesign
3. Educational Environment Modeling Language
4. Modeling with Object Types +

l'arithmétique [AMSK09], ou l'ingénierie [Hsi01] pour lesquels des environnements de conception spécifiques ont pu être mis au point. Le fait que le domaine géographique soit assez peu structuré soulève ainsi des difficultés supplémentaires pour mettre au point des traitements automatiques facilitant l'intégration du domaine au sein d'activités pédagogiques implantées dans un EIAH.

1.3.2 Objectifs

Le but des travaux de thèse est de proposer un cadre de conception destiné à des enseignants et répondant aux trois constats précédents. Pour la prise en compte du premier constat, il s'agit de proposer une approche couplant :

- des aspects méthodologiques accessibles à un non-informaticien et permettant de guider la conception sans toutefois imposer un cadre trop fermé bridant la créativité (source de richesse pédagogique et didactique) ;
- des langages accessibles à un non-informaticien lui permettant de spécifier une activité pédagogique de manière suffisamment précise (contenus manipulés, manières de les présenter et comportements associés) mais sans détails d'implémentation ;
- des modèles supports permettant de traduire automatiquement la spécification donnée sous forme d'une application exécutable et évaluable immédiatement.

Vis à vis du deuxième constat, nous souhaitons proposer des modèles et des langages permettant à l'enseignant-concepteur de spécifier des interactions sur des objets d'apprentissage moins classiques que ne le sont des QCM ou des textes à trous par exemple. Les stratégies pédagogiques privilégiées au sein de l'application sont de type constructivistes basées sur les situations de lecture active guidée par des buts. Les activités de lecture et d'annotation de documents permettront de répondre à des questions que se pose l'apprenant et pourront aller jusqu'à la résolution de situations-problèmes [Nod05].

Vis à vis du troisième constat, nous avons choisi d'examiner les interactions qu'il est possible de spécifier et d'implanter dans le domaine de la géographie. Pour valoriser les travaux précédents de l'équipe de recherche (*cf.* Annexe A), nous avons limité notre domaine d'étude qui intègre des concepts géographiques qui sont difficiles à formaliser mais qui présentent un intérêt certain en tant qu'objet d'apprentissage : comment le concepteur pédagogue peut-il manipuler et faire apprendre des relations d'orientation, de distance (telles que "*l'ouest de Paris*"), des relations entre deux entités géographiques (telle que "*la ligne Bayonne - Paris*"), etc. sachant bien sûr que ces relations peuvent en plus se combiner (Comment permettre à l'apprenant de se forger une représentation mentale de la région se trouvant à l'ouest de la ligne Bayonne - Paris).

Nos contributions visent avant tout les enseignants de terrain de niveau cycle 3 (élèves de CM1 et CM2 âgés de 9-10 ans) et collège. Les moyens mis à leur disposition doivent leur permettre de bâtir de "petites" applications de manière autonome afin qu'ils puissent les exploiter dans leurs classes. La taille des applications visées reste modeste dans le sens où il s'agit de produire des applications servant de support à des activités pédagogiques

et non pas des EIAH complets. Chaque application bâtie peut être considérée comme une brique support à une activité pédagogique, plusieurs de ces briques pouvant ensuite être fabriquées pour être intégrées par exemple dans un LCMS [5] tel que Moodle afin de les séquencer. Les dimensions pédagogiques telles que le diagnostic, le tutorat, le travail collaboratif, etc. ne sont pas prises en compte dans les travaux proposés. Dans le cadre de cette thèse, ces dimensions restent confiées aux plateformes dédiées sachant qu'elles constituent des problématiques de recherche à part entière.

D'un point de vue technologique, l'environnement de conception visé, ainsi que les applications géographiques qu'il permettra de générer seront exclusivement de type web. En effet, nous souhaitons éviter à l'enseignant-concepteur tout problème d'installation et de déploiement (non seulement de l'environnement de conception mais aussi des applications générées). De manière concrète, nous imaginons proposer un environnement de conception accessible depuis un navigateur et dont les applications générées pourront être téléchargées puis exécutées sur des postes clients bénéficiant d'une simple connexion internet.

1.3.3 Verrous / Difficultés

Par rapport aux objectifs précédemment décrits, nous identifions les deux principaux verrous scientifiques suivants.

Verrou 1 : Comment permettre à un non-informaticien de penser et modéliser une application ?

Les informaticiens le savent, la conception d'une application informatique, nécessite une rigueur et des modes de pensée spécifiques qui ont donné naissance à de nombreux travaux et contributions dans le domaine du génie logiciel. Ces techniques de génie logiciel s'acquièrent progressivement et avec de l'expérience ; elles font partie intégrante du métier d'informaticien. Bien qu'elles soient incontournables, ces techniques de spécification ne sont pas en général à la portée d'un concepteur non-informaticien et peuvent même aller à l'encontre de leurs schémas de pensée en bridant leur créativité.

La première difficulté que nous identifions est donc d'ordre méthodologique : Peut-on proposer une approche de conception qui soit suffisamment formalisée pour aller jusqu'à la production d'une application et qui soit également suffisamment flexible pour être acceptée par des non-spécialistes ?

Une deuxième difficulté réside dans la manière d'exprimer les interactions que pourra avoir un apprenant dans la manipulation d'une application à connotation géographique : Quels moyens d'expression donner à un non-informaticien pour qu'il spécifie clairement chacune des interactions qu'il imagine utiles lors d'une activité (pédagogique) ? Peut-

5. "*Learning Content Management System*" ou plateforme (de gestion) d'apprentissage en ligne

on proposer des langages offrant une syntaxe et une sémantique adaptées à ce type de concepteur ? Peut-on implanter ce type de langage dans un environnement-auteur ?

Comme précisé antérieurement, nous souhaitons ici sortir du cadre qui consiste à mettre à disposition de l'enseignant un ensemble d'objets d'apprentissage avec des interactions "pré-cablées" (textes à trous, QCM, etc.). L'objectif est d'offrir à l'enseignant des moyens de fabriquer ses propres interactions pour que l'activité pédagogique qu'il imagine puisse être implantée conformément à ses attentes. Nous savons que la dimension interactive d'une application demeure souvent une couche complexe, même pour un informaticien.

Verrou 2 : Comment permettre à un non-informaticien d'implanter une application tout en masquant la complexité technologique sous-jacente ?
Tous ceux qui s'y sont essayés savent que concevoir et coder des applications interactives en géographie telles que celles présentées dans la partie 1.2.2.2 n'est pas une tâche facile. En effet, il faut tout à la fois :

- avoir des compétences en programmation (par exemple, JavaScript, AJAX) pour exploiter les API de programmation des composants cartographiques (par exemple Google Maps ou OpenLayers...), des composants calendaires (par exemple SIMILE Timeline...) ;
- avoir des compétences dans le domaine des systèmes d'informations (géographiques) pour interroger les bases de données disponibles (par exemple des BD de l'IGN ou des BD dédiées) ;
- avoir des compétences techniques suffisantes pour pouvoir interroger des services Web donnant accès à des informations géographiques communautaires (par exemple Geonames ou encore OpenStreetMap) ou à des moteurs capables d'extraire la sémantique de textes prédéfinis ou saisis par un utilisateur ;
- être en mesure de programmer des interactions homme-machine tirant parti des composants choisis et des informations rendues disponibles grâce à ces services et outils.

La troisième difficulté que nous identifions donc est la suivante : Comment éviter à l'enseignant-concepteur toutes ces contraintes techniques afin qu'il se focalise sur la dimension pédagogique de l'application et non sur les aspects technologiques ?

1.3.4 Hypothèses

L'ensemble de ces considérations nous amène à proposer les orientations scientifiques de ce travail de thèse. D'après [GRMD07], "*une façon de fournir des supports technologiques pour les communautés d'enseignants est d'aider les participants à produire, à structurer et à partager les informations*". Nous pensons qu'il est nécessaire de proposer un cadre de conception spécifique pour la description et l'opérationnalisation d'applications géographiques par des non-informaticiens, en particulier des enseignants.

Hypothèse 1 : Une approche de conception guidée par les contenus et par les interactions et basée sur des cycles courts est une approche adaptée à des non-informaticiens et à des enseignants.

Nous partons ici du principe que des approches de conception classiques, issues du génie logiciel, ne peuvent être mises en œuvre par des non-experts en informatique et qu'il est donc nécessaire de s'orienter vers d'autres solutions. L'hypothèse ici défendue est qu'une conception guidée par les contenus et les interactions peut être acceptée et mise en œuvre par notre public cible :

- Lorsqu'un non-expert spécifie une application à élaborer, la spécification porte principalement sur ce que le non-expert perçoit au niveau de l'interface utilisateur graphique de l'application, à savoir les contenus présentés, la manière dont ils sont présentés et la façon dont le système réagit lorsque l'utilisateur interagit avec ces contenus.
- Lorsqu'un enseignant élabore un cours, une approche classique consiste à formaliser les contenus à enseigner, à penser la manière de les présenter mais aussi la manière de les faire acquérir au travers d'exercices interactifs pratiques. Nous notons également que dans cette approche, les ressources textuelles (livres, articles...) servent souvent de point de départ pour identifier et structurer les contenus à enseigner. Il nous semble donc important de disposer de moyen permettant de chercher et de structurer des informations enfouies dans des textes de nature géographique (textes patrimoniaux, topo-guides, récits de voyage, etc.)

Dans une approche constructiviste, l'apprenant construit ses savoirs en agissant sur des objets. Dans le cadre d'une stratégie basée sur des situations-problèmes [Nod05], l'apprenant est confronté à des problèmes dont la résolution implique la mise en œuvre de savoirs et savoir-faire particuliers. Dans ce contexte, c'est leur mise en œuvre qui sert de levier pour faire acquérir les connaissances et compétences sous-jacentes. Lorsque la situation-problème devient outillée par une application informatique, cette dernière doit permettre à l'apprenant d'interagir avec des données particulières pour construire son savoir. Dans ce cas, ce sont les possibilités interactives de l'application qui servent de levier pour permettre à l'apprenant d'acquérir les connaissances. Dès lors qu'un contenu intervient dans une interaction, il est valorisé car il est découvert / appréhendé / mis en avant / manipulé par l'utilisateur. En tant que telle, l'interaction devient un moyen mis à la disposition de l'enseignant pour définir des stratégies pédagogiques facilitant l'acquisition de savoirs chez les apprenants.

Ces différentes raisons nous amènent donc à défendre une approche qui incite le concepteur à penser son application en termes de contenus et d'interactions avec ces contenus.

La conception doit être suffisamment souple pour autoriser des approches empiriques laissant part à la créativité pédagogique. Par approche empirique, nous entendons une conception fondée à la fois sur l'expérience et sur l'essai-erreur. Nous considérons que

l'enseignant sait ce qu'il fait d'un point de vue pédagogique (conception fondée sur l'expérience). Par contre, sa maîtrise relative (ou sa non-maîtrise) des outils techniques induira inévitablement une démarche de type essais-erreurs dans laquelle l'enseignant prévoira un comportement supposé de son outil de conception et aura besoin de vérifier par des retours immédiats la véracité de ses hypothèses. En proposant une approche de conception avec des cycles courts, via des outils qui permettent, à tout instant du processus de conception, de visualiser et tester ce qui est conçu, nous favoriserons les processus de conception dans lesquels l'enseignant a une idée, formalise cette idée puis demande à tester l'application concrétisant son idée de départ.

Hypothèse 2 : Des modèles de conception exécutables permettent d'éviter à l'enseignant-concepteur de se préoccuper des contraintes technologiques induites par les applications cibles.

L'absence de compétences techniques chez les enseignants écarte d'emblée la possibilité de les impliquer dans des activités de production ou de modification de code. La réponse à cette contrainte nous semble se trouver dans les techniques d'ingénierie et de transformation de modèles [Sei03a], [JBB+05]. L'activité de conception doit être supportée par des modèles de conception dont les instances doivent pouvoir être transformées en code exécutable par le biais de techniques de transformation de modèles. En d'autres termes, tout choix de conception réalisé par l'enseignant-concepteur doit pouvoir être traduit systématiquement et automatiquement sous forme de code exécutable.

Les modèles de conception proposés doivent être construits avec la contrainte forte suivante. Un concept jugé intéressant (d'un point de vue du domaine applicatif) ne sera retenu que s'il existe une manière de transformer ce concept en un ou plusieurs concepts pouvant se traduire en code exécutable. Si ce n'est pas le cas, le concept ne sera pas intégré au modèle de conception afin de conserver la propriété de génération automatique de code qui demeure inévitable vis-à-vis :

- du public de concepteurs que nous ciblons ;
- des cycles courts que nous souhaitons privilégier dans notre approche.

Nous restons conscients qu'en retirant un concept du modèle nous appauvrissons les possibilités créatrices du concepteur-pédagogue et nous introduisons donc ici une première limitation de notre proposition.

Hypothèse 3 : Un langage visuel permettant de décomposer l'interaction et de la décrire à partir d'éléments de l'interface permet à un non-expert de concevoir seul ses propres interactions.

L'implantation d'interactions au sein d'une application demeure une tâche difficile mais nous souhaitons que cette activité soit à la portée de l'enseignant-concepteur et qu'il puisse définir les interactions pertinentes qui viendront supporter ses stratégies pédagogiques. Pour traiter ce problème difficile, nous nous positionnons de la manière suffisante :

- Partant du principe que le concepteur n'est pas un expert en informatique, nous défendons l'idée que le langage mis à sa disposition pour décrire ses interactions doit être un langage visuel au sens de [Mye90], [Shu99], [KH93] et [NH98] ;
- Partant du principe qu'un non-informaticien décrit une interaction selon sa dimension visuelle (c'est à dire à partir de ce qui se voit à l'écran), nous défendons l'idée que la description de l'interaction doit se faire à partir de contenus affichés à l'écran, de composants d'interface affichant ses contenus et de réactions visuelles du système modifiant / valorisant de nouveaux contenus ;
- Partant du principe qu'une interaction peut être complexe, le langage utilisé pour la décrire doit pouvoir décomposer cette complexité et décrire l'interaction en plusieurs "morceaux" ou étapes. Le langage descriptif utilisé doit donc être suffisamment souple pour s'adapter aux capacités de modélisation du concepteur.

Hypothèse 4 : Des chaînes sophistiquées de traitements de contenu permettent de définir et donc de manipuler les concepts du domaine géographique.

Un enseignant doit disposer de moyens lui permettant de définir avec précision les concepts qu'il souhaite enseigner. Cette règle reste particulièrement importante si on considère que la géographie est un domaine mal défini et que l'enseignant souhaite pouvoir manipuler des concepts géographiques dont la définition est floue ou non standardisée.

Il est donc nécessaire d'offrir des moyens d'expression permettant à l'enseignant de préciser les caractéristiques de chaque concept et éventuellement de créer ses propres concepts. Dans notre approche qui consiste à penser une application en termes de contenus géographiques, de présentation de ces contenus et d'interactions avec ces contenus, il est primordial que l'enseignant puisse définir chaque contenu avec précision. A cette fin il nous semble important de proposer un outillage permettant :

- d'accéder et rapatrier des concepts géographiques connus et formalisés (au sein de bases de données géographiques dédiées, de ressources externes type DBpedia, etc.) ;
- d'agréger plusieurs données géographiques pour en produire de nouvelles ;
- de définir par calcul de nouvelles données géographiques ayant un sens dans l'activité pédagogique prévue par l'enseignant.

1.3.5 Synthèse des contributions proposées

L'ambition des travaux de thèse est de contribuer à lever les verrous précédemment identifiés en proposant des moyens méthodologiques et logiciels que des non-informaticiens sont susceptibles de mettre en œuvre. Les enseignants sont bien sûr la cible à atteindre mais nous ne prétendons pas dans cette thèse développer l'ensemble des briques logicielles attendues par la communauté éducative pour scénariser, déployer et évaluer des applications éducatives en géographie.

Nous proposons :
- d'élaborer une approche de conception permettant à l'enseignant de penser son application en termes de contenus et d'interactions avec ces contenus ;
- de produire des modèles de conception exécutables permettant d'une part d'éviter au concepteur toute activité de production de code et, d'autre part, de mener des cycles de conception courts ;
- de créer un outillage permettant au concepteur de créer, combiner, calculer, affiner des contenus fidèles aux concepts géographiques qu'il souhaite véhiculer. Une attention particulière sera portée pour proposer des outils permettant d'extraire des contenus géographiques issus de ressources textuelles brutes que l'enseignant jugera pédagogiquement intéressantes ;
- de concevoir un langage visuel permettant à un non-informaticien de spécifier des interactions selon des critères visuels présents sur un écran ;
- de développer un environnement de conception support à la démarche proposée, intégrant les modèles exécutables précédemment cités et des outils de spécification visuels. L'environnement proposé et les applications qu'il pourra générer seront exclusivement web pour éviter à l'enseignant tout problème d'installation et de déploiement.

Pour maximiser les chances d'aboutir à des contributions opérationnelles, nous réduisons notre domaine d'étude de la manière suivante :
- l'objectif ne consiste pas à proposer des outils permettant à des enseignants de déployer des EIAH. Il s'agit en revanche d'outiller les enseignants pour leur permettre de produire plusieurs "petites" applications, chaque application pouvant être support à une activité pédagogique particulière s'inscrivant dans un séquencement pédagogique pensé par l'enseignant et pouvant être scénarisé via un LCMS. Les applications visées seront suffisantes pour être support à une activité pédagogique précise.
- les moyens proposés ne permettront pas (dans le cadre de cette thèse)à l'enseignant d'intégrer une couche pédagogique au sein de ses applications. Ainsi, les activités de diagnostic, de tutorat et de séquencement des activités seront à la charge de l'enseignant ou du LCMS dans lequel seront intégrées les applications conçues.

1.4 Guide de lecture

Le plan que nous proposons est original dans la mesure où nous ne reprenons pas un schéma classique de thèse commençant par un état de l'art permettant de positionner nos travaux par rapport à l'ensemble de la communauté scientifique, suivi par une mise en exergue du détail de nos contributions puis une partie dédiée à l'implémentation et aux tests permettant de valider nos propositions et d'évaluer le travail. Nous avons choisi de distribuer ces différents points pour chacun des trois chapitres constituant la contribution de la thèse.

Ce manuscrit se compose de cinq chapitres. Le chapitre 1 a présenté la problématique de la thèse ainsi que les orientations scientifiques de la thèse.

Le chapitre 2 s'intéresse aux approches de conception centrée utilisateur et défend notre proposition consistant à penser une application en termes de contenus et d'interactions permettant de les valoriser. Ce chapitre met en évidence la place centrale de la dimension contenus dans le processus de conception et la nécessité de disposer de modèles de conception couplables pour créer des contenus, disposer de différents moyens permettant de les présenter mais aussi de les rendre interactifs. Ce chapitre 2 doit donc être lu en premier pour cerner les caractéristiques de la démarche de conception proposée et pour comprendre l'intérêt des contributions décrites dans les chapitres suivants.

Le chapitre 3 focalise sur les caractéristiques des contenus ciblés par notre domaine d'étude. L'information géographique est présentée sous différents angles avec pour objectif d'identifier les caractéristiques fortes permettant de décrire des contenus géographiques selon un modèle de conception opérationnalisable. Dans ce chapitre, le lecteur prend connaissance du modèle de contenu que nous proposons et des chaînes de traitement associées pour qu'un non-expert puisse définir ces propres concepts géographiques et que ceux-ci soient automatiquement formalisés par la machine. Ce chapitre s'intéresse également aux moyens permettant de définir une interface graphique présentant des contenus géographiques. Bien que le modèle d'interface présenté ne soit pas considéré comme une contribution à part entière, sa compréhension est nécessaire pour comprendre la contribution présentée au chapitre suivant.

Le chapitre 4 porte sur les moyens envisagés pour permettre à un non-expert de définir et d'opérationnaliser des interactions impliquant des contenus géographiques. Ce chapitre propose un modèle d'interaction et un langage visuel associé pour permettre à un concepteur de spécifier ses propres interactions. L'objectif étant de décrire des interactions essentiellement à partir d'éléments de l'interface, le modèle et le langage proposé s'appuient sur le modèle de contenu présenté au chapitre 3 (pour que l'interaction puisse être décrite sur des contenus géographiques particuliers) ainsi que sur le modèle d'interface de ce même chapitre (pour que l'interaction puisse être décrite par rapport à des composants d'interface spécifiques comme des textes ou des cartes). Ce chapitre doit donc être lu uniquement après une bonne compréhension des concepts énoncés au chapitre 3.

Enfin, le chapitre 5 a pour but de rappeler nos contributions, de présenter les premières évaluations pour nos travaux, de discuter des limites de nos travaux et de proposer les perspectives de la thèse.

Chapitre 2

Conception centrée utilisateur final

Sommaire

2.1 Introduction

Ces dernières années, les usages d'applications informatiques se sont rapidement accrus en nombre et en diversité, et cela concerne des utilisateurs finaux travaillant dans des domaines très diversifiés. En informatique, un concepteur prend en charge l'identification des besoins des utilisateurs et de leur spécification. Par la suite, un programmeur / développeur est un informaticien qui réalise des logiciels en mettant en œuvre les fonctionnalités de logiciel avec un langage de programmation. Enfin un utilisateur (final) est une personne qui utilise les logiciels. Nous pouvons aussi identifier une autre catégorie d'acteur concernés : les "utilisateurs avancés" qui sont des utilisateurs ayant quelques compétences informatiques leur permettant de comprendre des éléments techniques simples. De plus, ces populations ne sont pas toutes uniformes dans leurs usages des environnements informatiques [CFMP06].

Bien qu'ils ne soient pas experts en informatique, les utilisateurs finaux ont besoin d'utiliser efficacement ces applications informatiques. De surcroit, les développeurs professionnels ne peuvent pas répondre directement à tous leurs besoins par manque de connaissance du domaine et parce que leur processus de développement prend beaucoup de temps. Ceci est d'autant plus vrai lorsque cela concerne des applications qui ne seront pas déployées à des milliers d'exemplaires et pour lesquelles un retour sur investissement assez rapide pourrait être espéré.

L'intégration des utilisateurs finaux dans le processus de conception est donc de plus en plus importante. Dans ce cadre, les utilisateurs finaux manipulent non seulement leur application, mais aussi ils veulent la concevoir totalement ou partiellement pour mieux la personnaliser en fonction de leurs besoins. Ce changement a amené à une tendance de conception "centrée utilisateur final" dans laquelle les utilisateurs finaux jouent un rôle important dans toute la conception.

La conception centrée utilisateur final (EUD [6]) est "*un ensemble de méthodes, techniques et outils permettant aux utilisateurs de systèmes logiciels, qui agissent en tant que développeurs non-professionnels de créer, de modifier ou de prolonger un artefact logiciel*" [LPKW06]. L'EUD permet aux utilisateurs finaux de concevoir ou de personnaliser l'interface graphique et les fonctionnalités de leur application informatique. Ceci est un avantage parce que les utilisateurs connaissent leur propre contexte et leurs besoins mieux que quiconque, et ils suivent les changements de leur application en temps réel.

Cependant, cette approche est souvent insuffisante pour obtenir des résultats pleinement satisfaisants car les utilisateurs finaux ne maîtrisent pas les langages de programmation et ne sont pas nécessairement motivés pour programmer. Il faut donc des processus et des outils dédiés aux utilisateurs sans écriture des lignes de code.

L'EUD inclut deux concepts : la programmation centrée utilisateur final (EUP [7]) et le génie logiciel centré utilisateur final (EUSE [8]). L'EUSE met l'accent sur la qualité des applications de grande envergure que les utilisateurs finaux peuvent créer et modifier. L'EUSE se concentre sur les méthodes, les techniques et les outils qui favorisent la qualité de telles applications. L'EUP se définit comme "*la programmation pour achever le résultat d'une application, plutôt que l'application elle-même*" [KAB+11]. Pour notre problématique, nous nous intéressons donc à ce sous-ensemble de l'EUD. Car dans l'EUP, l'objectif du concepteur est d'utiliser l'application à concevoir alors que l'objectif du programmeur professionnel est de créer l'application pour que les autres l'utilisent.

Dans les travaux de thèse, nous essayons de confondre ces rôles (utilisateur, concepteur, programmeur) : un utilisateur / concepteur peut être un non-informaticien qui

6. End-User Design
7. End-User Programming
8. End-User Software Engineering

conçoit par lui-même ses applications. Ainsi nous proposons un processus de conception centré utilisateur avec lequel un utilisateur (qui joue le rôle de concepteur) peut réaliser une application conforme à ses besoins sans intervention des développeurs professionnels. Comme présenté dans le chapitre 1, les applications ciblées sont des applications Web géographiques.

Nous retenons quatre recommandations issues des travaux de Norman [Nor88] sur l'implication de l'humain dans le processus de conception. Ces recommandations permettent de placer les utilisateurs finaux au centre de la conception :

- à tout moment, rendre visible les résultats possibles, y compris les actions alternatives et le résultat des actions ;
- rendre facile l'évaluation de l'état courant du système ;
- suivre une relation logique entre les intentions et les actions attendues, entre les actions et les effets en résultant ; et entre les informations qui sont visibles et les interprétations de l'état du système.

En ce sens nous rejoignons aussi la méthodologie WYSIWYT ("*What You See Is What You Test*") de [BCR04] avec une boucle de rétroaction ("*feedback*") permanente entre le système et son utlisateur permettant de conduire au résultat attendu.

Dans la section suivante, nous résumons et synthétisons les processus de conception centrée utilisateur final ainsi que des mécanismes de transformation de modèles pour transformer des spécifications de haut niveau d'abstraction en code exécutable (section 2.1.1) et les paradigmes de programmation centrée utilisateur final (section 2.1.2). Nous présentons également les Mashups qui sont des outils accessibles à un concepteur / utilisateur final (2.1.3).

Ensuite, dans la section 2.2, nous proposons un processus de conception souple en trois phases pour concevoir des applications Web géographiques. Ce processus va être considéré selon les points de vue d'un concepteur / utilisateur final et d'un informaticien.

2.1.1 Processus de conception centrée utilisateur final

Les principaux cycles de vie du logiciel, détaillés en Annexe C.1, que nous pouvons qualifier de "centrés utilisateur final" sont les suivants :

- *Cycle en V.* Il s'agit du cycle le plus ancien et son côté trop séquentiel/linéaire ainsi que la prépondérance de la documentation vis-à-vis de la réalisation participent à sa mise à l'écart.
- *Développement incrémental.* Ce cycle a été principalement proposé en réponse aux faiblesses des cycles séquentiels/linéaires. Grâce à des cycles répétés, les développeurs profitent progressivement du développement des parties précédentes. L'application est donc délivrée de manière incrémentale mais chaque incrément peut impliquer un cycle de vie classique et donc révéler un coût important au final.
- *Cycle en spirale.* Ce cycle vise à supprimer les défauts relevés précédemment (sans toutefois annuler le coût potentiellement élevé) mais le point de départ de chaque

spire réside dans une analyse des risques qui est un élément primordial qui se révèle difficile à appréhender par un non spécialiste en informatique.
Les deux points suivants ne concernent pas à proprement parlé des cycles de vie mais relèvent plutôt d'une approche de développement.

- *Développement rapide d'applications (RAD).* Dans cette méthode, l'objectif est d'obtenir une application dans un cycle court à partir d'un prototype impliquant l'utilisateur final. De nombreux outils logiciels (comme *WinDev* ou *JBuilder* par exemple) permettent de créer "rapidement" des applications à l'aide d'une interface graphique dédiée et très visuelle. Cet aspect de processus de conception rapide a été repris dans le cadre des méthodes agiles présentées ci-dessous.
- *Développement agile.* Il s'agit d'une méthode de développement visant à réduire le cycle de vie du logiciel tout en impliquant au maximum le client (l'utilisateur final) dans un processus itératif. La capacité d'adaptation aux changements de contexte et aux modifications de spécifications pendant le processus de développement s'en trouve renforcée. Les méthodes agiles promettent donc plus de clarté et de souplesse dans le développement du logiciel.

Cas particulier de l'Ingénierie Dirigée par les Modèles (IDM)

La figure 2.1, extraite de `www.theenterprisearchitect.eu/archive/2009/08/05/a-metaphor-for-model-driven-engineering`, compare sous forme de métaphore le processus de construction d'une maison avec celui d'un logiciel.

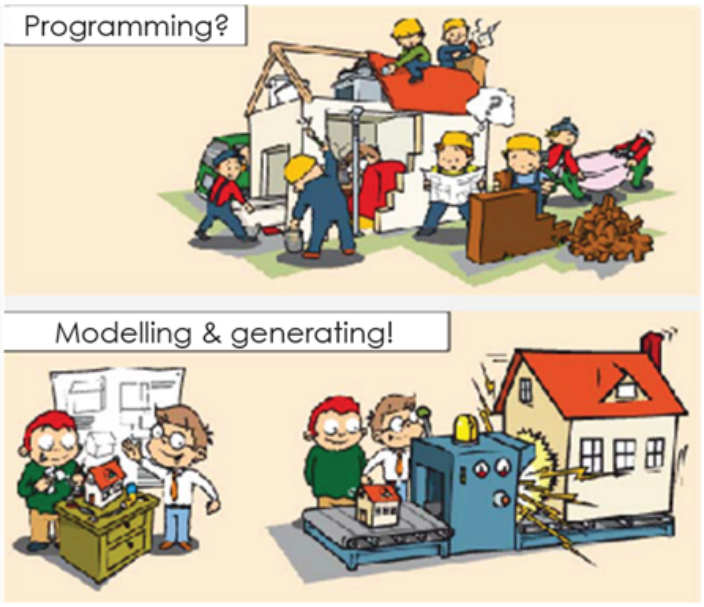


FIGURE 2.1 – Métaphore de l'IDM

Comme dans de nombreux domaines, le développement du logiciel s'oriente vers l'ingénierie dirigée par les modèles (IDM) [Com08]. L'IDM est un domaine de recherche en informatique permettant d'offrir des outils, concepts et langages pour créer et transformer des modèles. Le code source n'est plus considéré comme l'élément central d'un logiciel mais comme un élément dérivé d'éléments de modélisation. Cette démarche met le modèle au centre de la préoccupation des concepteurs. Leur élaboration devient donc centrale et le choix du formalisme revêt une importance capitale. Les technologies d'IDM et notamment de DSL (*Domain Specific Language*) offrent aussi des techniques et des outils pour automatiser le cycle de développement.

Dans l'IDM, le concept central est celui de "modèle" pour lequel nous n'avons pas trouvé facilement une définition unanimement reconnue. Certains travaux [BG01,Sei03b, MFB09, JCV12] proposent la définition suivante : "*Un modèle est un ensemble de faits caractérisant un aspect d'un système dans un objectif donné. Un modèle représente donc un système selon un certain point de vue, à un niveau d'abstraction facilitant par exemple la conception et la validation de cet aspect particulier du système*". Typiquement, nous pouvons utiliser des modèles à la fois manipulables par la machine, et compréhensibles pour le concepteur ; mais ils offrent une certaine abstraction qui masquent au concepteur une partie de la complexité.

L'architecture dirigée par les modèles ou MDA (*Model Driven Architecture*) a été définie en 2000 par l'OMG [9]. D'après [Com08], le principe principal du MDA consiste à s'appuyer sur les standards d'UML pour décrire séparément des modèles pour les différentes phases du cycle de développement d'une application. Ainsi, le MDA consiste en la description des modèles :

- d'exigence (CIM pour *Computation Independent Model*) ;
- d'analyse et de conception (PIM pour *Platform Independent Model*) ;
- de code (PSM pour *Platform Specific Model*).

Le passage de PIM à PSM utilise des mécanismes de transformation de modèles et un modèle de description de la plateforme (PDM pour *Platform Description Model*). Ce processus peut être illustré (Figure 2.2) dans un cycle de développement "en Y" de MDD (*Model Driven Development*). Ce cycle de développement se base sur une plateforme qui se compose de :

- Un moteur de génération de code pour générer le modèle PSM. Dans un environnement Web, JavaScript est un langage souvent utilisé (*cf.* Annexe D).
- Un niveau intermédiaire (PIM) c'est à dire un modèle abstrait qui est indépendant du code opérationnel. Dans un environnement Web, cette "couche" intermédiaire est souvent encodée en XML ou JSON (*cf.* Annexe D).
- Des facilités de génération du PSM. Dans un environnement Web, une API (*Application Programming Interface*) masquant la complexité de la plateforme d'exécution est un atout.

9. The Object Management Group : www.omg.org

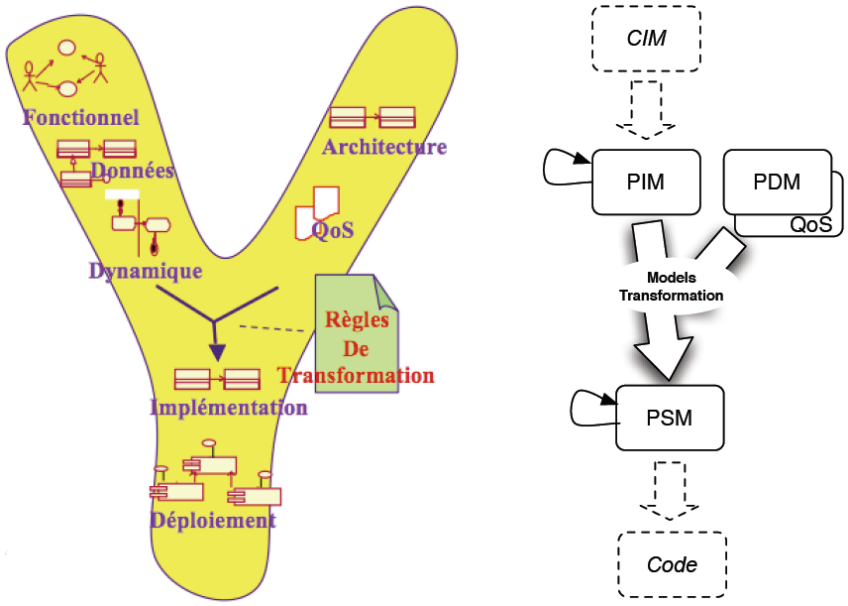


FIGURE 2.2 – MDA : Un processus en Y dirigé par les modèles [Com08]

La conception dirigée par les modèles n'est pas adaptée à des utilisateurs finaux non-informaticiens mais offre des techniques permettant de transformer automatiquement des spécifications de haut niveau d'abstraction (exprimables par des non-spécialistes) en code exécutable. Nous allons maintenant dresser un aperçu des différents paradigmes de programmation centrée utilisateur final afin de voir si nous pouvons y trouver des éléments réutilisables.

2.1.2 Paradigmes de programmation centrée utilisateur final

Concevoir une application informatique sous-entend aborder tôt ou tard une technique de programmation. L'objectif de la phase de programmation est de définir les instructions que le programme devra exécuter pour répondre aux attentes du concepteur. Plusieurs approches peuvent être envisagées pour définir ces instructions.

L'annexe C.2 décrit plusieurs approches de programmation que nous résumons ci-dessous :

- *Programmation par texte.* Il s'agit de la technique la plus traditionnelle de programmation qui demeure la plus largement utilisée pour sa concision et son efficacité. Toutefois il s'agit aussi de la technique la moins abordable par un non-spécialiste en informatique.
- *Programmation visuelle.* Dans ces environnements, la sémantique de l'application est exprimée à travers la présentation visuelle du programme. Citons ici les travaux du Laboratoire Media du MIT (`http://llk.media.mit.edu`) avec des langages

tels que *Scratch* où les primitives sont des briques visuelles. Citons aussi *LEGO MindStorms* [Pap80] ou encore *LabVIEW* (www.ni.com/labview). Un telle approche de programmation la rend accessible à des non-spécialistes en informatique.
- *Programmation par démonstration.* Parfois appelée *programmation par exemple*, c'est une technique de programmation permettant à l'utilisateur de montrer la logique du nouveau programme, à partir de l'environnement de programmation en déduisant un programme qui représente cette logique. *Microsoft Word* permet notamment de développer des macros en respectant ce principe. Une difficulté réside dans la représentation de l'application finale sous une forme utile pour l'utilisateur final. Ainsi cette approche est souvent combinée avec des langages visuels ou textuels.
- *Programmation en langage naturel.* Il s'agit d'une technique permettant au concepteur de spécifier son programme en langage naturel. Le texte exprimé en langue naturelle doit décrire à la fois les données manipulées et les comportements du programme. *AlgoBox* (www.xm1math.net/algobox) est un logiciel libre, multiplateforme et gratuit d'aide à l'élaboration et à l'exécution d'algorithmes dans l'esprit des nouveaux programmes de mathématiques du lycée. Le code de l'algorithme est construit à partir d'un mini-langage algorithmique ("pseudo-code") qui se veut simple à comprendre et à utiliser (mis à part les fonctions mathématiques, les instructions sont en français) ; l'utilisateur n'a donc pas à apprendre toute une syntaxe complexe. La limitation principale de cette approche de programmation en langage naturel est que l'outil de programmation ne peut souvent traiter correctement qu'un nombre restreint d'entrées et qu'il est difficile pour un utilisateur lambda de prédire quel programme va être généré.

Lorsque le programmeur est l'utilisateur final et que ce dernier n'est pas expert, nous situons chaque paradigme précédent de la manière suivante :
- la *programmation textuelle* traditionnelle présente un niveau d'abstraction trop faible, encore trop proche de la machine pour pouvoir la rendre accessible à un non-expert.
- la *programmation visuelle* présente un intérêt particulier du fait qu'elle privilégie un langage graphique souvent plus expressif qu'un langage textuel. Le niveau d'abstraction est légèrement supérieur aux techniques de programmation précédentes mais reste souvent insuffisant pour des non-experts.
- la *programmation par démonstration* reste une des approches les plus abordables pour un non-expert. Elle nécessite néanmoins l'existence préalable d'un système sur lequel le programmeur va simuler les comportements de l'application qu'il souhaite concevoir. Cette approche n'est donc pas adaptée pour la création d'applications *ex nihilo*.
- la *programmation en langage naturel* présente aussi un intérêt pour des non-experts car elle permet à ces derniers de décrire le programme dans un langage qu'ils maîtrisent. Elle présente toutefois des limites importantes car, pour obtenir des résultats intéressants, elle implique souvent que le concepteur connaisse le langage cible

afin de produire une description textuelle interprétable par le générateur de code associé.

Étant donné notre objectif, la piste qui nous semble la plus prometteuse reste la programmation visuelle, *à condition de pouvoir accroître le niveau d'abstraction mis à disposition du programmeur*. La dimension graphique du langage induit un degré d'expression supérieur à un langage textuel car elle permet au concepteur d'exprimer et d'assembler des concepts selon deux dimensions (contrairement à un langage textuel qui reste unidimensionnel). Si les concepts mis à disposition du programmeur s'éloignent du niveau machine pour se rapprocher du niveau métier du concepteur, alors il nous semble possible de proposer une approche de programmation visuelle accessible à un concepteur non-informaticien.

2.1.3 Les Mashups, outils de conception centrée utilisateur final

Les Mashups sont un cas particulier d'environnement de programmation visuelle qui nous semble particulièrement intéressant pour construire des applications centrées "métier". Au départ, le terme Mashup faisait référence aux activités de mélange musical. En informatique, un Mashup [Jhi06] "*est un site Web, un service ou une application qui combine de façon dynamique le contenu de sources différentes dans un produit intégré*". Les Mashups sont aujourd'hui très utilisés comme outils de conception pour créer de nouvelles applications avec peu ou pas de programmation. Sur le Web, les Mashups ont grandement amélioré la créativité des concepteurs, en leur permettant de combiner rapidement et simplement des informations provenant de diverses sources (éventuellement hétérogènes) puis de les intégrer dans de nouvelles applications. L'idée est que les informations disparates sont accessibles à distance grâce à des interfaces basées sur les services et peuvent ensuite être recombinées librement par l'utilisateur sur son navigateur.

Comme présenté dans [BP09], les principaux challenges à relever dans la construction de Mashups concernent l'*extraction de données*, l'*hétérogénéité* de ces données, leur *intégration*, l'*hétérogénéité des services* d'accès à ces données, la *qualité* des données extraites, la *maintenabilité* à long terme des Mashups et la *sécurité* globale.

Les Mashups [Jhi06, Kri07] reposent sur les caractéristiques principales suivantes :

- Réutilisation et intégration des codes et des contenus. Ceci implique de faibles coûts de conception et de développement. Un Mashup peut intégrer différents composants (services Web, sources de données...). Dans le cas des sites Web, le principe d'un Mashup est d'agréger du contenu provenant d'autres sites, afin de créer un nouveau site.
- Le logiciel en tant que service (en anglais *Software as a service* - SaaS). Il s'agit d'un modèle de livraison, reposant sur une architecture en couches, qui est à la fois simple et léger, accessible partout (via un navigateur), sans besoin d'installation. Il y a une séparation propre des parties relatives à l'interface utilisateur, des par-

ties d'intégration logique déployées par le moteur du Mashup, et des fournisseurs de services et sources de données. D'un point de vue technique, pour développer un Mashup, le programmeur peut utilisateur des API de services et des technologies comme AJAX, XML-RPC (*XML Remote Procedure Call*), JSON-RPC (*JSON Remote Procedure Call*), REST (*REpresentational State Transfer*)...

– *Do it yourself* (DIY) : n'importe qui peut être un auteur. Grâce à la capacité d'intégration des Mashups, le concepteur peut agréger plusieurs parties issues de sites Web différents pour créer un nouveau site Web sans écrire une ligne de code de programmation textuelle.

Ainsi, utiliser des outils visuels de type Mashup est un moyen de conception qui ne nécessite pas la connaissance des langages de programmation, des structures de données ou des architectures d'hébergement. Cette conception permet à des utilisateurs qui n'ont ni le temps ni l'envie d'étudier les langages de programmation d'aborder la programmation. Ces environnements de programmation de type Mashup permettent à l'utilisateur de s'appuyer sur des métaphores faciles à saisir par des non-programmeurs, comme par exemple des flux liant des blocs de traitement, la sélection visuelle par glisser-déposer de composants d'interface...

Les Mashups peuvent être classifiés selon les trois dimensions suivantes [ABCG09] illustrées dans la figure 2.3 :

1. La première dimension est **la nature des Mashups utilisés par la plateforme**. Nous pouvons distinguer les *Mashups de données*, les *Mashups logiques* et les *Mashups de présentation*. Un *Mashup de données* se consacre principalement à combiner les données provenant de sources différentes. Un *Mashup logique* présente les manières de faire collaborer des services différents par rapport à l'objectif de l'application. Enfin, un *Mashup de présentation* affiche les données ou les services récupérés à l'aide d'une interface commune.
2. La deuxième dimension est **le type d'utilisateur capable d'utiliser la plateforme de Mashup**. Nous pouvons distinguer le *développeur*, l'*utilisateur avancé* et l'*utilisateur final*. Le *développeur* a une connaissance profonde de la technologie Web, ainsi que des compétences fortes en programmation. Il considère les Mashups comme un outil de programmation plus rapide. L'*utilisateur avancé* n'a pas de compétences en programmation, mais est capable de comprendre les problèmes technologiques sous-jacents. Enfin, l'*utilisateur final* ne connait le problème qu'au niveau fonctionnel, et n'a pas de compétences techniques.
3. La troisième dimension concerne **"comment" et "où" le Mashup est exécuté**. Nous pouvons distinguer le Mashup côté *client* qui tire avantage des ressources côté client via le navigateur Web de l'utilisateur. Dans ce cas, l'environnement d'exécution repose généralement sur les technologies du Web 2.0 telles que JavaScript, XML et JSON (*cf.* Annexe D pour plus de détails). Le Mashup côté *serveur* tire avantage des technologies du serveur telles que WSDL ou encore SOAP.

Vis-à-vis de notre approche de conception, les Mashups permettent donc de définir des contenus (données) d'un programme, de préciser la manière de les présenter (en y associant souvent des interactions prédéfinies au sein des afficheurs), le tout sans phase de programmation formelle et avec des capacités à obtenir des retours immédiats.

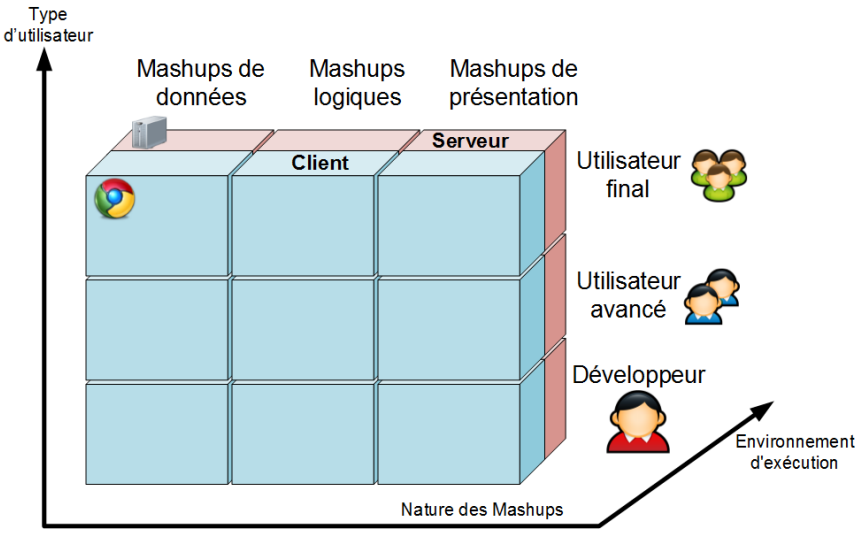


FIGURE 2.3 – Trois dimensions caractérisant les Mashups

La Figure 2.4 extraite de [BG10] montre un exemple avec MashLight où le concepteur définit des données géographiques pour les présenter sur une carte proposant des interactions précablées comme le zoom ou le changement de fond de carte par exemple.

La Figure 2.5 extraite de [BP09] présente un exemple similaire sous Eclipse avec JOpera (`www.jopera.org`). La partie gauche présente un workflow définissant les données traitées (récupération en temps réel des fichiers de logs de connexion à un site Web puis calcul des géoréférencements des serveurs à partir de leur adresse IP) et la partie droite est un afficheur (carte GoogleMaps) dans lequel les données sont projetées.

2.1.4 Synthèse et discussion

Comme présenté dans le chapitre 1, nous ciblons la conception / réalisation d'applications Web géographiques par des non-spécialistes de la programmation.

Pour atteindre cet objectif, il nous semble nécessaire de définir :
- une approche de conception accessible et acceptable par le public de concepteurs visés ;
- des langages de spécification compréhensibles et utilisables par ces concepteurs ;

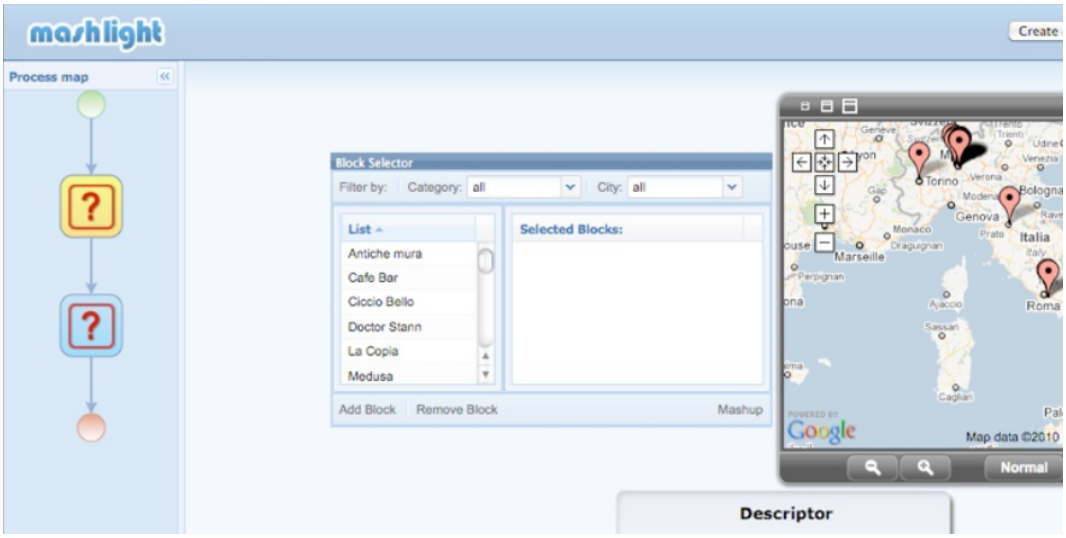


FIGURE 2.4 – Exemple de conception avec le mashup MashLight [BG10]

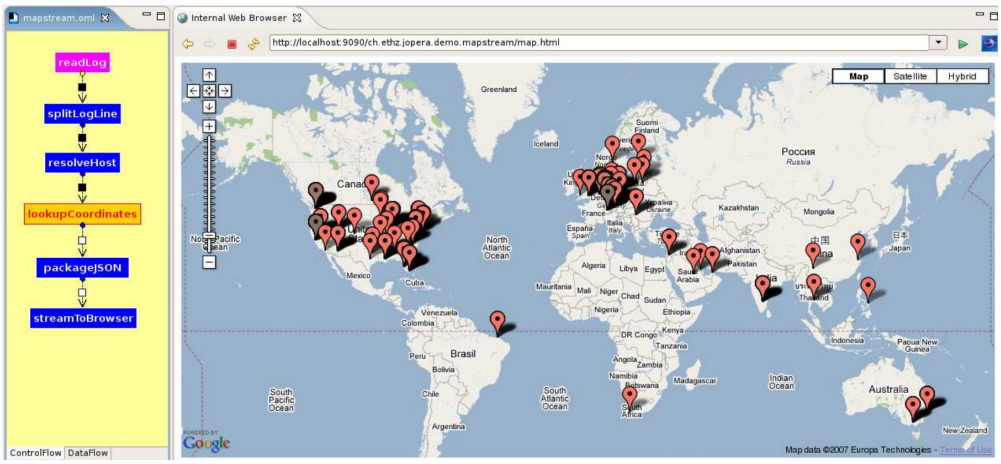


FIGURE 2.5 – Exemple de conception du mashup Mapstream avec Eclipse [BP09]

– un outillage permettant de transformer automatiquement les spécifications des concepteurs sous forme d'une application exécutable et donc évaluable.

Nous rappelons qu'une des hypothèses de départ de ce travail de thèse est de considérer qu'une conception guidée par les contenus et les interactions est une approche de conception accessible à des non-experts.

Parmi les différents cycles de vie envisagés pour dérouler un tel processus de conception, il nous semble préférable de privilégier les approches agiles. Elles préconisent des cycles itératifs courts et permettent donc au client / concepteur, qui est intégré dans le processus, d'avoir des retours rapides sur ce qu'il conçoit. Cette propriété de feedback immédiat nous semble primordiale lorsque le concepteur n'est pas un expert du développement informatique.

La mise en place de ces cycles courts sous-entend de pouvoir traduire des spécifications de haut niveau en code machine pouvant être testées et évaluées immédiatement. Le concepteur n'étant pas un développeur, la traduction de ses spécifications sous forme de code exécutable devra être réalisée de manière automatique et sous-entendra donc l'usage de techniques issues de l'IDM.

Pour pouvoir mettre en œuvre des techniques de transformation de modèles, il est nécessaire de disposer d'un modèle de conception ayant deux propriétés :
- il intègre des concepts d'un niveau métier qui sont porteurs de sens pour un concepteur non-expert ;
- les modèles de conception qu'il permet de produire sont transformables en modèles susceptibles de donner lieu à de la génération de code.

Le concepteur mènera donc son activité de conception en spécifiant les propriétés de l'application à bâtir et le modèle de conception précédent servira de cadre de réflexion pour guider cette activité de conception. Le travail de spécification consistera à définir des contenus géographiques et des interactions sur ces contenus et cette activité de spécification devra donc être menée avec un langage adapté.

Parmi les différents paradigmes de programmation cités dans ce chapitre, nous retenons les potentialités de la programmation visuelle qui propose au concepteur de spécifier son programme de manière visuelle. La richesse sémantique des représentations graphiques représente un atout qu'il nous semble judicieux d'exploiter pour apporter du sens dans l'activité de conception et permettre au non-expert de s'approprier plus facilement les briques de conception qu'il devra combiner pour bâtir son application. Le langage visuel associé devra donc être pensé pour permettre au concepteur de manipuler les concepts du modèle cité précédemment et donc de travailler à un niveau d'abstraction qui soit le plus élevé possible.

D'un point de vue des outils logiciels, nous retenons les capacités intéressantes des Mashups qui permettent de créer des applications Web de manière visuelle et sans programmation (ou presque). Ces outils apportent déjà un début de réponse à notre problématique dans le sens où ils permettent (de manière visuelle) de définir des données (contenus) et de les présenter dans des afficheurs spécialisés (cartes géographiques, tableaux triables, etc). Ils présentent donc un intérêt significatif pour la définition de données (contenus) et pour la conception d'interfaces permettant de les présenter. Nous

notons néanmoins leurs limites dès lors qu'il s'agit de définir et de personnaliser les interactions qui pourront être déclenchées sur ces données.

Dans la section suivante, nous proposons de détailler l'approche de conception que nous défendons pour permettre à des non-experts de bâtir une application en raisonnant en termes de contenus à présenter et d'interactions déclenchables sur ces contenus. Nous considérerons à la fois le niveau méthodologique mais aussi les contraintes induites au niveau de l'outillage support à cette approche de conception.

2.2 Conception des applications géographiques par les contenus et les usages

Rappelons tout d'abord notre première hypothèse relative aux objectifs scientifiques de la thèse (*cf.* Chapitre 1 section 1.3). Quand un concepteur non-informaticien décrit l'application qu'il souhaite mettre en œuvre, le comportement de l'application (les contenus et les usages) est au centre de sa description. Pour concevoir des applications géographiques, les concepteurs partent d'un ensemble de contenus sur lesquels ils veulent définir des interactions.

D'un point de vue de la conception, notre objectif est de fournir aux utilisateurs un environnement visuel en ligne qu'ils peuvent utiliser pour concevoir et évaluer des applications Web géographiques. Voici un exemple de besoin que V. Paillas a pu récolter auprès des enseignants durant son stage de Master 2 Recherche en Sciences de l'Education [Pai11].

"*Dans le cadre de mes enseignements autour de la géographie, je souhaiterais faire travailler mes élèves sur la représentation spatiale des concepts géographiques mentionnés dans un texte. A cette fin, je souhaiterais disposer d'une application permettant aux élèves de faire de la lecture active de la manière suivante. Cette application présenterait un texte (que je pourrais choisir moi-même) qui décrirait un territoire. Je souhaiterais que les élèves puissent étudier ce texte et obtenir une représentation spatiale concrète des différents lieux cités, en considérant aussi bien les villages, les rivières, les monts, etc. Il faudrait donc que l'application intègre une carte du territoire décrit dans le texte et, pour assister les élèves dans leur activité de lecture, j'aimerais qu'ils puissent pointer n'importe quel lieu cité dans le texte et que le système affiche ce lieu sur la carte. Il faudrait aussi que les élèves puissent jouer avec différents niveaux de zoom sur la carte afin de pouvoir localiser un lieu par rapport à d'autres lieux qu'ils connaissent où qui ont été précédemment cités dans le texte.*"

L'application doit être ainsi décrite à un niveau "surfacique" (relatif aux éléments visibles sur l'interface) en soulignant ce qui est perceptible par l'utilisateur final. Ce type de description reste informel et demeure le seul niveau de description accessible à un non-informaticien. Dans l'exemple ci-dessus, l'enseignant décrit de manière désordonnée :

- les contenus géographiques qu'il souhaite faire travailler à ses élèves : un ensemble de lieux cités dans un texte qu'il a sélectionné ; ces lieux pouvant être de nature diverse (villages, rivières, etc.) ;
- la manière dont ces contenus seront présentés à l'écran : les lieux sont initialement présentés dans un contexte textuel via le récit mis à disposition par l'enseignant. Mais ils pourront également être présentés sur une carte géographique à la demande des élèves ;
- les interactions qui pourront être menées sur les contenus affichés : possibilité de demander une représentation cartographique de tout lieu pointé dans le texte, possibilité de situer un lieu par rapport à un autre via un outil de zoom intégré dans la carte.

Ces éléments nous amènent à proposer une démarche de conception permettant au concepteur de penser une application en termes de contenus, de présentation de ces contenus et d'interactions avec ces contenus. Les sections qui suivent décrivent le détail de l'approche que nous proposons.

2.2.1 Un processus en trois phases (*Contenu*, *Interface*, *Interaction*)

Les concepts d'interaction et d'interactivité dans l'application doivent être clairement définis. Nous pouvons également noter qu'ils sont fortement liés à des éléments visuels sur l'interface. Ces éléments constituent des données ou des contenus à manipuler dans les applications via des interfaces adaptées sur lesquelles les contenus sont dotés d'un comportement adéquat.

Ainsi, l'idée consiste à guider l'activité de conception à partir des données présentées à l'utilisateur et des possibilités interactives proposées sur ces données. Le travail du concepteur consiste à définir :

1. quelles sont les données qui doivent être manipulées,
2. la façon de présenter ces données et
3. les interactions qui seront disponibles pour permettre à l'utilisateur d'interagir avec elles.

Comme les applications conçues sont guidées par les contenus, nous proposons (Figure 2.6) un processus de conception d'applications interactives composé de trois phases complémentaires [LNLM09, LLN11] :

1. **Contenu** : identification des données manipulées par l'application à concevoir. Les données sont à connotation géographique ; certaines informations sont automatiquement ou semi-automatiquement extraites et d'autres sont calculées.
2. **Interface** : spécification de l'interface graphique de l'application. Cette disposition peut être composée de plusieurs afficheurs, tels que des afficheurs textuels, des listes, des afficheurs cartographiques ou calendaires.

3. **Interaction** : définition des interactions potentielles de l'utilisateur sur les données présentées à l'intérieur des afficheurs.

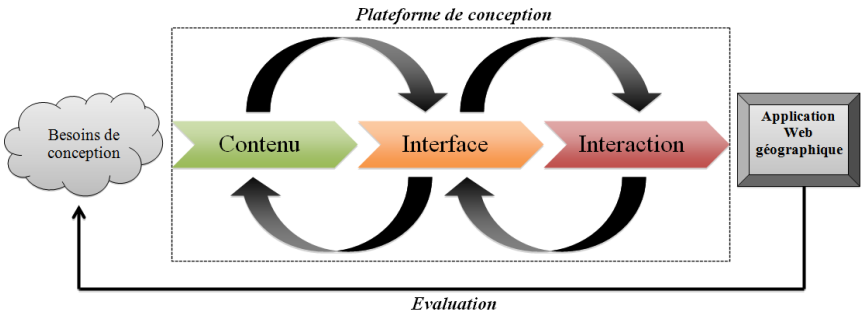


FIGURE 2.6 – Trois phases du processus de conception - ordonnancement 1

Au sein de ce processus, il faudrait également que les facettes / phases soient aussi indépendantes que possibles les unes des autres [Voj12] et que le concepteur puisse librement aller d'une phase à la suivante en choisissant même la phase de départ qu'il juge la plus pertinente.

2.2.2 Un processus "agile"

Comme présenté dans la partie précédente, le concepteur souhaite pouvoir mettre en œuvre un processus souple lui permettant de rapidement évaluer / modifier son application (Figure 2.7).

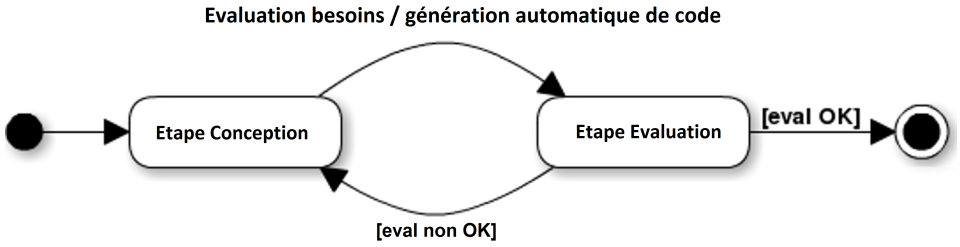


FIGURE 2.7 – Un processus souple

Suite à une première proposition de spécification de l'application, des outils de génération de code doivent lui permettre de pouvoir immédiatement évaluer le résultat de sa spécification. S'il en est satisfait, son travail est terminé mais s'il constate que l'application générée ne correspond pas parfaitement à ses besoins, il doit pouvoir revenir en phase de conception pour reprendre son travail. Cette souplesse doit également s'exprimer au travers de l'ordonnancement des phases de définition des contenus, des interfaces

et des interactions. L'exemple d'application cité en début de section 2.2 pourrait ainsi être conçu selon différents angles d'attaque.

Scénario 1 : L'enseignant définit d'abord les contenus à valoriser, à savoir, le texte de départ présenté aux élèves duquel il extrait les lieux cités. Il pourrait ensuite bâtir l'interface de son application en assemblant un composant textuel dans lequel il placerait son texte et un composant cartographique dans lequel il définirait la région à afficher. Il pourrait ensuite définir les interactions à opérationnaliser à savoir l'affichage sur la carte de tout lieu désigné dans le texte et l'intégration de la fonctionnalité de zoom sur le composant cartographique.

Scénario 2 : L'enseignant construit d'abord l'interface de son application en assemblant un composant textuel vide et un composant cartographique qu'il place et dimensionne selon ses souhaits. Il choisit ensuite un texte qu'il place dans son composant textuel et ajuste la carte affichée pour qu'elle corresponde à la région relatée dans le texte. Il définit enfin les interactions de la même manière que précédemment.

Scénario 3 : L'enseignant construit une première ébauche de son interface en plaçant un composant textuel vide puis sélectionne le contenu textuel à présenter. Il affine ensuite son interface en y ajoutant un composant cartographique qu'il positionne par rapport au composant textuel. Il définit ensuite les interactions possibles entre ces deux composants.

Dans ce cadre, la dimension contenu et la dimension interface peuvent, aussi bien l'une que l'autre, servir de point d'entrée pour la conception de l'application et il nous semble tout à fait envisageable de permettre au concepteur de mener en parallèle le développement de ces deux dimensions (*cf.* Figure 2.8).

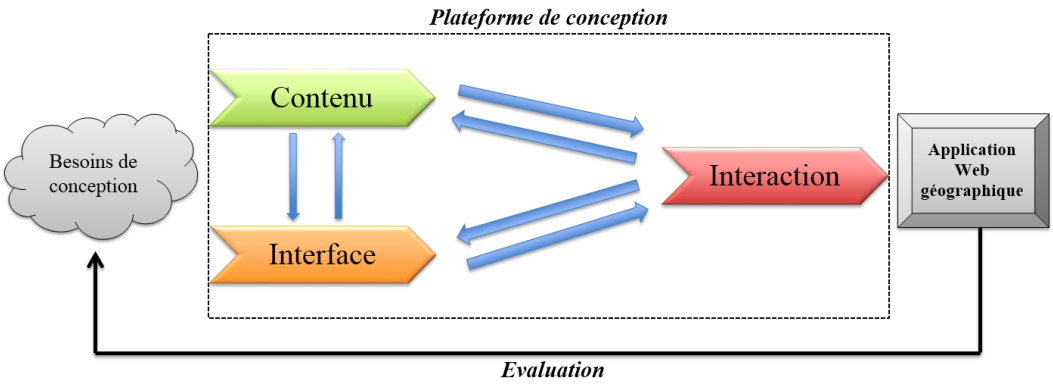


FIGURE 2.8 – Trois phases du processus de conception - ordonnancement 2

La dimension interactive reste plus difficile à appréhender en tant que point de départ car nous faisons le choix de la décrire en termes de contenus rendus interactifs sur une interface et de contenus valorisés suite à une réaction du système (présentation de nouveaux contenus, mise en évidence de contenus particuliers, etc.). Il faut donc qu'un minimum de contenus et qu'une ébauche d'interface aient été réalisés avant de pouvoir spécifier des éléments d'interaction. La dimension interactive est donc fortement couplée avec les dimensions contenu et interface au point que la suppression d'un contenu ou d'un composant d'interface entraîne l'invalidité de toutes les interactions où il était sollicité.

Afin de mettre en œuvre ce processus "agile", chacune de ces trois dimensions sera formellement représentée par un modèle. De plus dans une démarche MDA, ceci permettra une meilleure séparation des problèmes/préoccupations ("*separation of concerns*") entre les aspects métiers et les éléments techniques, propice à l'adoption de l'agilité.

Le *modèle* que nous proposons est structuré en trois parties (sous-modèles) correspondant aux trois phases précédentes :

1. Le *modèle de contenu* décrit les données géographiques qui sont manipulées par l'application ;
2. Le *modèle d'interface graphique* définit l'organisation de la mise en page de l'application ;
3. Le *modèle d'interaction* spécifie le comportement de l'application et des différents contenus sur l'interface.

2.2.3 Un environnement-auteur support du processus de conception

Comme annoncé en introduction, notre objectif vise, dans la mesure du possible, à libérer le concepteur des contraintes techniques pour lui permettre de recentrer son travail de conception sur les aspects interactifs de l'application à construire. En ce sens, nous souhaitons proposer un environnement de conception capable :

- de supporter l'approche de conception que nous proposons ;
- de générer le code de l'application spécifiée (afin d'évacuer les problèmes techniques en fin de processus de conception).

Notre objectif est de proposer un environnement visuel reposant sur une approche à base de Mashup (Figure 2.9). Cet environnement doit faciliter la conception des applications Web interactives géographiques conformément à notre approche de conception par tissage de modèles et par phases (*cf.* sections 2.1.1 et 2.2.1).

Le processus de conception est divisé en trois phases successives : *Contenu*, *Interface* et *Interaction* ; chaque phase faisant l'objet d'une spécification visuelle la plus intuitive possible ①.

L'activité de spécification du concepteur se traduit par des instanciations ② des modèles de contenu, d'interface et d'interaction ; ces instanciations ③ constituant les spécifications formelles de l'application à générer.

Ces spécifications, indépendantes de toute technologie, peuvent ensuite servir de point d'entrée à un moteur de génération de code spécifique ④.

Dans notre cas, ce moteur traduira (*cf.* Annexes D et F pour plus de détails), les spécifications du concepteur sous forme de code HTML, CSS et Javascript ⑤ [10] permettant au concepteur d'évaluer directement son application ⑥.

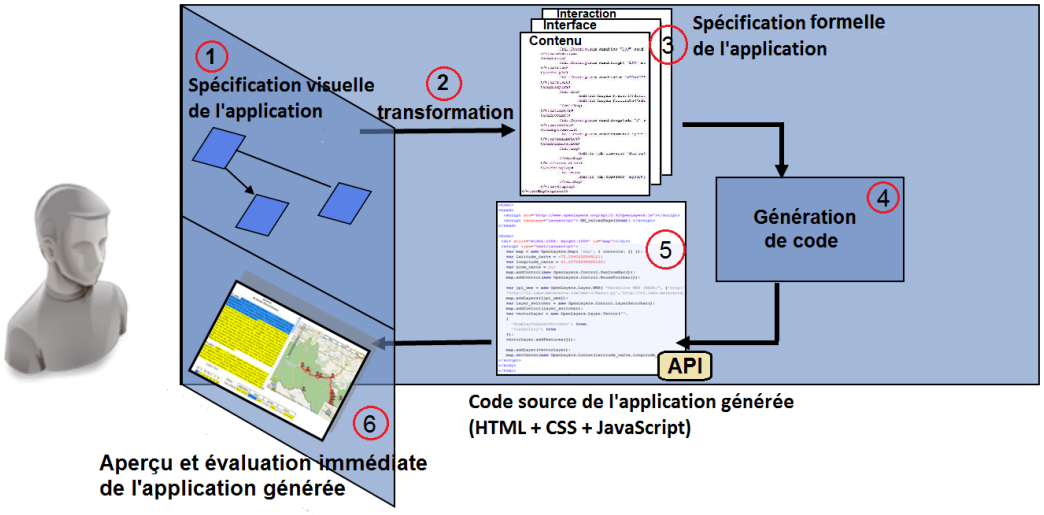


FIGURE 2.9 – Environnement de conception utilisable par le concepteur

2.3 Conclusion

Dans ce chapitre nous avons focalisé notre étude sur le processus de conception à privilégier pour permettre à un concepteur novice de bâtir une application géographique en toute autonomie.

Pour définir les caractéristiques de cette approche de conception, nous nous sommes intéressés, dans une première partie, aux approches traditionnelles issues du génie logiciel, aux paradigmes de programmation accessibles à des non-experts et aux outillages

10. De plus, ces applications générées pourront tirer parti d'une API permettant de masquer la complexité de codage inhérente aux technologies utilisées.

permettant de produire des applications sans connaissance technique spécifique. En bilan de cette étude, nous avons retenu l'intérêt et les potentialités :

- des approches de conception agile qui préconisent de travailler par cycles courts en impliquant l'utilisateur final dans l'activité de conception.
- des langages visuels de programmation qui, par leur pouvoir d'expression graphique, peuvent hypothétiquement être pris en main par des non-experts.
- des outils de type Mashup qui permettent de bâtir de petites applications de manière visuelle et sans phase de programmation.

À partir de cette étude nous avons, dans une seconde partie, défini plus précisément les caractéristiques d'une approche de conception adaptée à des non-informaticiens. Les idées défendues sont les suivantes :

- *un concepteur non-informaticien ne peut décrire une application que d'un point de vue de ce qu'il perçoit visuellement.* Selon cette idée, nous avons proposé de guider la conception selon trois facettes : une facette *contenu* dans laquelle le concepteur définit les données qu'il souhaite présenter à l'écran, une facette *interface* dans laquelle le concepteur précise la manière selon laquelle il souhaite présenter les contenus définis précédemment et enfin une facette *interaction* dans laquelle le concepteur décrit les réactions du système lorsque l'utilisateur interagit avec un contenu présenté à l'écran.
- *un concepteur non-informaticien conçoit une application selon une approche essai-erreur dans le sens où il ne peut prédire comment ses spécifications seront traduites par la machine.* En ce sens, l'approche de conception proposée doit permettre au concepteur d'avoir des retours immédiats sur ce qu'il conçoit et donc être basée sur des cycles courts. Ces cycles courts ne sont possibles que si les spécifications fournies par le concepteur peuvent être traduites automatiquement sous forme de code exécutable. Il est donc nécessaire ici de faire appel à des techniques issues de l'IDM pour élaborer des outils de génération de code capables de transformer toute spécification de haut niveau en langage machine.
- *un concepteur non-informaticien ne peut spécifier les caractéristiques de l'application à bâtir à l'aide d'un langage de programmation classique.* Il faut donc être en mesure de lui proposer un langage de spécification qui soit accessible et la programmation visuelle nous semble apporter un début de réponse à cette problématique. Les concepts manipulables par le concepteur au travers de ce langage de programmation doivent être hissés à un niveau d'abstraction le plus élevé possible et se rapprocher du niveau métier avec lequel le concepteur est familier.

Ces différents points impliquent qu'une telle approche de conception ne peut être menée sans environnement de conception dédié. Pour être support à cette approche, l'environnement-auteur associé doit posséder les caractéristiques suivantes :

- proposer un ensemble de services permettant de définir les contenus à intégrer dans l'application. La géographie étant considérée comme un domaine mal défini ("*ill-structured domain*" - *cf.* Chapitre 1 section 1.3), les services proposés devront

permettre au concepteur de caractériser chaque donnée selon son propre point de vue. Les données géographiques doivent pouvoir être extraites à partir de base de données géographiques ou de textes à caractère géographique (récits de voyage, topoguides...). Il faut également prévoir des services permettant au concepteur d'affiner chaque donnée extraite ou de combiner plusieurs données pour créer de nouvelles données porteuses de sens pour le concepteur.

- proposer un ensemble de composants permettant de bâtir des interfaces capables d'afficher des contenus géographiques. Étant donnée la nature tripartite de ces contenus (*cf.* section 1.1), ces composants d'interface devront permettre de représenter aussi bien la dimension spatiale, temporelle que thématique de chaque donnée géographique manipulée par le concepteur. Ce dernier devra donc être en mesure de bâtir des interfaces présentant des cartes (pour valoriser la dimension spatiale d'une donnée), des calendriers ou frises chronologiques (pour valoriser la dimension temporelle) et des objets textuels (pour représenter n'importe quelle dimension).
- proposer un langage visuel permettant au concepteur de préciser le détail de chaque interaction qu'il souhaite opérationnaliser. Ce langage devra permettre au concepteur d'implanter ses propres interactions afin qu'il ne soit pas limité aux interactions pré-cablées qu'il pourra trouver dans les composants d'interface qu'il assemblera.
- permettre, dans la mesure du possible, de bâtir l'application selon différents points d'entrée : définir d'abord les contenus à valoriser, la manière de les présenter sur une interface puis les interactions associées ou bien bâtir d'abord l'interface, puis les contenus que l'interface devra présenter et enfin les interactions associées. Cette souplesse permet de prendre en compte partiellement les approches de conception empiriques que peuvent adopter des concepteurs non-informaticiens.
- permettre de générer automatiquement le code de l'application spécifiée et, si possible, quelque soit le stade d'avancement de la conception. La génération automatique du code représentera le moyen de supporter des cycles de conception courts et d'offir au concepteur des retours immédiats sur ce qu'il a conçu.

Les chapitres qui suivent présentent les modèles conceptuels nécessaires à cette démarche. Le chapitre 3 présente un modèle permettant de définir les contenus que le concepteur souhaite intégrer à son application, ainsi qu'un modèle d'interface simple permettant de présenter ces contenus. Le chapitre 4 présente le modèle d'interaction sur lequel le concepteur pourra s'appuyer pour définir le comportement de son application ainsi que le langage visuel qui sera mis à sa disposition pour spécifier le détail de chaque interaction.

Chapitre 3

Contenus manipulés dans les applications Web géographiques

Sommaire

3.1 Introduction

Actuellement, de nombreux contenus géographiques sont disponibles en ligne. Ils peuvent être traités automatiquement par des services Web pour récupérer et rendre explicite l'information géographique qu'ils contiennent au sein de méta-données. Par

exemple, un texte brut peut être traité par des services Web extrayant automatiquement des entités spatiales et temporelles citées dans le texte. De même, des données structurées sur le Web (par exemple, *Linked Data*) peuvent être reliées entre elles pour construire un réseau global d'informations ; ceci facilitant l'interrogation ainsi que la visualisation des données. D'autre part, de plus en plus d'applications Web géographiques ont été développées grâce à des technologies issues du Web 2.0 ainsi que des services de cartographie en ligne (par exemple, Google Maps, OpenStreetMap...)

Développer des applications Web géographiques est une tâche difficile car il est nécessaire d'exploiter plusieurs composants visuels (par exemple, des cartes, des contenus multimédias, des services Web, des bases de données...). De plus, les développeurs doivent avoir de nombreuses connaissances sur les différentes structures et modèles de contenus. Enfin, ces composants visuels doivent interagir entre eux et avec l'utilisateur, par exemple, lors d'un clic de l'utilisateur sur un contenu affiché dans un composant visuel, les autres composants se synchronisent (affichent les informations correspondant à ce contenu).

Ce chapitre est consacré aux contenus géographiques manipulés dans les applications Web à concevoir et montre comment modéliser ces contenus géographiques pour développer des applications Web géographiques satisfaisant des besoins. Nous commençons par synthétiser les types d'applications Web géographiques existantes ainsi que les environnements Web, de type Mashup, dédiés à la conception d'applications interactives. Nous abordons ensuite les verrous scientifiques relatifs à la conception d'applications Web géographiques. Pour résoudre ces problématiques, nous proposons les modèles des phases *Contenu* et *Interface* du processus proposé dans le chapitre 2. Nous implémentons ces deux modèles dans l'API WIND et les mettons en œuvre au sein d'un environnement nommé WINDMash pour plus de détails) permettant à l'utilisateur de concevoir lui-même des applications Web géographiques. Nous concluons ce chapitre avec un bilan et des limites.

3.2 État de l'art : Information géographique

Le paysage des technologies Web utilisées dans le domaine de la cartographie a radicalement changé depuis 2005. De nouvelles techniques sont utilisées et de nouveaux termes ont été inventés. Toute une gamme de sites Web ainsi que des communautés de Google Maps à OpenStreetMap, ont également émergé. Ces nouvelles applications représentent une nouvelle étape dans l'évolution de l'aire d'applications Web géographiques (que certains ont appelé GeoWeb) [ST07]. La nature de ce changement justifie une explication et un aperçu à cause des implications tant pour les géographes que pour les utilisateurs de géographie du grand public. Cette section fournit une analyse critique de ce paysage émergent, à commencer par une introduction aux concepts, aux technologies et aux structures du GeoWeb.

3.2.1 Étude des principaux systèmes Web géographiques

Dans cette partie, nous présentons certains systèmes Web géographiques. Notre but n'est pas de donner une liste exhaustive de tous les systèmes existants. Nous divisons notre présentation en deux catégories de système : les systèmes propriétaires (3.2.1.1) et les systèmes libres (3.2.1.2).

3.2.1.1 Principaux systèmes propriétaires

A. Google Maps

Google Maps [11] est un service gratuit de cartes géographiques en ligne. Le service a été créé par l'entreprise Google. Lancé en 2004 aux États-Unis et au Canada, et en 2005 en Grande-Bretagne (sous le nom de Google Local), Google Maps a été lancé le jeudi 27 avril 2006, simultanément en France, Allemagne, Espagne et Italie. Ce service permet, à partir de l'échelle du monde, de pouvoir zoomer jusqu'à l'échelle d'une rue. Divers types de plan sont disponibles et notamment : un plan classique avec les noms des rues, quartiers, villes et un plan en image satellite qui couvre aujourd'hui le monde entier. Ce service n'est plus en version bêta depuis le 12 septembre 2007 et a été ajouté aux liens traditionnels de la page d'accueil de Google.

Google offre également un service intitulé Google Earth [12] qui est un logiciel que l'on peut installer sur des ordinateurs ou des dispositifs mobiles. Ce service permet de visualiser la Terre en 3D avec un assemblage de photographies satellites. Google fournit également le service Street View [13] afin de compléter Google Maps et Google Earth. Ce service permet de visualiser une rue en 360 degrés. Les développeurs peuvent explorer les fonds et les données cartographiques de Google en utilisant l'API Google Maps qui repose sur ces services.

B. Yahoo! Maps

Yahoo! Maps [14] est un portail de cartographie en ligne gratuit fourni par Yahoo! depuis 2007. L'utilisateur peut explorer le monde entier à travers des cartes routières et des vues par satellite. L'API de Yahoo! Maps permet d'intégrer facilement des cartes riches et interactives dans les applications Web en utilisant une des plates-formes suivantes : Flash, Ajax et API Image Map. Yahoo! fournit également d'autres API que l'utilisateur peut utiliser pour ajouter des informations géographiques au sein de cartes.

11. http://maps.google.fr
12. http://www.google.fr/intl/fr/earth/index.html
13. http://www.google.com/intl/fr/help/maps/streetview/index.html
14. http://fr.maps.yahoo.com

C. Bing Maps

Bing Maps for Enterprise[15] (anciennement *Microsoft Virtual Earth*) est une plateforme de cartographie géospatiale qui permet aux développeurs de créer des applications qui superposent des données de géolocalisation au-dessus de la carte de *Bing Maps for Enterprise*. Cela inclut les images prises par les satellites d'observation, les caméras aériennes (y compris les images aériennes du type "Bird's eye" prises avec un angle de vue de 45 degrés, dans le but de montrer les façades de bâtiments ainsi que les entrées), mais également des modélisations 3D de villes ou de terrains. La plateforme Bing Maps for Enterprise fournit également une base de données complète de points d'intérêts et la capacité de rechercher des personnes, bâtiments ou adresses. Microsoft utilise *Bing Maps for Enterprise* pour propulser sa plateforme Bing Maps.

D. IGN Géoportail

Le Géoportail[16] est un portail Web public permettant l'accès à des services de recherche et de visualisation de données géographiques ou géolocalisées. Il a notamment pour but de publier des données géographiques de référence de l'ensemble du territoire français. Il est mis en œuvre par deux établissements publics : l'IGN[17] et le BRGM[18], et a été officiellement inauguré le 23 juin 2006.

Qualifié de rival ou de concurrent de Google Maps, le Géoportail IGN a des conceptions voisines pour montrer des cartes au moyen d'une interface en ligne, mais avec des données géographiques différentes. Le Géoportail présente des données publiques de référence avec une qualité technique définie au préalable. Le Géoportail couvre l'ensemble du territoire français selon le principe d'égalité et de satisfaction de l'intérêt général tandis que Google Maps, Yahoo! Maps et Bing Maps couvrent le monde entier avec des résolutions variant en fonction de l'intérêt du lieu. En conséquence, les données couvrant la France proposées par le Géoportail en dehors des villes sont de meilleures qualités que celles de Google, Yahoo! et Bing ; tandis que dans les zones urbaines les données de Google, par exemple, sont parfois meilleures. Les couches cadastrales, les couches cartes, le traitement des bâtiments en 3D ou celui des départements et collectivités françaises d'outre-mer restent des points de différenciation forts du Géoportail.

15. http://www.bing.com/maps/
16. http://www.geoportail.fr
17. Institut national de l'information géographique et forestière
18. Bureau de Recherches Géologiques et Minières

3.2.1.2 Principaux systèmes open-source

A. OpenLayers

OpenLayers[19] est un logiciel libre, publié sous licence BSD. Il constitue une bibliothèque de fonctions JavaScript permettant la mise en place d'applications cartographiques fluides. OpenLayers permet d'afficher des fonds cartographiques tuilés ainsi que des marqueurs provenant d'une grande variété de sources de données.

Il est, par exemple, utilisé pour l'affichage de cartes dont les fonds cartographiques proviennent de Web Mapping Service (WMS), Web Feature Service (WFS), Google Maps, OpenStreetMap, Virtual Earth (Bing Maps), Yahoo! Maps...

B. Open Street Map

OpenStreetMap[20] est un projet sous licence libre. Il est destiné à réaliser la carte du monde grâce à la contribution de tous les utilisateurs. Ceux-ci peuvent utiliser et participer à l'amélioration de la carte. Ce projet est implémenté en se basant sur l'API OpenLayers.

C. Web Mapping Service

Web Mapping Service ou WMS permet de produire des données géographiques à partir de différents serveurs de données. Cela permet de mettre en place un réseau de serveurs cartographiques pour que les utilisateurs puissent construire des cartes interactives.

Un service WMS sert à retourner une image visualisable aux formats JPEG, PNG ou sous forme vectorielle avec SVG. Les WMS peuvent être appelés en utilisant un navigateur web standard en soumettant des demandes directement dans l'URL. Le contenu d'une telle URL dépend de l'opération souhaitée. En particulier, lorsque l'on demande une carte, l'URL doit contenir les informations que l'on désire voir sur la carte :

- Couches à tracer parmi celles disponibles.
- Styles des couches.
- Système de référence à utiliser.
- Taille de l'image produite.
- Étendue de la carte souhaitée.

Lorsque deux cartes ou plus sont produites avec la même localisation géographique, on obtient une carte composée. L'utilisation de formats d'image supportant la transpa-

19. http://openlayers.org
20. http://www.openstreetmap.fr/

rence (comme le GIF ou PNG) permet d'améliorer la superposition et la lisibilité des cartes. En outre, différentes cartes peuvent être demandées sur différents serveurs. Le WMS permet ainsi la création d'un réseau de serveurs rendant disponibles un ensemble de cartes.

Un WMS n'est pas habituellement appelé directement à travers un navigateur Web. Le plus souvent, il est appelé par une application cliente qui fournit à l'utilisateur des commandes interactives.

3.2.1.3 Comparaison des principaux systèmes Web géographiques

Nous comparons les systèmes Web cartographiques dans le tableau suivant (Figure 3.1). Nous choisissons quelques dimensions pour la comparaison :

- **Licence** : propriétaire ou libre ;
- **Niveau de zoom** : nombre d'échelles de zoom sur l'interface utilisateur ;
- **Types** : types de fonds cartographiques affichés aux utilisateurs (Route, Satellite...) ;
- **Technologies utilisées** pour mettre en place les services (JavaScript, XML, JSON...) ;
- **Fournisseurs de données** : entités fournissant ou créant les données cartographiques ;
- **Recherche d'itinéraire** : possibilité (ou non) de fournir un service pour trouver des itinéraires ;
- **API disponible** (ou pas) pour les développeurs.

Nous constatons que JavaScript est indispensable pour créer et manipuler des cartes interactives en ligne. Nous prenons aussi en compte les avantages d'OpenLayers car il fournit une API libre, ouverte pour manipuler tous les fonds cartographiques dont ceux de Googles Maps, de l'IGN, de Yahoo! Maps, de Bing Maps et d'OpenStreetMap.

3.2.2 Information géographique et outils dédiés

Cette partie a pour objectif de présenter les caractéristiques des informations géographiques et les outils dédiés pour les traiter.

3.2.2.1 Caractéristiques de l'information géographique

Selon [LGMR05], "*le problème fondamental de l'information géographique est que celle-ci lie un espace, souvent un instant et quelquefois des propriétés descriptives*". L'**information géographique**, peut donc se définir comme un ensemble de trois facettes : thème, espace et temps [Use96, Gai01]. Elle peut se représenter sous différentes formes : représentation graphique (pour le spatial par exemple en 2D (carte) ou 3D (avec les élévations)), représentation textuelle (sous forme d'expression) ou encore représentation sous forme de données (tuples dans une base de données).

	Google Maps	Yahoo! Maps	Bing Maps	IGN	OpenLayers	OpenStreetMap
Licence	Propriétaire	Propriétaire	Propriétaire	Propriétaire	Libre	Libre
Niveau de zoom	22	17	19	18	varié	N/A
Types	6: Road, Satellite, Hybrid, Street View, Traffic, 3D	3: Road, Satellite, Hybrid, Traffic (US)	9: Road, Aerial, Hybrid, Bird's Eye, Traffic, 3D	plusieurs	tous	Road
Technologies utilisées	JavaScript, Ajax, JSON, XML, WebGL	JavaScript, Ajax, XML, JSON	JavaScript, Ajax, .NET	JavaScript, Ajax	JavaScript	JavaScript
Fournisseurs de données	MAPIT, TeleAtlas, DigitalGlobe, MDA Federal	NAVTEQ, TeleAtlas, i-cubed, Public domain	NAVTEQ, Intermap, Pictometry International, NASA	IGN	tous	Contribution de l'utilisateur
Recherche d'itinéraire	Oui	Oui	Non	Non	Non	Non
API disponible	Oui	Oui	Oui	Oui	Oui	Non

FIGURE 3.1 – Comparaison des systèmes Web cartographiques

Dans notre cas, nous travaillons sur l'information géographique représentée sous forme textuelle. La figure 3.2 reprenant la figure 1.1 illustre l'information géographique avec un exemple textuel sur "**Vignobles au sud de Pau au XXe siècle**". Dans cet exemple, l'information spatiale est représentée par "**au sud de Pau**". Pour le temporel, l'expression "**XXe siècle**" est représentée par un intervalle de temps et permet de retourner toutes les dates ou périodes qui s'y rapportent (par exemple : 1905, été 1960...). Le thème traité dans cet exemple peut être "**vignoble**".

FIGURE 3.2 – Information géographique selon trois facettes : spatiale, temporelle et thématique

3.2.2.2 Ontologies géographiques

Les dictionnaires spatiaux (*gazetteers* ou *gazetiers*) peuvent être manipulés via des outils dédiés tel que les Systèmes d'Information Géographique (SIG). Un *gazetier* est une liste de noms de lieux associés à leur localisation (coordonnées). Ces lieux peuvent aussi préciser diverses caractéristiques (par exemple, statistiques telles que la population, ou physiques telles que le relief).

Prenons l'exemple du gazetier Geonames [21]. Chaque information géographique y est décrite par un nom, un pays, un type (parc, lac, montagne, ville...), une latitude et une longitude. La figure 3.3 montre un exemple de Geonames sur la ville de Biarritz avec le format RDF/XML : le nom de la ville est "Biarritz", elle est de type "P" ou plus précisément "P.PPL" qui est défini dans l'ontologie de Geonames et signifie "une place habitée", elle est dans le pays dont le code est "FR", elle a 33188 habitants, le code postal est 64200 et elle se situe aux coordonnées : -1,55558 de longitude et 43,48012 de latitude.

21. http://www.geonames.org/

```
<rdf:RDF
   xmlns:gn="http://www.geonames.org/ontology#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:wgs84_pos="http://www.w3.org/2003/01/geo/wgs84_pos#">
   <gn:Feature rdf:about="http://sws.geonames.org/3032797/">
      <gn:name>Biarritz</gn:name>
      <gn:featureClass rdf:resource="http://www.geonames.org/ontology#P"/>
      <gn:featureCode rdf:resource="http://www.geonames.org/ontology#P.PPL"/>
      <gn:countryCode>FR</gn:countryCode>
      <gn:population>33188</gn:population>
      <gn:postalCode>64200</gn:postalCode>
      <wgs84_pos:lat>43.48012</wgs84_pos:lat>
      <wgs84_pos:long>-1.55558</wgs84_pos:long>
   </gn:Feature>
</rdf:RDF>
```

FIGURE 3.3 – Exemple de Geonames

Un système d'information géographique permet d'une part de stocker des données spatiales et d'autre part d'utiliser des opérateurs pour les manipuler (intersection, distance...). Les données spatiales peuvent être plus ou moins précises : uniquement des points (latitude/longitude par exemple), seulement les coordonnées du rectangle délimitant l'information spatiale (on parle de boîte englobante ou MBR pour "*Minimum Bounding Rectangle*" en anglais) ou encore la forme géométrique fine (tel qu'un polygone) (Figure 3.4).

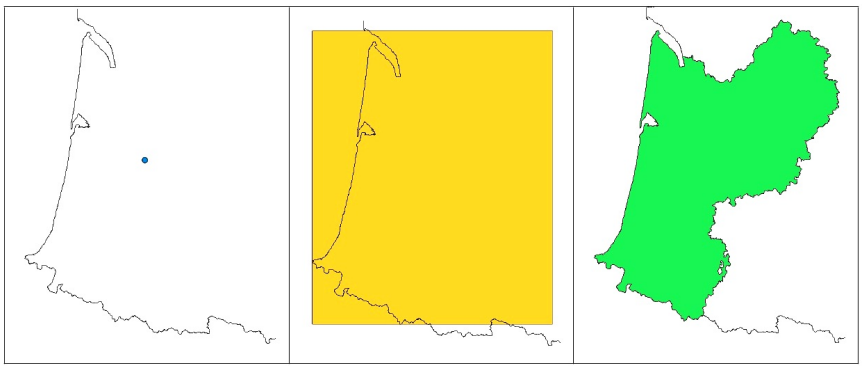


FIGURE 3.4 – Exemple de représentations spatiales possibles pour Aquitaine, respectivement point, boîte englobante (MBR) et polygone

Pour l'information temporelle, le principe est le même : détectée dans un syntagme nominal, elle est successivement représentée sous forme textuelle, symbolique puis numérique (ici ce sont des instants ou intervalles de temps et non des géométries).

Enfin concernant la facette thématique, l'information reste généralement limitée aux termes utilisés. Néanmoins, des termes différents peuvent couvrir des thèmes identiques (exemple : automobile et voiture). Cette approche peut être complétée par des ressources externes (thésaurus, ontologies) contenant des liens de synonymies ou hiérarchiques (Figure 3.5).

Dans ce chapitre, nous considérons donc que l'information géographique est une combinaison des facettes spatiales, temporelles et thématiques. Cependant, nous ne détaillerons pas davantage la facette thématique qui se limitera à l'exploitation de termes.

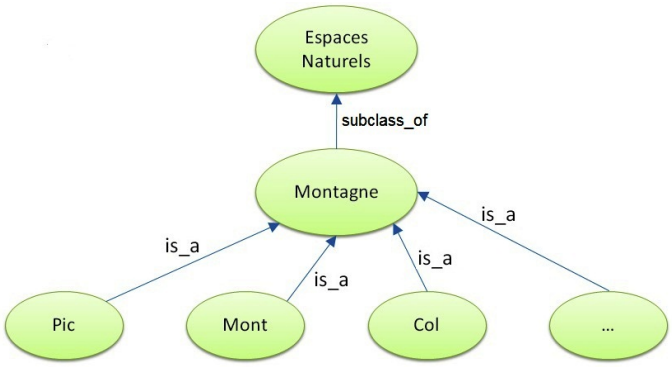


FIGURE 3.5 – Exemple de liens hiérarchiques pouvant être exprimés dans une ontologie

3.2.2.3 Outils développés pour traiter les informations géographiques

Notre domaine d'application contribue clairement à permettre la conception d'applications géographiques avec :

– un ensemble d'outils et de composants logiciels pour extraire automatiquement les informations géographiques au sein de documents textuels. Ces informations géographiques sont composées de trois facettes complémentaires : spatiale, temporelle et thématique [GSE+08]. La "chaîne spatiale" produit une indexation où chaque élément spatial (*Spatial Feature* - SF) est associé à une ou plusieurs géométries. De la même manière, la "chaîne temporelle" associe les éléments temporels (*Temporal Feature* - TF) à un ou plusieurs intervalles de temps et la "chaîne thématique" est basée sur les critères statistiques (e.g., la fréquence des termes). Les fonctions avancées de l'élément spatial relatif sont "**près de Paris**" ou "**à environ 10 km au sud de Paris**" ou encore "**entre Paris et Versailles**" par exemple. De cette façon, l'interprétation de la sémantique de l'information géographique conduit à la construction des indexations spatiale, temporelle et thématique [GSE+08]. Actuellement, deux versions de ces chaînes automatiques sont exploitées : GeoStream [SRL+09] est un service Web qui peut annoter automatiquement les SF dans les documents

textuels, tandis que PIIR [LNG09] est un autre service Web qui peut marquer des verbes de mouvement et les SF afin de trouver des itinéraires spécifiques au sein documents textuels : récits de voyages, guides de voyage...
- un ensemble d'outils et de composants logiciels (composant textuel, composant cartographique, composant calendaire et composant thématique) qui peuvent être paramétrés et combinés [LENM09] avec les services Web précédents afin de concevoir des applications géographiques en relation avec la découverte d'un territoire. Nous mettons en place deux caractéristiques spécifiques de ces applications géographiques par rapport aux applications Web cartographiques disponibles actuellement :
  1. L'accent est mis sur l'interaction et non pas uniquement sur la visualisation des données ;
  2. La carte n'est plus l'élément central, ni le texte, ni le calendrier : l'utilisateur (un apprenant par exemple) a besoin d'interagir sur un de ces composants et le système doit réagir sur un ou plusieurs composants.

Nous présentons ci-dessous quelques outils dédiés à l'extraction des informations géographiques.

A. Pyrénées Itinéraires Virtuels (PIV)

Ce projet a été initié au sein de notre équipe de recherche en 2005 en collaboration avec une médiathèque locale (MIDR) qui a pour ambition de revaloriser son fonds documentaire territorial. L'objectif général de ce projet est de développer des techniques et des outils informatiques de marquage sémantique à des fins d'indexation par les contenus territorialisés des documents, à la fois dans leurs composantes textuelles et imagées. Ces outils permettront à un lecteur-utilisateur de découvrir et de s'approprier un document ou un ensemble documentaire selon une arrière pensée "territoriale". Cette finalité nécessite un marquage des contenus documentaires selon les deux facettes : spatiale et/ou temporelle.

Deux grandes problématiques ont été explorées dans ce projet, tant du point de vue spatial (dans [LGL06, LSG06, LL06, GSE+08]) que temporel (dans [LPLGS07]) :
- la problématique d'Extraction d'Information et de modélisations des informations recherchées ;
- la problématique de Recherche d'Information qui consiste à produire des index performants pour répondre à des requêtes qui prennent en compte le territoire (du point de vue spatial et/ou temporel).

Le prototype développé (Figure A.1 en Annexe A, section A.1) permet d'indexer un fonds documentaire par le biais de l'information géographique qu'il contient. Cette information géographique apparaît sous forme de syntagme nominal. Elle est interprétée d'abord d'un point de vue qualitatif (par exemple l'entité *au sud de Pau* est une entité en

relation d'orientation par rapport à l'entité *Pau*) puis d'un point de vue quantitatif afin d'être indexée par l'intermédiaire de sa représentation géométrique dans un SIG. Cette indexation par la représentation permet de s'abstraire de la façon dont un territoire est évoqué afin de le retrouver lors du requêtage du système. Par exemple à la requête "je souhaite obtenir les documents qui évoquent *le sud de Pau*" le système est en mesure de renvoyer les documents qui évoquent *Gan*, *Jurançon* (communes effectivement au sud de Pau) ou encore des documents évoquant *le nord de Laruns* (commune effectivement au sud de Pau mais plus lointaine).

B. Prototype Interprétation Itinéraires dans des Récits (PIIR)

PIIR [Lou08] est un Prototype pour l'Interprétation d'Itinéraires dans des Récits (Figure A.2 en Annexe A, section A.1). La démarche générale consiste à construire une chaîne de traitement linguistico-géographique, capable d'extraire les déplacements de manière locale au niveau phrastique dans les textes puis de reconstruire l'itinéraire en utilisant des ressources géographiques. Cette chaîne de traitement utilise le langage XML qui permet d'enchaîner facilement différents traitements en ajoutant à l'information extraite dans une phase n l'information extraite à une phase $n+1$.

La figure 3.6 illustre les chaînes de traitement de PIIR. Les deux parties principales sont l'extraction des déplacements et la reconstruction de l'itinéraire.

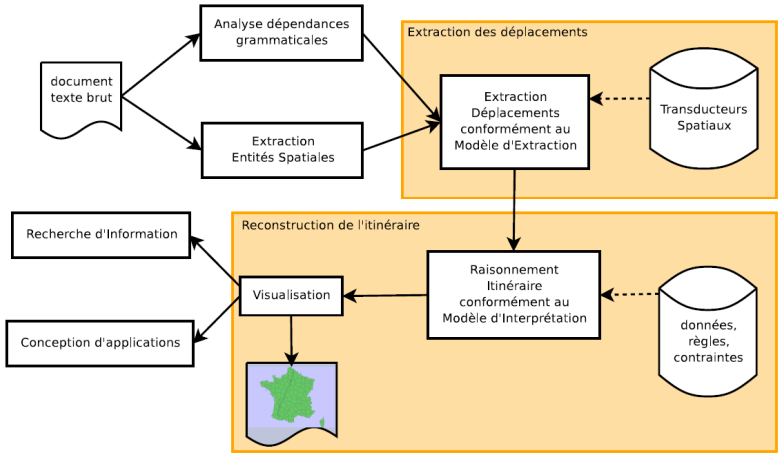


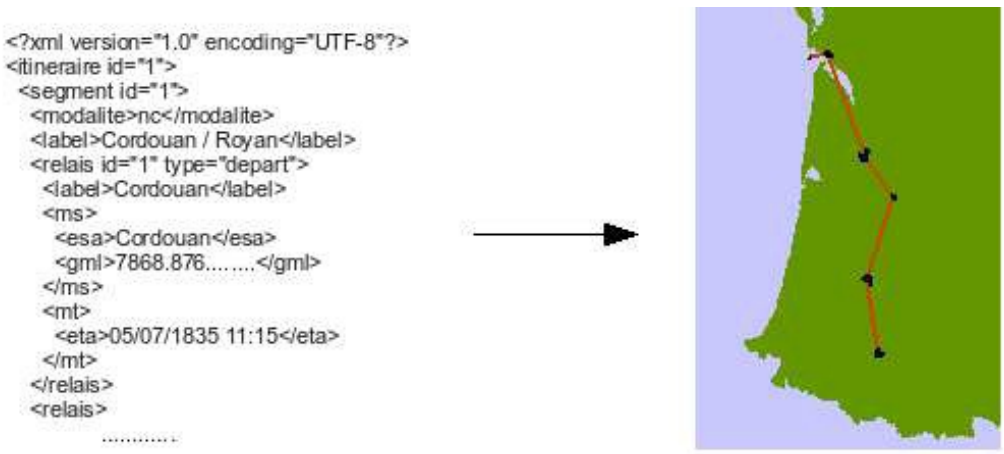
FIGURE 3.6 – Chaîne de traitement de PIIR [Lou08]

L'*extraction des déplacements* est une chaîne de traitement linguistique à part entière. Elle est constituée des grandes phases que nous décrivions dans la section précédente et a été implémentée grâce à la plate-forme de traitement linguistique Linguastream de Widlöcher et Bilhaut (2005). Au sein de celle-ci, l'analyse morpho-syntaxique est ef-

fectuée par Tree-Tagger, l'analyseur morphosyntaxique éprouvé de Schmid (1994). Les transducteurs des verbes de déplacements sont traduits en grammaires DCG [22] et l'analyse basée sur ces grammaires est confiée à Prolog afin de profiter des mécanismes de déduction et d'unification de ce langage.

La *reconstruction de l'itinéraire* est un module permettant de passer des déplacements extraits à un itinéraire. Il prend en entrée un fichier XML dans lequel les déplacements sont représentés selon certains critères. Il utilise le SIG PostGIS [23] afin de mettre en application les règles de raisonnement spatial précédemment évoquées.

Pour chaque déplacement extrait, on cherche tout d'abord à produire une zone probable de localisation de l'acteur. Pour cela, le système fait appel aux règles concernant la polarité aspectuelle et la modalité du transport. Considérons le déplacement de l'exemple suivant : "Nous avons quitté Bordeaux pour Langon en voiture.". Celui-ci a pour origine Bordeaux et pour destination Langon. Il fait donc émerger un segment reliant les deux relais que sont Bordeaux et Langon. La modalité du déplacement étant voiture, le système fait appel à un algorithme de calcul de trajet dans le réseau routier pour déterminer la route probablement empruntée par l'acteur du déplacement.


<?xml version="1.0" encoding="UTF-8"?>
<itineraire id="1">

<modalite>nc</modalite>
<label>Cordouan / Royan</label>
<relais id="1" type="depart">
<label>Cordouan</label>
<ms>
<esa>Cordouan</esa>
<gml>7868.876.....</gml>
</ms>
<mt>
<eta>05/07/1835 11:15</eta>
</mt>
</relais>
<relais>
.....

FIGURE 3.7 – Sortie simplifiée du prototype PIIR : un fichier XML contenant l'interprétation de l'itinéraire

En sortie du prototype PIIR, nous obtenons une instance du modèle des attendus au format XML. Ce fichier XML contient l'ensemble des objets qui permettent de décrire un itinéraire : des segments, des relais et des étapes. Tous ces objets sont des entités géographiques (EG) ; ils contiennent donc un géocodage au format GML et une marque

22. Definite Clause Grammar
23. PostGIS : extension GIS du système de gestion de base de donnée libre PostgreSQL

temporelle. Ces interprétations d'itinéraires peuvent être alors visualisées par des outils de cartographie comme montré dans la figure 3.7. Ils peuvent également intervenir dans la phase de *Contenu* de la conception d'applications Web géographiques avec notre environnement WINDMash.

C. Services Web d'indexation automatique

GéoTopia est une plate-forme en ligne (`http://geotopia.univ-pau.fr/`) destinée à promouvoir la publication, la distribution, l'échange, la discussion et l'analyse des données archivées. Elle intègre des fonctions pour la gestion spatiale et temporelle des images, des dessins et des textes. Une interface cartographique permet de consulter et de manipuler des documents géoréférencés au moyen d'une barre de zoom et d'une *timeline*.

Deux services Web avec des paramètres configurables, GeoStream et TempoStream, ont été développés et sont respectivement dédiés aux indexations spatiale et temporelle des textes. Ces services Web sont intégrés dans la plate-forme GéoTopia.

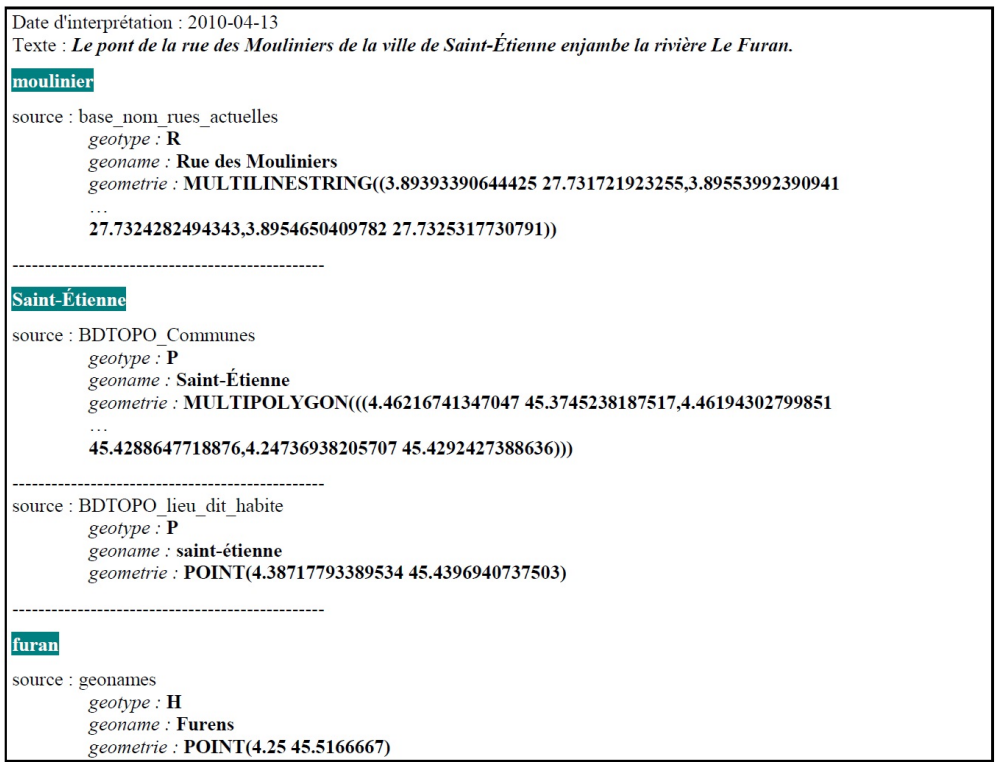
Le service Web d'indexation spatiale appelé **GeoStream** [SRL+09] est un service Web qui peut taguer automatiquement les SF dans les documents textuels. Les traitements du service sont décrits dans un fichier GDD (*Geostream Description Document*). Pour GeoStream, le géoréférencement est le traitement le plus important. Il fait appel à certaines ressources (*gazetteer*) telles que *bdnymes* de l'IGN ou Geonames. Nous montrons ci-dessous la sortie XML (Figure 3.8) mise en forme ici à l'aide d'une feuille de style pour en faciliter la lecture.

Dans le fichier résultat de la figure 3.8, le mot "**Saint-Étienne**" a le *geotype* P (*place*). La ressource `BDTOPO_Communes` a fourni le geoname `Saint-Étienne` ainsi que sa géométrie `MULTIPOLYGON` et la ressource `BDTOPO_lieu_dit_habite` a fourni le geoname `saint-étienne` ainsi que sa géométrie `POINT`.

Le service Web d'indexation temporelle, nommé **TempoStream**, est basé sur le même principe que GeoStream. Un exemple de résultat de TempoStream est présenté dans la figure 3.9.

Le terme "*2000*" est considéré comme une année. Le grain de cette entité temporelle est donc l'année (autres possibilités : jour, mois). Cette année commence le 01/01/2000 et se termine le 31/12/2000.

L'expression "*années 2000*" conduit à une relation d'adjacence de sens "*annee*" qui amène un calcul de période intermédiaire débutant au 01/01/2000 et s'achevant le 31/12/2009.


Date d'interprétation : 2010-04-13
Texte : ***Le pont de la rue des Mouliniers de la ville de Saint-Étienne enjambe la rivière Le Furan.***

moulinier

source : base_nom_rues_actuelles
geotype : **R**
geoname : **Rue des Mouliniers**
geometrie : **MULTILINESTRING((3.89393390644425 27.731721923255,3.89553992390941**
…
27.7324282494343,3.8954650409782 27.7325317730791))

Saint-Étienne

source : BDTOPO_Communes
geotype : **P**
geoname : **Saint-Étienne**
geometrie : **MULTIPOLYGON(((4.46216741347047 45.3745238187517,4.46194302799851**
…
45.4288647718876,4.24736938205707 45.4292427388636)))

---

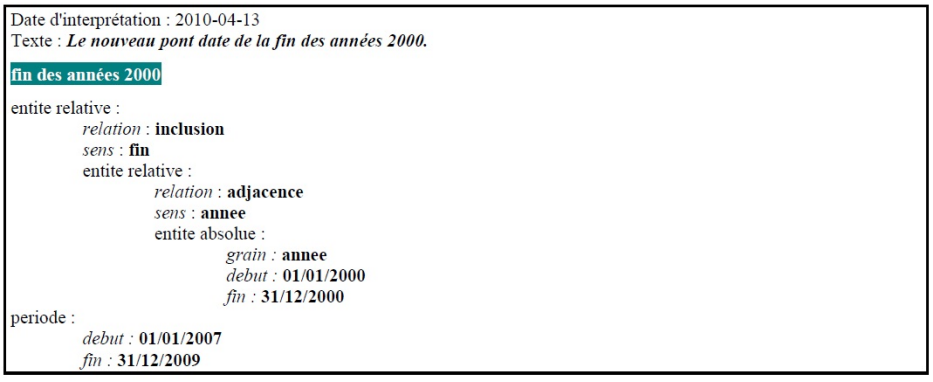
source : BDTOPO_lieu_dit_habite
geotype : **P**
geoname : **saint-étienne**
geometrie : **POINT(4.38717793389534 45.4396940737503)**

---

furan

source : geonames
geotype : **H**
geoname : **Furens**
geometrie : **POINT(4.25 45.5166667)**

FIGURE 3.8 – Un exemple de sortie fournie par GeoStream


Date d'interprétation : 2010-04-13
Texte : ***Le nouveau pont date de la fin des années 2000.***

fin des années 2000

entite relative :
relation : **inclusion**
sens : **fin**
entite relative :
relation : **adjacence**
sens : **annee**
entite absolue :
grain : **annee**
debut : **01/01/2000**
fin : **31/12/2000**
periode :
debut : **01/01/2007**
fin : **31/12/2009**

FIGURE 3.9 – Un exemple de résultat sorti de TempoStream

Enfin, l'expression "*fin des années 2000*" est interprétée par une relation d'inclusion de sens "*fin*". La période recherchée couvre donc le dernier tiers de la période intermédiaire, ainsi elle commence le 01/01/2007 et se termine le 31/12/2009.

D. Quelques autres systèmes dédiés à l'information géographique

De nombreux systèmes existent afin de marquer les toponymes contenus dans des documents à des fins de Recherche d'Information. Ces travaux allient les approches statistiques à d'autres techniques (notamment linguistiques et conceptuelles) afin de répondre à des problématiques spécifiques à l'information géographique.

Le projet Gipsy [WP94] propose par exemple une méthode d'indexation de documents textuels basée sur l'agrégation des géolocalisations des entités spatiales contenues dans les documents. Le document est alors indexé par la zone géographique la plus représentative déduite à partir de l'agrégation des géolocalisations des entités spatiales marquées.

Le projet SPIRIT de [JPR+02] est un projet plus important d'extraction de localisations géographiques dans des pages web. La problématique de ce projet est l'accès à l'information géographique sur le web et la constitution de système de recherche d'information spatiale. Les apports majeurs sont une ontologie géographique, une réflexion sur le classement de pertinence géographique de documents web, une interface multi-modale spécifique et une méthode d'enrichissement des méta-données géographiques. L'ontologie définit une entité géographique (*Geographical Feature*) liée à un nom unique d'entité et à une ou plusieurs variantes de dénomination ainsi qu'à un type géométrique (pour sa représentation) et géographique (de type hôtel, restaurant...). De plus, une entité géographique peut être reliée à elle-même via une ou plusieurs relations spatiales. Ces relations sont utilisées au niveau de la requête et servent à définir l'entité spatiale selon qu'elle soit à l'intérieur, à l'extérieur ou "près" de l'entité nommée.

Le projet GeoSem [24] [EBG+06], créé dans le cadre d'un appel à projet CNRS, a repris cette problématique générale d'accès à l'information géographique dans des documents textuels. Une méthode d'indexation générique des contenus textuels a été proposée. Elle est basée sur une structuration du sens par traits sémantiques. Les index sont donc constitués d'une telle structure pour chaque facette de l'information géographique (espace, temps, thématique). Un prototype de recherche d'information multi-critères (spatiaux, temporels, thématiques) a été développé à partir de ce modèle. Cependant, la volonté de généricité de cette indexation a l'inconvénient de ne pas être optimale. En particulier pour l'information spatiale, les index spatiaux sont stockés sous forme de flux XML et l'appariement se fait sans l'aide d'outils dédiés comme les Systèmes d'Information Géographique (SIG).

24. https://www.info.unicaen.fr/geosem/

3.2.3 Synthèse et discussion

Cette section a permis d'introduire les caractéristiques liées aux informations géographiques, les outils dédiés à les traiter et les systèmes Web pour les visualiser. Nous avons également vu que l'information géographique se compose de trois facettes (spatiale, temporelle et thématique).

Nous pouvons noter que la majorité des systèmes traitent de la facette spatiale. Néanmoins, la plupart se limitent aux informations toponymiques (lieux). Le système SPIRIT [VJJS05] travaille sur des adresses postales (des lieux précis) et gère les relations spatiales exprimées dans des champs spécifiques de l'interface d'interrogation (il ne les gère pas au niveau de l'indexation). Le système STEWARD [LSSS07], utilise deux gazetteers (GNIS Geographic Names Information System pour les USA et GNS - GEOnet Names Serveur pour le reste du monde) mais n'a pas accès à des opérateurs spatiaux pour traiter des relations spatiales. Le système GeoSem [BDEH07], malgré l'absence de SIG, gère certaines relations spatiales (orientation, proximité) définies par des règles.

Peu de systèmes traitent du temporel. Les systèmes DIGMAP [MMBS09] et CITER [PEH+09] gèrent uniquement les dates, tandis que les systèmes GeoSem [BDEH07] et PIV [LGS07] traitent à la fois les dates et les relations temporelles. Dans ces systèmes, les informations temporelles sont représentées par des intervalles de temps (une date de départ et une date de fin).

Seul CITER [PEH+09] utilise des concepts provenant d'une ontologie. L'utilisateur doit sélectionner un thème dans la liste disponible mais ne peut pas fournir d'autres mots-clés comme c'est le cas dans les autres systèmes qui utilisent des approches statistiques standards de RI sur les termes.

Concernant les représentations des informations spatiales, la plupart des systèmes travaillent avec des boîtes englobantes (MBR). Seuls DIGMAP [MMBS09] et CITER [PEH+09] se basent sur des données ponctuelles, et PIV [LGL06, SGPL08] sur des données géométriques (polygones).

Dans les parties qui suivent, nous allons nous intéresser à comment gérer et afficher les informations géographiques sur le Web.

3.3 État de l'art : Environnements Web pour la gestion et l'affichage des contenus

Les environnements Web permettent à l'utilisateur de n'installer aucun programme supplémentaire sauf à utiliser un navigateur qui permet d'accéder aux données de n'importe où via le Web. En effet, la popularité du Web l'a fait évoluer depuis des pages

affichant de l'information en une véritable plate-forme permettant aux utilisateurs de récupérer et de combiner les informations à distance.

3.3.1 Étude des outils de Mashup

Dans cette section, nous présentons certains outils Web permettant de faciliter la gestion et l'affichage de données telles les informations géographiques. Nous divisons les outils selon le type d'utilisateurs qui les utilisent (développeurs, utilisateurs avancés et utilisateurs finaux).

3.3.1.1 Mashups pour les développeurs

A. Google Mashups Editor

Google Mashup Editor [25] (GME) était un environnement de développement Web 2.0 pour l'intégration de certains services de Google, tels que Google Maps et Google Calendar. GME permettait aux développeurs de créer rapidement et facilement des applications Web simples et Mashups avec des services Google.

Créer des applications avec GME est "intuitif" si l'utilisateur a des compétences de programmation et est familier avec les technologies Web, telles que HTML, CSS, XML et JavaScript. Les applications GME sont décrites sous forme de fichiers XML qui sont interprétés par le moteur de Google. Ainsi, GME est plutôt un outil de développement logiciel et non pas un environnement de Mashup pour l'utilisateur final.

B. Exhibit

Exhibit [HKM07] est une plate-forme pour la publication de données structurées sur des serveurs Web standards et ne nécessite aucune installation, ni base de données ou programmation. Exhibit permet aux auteurs d'écrire des pages Web interactives qui exploitent la structure de leurs données pour une meilleure navigation / visualisation.

Exhibit permet aux concepteurs qui ne connaissent que le langage HTML de créer des pages Web contenant des contenus riches et des visualisations dynamiques de données structurées. Ces concepteurs n'ont pas besoin d'installer, de configurer et de maintenir une base de données ou bien d'écrire des lignes de code côté serveur. Ils ne s'intéressent qu'à des données structurées qu'ils souhaitent publier et exposer dans leurs applications HTML.

Dans la figure 3.10, les informations sur les présidents américains [26] se présentent selon plusieurs représentations : carte, frise chronologique (*timeline*) et liste.

25. Google Mashup Editor a été arrêté et migré dans Google App Engine.
26. http://www.simile-widgets.org/exhibit/examples/presidents/presidents.html

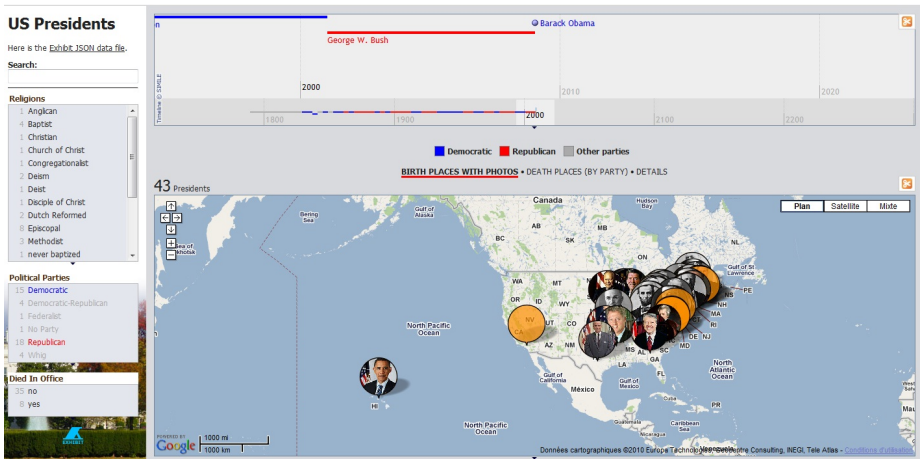


FIGURE 3.10 – Page Web intégrant Exhibit pour afficher des informations sur les présidents américains

Avec Exhibit, les données peuvent être visualisées en plusieurs vues : mosaïque, miniature, tabulaire, frise chronologique et cartographique. Exhibit fournit actuellement des exportateurs aux formats RDF/XML, JSON, code wiki sémantique d'extension MediaWiki [VKV+06] et Bibtex. L'objectif de ces exportateurs est de faciliter et d'encourager la réutilisation de données structurées en offrant des avantages pour les utilisateurs. Créer une application Exhibit consiste en deux tâches : la création des données et la création de leur présentation. Exhibit ne fournit pas un environnement ou outil visuel pour la création de l'application. Les données Exhibit peuvent être éditées dans un éditeur de texte. Exhibit peut lire des données au format JSON ou au format de flux RSS d'une feuille de calcul de Google. Pour créer une application Exhibit, les concepteurs ont donc besoin de connaissances avancées sur la programmation Web. Une application Exhibit fournit les interactions entre les composants mais de manière prédéfinie. Par exemple, cliquer sur un élément d'une liste permet de synchroniser la carte avec cet élément sélectionné ; cliquer sur une image d'un président sur la carte fait apparaître une fenêtre avec des informations complémentaires sur ce président...

C. Chickenfoot

Chickenfoot [BWR+05] a été développé par le laboratoire CSAIL [27] (*Computer Science and Artificial Intelligence Laboratory*) du MIT (*Massachusetts Institute of Technology*) comme une extension de Firefox. Le script de Chickenfoot est écrit en JavaScript et nécessite une certaine expérience en programmation. Ainsi, l'abstraction dans Chickenfoot a été bien accomplie car les utilisateurs peuvent écrire le script en utilisant des mots-clés anglais au lieu d'utiliser la manipulation DOM en JavaScript.

27. http://www.csail.mit.edu/

Par exemple, dans la figure 3.11, pour saisir la valeur “windmash” dans la zone de recherche, un programmeur Chickenfoot peut écrire : `enter("windmash")` au lieu d'écrire :

```
document.getElementsByTagName("input")[0].value = "windmash";
```

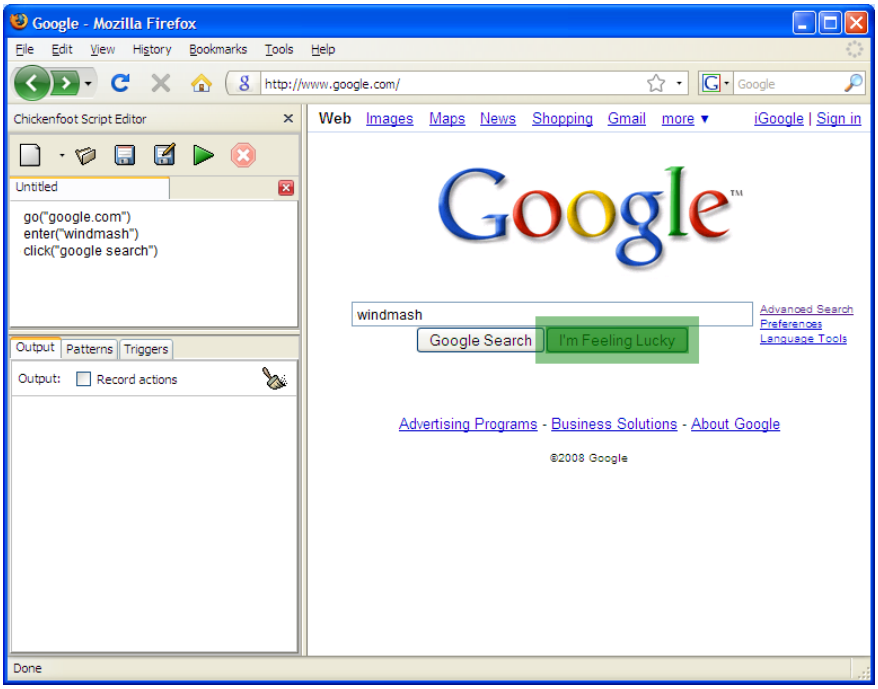


FIGURE 3.11 – Environnement de développement de Chickenfoot sous Firefox

L'inconvénient principal de l'utilisation de mots-clés est qu'il y a peu d'indication sur les mots-clés utilisés disponibles dans le système. L'utilisateur doit consulter la documentation et le processus est similaire à l'utilisation d'une interface système de type ligne de commande (*command-line*), telle que *bash shell*.

D. Piggy Bank

Piggy Bank [HMK05] montre comment les utilisateurs peuvent utiliser plus efficacement les données existantes en combinant des données provenant de différents sites ou en combinant des données publiques et privées.

Bien que la promesse du Web sémantique soit présentée depuis plus d'une décennie, les utilisateurs finaux n'ont pas encore tiré parti des technologies qui en sont issues. L'utilisation des données RDF (*Resource Description Framework*) est rarement effectuée par les webmasters. Pour montrer un exemple d'utilisation du Web sémantique, Piggy Bank

a été implémenté sous une extension Firefox permettant aux utilisateurs d'extraire les informations depuis les pages Web et de sauvegarder les données extraites en RDF. Les aspects de la programmation de Piggy Bank sont les suivants : les utilisateurs peuvent écrire et soumettre les capteurs de données d'écran [28] (*screen scrapers*) en JavaScript pour les sites qu'ils visitent souvent, et les utilisateurs peuvent partager l'information qu'ils créent dans Mashups Piggy Bank avec d'autres utilisateurs, devenant ainsi les auteurs.

Bien que Piggy Bank soit l'un des plus riches outils de description de données en RDF, il ne semble pas avoir beaucoup de considération sur les expériences d'utilisation du Web sémantique. De plus, il n'y a pas de modèle générique sur les données capturées du Web. Il reste donc inaccessible aux utilisateurs qui voudraient utiliser les informations recueillies d'une manière significative parce que les utilisateurs ont besoin de connaissance sur le langage de programmation JavaScript pour écrire et utiliser les capteurs de données.

3.3.1.2 Mashups pour les utilisateurs avancés

A. Yahoo! Pipes

Yahoo! Pipes [29] fournit des outils permettant de spécifier un flux de données pour combiner les flux RSS ou Atom et les données XML ou JSON.

Yahoo! Pipes est un environnement visuel permettant de récupérer et fusionner des données provenant de différentes sources. Il ne nécessite aucune connaissance des langages de programmation mais il nécessite encore une bonne compréhension d'un des formats de données (par exemple : JSON ou XML).

Cet outil s'exécute dans un navigateur et est basé sur des technologies Web standards. L'interface de création Mashup est visuellement divisée en trois zones (Figure 3.12) :

- à gauche, il y a une bibliothèque qui liste tous les modules fonctionnels qui peuvent être tirés de l'Internet comme par exemple Yahoo! Search, Flickr...
- au centre, c'est la zone de travail sur laquelle l'utilisateur peut glisser et déposer des modules sélectionnés de la bibliothèque. Les modules sont liés avec les connecteurs ou "tuyaux" qui définissent le flux de données.
- En bas au centre, il y a une zone de débogueur qui permet de vérifier les résultats intermédiaires.

Il y a plusieurs opérateurs traitant le flux de données. On peut combiner plusieurs flux en un seul, puis trier, filtrer le flux... La sortie de Yahoo! Pipes peut être visualisée sur des types d'interfaces différents tels qu'une carte interactive ou un site Web. Cependant, les données visualisées sur les composants d'interface ne se sont pas synchronisées ensemble

28. La capture de données d'écran est une technique par laquelle un programme récupère les données envoyées à un dispositif de sortie (généralement un moniteur) par un autre programme.
29. http://pipes.yahoo.com/pipes/

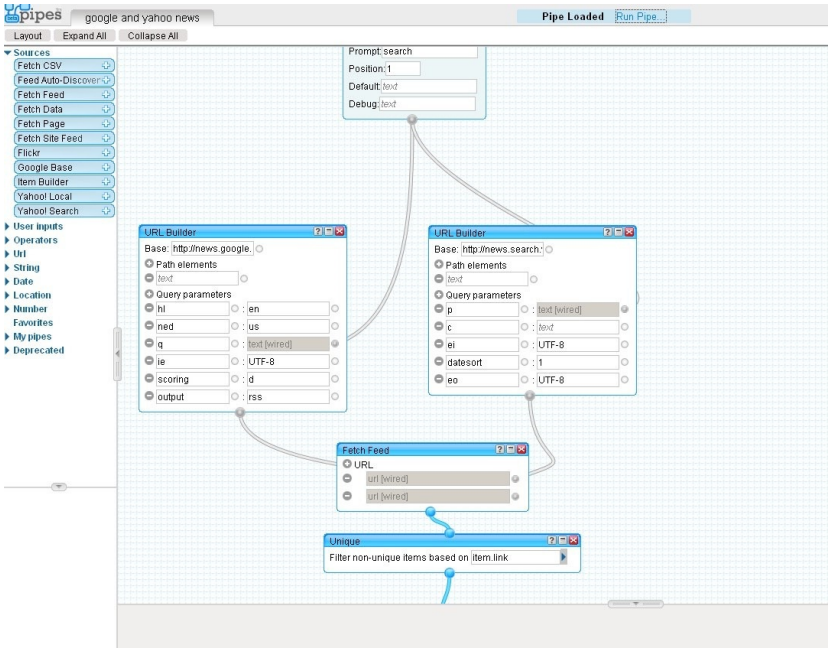


FIGURE 3.12 – Environnement de création de Mashups Yahoo! Pipes

(par exemple : lors d'un clic sur un nom de lieu dans une liste, la carte effectue un zoom avant sur ce lieu).

B. Popfly

Microsoft Popfly [30] était un outil très puissant qui pouvait être utilisé pour créer différents types de Mashups. Il fournissait les composants de base fonctionnels de diverses catégories.

Microsoft Popfly permettait à l'utilisateur de créer et de partager des sites Web. Il comportait deux parties : le réseau social appelé "Popfly Space" et l'outil en ligne appelé "Popfly Creator" pour créer différents types d'expériences.

Comme Yahoo! Pipes, il avait un menu avec des blocs fonctionnels à gauche et la zone de travail à droite. Les modules étaient liés avec des connecteurs les uns aux autres sur la zone de travail (Figure 3.13). Les connexions entre blocs visuels dans Popfly étaient mises en œuvre en utilisant la technologie Silverlight de Microsoft, ce qui est un inconvénient car ceci nécessitait une installation du plug-in Silverlight avant son utilisation.

30. Popfly de Microsoft a pris fin à partir de l'été 2009.

FIGURE 3.13 – Création des Mashups avec Popfly

Les modules fournissaient une recommandation au sujet des liens possibles avec les autres blocs. L'utilisateur pouvait choisir une des multiples options pour visualiser les résultats. Par exemple, il était possible de visualiser l'ensemble d'images sous forme d'album, de carrousel, de livre...

C. Damia

Damia [ABC+07] est un service d'intégration de données pour que des utilisateurs professionnels puissent créer des flux de données consommés par des applications d'entreprise. Il se compose :

- d'un navigateur de l'interface utilisateur (Figure 3.14) permettant la spécification de Mashups de données sous forme de flux de données en utilisant un ensemble d'opérateurs ;
- d'un serveur avec un moteur d'exécution ;
- des API pour la recherche, le débogage, l'exécution et la gestion des Mashups.

La figure 3.14 montre un aperçu de l'éditeur de Mashup Damia, qui a été mis en œuvre avec le Toolkit Dojo [31]. Il communique avec le serveur via un ensemble d'interfaces API REST, comme l'illustre la figure 3.14. L'interface graphique permet à l'utilisateur de glisser et de déposer des blocs, qui représentent les opérateurs Damia, sur la zone de travail et de les connecter pour créer les flux de données entre les opérateurs. L'utilisateur peut utiliser une fonction de prévisualisation pour voir le résultat du flux de données à n'importe quel point du processus. Une fois le flux de données bien exécuté, le résultat est sérialisé en un document XML et transmis au serveur pour un traitement ultérieur.

31. http://dojotoolkit.org/

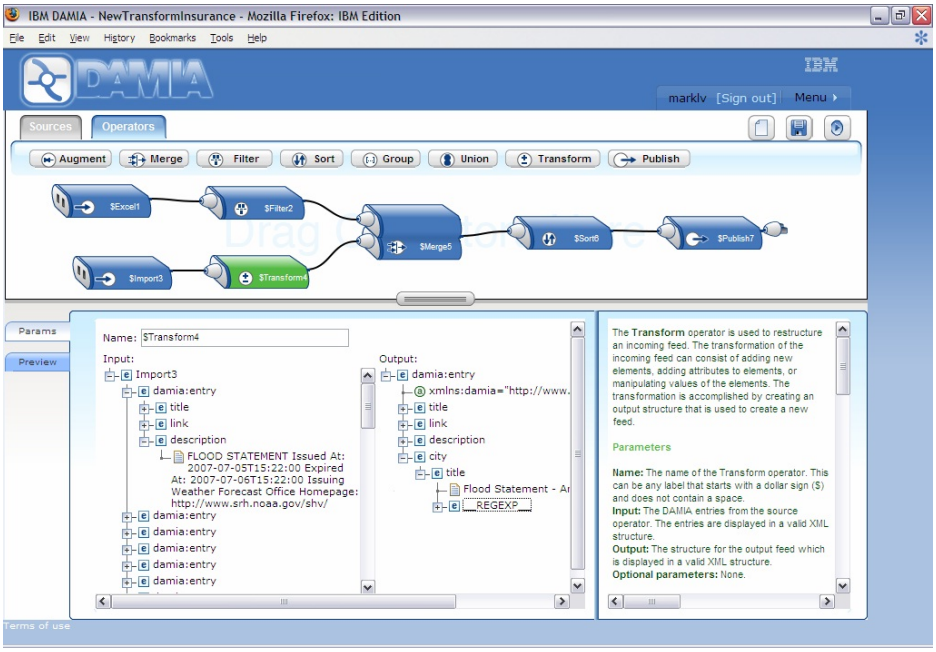


FIGURE 3.14 – Editeur Mashup de Damia

Damia va au-delà de Yahoo! Pipes de plusieurs façons : Damia dispose d'un modèle de données plus général que celui de Yahoo! Pipes ; Damia se centre sur les données d'entreprise et permet l'ingestion d'un grand ensemble de sources de données ; le modèle de données de Damia permet de regrouper plus de sources de données Web.

Comme Yahoo! Pipes, Damia ne prend en compte que le traitement des flux de données. Il ne supporte pas la synchronisation entre les données dans des composants d'interface.

3.3.1.3 Mashups pour les utilisateurs finaux

Les environnements de programmation de type Mashup pour l'utilisateur final sont une nouvelle génération d'outils visuels en ligne permettant aux utilisateurs de rapidement créer, par exemple, des applications Web. Ils s'appuient sur des métaphores qui sont faciles à saisir par des non-programmeurs professionnels.

A. Afrous

Afrous[32] est un exemple de plate-forme de Mashup permettant aux utilisateurs de créer et d'exécuter leurs applications via un navigateur Web.

32. http://www.afrous.com/en/

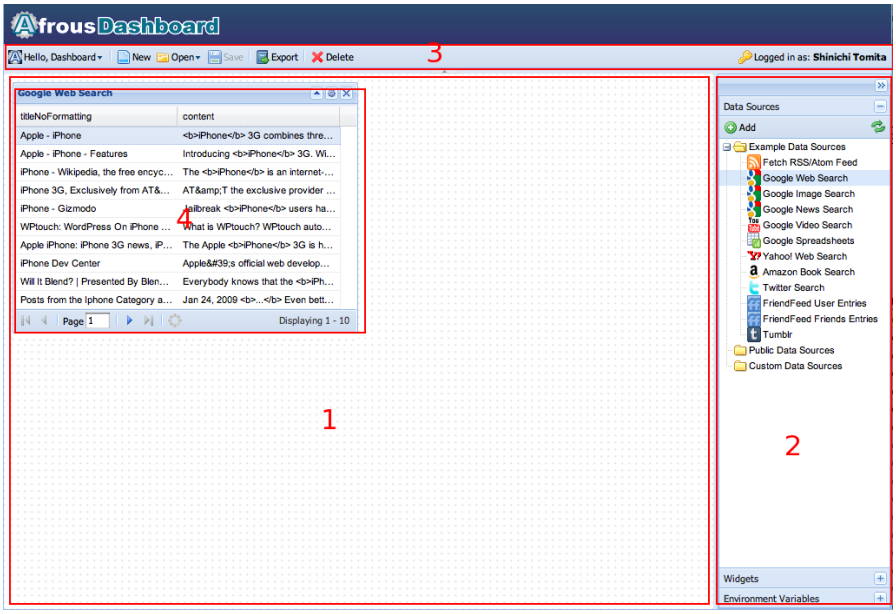


FIGURE 3.15 – Tableau de bord d'Afrous

Comme illustré dans la figure 3.15, le tableau de bord d'Afrous se compose des composants suivants :

1. La zone de travail du tableau de bord (*Dashboard Main Panel*) : une zone principale qui contient des widgets à afficher dans une disposition fixe. La présentation est définie lors de la création de nouveaux widgets, et peut être changée via le menu "Layout Change" dans la barre de menu.
2. Le tiroir (*Drawer*) : un panneau avec une barre latérale qui contient plusieurs éléments de personnalisation du tableau de bord tels que des sources de données ou des widgets.
3. La barre de menu (*Application Menubar*) : elle contient des opérations de base d'administration (par exemple : Nouveau, Ouvrir, Enregistrer ou Supprimer).
4. Widget : un cadre rectangulaire avec une zone de la barre de titre qui affiche les contenus. Il peut être déplacé à l'intérieur de la zone de travail.

B. Marmite

Marmite [WH07] est un outil permettant aux utilisateurs finaux de créer des Mashups en extrayant facilement le contenu depuis une page Web, en la traitant à la manière d'un flux de données, en l'intégrant avec d'autres sources de données et en l'orientant vers d'autres outils, tels que des bases de données ou encore des services de fonds cartographiques.

Marmite, comme Chickenfoot, est implémenté comme un plugin de Mozilla Firefox écrit en XUL (*XML-based User interface Language*) et JavaScript. Marmite se concentre sur trois aspects :

1. la facilité de sélection du contenu à analyser ;
2. le développement d'un flux de données (*dataflow*) hybride, l'interface utilisateur montrant le contenu extrait et comment le contenu est transformé ;
3. le développement des techniques de gestion des exceptions dans le flux de données.

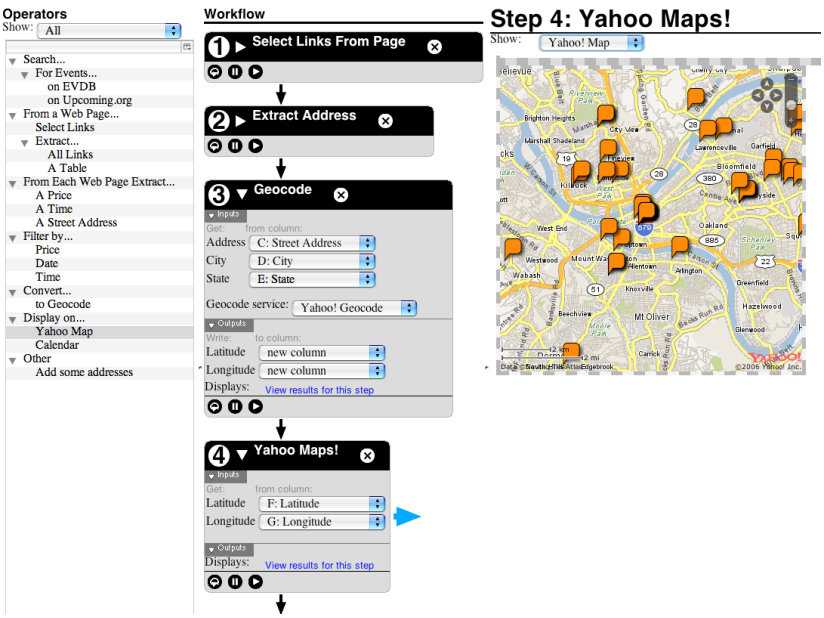


FIGURE 3.16 – Marmite, un outil de programmation de Mashups pour l'utilisateur final

Dans la figure 3.16, on visualise à gauche des opérateurs, au milieu le flux de données et à droite une visualisation cartographique. Cette figure illustre un scénario de conception avec Marmite : "*trouver toutes les adresses sur un ensemble de pages Web, puis utiliser le service de géolocalisation de Yahoo! et enfin afficher ces lieux sur une carte Yahoo! Map*".

Marmite est une solution de Mashup inspirée par Apple Automator, un outil visuel pour automatiser des tâches répétitives sous le système d'exploitation MacOS. Les utilisateurs peuvent choisir à partir d'un certain nombre d'activités d'extraction de données des sites Web et des bases de données locales ou distantes.

L'idée est que la structure d'un Mashup Marmite comprend des sources, les transformateurs et les afficheurs. Les sources permettent d'ajouter des données dans Marmite en

interrogeant des bases de données, extrayant des informations à partir de pages Web... Les transformateurs permettent de modifier, de combiner ou de supprimer des résultats existants. Les afficheurs peuvent afficher la sortie du flux de données de Marmite.

C. MashMaker

Mashmaker [EG07] est un outil simple adapté aux utilisateurs finaux intéressés à la création de pages Web combinant des informations provenant de diverses sources.

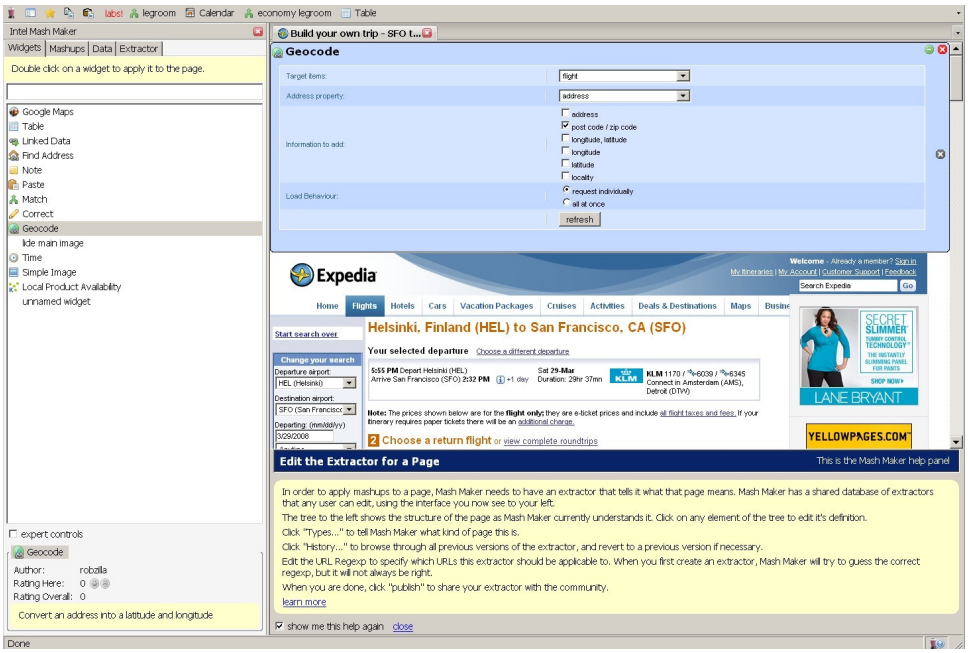


FIGURE 3.17 – MashMaker intégré dans le navigateur

Intel MashMaker est une extension pour Firefox qui permet à l'utilisateur de facilement ajouter dans la page qu'il parcourt des informations provenant d'autres sites. Lorsqu'il navigue sur le Web, la barre d'outils MashMaker suggère des Mashups qu'il peut intégrer à la page en cours afin de l'enrichir avec plus d'informations pour l'utilisateur. Par exemple : "*trouver toutes les villes autour du centre de la carte en cours d'affichage*".

MashMaker a été mis en œuvre pour les utilisateurs finaux. N'importe qui peut créer de nouveaux Mashups avec MashMaker, en utilisant une simple interface par copier-coller. Une fois qu'un utilisateur a créé un mashup, ce mashup sera automatiquement proposé à d'autres utilisateurs [EG07].

Cependant, il n'y a pas de page Web dédiée où l'utilisateur peut visiter et construire l'application Mashup. L'utilisateur doit installer la barre d'outils dans le navigateur Web et commencer à naviguer sur le Web (Figure 3.17). Ce paradigme de programmation pourrait être appelé "*annoter et mélanger pendant la navigation*". Le service est en version bêta et n'est pas disponible au grand public pour le moment.

D. Mashlight

Dans [ABCG09], les auteurs présentent une plate-forme baptisée Mashlight permettant de créer et d'exécuter des Mashups. Elle fournit une interface simple pour créer des applications Web 2.0 de Mashup. Les utilisateurs qui n'ont aucun savoir-faire technique (non-informaticiens) peuvent enchaîner des blocs de construction à partir d'une bibliothèque pour définir l'application qu'ils veulent. Cette plateforme est implémentée en utilisant les technologies Web 2.0 et en se basant sur les concepts de *Mashlight block* et *Mashlight process*.

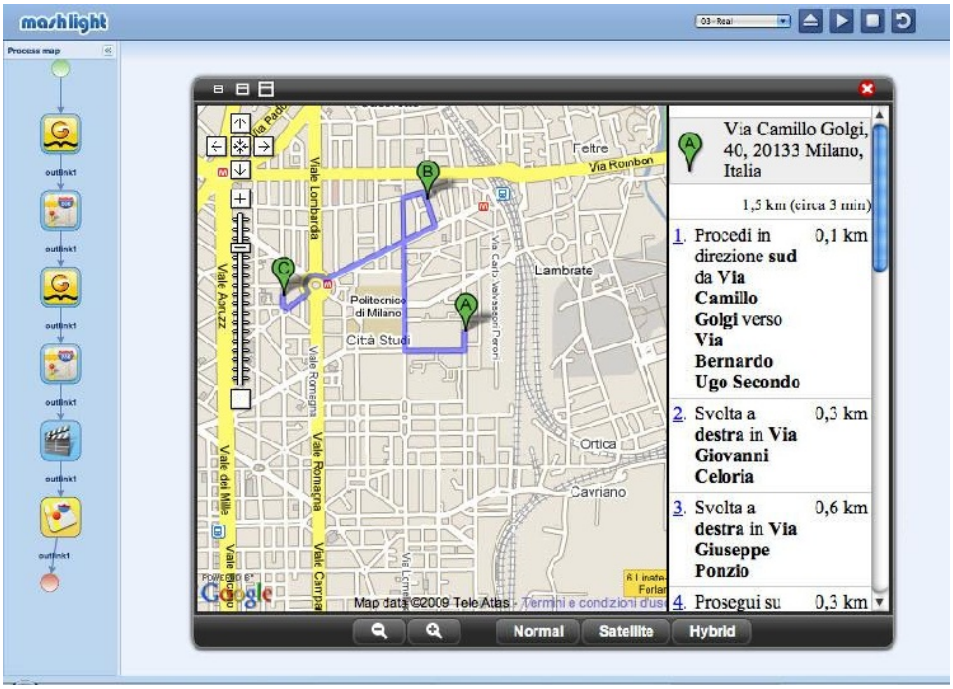


FIGURE 3.18 – Interface de Mashlight

La figure 3.18 illustre un exemple de Mashlight avec lequel l'utilisateur peut trouver un restaurant et un cinéma à Milan.

3.3.2 Synthèse et discussion

Nous comparons les environnements de Mashup dans le tableau suivant (Figure 3.19) pour lequel nous avons choisi les dimensions suivantes pour la comparaison :

- **Paradigme** : c'est le paradigme de programmation ou la manière de création du Mashup (par exemple : flux de données, codage...) ;
- **Fonctionnalité** : la richesse des fonctions des outils ;
- **Utilisation** : le type d'utilisateur (développeurs, utilisateurs avancés ou finaux) ;
- **Technologie** : les technologies utilisées pour implémenter les outils de Mashup (par exemple : AJAX, Silverlight...) ;
- **Intégration** : la façon pour intégrer les Mashups dans un site Web ;
- **Service Web** : indique s'il supporte un(des) service(s) Web.

Comme présenté dans le tableau 3.19, nous pouvons constater que les environnements Mashup utilisent la technologie JavaScript, ce qui facilite la spécification de l'interface des applications Web. De même, les flux de données sont beaucoup utilisés pour créer les contenus dans les applications Mashup.

Toutefois, le développement d'une application géographique avec ces environnements Mashup est encore difficile, même s'ils sont génériques et s'appliquent à plusieurs domaines. En effet, ces systèmes sont génériques et ne sont pas exclusivement conçus pour construire des applications géographiques, d'où le fait qu'ils ne proposent pas de plate-forme pour la conception de ce type d'application. En outre, beaucoup d'entre eux ne prennent pas en compte la spécification d'interaction entre les contenus affichés sur l'interface de l'application.

En particulier, nous trouvons que les deux phases *Contenu* et *Interface* de notre processus de conception (présenté dans la figure 2.6) sont dans une approche similaire à celle de Marmite. Néanmoins, ce système ne propose pas de modèle générique permettant de représenter les dimensions spatiales, temporelles et thématiques d'une information géographique[33].

Afin de prendre en compte l'information géographique dans sa globalité au sein de dans la phase *Contenu* du processus de conception (*cf.* Chapitre 2), il nous faut proposer dans la section suivante le modèle des concepts concernant l'information géographique. Ce modèle permet à notre plate-forme de générer des données normalisées (après la manipulation d'un outil de type Mashup) qui sont utilisables dans les phases suivantes du processus de conception.

33. En outre, Marmite ne permet pas de définir les interactions dans l'application générée.

	Paradigme	Fonctionnalité	Technologie	Intégration	Service Web
Pour les développeurs					
GME	Codage	Riche	XML	Code XML à intégrer dans iGoogle	N/A
Exhibit	Codage	Riche	HTML, JSON	Non	Non
Chickenfoot	Plug-in de Firefox	Pauvre, ligne de commande	JavaScript	Non	Non
PiggyBank	Plug-in de Firefox	Pauvre	RDF, JavaScript	Non	Non
Pour les utilisateurs avancés					
Yahoo! Pipes	Flux de données	Riche	Web standard, YUI	Code HTML à intégrer	Oui
Popfly	Flux de données	Riche, intégrée aux services	Silverlight	Code HTML à intégrer	Oui
Damia	Flux de données	Riche, prévisualisation	REST, XML	Non	Oui
Pour les utilisateurs finaux					
Afrous	Application on canvas	Riche	JavaScript	Code HTML	Oui
Marmite	Flux + spreadsheet	Pauvre	Plugin de navigateur	N/A	Oui
MashMaker	Enrichissement de navigation	Riche	Plugin de navigateur	N/A	Oui
Mashlight	Flux de données	Pauvre	JavaScript, XML	Non	Oui

FIGURE 3.19 – Comparaison des environnements de Mashup

3.4 Contribution : Phase *Contenu*

Cette phase se concentre sur les contenus (à connotation géographiques) que les concepteurs voudraient présenter à l'utilisateur lors de l'exécution d'une application. Les contenus représentent le concept central de notre processus car notre approche de conception a pour but de valoriser des données choisies par le concepteur. Cette valorisation se concrétise par :

- une présentation de ces contenus (à l'écran) selon des modes de représentation variés (*cf.* la phase *Interface* - Section 3.5);
- une mise en valeur de ces données auprès de l'utilisateur via des interactions (*cf.* la phase *Interaction* - Section 4.3).

3.4.1 Modèle conceptuel

L'annotation est un principe qui permet d'ajouter des informations à un document selon certains niveaux comme un mot, une phrase, un paragraphe, une section ou tout le document. Cette information est également appelée une "méta-donnée" qui est une donnée au sujet d'une autre donnée.

Dans ce qui suit, nous rappelons quelques définitions de l'annotation :

- Une annotation est un commentaire sur un objet tel que le commentateur veut qu'il soit perceptiblement distinguable de l'objet lui-même et le lecteur l'interprète comme perceptiblement distinguable de l'objet lui-même (dans le contexte des Interfaces Homme Machine) [BCGP00].
- L'annotation est l'activité du lecteur qui consiste à poser des marques graphiques ou textuelles sur un document papier, et ce suivant plusieurs objectifs [Hua96].
- Une annotation est une note particulière attachée à une cible. La cible peut être une collection de documents, un document, un segment de document (paragraphe, groupe de mots, mot, image ou partie d'image...), une autre annotation. À une annotation correspond un contenu, matérialisé par une inscription, qui est une trace de la représentation mentale que l'annotateur se fait de la cible. Le contenu de l'annotation pourra être interprété à son tour par un autre lecteur [BBC03].
- L'annotation est l'action d'annoter ou résultat de cette action. L'action d'annoter est définie comme accompagner un texte de notes ou de remarques [tre92].

[Mil05] a proposé une synthèse sur l'annotation :

- L'annotation peut être considérée comme une trace de l'activité de lecture. L'activité d'annotation est donc considérée comme une des activités de la lecture active.
- Définir l'objet d'annotation revient souvent à en décrire son aspect graphique (les formes visuelles possibles).
- L'objectif d'une annotation est fondamental : quand une personne annote c'est pour un but précis.
- Les annotations peuvent être textuelles ou graphiques.

Ainsi selon [Mil05], *une annotation est une trace de l'activité du lecteur, perceptible sur un document en tant que marque, placée dans un but spécifique, et en un lieu spécifique dont elle ne peut être dissociée.*

Nous retenons la définition qu'une annotation peut être un objet que l'utilisateur ajoute au document sous la forme textuelle ou graphique. Notre concept de l'annotation est étendu pour le concepteur ainsi que l'utilisateur final, non seulement dans un document textuel, mais aussi dans un document se composant de textes, de cartes, de calendriers et de photos.

Dans cette section, nous proposons un modèle générique (Figure 3.20) pour décrire les contenus d'applications Web géographiques [LLN11]. En effet, nous souhaitons nous abstraire des différents formats de métadonnées existants pour pouvoir les combiner, les confronter... Nous considérons des contenus (`Content`) ① qui peuvent être composés de plusieurs segments (`Segment`) ②. Comme illustré dans la figure 3.20, un contenu peut détenir plusieurs annotations (`Annotation`) ③ faisant référence à des segments spécifiques. Une annotation peut être connectée avec d'autres annotations grâce à des propriétés (`property`) ④ différentes. Par exemple, une annotation sur "Biarritz" est liée à une annotation sur "Anglet" par la propriété dont l'`URI` est `wind:nextTo` (à côté de).

Actuellement, nous envisageons exclusivement des contenus textuels comme point de départ dans une application géographique. Par conséquent, comme présenté dans la figure 3.20, les textes (`Text`) ⑤ peuvent être segmentés en paragraphes (`Paragraph`) ⑥ et tokens (`Token`) ⑦ qui héritent d'un segment textuel (`TextSegment`) ⑳. Un segment textuel peut avoir une position dans le texte de départ qui a servi à l'extraire (par exemple, 6e mot, 4e paragraphe), une valeur présentée sous forme d'une chaîne de caractères (par exemple, "*Paris*", "*J'habite à 10 kilomètres au sud de Paris.*").

Par ailleurs, dans notre modèle de *Contenu*, l'information géographique (`Geographic-Information`) ⑧ est une annotation ③ (lien d'héritage). Comme annoncé dans la partie 3.2.2.1, l'information géographique est composée de trois facettes : spatiale, temporelle et thématique. Ainsi, une information géographique (`GeographicInformation`) ⑧ peut avoir plusieurs représentations (`Representation`) ⑨ de nature spatiale (`Spatial-Information`) ⑩, temporelle (`TemporalInformation`) ⑪ ou thématique (`Thematic-Information`) ⑫.

La représentation spatiale d'une annotation se traduit toujours par des coordonnées géographiques permettant de situer l'annotation sur une carte. Elle est décrite selon une forme qui peut être un point (par exemple, une localisation ou un endroit), une ligne (par exemple, une rivière ou une route), un polygone (par exemple, une zone ou une ville) ou un ensemble de coordonnées géographiques correspondant à cette forme. La représentation temporelle d'une annotation peut être un instant dans le temps (par

exemple, la date "*2011-10-26*"), ou une période (par exemple, "*du 1er octobre 2011 au 31 décembre 2011*").

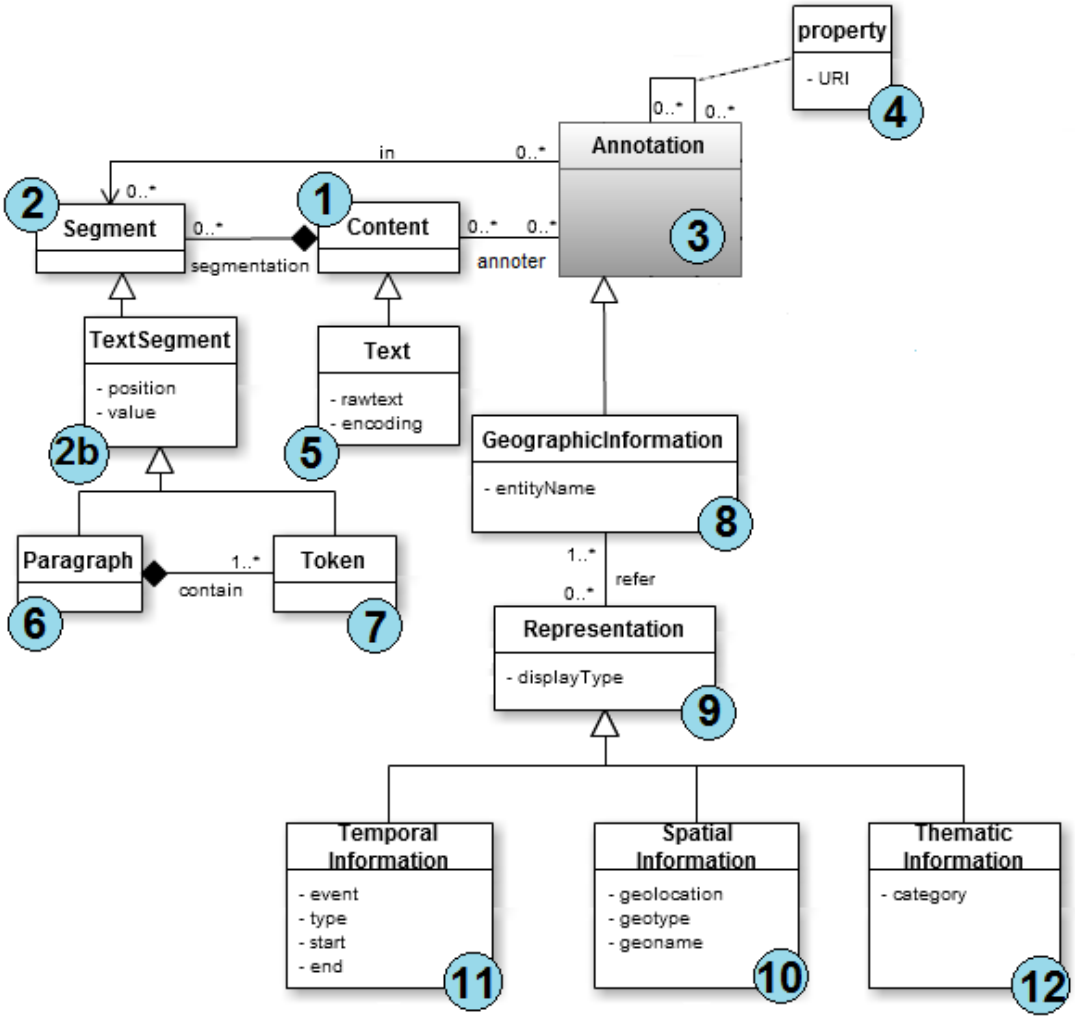


FIGURE 3.20 – Modèle conceptuel de la phase *Contenu*

3.4.2 Instanciation en RDF

Afin de coder, de regrouper, de partager et de réutiliser les descriptions de contenus de cette phase, nous proposons une sérialisation RDF / XML. Grâce à ce standard de description de ressources sur le Web, chaque phase du processus de conception peut être décrite de façon indépendante et peut se référer aux descriptions des autres phases. En outre, des requêtes sémantiques peuvent être exécutées sur ces descriptions RDF.

```
<rdf:RDF>
<rdf:Description rdf:about="&ex;text.txt">
  <wind:annotation>
    <wind:GeographicInformation rdf:about="&ex;data.rdf#
        Annotation1">
      <wind:entityName>Mauléon-Licharre</wind:entityName>
      <wind:in>
        <rdf:Bag>
          <rdf:li>
            <rdf:Description>
              <wind:start rdf:resource="&ex;text.txt#Par1-
                  Token10"/>
              <wind:end rdf:resource="&ex;text.txt#Par1-Token10"
                  />
            </rdf:Description>
          </rdf:li>
        </rdf:Bag>
      </wind:in>
      <wind:refer>
        <rdf:Bag>
          <rdf:li>
            <wind:SpatialInformation rdf:about="&geostream#
                MAULEON">
              <wind:geoname>MAULEON-LICHARRE</wind:geoname>
              <wind:geolocation>MULTIPOLYGON((...))</
                  wind:geolocation>
              <wind:geotype rdf:resource="&geotopia#Town"/>
            </wind:SpatialInformation>
          </rdf:li>
          <rdf:li>
            <wind:TemporalInformation rdf:about="&tempostream#
                ET1">
              <wind:start>2011-07-14</wind:start>
              <wind:end>2011-07-14</wind:end>
              <wind:event>parti pour Mauléon-Licharre</
                  wind:event>
              <wind:type rdf:resource="&wind#Instance"/>
            </wind:TemporalInformation>
          </rdf:li>
          <rdf:li>
            <wind:ThematicInformation rdf:about="&theme#Th1">
              <wind:category>Voyage</wind:category>
            </wind:ThematicInformation>
          </rdf:li>
        </rdf:Bag>
      </wind:refer>
    </wind:GeographicInformation>
        ...
  </wm:annotation>
 </rdf:Description>
</rdf:RDF>
```


FIGURE 3.21 – Extrait RDF/XML d'une instance du modèle *Contenu*

La figure 3.21 présente un exemple RDF/XML correspondant au modèle de contenu décrit dans la figure 3.20. Cette description indique que le fichier `text.txt` [34] contient une annotation (`Annotation1`) au sujet d'une entité nommée `Mauléon-Licharre` (lignes 4 - 5). Les objets des propriétés `wind:start` et `wind:end` sont des ressources définies dans un autre fichier RDF/XML (Figure 3.22). Cela permet de trouver que l'entité géographique est située dans le premier paragraphe du texte au 10e token (lignes 8 - 11 dans la figure 3.22). Pour une annotation, nous pouvons avoir plusieurs représentations (les objets de la propriété `wind:refer`) qui sont une entité spatiale (`wind:SpatialInformation`), une entité temporelle (`wind:TemporalInformation`) ou une entité thématique (`wind:Thematic-Information`). Dans l'exemple de la figure 3.21, l'annotation `Annotation1` possède trois représentations (lignes 16 - 39 dans la figure 3.21) :

- une information spatiale qui a un geotype (`Town`) et une géolocalisation (`MULTIPOLYGON((...))`). Cette géolocalisation est représentée par une chaîne de coordonnées géographiques au format WKT [35] qui est un format standard en mode texte utilisé pour représenter des objets géométriques vectoriels issus des systèmes d'informations géographiques (SIG) défini par l'OGC [36] ;
- une information temporelle concernant le départ pour Mauléon-Licharre le 14 juillet 2011 ;
- une information thématique qui fait partie de la catégorie “Voyage”.

```
1  <wind:Paragraph rdf:about="&ex;text.txt#Par1">
2     <wind:position>1</wind:position>
3     <wind:value>Durant l'été, je suis parti pour Mauléon-Licharre le
          14 juillet 2011. Je suis rentré à Bayonne deux jours
          après.</wind:value>
4     <wind:contain>
5        <rdf:Seq>
6           ...
7           <rdf:li>
8              <wind:Token rdf:about="&ex;text.txt#Par1-Token10">
9                 <wind:position>10</wind:position>
10                <wind:value>Mauléon-Licharre</wind:value>
11             </wind:Token>
12          </rdf:li>
13          ...
14       </rdf:Seq>
15    </wind:contain>
16 </wind:Paragraph>
```

FIGURE 3.22 – Extrait RDF/XML décrivant la tokenisation du texte

34. Contenu du fichier `text.txt` : “Durant l'été 2011, je suis parti pour Mauléon-Licharre le 14 juillet 2011. Je suis rentré à Bayonne deux jours après.”
35. Well-known text
36. Open Geospatial Consortium

Naturellement, la description de la figure 3.21 pourrait être améliorée avec des informations supplémentaires. Par exemple, nous pouvons associer cette description RDF à une autre description RDF qui exprime que Pau, qui est une autre entité spatiale, est la "préfecture"[37] de Mauléon-Licharre. Comme Pau n'est pas une annotation contenue dans le fichier `text.txt`, cette entité ne sera pas reliée à un segment spécifique. Par ailleurs, la description de la figure 3.21 peut décrire d'autres informations géographiques contenues dans le texte, comme Bayonne.

3.5 Contribution : Phase *Interface*

Cette phase permet au concepteur d'organiser l'interface de l'application générée (taille, position, fournisseur de la carte, le niveau de zoom...). Le concepteur peut définir le rendu visuel de l'application géographique finale composée des différents afficheurs adaptés aux contenus géographiques manipulés. Une interface se compose d'afficheurs qui affichent les informations (contenus géographiques) de la phase *Contenu* et le concepteur peut décider où et comment chaque afficheur est présenté à l'écran.

Cette section a pour but de présenter le modèle élaboré pour présenter les contenus dans l'interface graphique de l'application finale [LEM11]. L'interface de l'application est vue comme une couche de visualisation permettant de présenter des contenus à l'utilisateur sous diverses formes. Une interface est construite par l'assemblage de plusieurs composants d'interface, chaque composant est spécialisé pour présenter des contenus sous une forme particulière.

Bien que la phase *Interface* ne soit pas une contribution forte en terme de recherche, elle est nécessaire pour prendre en compte les éléments de la phase *Contenu* jusqu'à l'exécutabilité.

3.5.1 Modèle conceptuel

Une application Web géographique contient une interface utilisateur graphique (`GUI`) ①. Celle-ci peut être composée de plusieurs composants d'interface (`GUIComponent`) ② organisés dans la mise en page de l'application. Actuellement, nous considérons quatre types de composant d'interface : texte (`TextComponent`) ③, carte (`MapComponent`) ④, frise chronologique (`TimelineComponent`) ⑤ et liste (`ListComponent`) ⑥ qui sont les sous-classes de `GUIComponent` ② dans la figure 3.23. Chaque composant d'interface prend en charge ses propres paramètres de configuration et d'affichage. Une annotation ⑦ peut apparaître dans un ou plusieurs composants d'interface ② qui peuvent eux-mêmes contenir une ou plusieurs annotations ⑦ et afficher les représentations ⑧ de ces annotations qui sont compatibles avec le type de composant d'interface :

– Pour un `TextComponent`, la représentation du contenu est textuelle sous forme de chaîne de caractères ou encore sa (ou ses) position(s) dans le texte. Les élé-

37. "préfecture" qui est définie dans l'ontologie, est une propriété entre deux annotations.

ments textuels sont automatiquement marqués par les services Web qui extraient automatiquement les entités nommées.
– Pour un `MapComponent`, les entités géographiques sont marquées comme des géométries sur la carte. Un point représente un endroit, un lieu... Une ligne représente une route, une rivière ou un itinéraire par exemple. Un polygone représente une région, une ville... Le concepteur peut choisir les fonds cartographiques pour son application. Nous mettons en évidence la puissance de la prise en charge multi-couches de `MapComponent`, qui peut supporter plusieurs couches de différents fournisseurs.
– Le `TimelineComponent` affiche les informations temporelles (par exemple, date, période) dans une frise chronologique.
– Le `ListComponent` affiche les annotations sous forme de liste à puces.

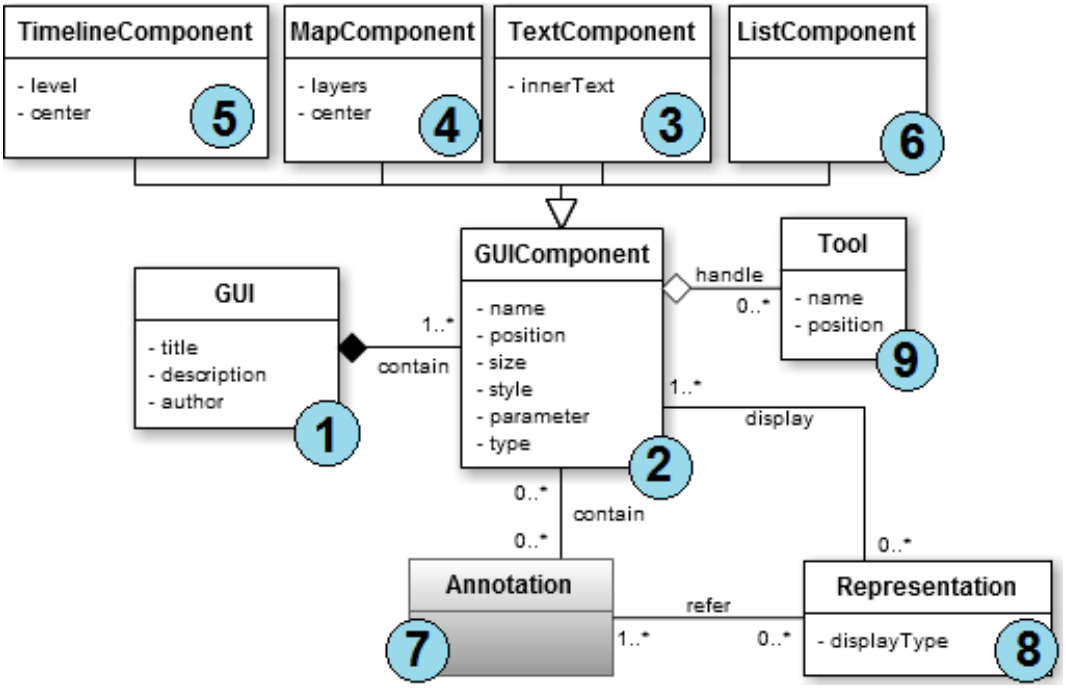


FIGURE 3.23 – Modèle conceptuel de la phase *Interface*

Chaque composant d'interface peut posséder un ou plusieurs outils (`Tool`) ⑨ permettant à l'utilisateur de créer une nouvelle annotation lors de la manipulation de ce composant d'interface dans l'application finale. Par exemple, un outil peut correspondre à un bouton d'annotation manuelle dans un composant textuel. Un outil pourrait aussi correspondre à un bouton pour dessiner des lignes, des polygones... sur un composant cartographique, etc.

Lorsqu'une annotation est projetée sur un composant d'interface et que cette annotation peut avoir plusieurs représentations possibles sur ce même composant, le concepteur doit préciser les représentations à utiliser pour représenter l'annotation sur ce composant d'interface. Par exemple, une annotation concernant "Bayonne" peut avoir trois représentations :

- une représentation textuelle au 5e token dans le premier paragraphe d'un texte donné [38] ;
- une représentation cartographique qui est un point dont la longitude est -1.475 et la latitude est 43.4936, sa géolocation est donc `POINT(-1.475 43.4936)` ;
- une autre représentation cartographique dont la géolocation est sous forme d'un polygone : `MULTIPOLYGON(((-1.49222966988623 43.5088419085818,-1.49223686780896 43.5090127768587,...,-1.49222966988623 43.5088419085818)))`. Cette représentation cartographique est plus intéressante car elle couvre la représentation ponctuelle.

3.5.2 Instanciation en RDF

La figure 3.24 présente un exemple RDF/XML correspondant au modèle d'interface graphique décrite dans la figure 3.23. Cette description montre que l'interface graphique contient un composant cartographique avec une position spécifiée par ses attributs `top, left, width, height` (lignes 10 - 15). Ce composant d'interface affiche une de deux couches cartographique de Google (*Street* ou *Satellite*) (lignes 19 - 24) et se centre aux coordonnées (2,45) à niveau de zoom 6 (ligne 26). Il est autorisé de faire un zoom ou un déplacement sur la carte (lignes 28 - 30).

Le composant d'interface contient une annotation (`Annotation1`) (ligne 33). Comme la fusion des instances du modèle est faite facilement en utilisant les propriétés du formalisme RDF, à partir des exemples RDF/XML de la figure 3.21 et de la figure 3.24, il est possible d'afficher sur la carte la géolocalisation de Mauléon-Licharre.

Notons aussi que les lignes 44 - 51 décrivent que ce composant cartographique dispose d'un outil qui est représenté par une icône (*polygonicon.png*) sur la carte et a pour but de permettre à l'utilisateur de dessiner un polygone.

38. Je suis allé à Bayonne le 15 août 2012.

```
<rdf:RDF>
<wind:GUI rdf:about="&ex;gui.rdf">
  <wind:title>Simple example</wind:title>
  <wind:author>nhan</wind:author>
  <wind:contain>
    <rdf:Bag>
      <rdf:li>
        <wind:MapComponent rdf:about="&ex;gui.rdf#Map1">
          <wind:name>Carte</wind:name>
          <wind:position>
            <rdf:Description wind:top="100" wind:left="170"/>
          </wind:position>
          <wind:size>
            <rdf:Description wind:height="300" wind:width="400"/>
          </wind:size>
          <wind:style>
            <rdf:Description wind:color="#3366CC" wind:border="#3366CC 2px
                solid" wind:icon="mvizicon.png"/>
          </wind:style>
          <wind:layers>
            <rdf:Seq>
              <rdf:li>Google Street</rdf:li>
              <rdf:li>Google Satellite</rdf:li>
            </rdf:Seq>
          </wind:layers>
          <wind:center>
            <rdf:Description wind:longitude="2" wind:latitude="45"
                wind:zoom="6"/>
          </wind:center>
          <wind:parameter>
            <rdf:Description wind:zoomable="yes" wind:pannable="yes"/>
          </wind:parameter>
          <wind:annotation>
            <rdf:Bag>
              <rdf:li rdf:resource="&ex;data.rdf#Annotation1"/>
            </rdf:Bag>
          </wind:annotation>
          <wind:display>
            <rdf:Bag>
              <rdf:li rdf:resource="&geostream#MAULEON"/>
            </rdf:Bag>
          </wind:display>
          <wind:handles>
            <rdf:Seq>
              <rdf:li>
                <wind:Tool>
                  <wind:name>polygon</wind:name>
                  <wind:position>top</wind:position>
                  <wind:image>polygonicon.png</wind:image>
                  <wind:allows>
                    <wind:PolygonDrawing
                        rdf:about="ex;gui.rdf#Map1-event1"/>
                  </wind:allows>
                </wind:Tool>
              </rdf:li>
            </rdf:Seq>
          </wind:handles>
        </wind:MapComponent>
      </rdf:li>
      ...
    </rdf:Bag>
  </wind:contain>
</wind:GUI>
</rdf:RDF>
```


FIGURE 3.24 – Extrait RDF/XML instanciant le modèle *Interface*

3.6 Exécutabilité des phases *Contenu* et *Interface*

Pour assurer l'exécutabilité des deux modèles de *Contenu* et d'*Interface* (présentés dans les sections 3.4 et 3.5), nous avons implémenté l'API WIND ("*Web INteraction Design*") et l'environnement WINDMash que nous présentons dans les sections suivantes.

3.6.1 API WIND

Les API sont en train de révolutionner la distribution d'informations et de services. À la base, une API n'est rien d'autre qu'un morceau de code permettant d'accéder à certaines fonctionnalités d'une application pour les intégrer dans une autre application, ce qui évite de les redévelopper. Le double phénomène de l'édition et de l'utilisation d'API (de contenu ou de services) s'est étendu à de nombreux secteurs, contribuant au développement d'une myriade de services et d'usages totalement inimaginables il y a quelques années.

Plusieurs travaux de recherche ont récemment approuvé l'efficacité des applications Web géographiques dans différents domaines, tels que tourisme, éducation ou encore surveillance. La plupart des services Web géographiques actuels fournissent une API permettant aux programmeurs d'ajouter ses propres informations. Les API principales de cartographie sont Google Maps, OpenLayers... Toutefois, les développements exigent des compétences avancées en informatique pour exploiter ces API. Par exemple, pour coder une application Web affichant une carte Google Maps, il faut que le développeur maîtrise bien les fonctionnalités dans l'API de Google Maps.

Dans cette section, nous présentons une bibliothèque JavaScript nommée WIND permettant de décrire les contenus géographiques intégrés dans les composants visuels de l'application. Les composants peuvent être textuels, cartographiques, temporels... Cette API facilite la programmation des applications Web géographiques.

3.6.1.1 Caractéristiques de l'API WIND

Les langages de programmation utilisés pour les outils cartographiques jouent un rôle important. JavaScript est un langage de programmation pour l'interaction des objets HTML et CSS dans une page Web, il permet aux programmeurs de créer de nouveaux éléments HTML, de les modifier, de les supprimer, de les associer avec des attributs CSS et de manipuler les événements sans rafraîchir l'ensemble de la page Web. Profitant de JavaScript, WIND est une API permettant aux programmeurs d'implémenter chaque interaction décrite au niveau conceptuel. Cette API propose une couche homogène construite sur les API de niveau inférieur, spécialisée dans la manipulation des éléments textuels, cartographiques et calendaires (Figure 3.25). Ainsi, les fonctions WIND permettent aux programmeurs de gérer les éléments cartographiques et leurs interactions associées à l'aide des API open-source, telles que OpenLayers ou des API propriétaires, telles que l'IGN Geoportail. Les programmeurs, toutefois, ne gèrent que les fonctions de

l'API WIND et n'ont pas besoin de connaître les caractéristiques techniques de l'API de niveau inférieur.

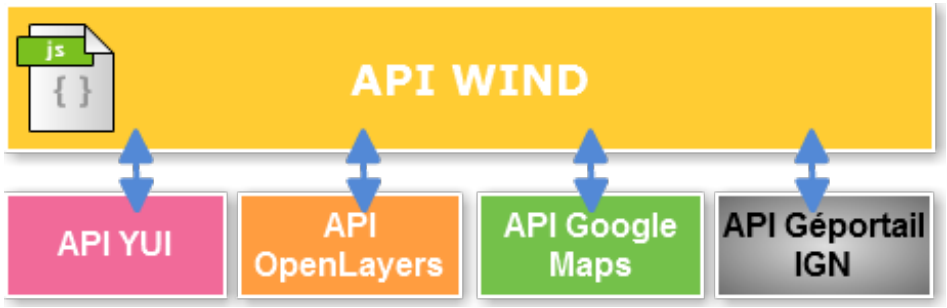


FIGURE 3.25 – Architecture de l'API WIND

L'intérêt de l'API WIND est de permettre aux programmeurs de développer les applications avec quelques lignes de code en utilisant les fonctionnalités spécifiques définies dans l'API.

Les principales caractéristiques, qui seront illustrées dans la section suivante, de l'API WIND sont :

- WIND combine les composants textuels, cartographiques et calendaires ; il associe également des services Web de cartographie.
- WIND est entièrement exécutable (grâce à l'API JavaScript de WIND).
- WIND favorise la programmation légère car peu d'instructions permettent de programmer des comportements évolués.
- WIND est orientée objet : l'implémentation du code des applications basées sur l'API WIND est homogène quel que soit le type de ses composants visuels.
- WIND respecte une approche déclarative permettant aux utilisateurs de concevoir des interactions entre les composants textuels, cartographiques et calendaires (Section 4.4).

3.6.1.2 Principes de codage d'une application Web avec l'API WIND

L'application présentée sur la figure 3.26 illustre les activités d'un extrait d'un récit de voyage. L'interface de l'application se compose de trois composants d'interface : texte, carte et frise chronologique. Le composant textuel affiche un extrait du récit de voyage. La carte montre les lieux concernés par les activités citées dans le récit de voyage, tandis que la frise chronologique présente les moments où les activités ont lieu.

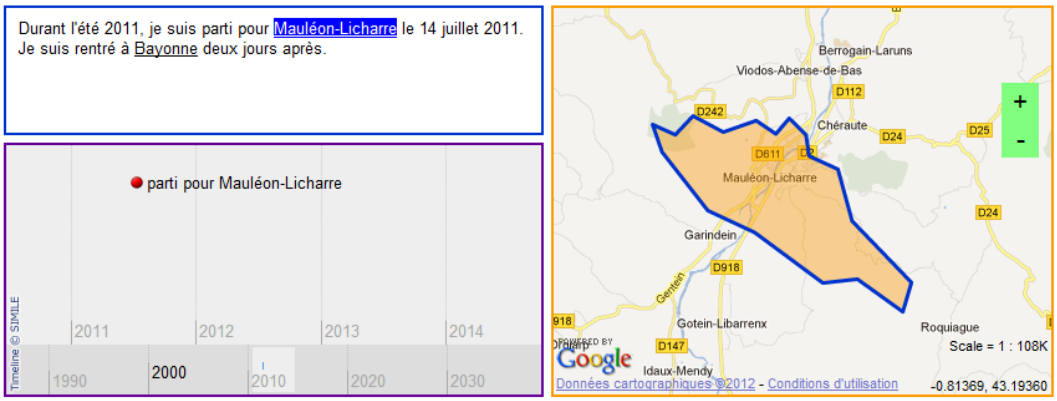


FIGURE 3.26 – Exemple d'association de trois composants d'interface (texte, carte, frise)

L'application présentée ci-dessus peut être codée comme suit. D'abord, afin d'utiliser l'API WIND, il faut l'importer ainsi que les autres API dont elle dépend au sein de la balise `<head>` dans le code HTML (Figure 3.27).

```
1 <script type="text/javascript"
      src="http://erozate.iutbayonne.univ-pau.fr/Nhan/windapi/
2         lib/openlayers/OpenLayers.js"></script>
3 <script type="text/javascript"
      src="http://api.ign.fr/geoportail/api?v=1.1-m&amp;key=***
4         &amp;includeEngine=true&amp;"></script>
5 <script type="text/javascript"
      src="http://maps.google.com/maps?file=api&amp;v=2.124&amp;
6         key=***"></script>
7 <script type="text/javascript"
      src="http://erozate.iutbayonne.univ-pau.fr/Nhan/windapi/
8         WINDv2.0.js"></script>
```

FIGURE 3.27 – Importation des librairies JavaScript

Étape 1 : Création de composants d'interface de l'application (Figure 3.28). Nous créons tout d'abord une interface de l'application en utilisant le constructeur de la classe `WIND.GUI`. Ensuite, nous créons successivement les composants d'interface (`afficheurs`) en utilisant la méthode `createDisplayer` de la classe `WIND.GUI` qui donne les objets `WIND.Text`, `WIND.Map` et `WIND.Timeline`. Pour chaque afficheur, nous mettons des paramètres correspondants dans la méthode. Nous pouvons ajouter un texte brut dans l'afficheur textuel en utilisant les méthodes `createParagraph` de la classe `WIND.Text` et `setContent` de la classe `WIND.Paragraph`.

```
1  // Création de l'interface de l'application
2  var mydoc = new WIND.GUI("main", {"title": "Mon Application",
       "description": "..."});
3
4  // Création du composant Texte
5  var t = mydoc.createDisplayer('text', {'top': 100, 'left': 10, 'width':
       500, 'height': 200, 'draggable': false, 'resizable' : false, 'color':
       '#0033CC', 'border': '#0033CC 2px solid', 'header': false, 'removable':
       false, 'configurable': false});
6  var p = t.createParagraph();
7  p.setContent('Durant l'été 2011, je suis parti pour Mauléon-Licharre le 14
       juillet 2011. Je suis rentré à Bayonne deux jours après.');
8
9  // Création du composant Carte
10 var m = mydoc.createDisplayer('map', {'top': 100, 'left': 520, 'width':
       600, 'height': 450, 'name': "toto", 'type': 'Google Street',
       'longitude': -0.32, 'latitude': 43.45, 'zoom': 8, 'draggable': false,
       'resizable' : false, 'color': '#FF9900', 'border': '#FF9900 2px solid',
       'header': false, 'removable': false, 'configurable': false});
11
12 // Création du composant Frise chronologique
13 var tl = mydoc.createDisplayer('timeline', {top:310, left:10, width:500,
       height:240, color: "#FF6600", border: "#FF6600 1px solid", 'header':
       false, 'removable': false,'configurable': false});
```

FIGURE 3.28 – Création de composants d'interface de l'application

Les 5e, 10e et 13e lignes de code dans la figure 3.28 ont pour but de créer les afficheurs textuel, cartographique et frise chronologique. Leur position (`top` et `left`) sur écran, leur taille (`width` et `height`) en pixel et leurs autres paramètres (couleur, bordure...) sont définis dans la méthode de création d'afficheur.

Étape 2 : Création des annotations (Figure 3.29). Dans cette application, le développeur veut présenter les annotations au sein des afficheurs pour mettre en valeur un contenu géographique. Concrètement, il veut souligner les noms de lieu dans l'afficheur textuel (i.e., Mauléon-Licharre et Bayonne) ainsi qu'afficher ces lieux sur la carte. Il veut également marquer l'événement "*parti pour Mauléon-Licharre*" à la date du 14 juillet 2011 sur la frise chronologique.

La 2e ligne de code dans la figure 3.29 a pour but de créer une annotation ayant la sémantique "*Town*" (ville) sur l'entité "Mauléon-Licharre" qui se trouve au 10e mot du 1er paragraphe du texte. La 5e ligne de code dans la figure 3.29 a pour but de créer une annotation sur la carte ayant la sémantique "*Town*" (ville) avec l'entité "Mauléon-Licharre" qui est présentée sous une forme géométrique "MULTIPOLYGON" sur la carte. La 8e ligne de code dans la figure 3.29 a pour but de créer une annotation temporelle sur la frise pour l'événement "*parti pour Mauléon-Licharre*" à la date du 14 juillet 2011.

```
1 // Création des annotations
2 var annot1 = t.createAnnotation("Town", "Mauléon-Licharre", 1, 10, 10);
3 var annot2 = t.createAnnotation("Town", "Bayonne", 1, 20, 20);
4
5 var annot3 = m.createAnnotation("Town", "Mauléon-Licharre",
      MULTIPOLYGON(((...))));
6 var annot4 = m.createAnnotation("Town", "Bayonne", MULTIPOLYGON(((...))));
7
8 var annot5 = tl.createAnnotation("Voyage", "parti pour Mauléon-Licharre",
      "14/07/2011", "14/07/2011");
```

FIGURE 3.29 – Affichage des annotations dans les composants visuels

3.6.2 WINDMash

Nous présentons ici un exemple de manipulation de notre environnement WINDMash (*cf.* Annexe F pour plus de détails) pour créer une application Web géographique.

Nous reprenons l'exemple qui a été codé par l'API WIND dans la section 3.6.1.2, mais nous allons concevoir cette application sans codage manuel en utilisant WINDMash. En effet, elle contient le texte en français, une carte contenant les annotations spatiales et une frise chronologique.

Comme illustré dans la figure 5.5, l'architecture WINDMash est composée de trois phases qui concernent, respectivement, la gestion des contenus et des annotations, l'organisation de l'interface graphique et la spécification des interactions d'utilisateur. Par conséquent, trois outils ont été mis en œuvre dans notre prototype WINDMash. Plus précisément, ces outils sont les suivants :

1. Un éditeur de flux qui permet de combiner différents services et de filtrer les données géographiques manipulées par l'application (Section 3.6.2.1);
2. Un éditeur de présentation graphique qui est utilisé pour organiser, par exemple, des composants cartographiques ou des contenus multimédias (Section 3.6.2.2);
3. Un constructeur inspiré par le diagramme de séquence UML qui permet de spécifier le comportement des interactions sur l'application (Section 4.4.2).

3.6.2.1 Phase *Contenu*

Afin de gérer les données (c'est à dire, le contenu et les annotations) qui doivent être manipulées par l'application Web géographique, nous avons développé un éditeur de flux de données. Cet outil a été inspiré de l'éditeur Yahoo! Pipes [39] et permet au concepteur de créer une chaîne de traitement contenant différents services (Figure 3.30).

39. http://pipes.yahoo.com

À partir d'un ou de plusieurs textes bruts, le concepteur peut facilement créer une chaîne de traitement en sélectionnant des modules dédiés. Cette chaîne de traitement peut transformer automatiquement un texte en entrée en un document structuré selon le modèle de *Contenu* qui peut être visualisé ultérieurement (phase *Interface*) par les afficheurs (`Displayer`) dédiés : textuel, cartographique et calendaire. Les modules disponibles peuvent être paramétrés par le concepteur pour atteindre un objectif spécifique. Nous divisons deux groupes de modules : **Conteneurs de données** et **Services**.

- Un conteneur de données peut être soit un texte brut, soit une base de données IGN, soit des données Web à connotation géographique (*LinkedGeoData*). Un texte brut est un document textuel (par exemple, le récit de voyage) utilisé par le concepteur et est relié à un module `Service`. Une base de données IGN comprend des tables d'informations géographiques (communes, départements, montagnes, cours d'eau...). Le module de données *LinkedGeoData* permet d'exploiter des données Web géographiques sur le Web (`http://linkedgeodata.org`).
- Les modules services (e.g., Annotation manuelle, Extraction lieu, Extraction temps, Service SPARQL...) permettent de traiter un conteneur de données et produire des contenus générés (données structurés selon le modèle *Contenu* présenté dans la section précédente). Un module *Extraction lieu* implémente le service Web GeoStream [SRL+09] pour extraire les informations géographiques dans les documents textuels et un module *Extraction temps* implémente le service Web TempoStream [LNG09] pour extraire les informations temporelles dans les documents textuels. Un module *Annotation manuelle* permet à l'utilisateur d'annoter par lui-même les entités sur un texte selon trois facettes : spatiale, temporelle et thématique. Le *service SPARQL* est utilisé avec les données Web (*LinkedGeoData*) pour extraire quelques entités géographiques en utilisant une requête SPARQL sur des descriptions RDF en ligne.

Les modules fondamentaux de WINDMash sont prédéfinis dans la plateforme. Ils sont décrits au format JSON [40] pour faciliter l'implémentation de WINDMash qui est programmée principalement en JavaScript. Ils sont également décrits sous la forme de texte de description (compréhensible pour les concepteurs) lorsque ils effectuent un survol sur les modules.

La figure 3.30 illustre une chaîne de traitement possible qui correspond au cas d'utilisation présenté. En fait, à partir d'un texte donné Ⓐ, nous tenons à extraire automatiquement les lieux Ⓑ (en occurrence des "villes") et les références temporelles Ⓒ (nommées "temps"). Pour cet objectif, nous invoquons le service Web `Extraction lieu` qui a été détaillé dans [SRL+09] et le service Web `Extraction temps`. De plus, le service Web `Extraction lieu` peut identifier et marquer les types des entités extraites ; dans ce cas les entités *Mauléon-Licharre* et *Bayonne* sont identifiées comme des villes (`Town`).

40. JavaScript Object Notation

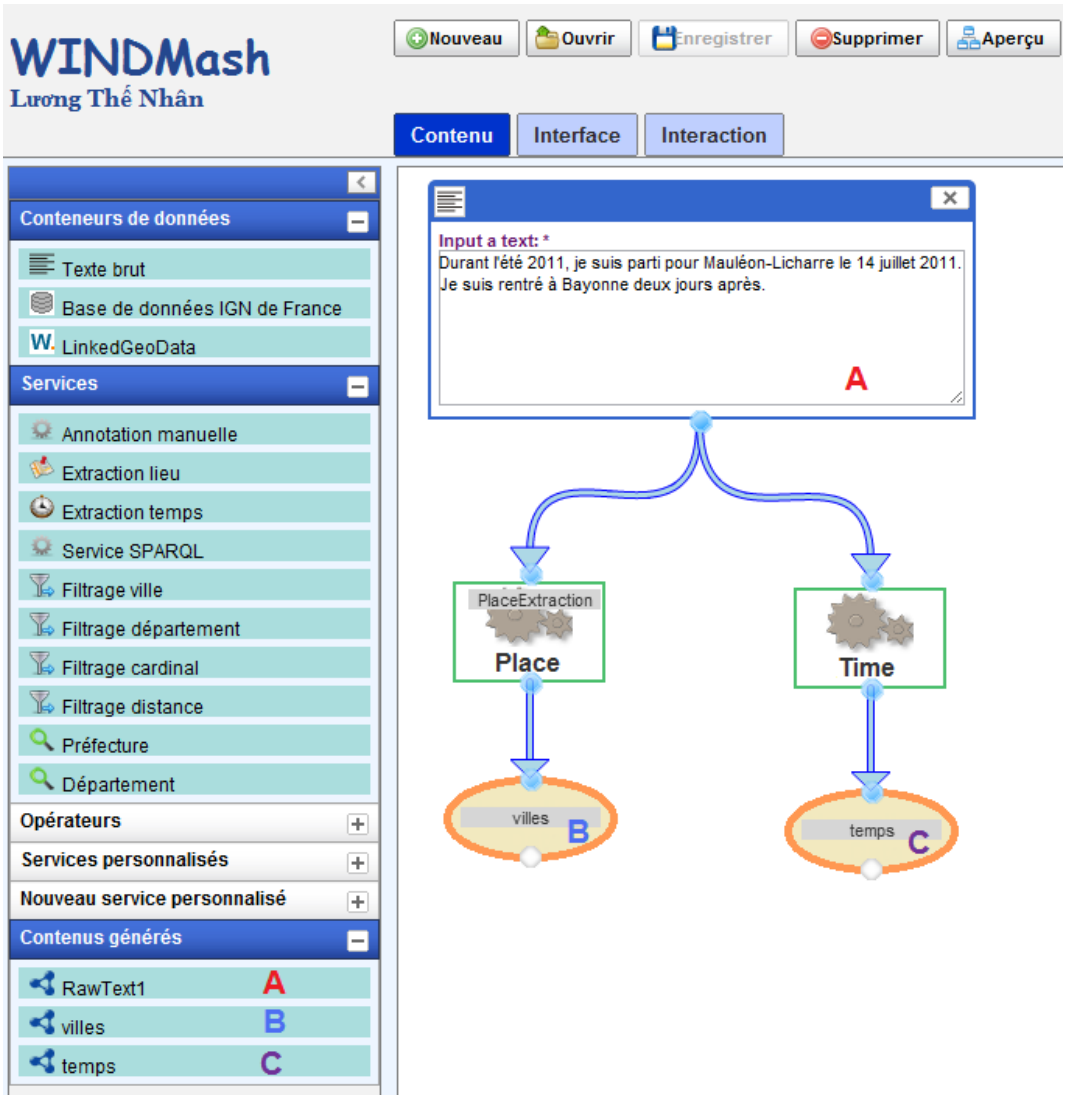
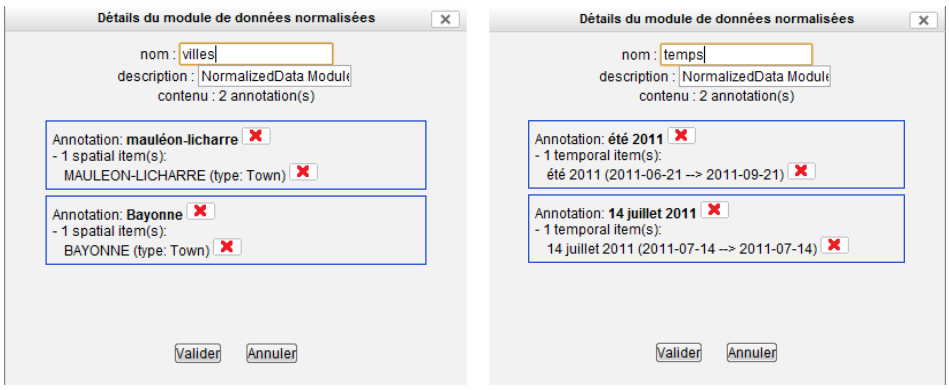


FIGURE 3.30 – Éditeur de flux de WINDMash (Phase *Contenu*)

Une fois la construction des flux réalisée, il est possible de visualiser à tout moment les données calculées en sélectionnant avec un double clic, les éléments “villes” Ⓑ (Figure 3.31(a)) et “temps” Ⓒ (Figure 3.31(b)).



(a) (b)

FIGURE 3.31 – Visualisation des contenus générés

Chaque fois qu’un ensemble de données est calculé, tel que la liste des lieux extraits Ⓑ, WINDMash génère une description RDF/XML qui correspond à la phase de *Contenu* présentée dans la figure 3.21. Ces descriptions sont également accessibles en bas à gauche du prototype WINDMash dans la zone des *Contenus générés* (Figure 3.30).

Nous montrons dans le paragraphe suivant, que ces descriptions peuvent être utilisées dans notre éditeur de présentation graphique pour afficher les données à l’intérieur des afficheurs.

3.6.2.2 Phase *Interface*

Notre éditeur de présentation graphique permet à un concepteur de spécifier l’interface utilisateur graphique de son application Web géographique. En effet, le concepteur décide quel type d’afficheur il souhaite dans son application (par exemple, `Afficheur texte`, `Afficheur carte`, `Afficheur liste`, `Afficheur frise`) et comment ces afficheurs sont organisés à l’intérieur de la présentation graphique (taille et position).

La figure 3.32 illustre comment un concepteur peut préciser, grâce à notre outil, la présentation graphique de son application. Le menu à gauche indique le type des afficheurs qui peuvent être manipulés par le concepteur, l’ensemble des données disponibles qui ont été calculées avec notre éditeur de flux (voir section 3.6.2.1) et les afficheurs qui sont actuellement utilisés. Ici aussi, les différents afficheurs et contenus générés (issus de la phase *Contenu*) sont placés dans la zone de travail par glisser-déposer du concepteur.

Ce dernier choisit la taille et la position de chaque afficheur qu'il peut également paramétrer (*cf.* Figure 3.33) en précisant notamment le nom qui s'affichera aussi dans la zone des *Afficheurs générés* (et sera utilisé ultérieurement dans la phase *Interation* - *cf.* Section 4.4.2). Dans la figure 3.32, trois afficheurs ont été spécifiés : un afficheur textuel ①, un afficheur cartographique ② et un afficheur de frise chronologique ③.

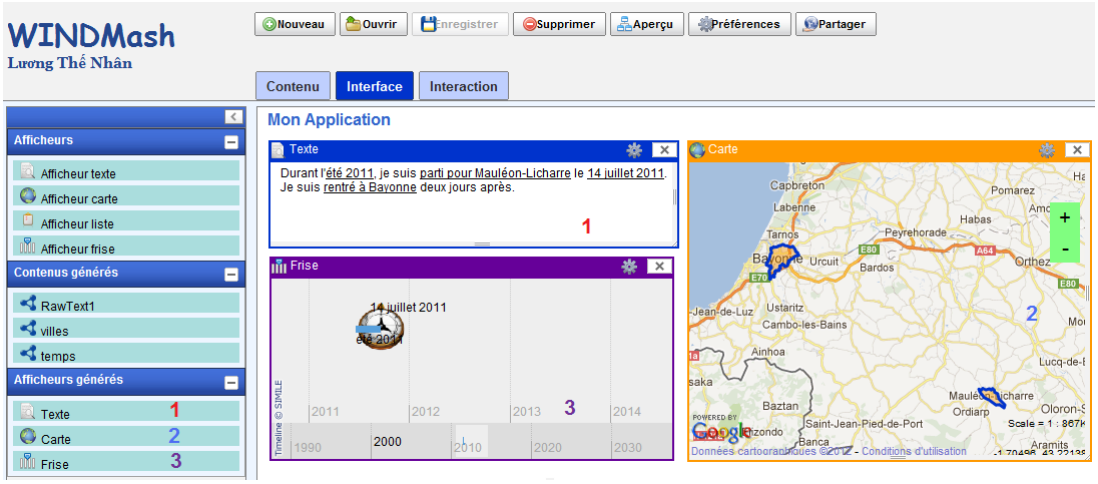


FIGURE 3.32 – Éditeur de la mise en page graphique de WINDMash (Phase *Interface*)

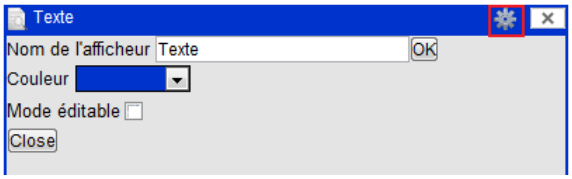


FIGURE 3.33 – Configuration de l'afficheur-Texte

Initialement, lorsque le concepteur glisse et dépose un afficheur à l'intérieur de la zone de travail, cet afficheur est vide, à l'exception de l'afficheur cartographique qui contient une carte. Si le concepteur souhaite afficher quelques informations à l'intérieur des afficheurs, à partir du menu, il doit glisser les données calculées et les déposer dans un afficheur spécifique. Par exemple, si le concepteur veut voir à l'intérieur de l'afficheur textuel ① le texte qui a été écrit dans la figure 3.30, il doit glisser l'élément `RawText1` du menu et déposer cet élément à l'intérieur de l'afficheur `Texte`. Par la suite, si le concepteur veut souligner les lieux et les temps qui ont été extraits de ce texte, il doit glisser les éléments `villes` et `temps` du menu et le déposer dans l'afficheur `Texte`.

De la même manière avec les autres types d'afficheurs, si le concepteur veut voir les lieux sur la carte, il doit glisser l'élément `villes` et le déposer dans l'afficheur `Carte`. Enfin, si le concepteur veut voir les temps sur la frise chronologique, il doit glisser l'élément `temps` et le déposer à l'intérieur de l'afficheur `Frise`.

Jusqu'à cette étape, le concepteur a utilisé WINDMash pour générer automatiquement deux instances de deux modèles de *Contenu* et d'*Interface*. Il peut prévisualiser l'application Web géographique (statique) en cliquant sur le bouton `Aperçu` dans la barre de menu (en haut) de notre prototype. Il peut également sauvegarder et modifier l'application quand il le souhaite. L'application générée est dite "statique" car elle ne contient aucun comportement dynamique sensible aux actions de l'utilisateur, telles que le clic ou le survol. Toutefois, nous allons prendre en compte cet aspect dynamique dans le chapitre suivant (section 4.4.2).

3.7 Bilan, limites

Dans ce chapitre, nous avons présenté deux modèles *Contenu* et *Interface* afin de modéliser, de confronter, de gérer et d'afficher des informations à connotation géographique dans les afficheurs au sein d'une application Web. Actuellement, le modèle de *Contenu* ne traite que des contenus textuels. Cependant, il est suffisamment générique pour être étendu afin de traiter des contenus multimédias, tels que des vidéos, audios et images. Le modèle d'*Interface* peut être également étendu pour afficher ces contenus multimédias.

Bien entendu, nous aurions pu nous concentrer sur l'utilisation de certains formats de description de méta-données focalisés sur des contenus textuels, comme SpatialML (`http://sourceforge.net/projects/spatialml/`) ou TimeML (`http://www.timeml.org/site/index.html`). Ces deux formats permettent effectivement de décrire des informations spatiales et temporelles contenues dans des textes. Néanmoins, nous souhaitions pouvoir intégrer différents types de contenus, tels que des images, des vidéos ou encore des contenus sonores.

Dans ce contexte, nous aurions pu aussi exploiter de multiples formats de description de contenus multimédias. Cependant, ce choix aurait complexifié la gestion et la confrontation des informations géographiques. En effet, ces formats offrent différentes structures et ne se focalisent pas forcément sur les trois dimensions de l'information géographique.

Notre proposition de modèle générique de contenu permet donc d'intégrer et de faciliter la gestion des descriptions de contenus multimédias selon les trois facettes : temps, espace et thème. De plus, l'utilisation de ce modèle avec le support des technologies issues de la communauté du Web Sémantique, comme RDF/XML, nous permet de facilement agréger les informations mais aussi d'effectuer des requêtes sémantiques grâce au langage SPARQL (ce qui n'aurait pas été le cas avec de nombreux formats de méta-données tels que SpatialML et TimeML).

Nous avons implémenté l'API WIND, une bibliothèque JavaScript pour la programmation légère d'applications Web géographiques. Cette API permet aux développeurs de décrire les applications Web géographiques sans connaître les services Web de cartographie sous-jacents (par exemple, Géoportail IGN, Google Maps, OpenLayers...). L'intérêt de l'API WIND est de permettre au développeur de surmonter les complexités des API telles que Google Maps ou OpenLayers. L'API WIND a été implémentée pour un usage simple. Notons toutefois qu'elle doit évoluer à l'instar des API sur lesquelles elle se base.

Nous avons également présenté WINDMash qui est un environnement dédié à la conception d'applications Web interactives par les utilisateurs finaux. Cet environnement est actuellement disponible à cette URL (`http://erozate.iutbayonne.univ-pau.fr/Nhan/windmash/`). Notre environnement est à la fois utilisé pour la conception et l'évaluation / l'utilisation de l'application grâce à la génération automatique de code. Nous avons notamment implémenté les deux phases *Contenu* et *Interface* dans WINDMash. La phase *Contenu* peut démarrer à partir d'un document textuel géographique qui peut être traité par divers services tels que ceux d'extraction de lieux ou d'extraction d'itinéraires. La phase *Interface* permet au concepteur de définir l'affichage de l'application générée.

Pour nous situer par rapport aux différents environnements de Mashup présentés en section 3.3, un concepteur (par exemple, un enseignant) sans compétence informatique peut, à l'aide de WINDMash, décrire et générer par lui-même une application Web géographique à partir de ses besoins.

Nous allons ci-dessous (Figure 3.34) effectuer une comparaison deux à deux pour chacune de nos trois contributions présentées dans ce chapitre.

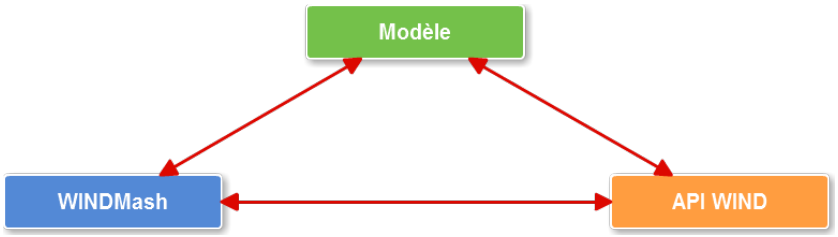


FIGURE 3.34 – Schéma des contributions du chapitre 3

Parmi ces propositions, les modèles de contenu et d'interface restent, par leur niveau d'abstraction, la contribution qui propose les concepts les plus nombreux pour gérer et afficher les données géographiques sur l'interface utilisateur. La figure 3.35 résume les concepts pris en compte dans les différentes contributions. La légende de la figure est la suivante :

- La lettre **A** signifie que le concept est intégré au niveau de l'API WIND (il existe une classe ou une méthode spécialisée pour gérer le concept) ;
- La lettre **W** signifie que le concept est pris en compte au niveau de l'environnement WINDMash (le concepteur peut manipuler graphiquement le concept); la lettre **w** signifie que le concept est en cours d'implémentation au niveau de WINDMash.

Comme présenté dans la figure 3.35, la plupart des concepts des modèles de contenu et d'interface ont été opérationnalisés au niveau de l'API WIND et de l'environnement WINDMash. Cependant, il y a quelques classes du modèle qui ne sont pas implémentées dans l'API WIND et ne sont pas manipulées dans WINDMash. Par exemple, le concept *Tool* n'est pas mis à disposition dans tous les composants d'interface dans l'API WIND et WINDMash. Nous ne l'avons implémenté que dans les composants textuel et cartographique. Les concepts `Segment` et `TextSegment` ne sont pas pris en compte dans l'API WIND et WINDMash parce que nous avons directement implémenté les concepts `Paragraph` et `Token`.

Comme abordé dans le chapitre 1, nous nous intéressons à une plate-forme permettant aux utilisateurs de concevoir les applications Web interactives. Nous avons présenté dans le chapitre 2 un processus de conception qui se compose de trois phases. Ce chapitre 3 a eu pour but de présenter deux phases *Contenu* et *Interface* permettant la conception / réalisation d'applications Web géographiques statiques. Le chapitre suivant est consacré à la phase *Interaction* dédiée à la modélisation du comportement (interactif) des applications Web géographiques (dynamiques). Les perspectives seront présentées plus en détail dans le chapitre 5.

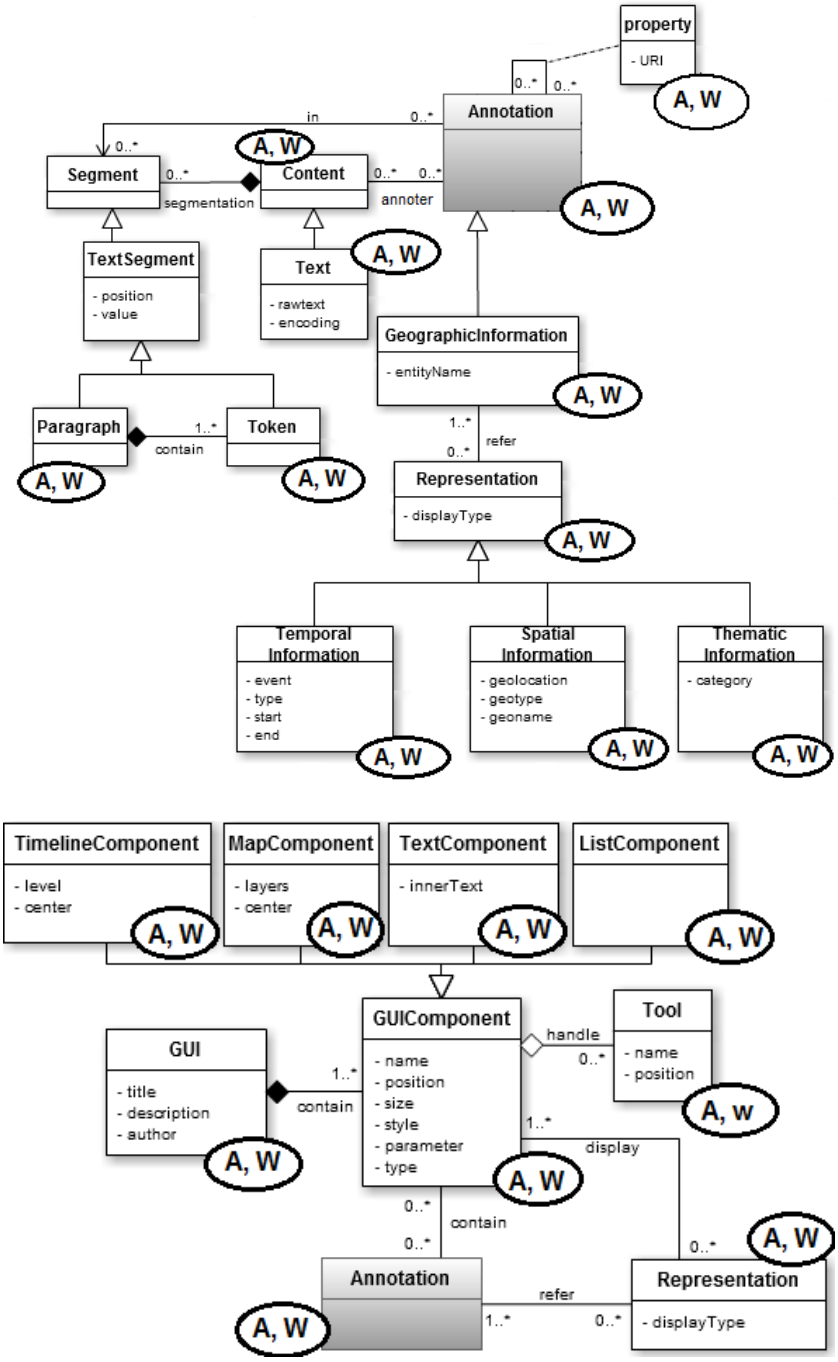


FIGURE 3.35 – Degré d'opérationnalisation des modèles de contenu et d'interface

Chapitre 4

Interactivité sur des contenus géographiques

Sommaire

4.1 Introduction

Ce chapitre concentre le processus de conception sur les dimensions interactives de l'application à concevoir. Il se focalise sur les moyens dont dispose un concepteur pour spécifier la dimension interactive d'une application. La section 4.2 présente des travaux de recherche visant à modéliser l'interaction selon différentes approches. Nous nous intéressons à la fois aux modèles dédiés à l'interaction mais aussi aux langages visuels permettant de la décrire.

A partir de cette étude et de la synthèse qui en résulte, nous proposons dans la section 4.3 un modèle d'interaction permettant de décrire les possibilités interactives d'une

application en raisonnant sur l'interface d'une application et plus particulièrement sur les contenus affichés ou saisis à l'interface. La propriété visée par ce modèle est de permettre au concepteur de décrire l'interaction en raisonnant de manière concrète à partir d'une interface, de sa composition des composants d'interface mais aussi des contenus affichés dans chacun d'eux. Pour faciliter la prise en main de ce modèle, nous proposons également un langage visuel permettant de spécifier l'interaction graphiquement sous forme de flux échangés entre l'utilisateur et le système.

Nous soulignons enfin dans la section 4.4 les capacités opérationnelles de ce modèle et de ce langage en présentant les possibilités de génération de code pour tester rapidement l'application décrite et les interactions élaborées.

4.2 État de l'art : Modèles et langages pour l'interaction

4.2.1 Interaction et conception d'interaction

Dans ce chapitre, notre travail porte principalement sur l'interaction homme-machine (IHM) et la conception d'interaction. L'exploration de la littérature relative aux IHM amène de la complexité dans la mesure où des dizaines de termes simples sont utilisés. Selon [Cur92] et [SJ07] il n'existe pas de définition convenue de ce qu'est une interaction. D'après [VR09], "*les interactions sont des comportements actifs mis en œuvre par des acteurs pour communiquer et échanger des informations*". L'échange d'informations est donc central et justifie la mise en œuvre de mécanismes d'interactions. L'interaction est donc une activité dynamique dans laquelle des échanges ont lieu entre deux entités et dont la finalité est d'obtenir de l'information.

Dans l'article de [Mar99] une interaction est définie comme "*une communication entre l'utilisateur et un ordinateur*", cette communication pouvant être *directe* ou *indirecte*. Une interaction *directe* sous-entend *un dialogue avec un feedback et un contrôle tout au long de la réalisation de la tâche*. Une interaction *indirecte* sous-entend la mise en œuvre *de tâches de fond ou de traitements par lots*. Cette définition souligne le fait qu'une interaction peut engendrer des actions du système perceptibles par l'utilisateur (via des boîtes de dialogue et de contrôle) mais aussi des actions internes au système (par exemple des calculs) dont la finalité est de répondre au besoin qui a déclenché l'interaction.

Nous gardons ces points de vue, en considérant que l'interaction est une communication entre l'utilisateur et le système. Cette communication est généralement initiée par une action de l'utilisateur qui produit une réaction du système. Cette réaction peut être directe et/ou indirecte telle que définie par [Mar99]. Ainsi, l'activité de conception d'interaction va consister à modéliser les actions que l'utilisateur sera en mesure d'effectuer sur le système ainsi que les réactions générées par celui-ci.

Il est également très difficile de définir la conception d'interaction... [RSP11] définit la conception d'interaction comme "*la conception d'interaction produit pour aider les gens dans la vie quotidienne et de travail*". Une définition plus large est donnée dans [Mog07] où la conception d'interaction est définie comme "*la conception de tout qui est à la fois numérique et interactif. Elle comprend la conception de toutes les interactions qui sont activées par la technologie numérique : ordinateurs, puces intégrées dans des produits ou dans les environnements, services ou Internet.*". Cette définition semble assez largement acceptée, car elle concerne à la fois le travail des professionnels en IHM, des informaticiens, des ingénieurs logiciels et des concepteurs.

La prise en compte de l'interaction lors du processus de conception d'un système sous-entend deux pré-requis. Il faut d'une part disposer de modèles définissant ce qu'est l'interaction ainsi que les propriétés fortes qui permettent de la décrire. D'autre part, il faut aussi des langages adaptés permettant aux concepteurs d'instancier les modèles précédents pour décrire les caractéristiques interactives des applications qu'ils souhaitent élaborer. En d'autres termes, les modèles sont nécessaires pour identifier *quoi* décrire et les langages répondent à la question du *comment* décrire.

Les sections qui suivent s'intéressent à ces deux aspects. Elles décrivent d'abord les modèles qui permettent de caractériser l'interaction (Section 4.2.2) puis les langages qui permettent de la décrire (Section 4.2.3).

4.2.2 Modèles permettant de caractériser l'interaction

Étant donné qu'il n'y a pas de définition du concept d'interaction, nous trouvons autant de modèles d'interactions que de définitions. Les travaux visant à définir un état de l'art sur la modélisation de l'interaction [Sil00,MPS04,CGGS09] soulignent néanmoins plusieurs familles de modèles permettant de décrire les différents aspects de l'interaction : les modèles de tâches, les modèles de dialogue, les modèles de présentation et les modèles architecturaux. Dans les sections qui suivent, nous listons ces différentes catégories de modèles pour cerner leur intérêt et identifier quels aspects de l'interaction chacun permet de décrire.

4.2.2.1 Modèles de tâches

La modélisation des tâches vise à identifier les actions que l'utilisateur peut effectuer sur le système final en vue d'atteindre des buts précis. L'objectif est de produire un modèle qui décrit ces différentes tâches mais également les relations qui les joignent.

Le premier résultat de l'analyse des tâches est une liste informelle de tâches que l'utilisateur doit mener pour atteindre un objectif donné [Pat99b]. Ces tâches ne sont pas uniquement "informatiques" : elles peuvent être physiques (comme sélectionner un élément dans un menu) mais également cognitives (comme synthétiser des informations présentées à l'écran).

De ce fait, la modélisation des tâches est un domaine de recherche à la croisée de l'informatique et des sciences cognitives.

Les sciences cognitives apportent une contribution forte sur les méthodes permettant d'identifier et de caractériser les tâches de l'utilisateur. Des méthodes telles que MAD [SPG89] ou GTA [VLB96] vont en ce sens et proposent des approches pour l'analyse et l'identification des tâches des utilisateurs. L'objectif de ces méthodes est de déterminer ce que font les utilisateurs, les outils qu'ils utilisent mais également les informations qu'ils doivent connaître pour manipuler la tâche [PP01].

La recherche en informatique s'intéresse quant à elle davantage à l'élaboration de modèles et de notations permettant de décrire ces tâches [HG92, PMM97] ou encore la façon d'exploiter ces notations pour faciliter le développement du système informatique final [JWMP93, PMM97]. C'est sur ce type de travaux que nous focalisons notre intérêt car notre objectif principal est de fournir des moyens permettant aux concepteurs de décrire et d'implanter des interactions.

Sur les trente dernières années de nombreux modèles de tâches ont été élaborés pour modéliser les tâches menées par un utilisateur en collaboration avec un système informatique. Parmi les plus célèbres, nous pouvons citer GOMS (*Goal, Operator, Method and Selection rules*) et ses dérivés [CMN83], MAD (*Méthode Analytique de Description*) [SPG89] et son extension MAD* [GS97], CTT (*ConcurTaskTree*) [PMM97], TOOD (*Task Object-Oriented Description*) [MAT01, OS04], AMBOSS [GMP+08], Diane+ [TB96], UAN (*User Action Notation*) [HH93], GTA (*Groupware Task Analysis*) [VLB96] ou encore UsiXML (*USer Interface eXtensible Markup Language*) [LV04] qui regroupe plusieurs modèles (dont un modèle de tâches) pour décrire l'interaction.

Tous ces modèles offrent les propriétés suivantes :
- la possibilité de décrire le but qui sera atteint lorsque la tâche sera réalisée ;
- la complexité des tâches est prise en compte en proposant des mécanismes de décomposition hiérarchique des tâches ;
- les opérateurs de séquentialité sont toujours présents pour décrire l'ordonnancement des tâches dans le temps.

Le pouvoir expressif des modèles diffère la plupart du temps sur les opérateurs proposés pour décrire l'agencement des tâches et des sous-tâches dans le temps. En plus de la séquentialité, la plupart des modèles proposent également des opérateurs permettant d'exprimer la répétition d'une tâche, le fait qu'une tâche soit optionnelle, qu'elle soit interruptible, ou encore que deux tâches soient équivalentes. Certains modèles de tâches comme celui intégré dans UsiXML [GGVGC08] proposent en plus des opérateurs pour exprimer le fait que deux tâches peuvent être concurrentes ou mises en parallèle ou encore qu'elles peuvent être réalisées en collaboration.

Les travaux de [LPV01, LV04] proposent une étude comparative intéressante des principaux modèles de tâches. Cette étude met en avant les concepts clés manipulés par chaque modèle et identifie les modèles ayant le plus grand pouvoir d'expression (Diane+, CTT, TOOD et UsiXML) mais qui sont également jugés comme étant les plus complexes ce qui, au final, peut influencer sur la qualité des descriptions obtenues.

Dans tous les cas, chaque modèle est bâti sur un méta-modèle qui définit les propriétés d'une tâche et sur un langage spécifique permettant au concepteur d'utiliser le méta-modèle pour décrire ses propres tâches. Nous remarquons d'ailleurs que les travaux qui décrivent ou exploitent un modèle de tâches se réfèrent généralement au *langage* permettant de décrire les tâches et moins souvent au méta-modèle qui définit ce qu'est une tâche ainsi que ses caractéristiques.

Nous pouvons noter que de nombreux points communs existent entre les différents méta-modèles de tâches cités précédemment mais les langages mis à disposition du concepteur pour décrire ses tâches peuvent varier sensiblement.

La figure 4.1 présente le méta-modèle de GOMS listant les caractéristiques que le concepteur peut utiliser pour décrire ses tâches. Nous y retrouvons les éléments classiques d'une tâche : l'identification de la tâche, son but (qui peut être décomposé en sous-buts), les outils nécessaires à la réalisation de la tâche, les relations temporelles entre les différentes tâches...

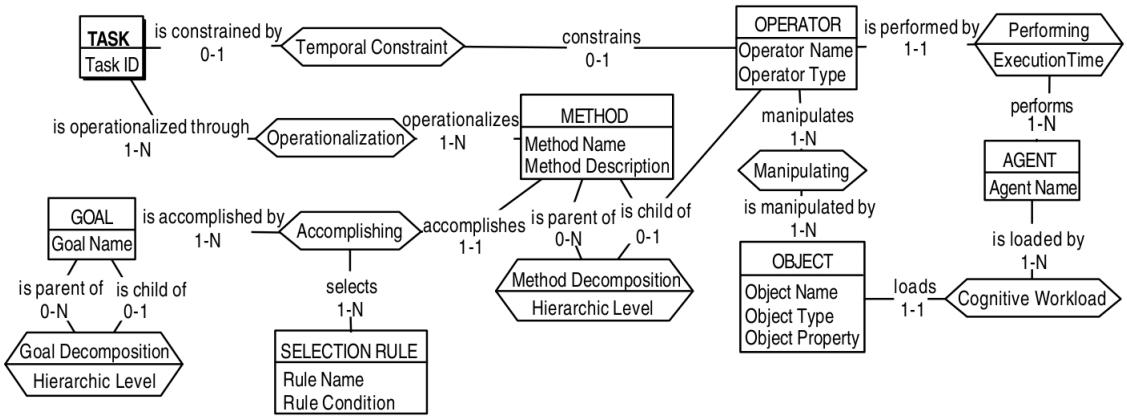


FIGURE 4.1 – Méta-modèle de GOMS [LPV01]

Le langage mis à disposition du concepteur pour décrire ses tâches (Figure 4.2) est de nature textuelle et structurée, la décomposition hiérarchique étant mise en évidence via des techniques d'indentation, comme pour un code source :

```
GOAL: EDIT-MANUSCRIPT
.     GOAL: EDIT-UNIT-TASK ... repeat until no more unit tasks
.     .     GOAL: ACQUIRE UNIT-TASK
.     .     .   GOAL: GET-NEXT-PAGE ... if at end of manuscript page
.     .     .   GOAL: GET-FROM-MANUSCRIPT
.     .     GOAL: EXECUTE-UNIT-TASK ... if a unit task was found
.     .     .   GOAL: MODIFY-TEXT
.     .     .   .   [select: GOAL: MOVE-TEXT* ...if text is to be moved
.     .     .   .       GOAL: DELETE-PHRASE ...if a phrase is to be deleted
.     .     .   .       GOAL: INSERT-WORD] ... if a word is to be inserted
.     .     .   .     VERIFY-EDIT
```

FIGURE 4.2 – Langage textuel pour décrire les tâches dans GOMS (extrait de [JK96])

La figure 4.3 présente le méta-modèle de tâches sur lequel est basé CTT : nous y retrouvons bien les notions de contraintes temporelles entre tâches, d'outils manipulés, de décomposition de tâches en sous-tâches...

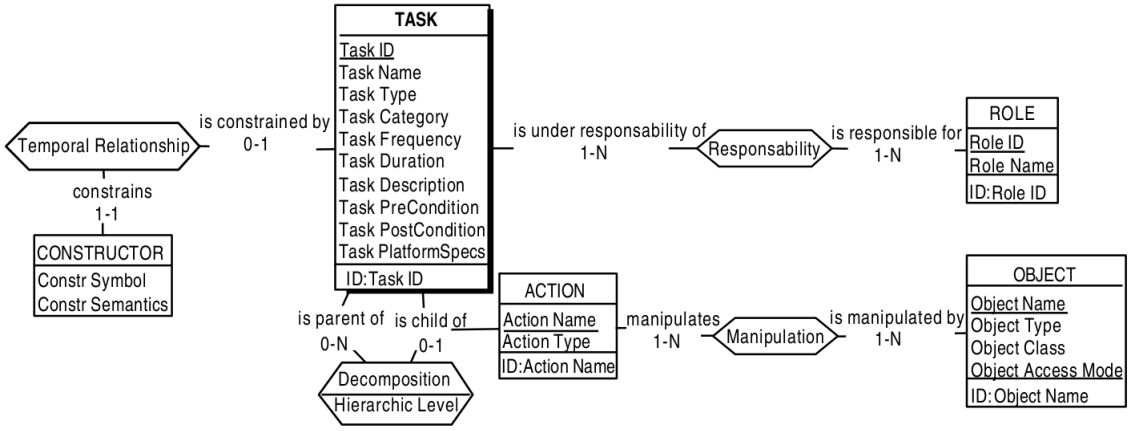


FIGURE 4.3 – Méta-modèle de CTT (extrait de [LPV01])

A l'inverse de GOMS, CTT propose au concepteur un langage graphique pour décrire ses tâches. L'organisation hiérarchique et temporelle des tâches est représentée sous forme arborescente (Figure 4.4). La syntaxe graphique simple à lire et à interpréter est une des clés du succès de CTT.

Si les modèles de tâches permettent de décrire les différentes tâches à enchaîner pour atteindre un objectif précis, ils ne définissent toutefois pas comment les tâches menées en interaction avec la machine sont réalisées. Ces modalités de mise en œuvre sont en général décrites par le biais de modèles de dialogue que nous décrivons dans la section suivante.

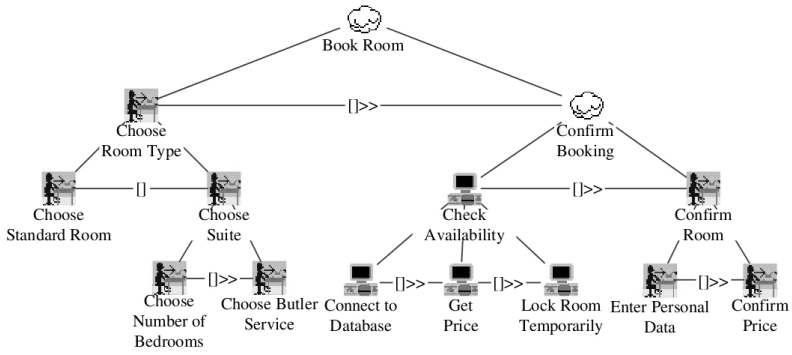


FIGURE 4.4 – Langage graphique pour décrire les tâches dans CTT (d'après [Pat99a])

4.2.2.2 Modèles de dialogue

Dans le cadre de l'interaction homme-machine, le modèle de dialogue vise à organiser la structure de l'interaction. L'objectif des modèles de dialogue est de décrire les échanges entre l'utilisateur et le système. En utilisant ces modèles, le concepteur spécifie quand l'utilisateur peut exécuter des commandes, sélectionner ou entrer des données mais aussi quand le système réagit et affiche des retours.

Les dialogues modélisés sont ceux mis en œuvre entre l'utilisateur et le système pour la réalisation d'une tâche. Les activités de dialogue visent généralement à demander des confirmations, à désambiguïser des situations, à gérer les cas d'erreurs, à demander ou saisir des informations manquantes...

Les travaux de [Gre86, CAS97, Bri87, Pal03] présentent un état de l'art intéressant sur les modèles de dialogue. Les trois familles les plus représentatives [CGGS09] sont : les modèles basés sur des grammaires, les modèles à base d'événements et les modèles à base d'automates.

A. Modèles de dialogue à base de grammaires

Les modèles basés sur les grammaires formelles visent à modéliser la structure du dialogue. La finalité de ces modèles est de rendre compte de la structure des dialogues en construisant des liens de composition, de succession, d'imbrication et de subordination entre les éléments constituants d'un dialogue.

Les règles de structuration du dialogue sont formalisées à l'aide de grammaires définissant des alphabets (terminaux et non terminaux), un axiome de départ et enfin un ensemble de règles permettant de vérifier si un enchaînement d'événements/mots ont un sens par rapport au dialogue modélisé. Le dialogue d'une application interactive est alors modélisé comme une séquence d'éléments terminaux, les éléments non-terminaux

ayant pour rôle de factoriser les sous-dialogues. [CGGS09] illustre comment modéliser une partie des interactions possibles avec un mailer avec un tel système de règles (Figure 4.5).

```
MAILER := DIALOG_ENVOI | DIALOG_CONSULTER | QUITTER
DIALOG_ENVOI := IDENTIFIER_EMAIL envoyer {send();}
IDENTIFIER_EMAIL := NOUVEL_EMAIL | EMAIL_BROUILLON
EMAIL_BROUILLON := select_brouillon {afficher();} MODIFIER_BROUILLON
MODIFIER_BROUILLON := modifier_dest{mettreDest();}MODIFIER_BROUILLON
                    | modifier_message{mettreMessage();}MODIFIER_BROUILLON
                    | 0
```

FIGURE 4.5 – Modélisation des dialogues avec un mailer via une grammaire [CGGS09]

La modélisation du dialogue à l'aide de grammaires a particulièrement été utilisée dans le domaine du dialogue homme-machine. Le modèle Genevois [RAM+85], souvent considéré comme la source et la référence des modèles structurels, a servi de base pour l'élaboration de plusieurs autres modèles similaires tels que SUNDIAL [Bil91] ou encore STUDIA [Che92]. Des études plus complètes sur les modèles de dialogue à base de grammaires sont disponibles dans les travaux de [Leh97] et [Ngu05].

B. Modèles de dialogue à base d'événements

Les modèles de dialogue à base d'événements décrivent les échanges possibles entre l'utilisateur et la machine en termes d'événements utilisateur et d'événements machine. Lorsqu'un événement se produit, il est capturé par un gestionnaire d'événements qui délègue son traitement vers la procédure chargée de le traiter.

Ce modèle de dialogue est utilisé par de nombreux langages de programmation, comme dans Java/Swing [ELW98], GTK [Gri99] ou de manière plus simplifiée dans JavaScript. Sa popularité dans les contextes de programmation événementielle vient de sa facilité d'utilisation mais aussi de son pouvoir d'exécutabilité. En 1986, Green [Gre86] mettait en évidence les qualités indéniables du modèle événementiel dès lors qu'il s'agit de décrire la dynamique dans des systèmes interactifs. La présence de cette gestion événementielle de l'interaction dans beaucoup de langages de programmation donne en partie raison à cette étude.

Toutefois, cette conclusion mérite d'être nuancée : les travaux autour de l'ingénierie dirigée par les modèles (IDM) soulèvent l'importance de pouvoir représenter les différents états du dialogue pour être en mesure de vérifier les propriétés du dialogue et en particulier la cohérence de son ensemble. Les modèles à base d'événements ne permettent pas de réaliser ce type de vérification car ces modèles ne décrivent pas les différents états dans lesquels le système peut se trouver, mais la manière d'y parvenir. Pour cette raison, les approches à base de modèles (MDA - *Model Driven Architecture*) privilégient géné-

ralement les modèles à base d'états tels que les diagrammes états-transitions d'UML par exemple.

C. Modèles de dialogue à base d'états

Les modèles de dialogue à base d'états permettent de formaliser le dialogue sous forme graphique. La spécification est réalisée à l'aide d'un ensemble d'états possibles pour le système (dont un état initial) et un ensemble de transitions qui décrivent les changements d'états du système en fonction d'événements qui surviennent (Figure 4.6).

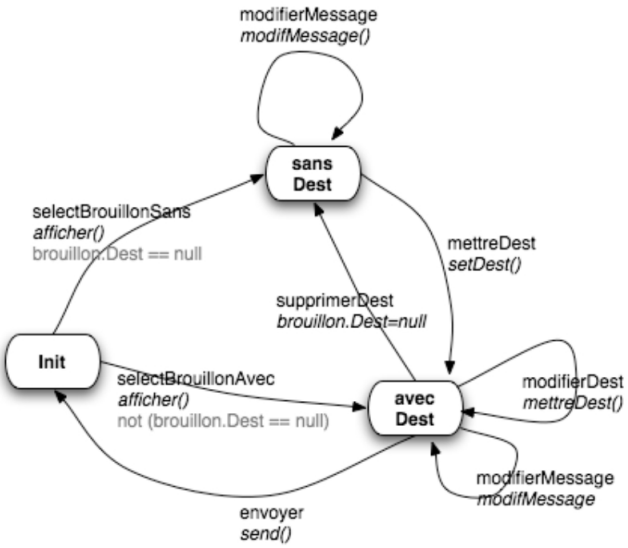


FIGURE 4.6 – Modélisation des dialogues avec un mailer via un automate [CGGS09]

Les diagrammes états-transitions d'UML demeurent sans doute les plus connus et leur formalisme a été repris / dérivé dans de nombreux travaux, y compris des travaux récents. Par exemple, HephaisTK [DLG+08] est une boîte à outils permettant à un concepteur de prototyper des applications multimodales basées sur un système multi-agents appelé JADE. La phase consistant à spécifier les propriétés du dialogue interactif est réalisée à l'aide d'un automate proche des diagrammes états-transitions d'UML (Figure 4.7).

L'intérêt des modèles à base d'automates est qu'ils reposent sur des modèles mathématiques. De ce fait, ils offrent la possibilité de parcourir les graphes résultants à des fins de raisonnement, de validation, ou encore de recherche d'incohérences. De plus, quand les graphes sont déterministes, cela permet de lever risque d'incohérences.

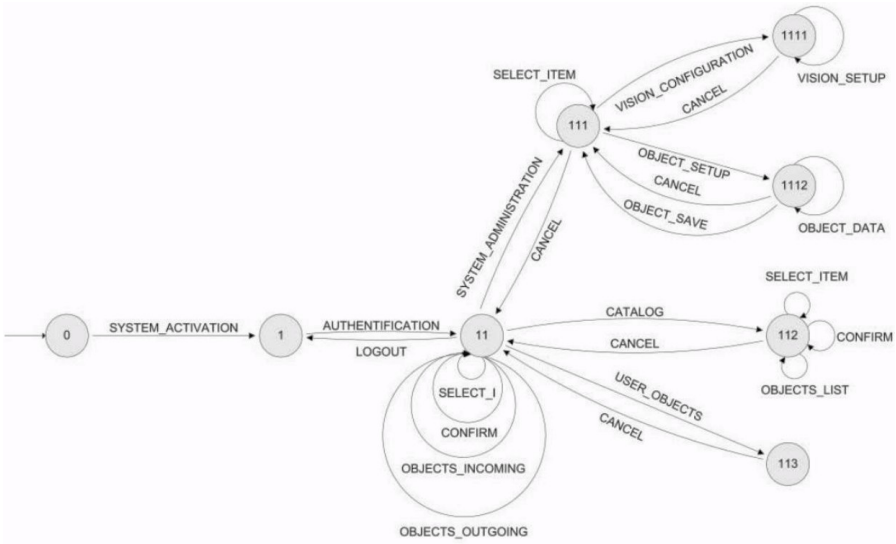


FIGURE 4.7 – Modélisation des dialogues d'une application avec HephaisTK [DLG+08]

4.2.2.3 Modèles de présentation

Les modèles de présentation visent à décrire les propriétés et les éléments composant l'interface utilisateur [GGGCVMA09]. Dans la plupart des travaux, les modèles de présentation se décomposent en plusieurs sous-modèles :

- les modèles de présentation abstraite qui permettent de décrire l'interface en termes d'objets abstraits ;
- les modèles de présentation concrète qui visent à décrire l'interface en termes de widgets.

UsiXML (*USer Interface eXtensible Markup Language*) [LV04] se base sur les travaux issus du projet Cameleon [CCB+02, CCT+03] et s'intéresse à la conception d'interfaces utilisateurs selon une approche basée sur les modèles. Le modèle de présentation est construit à partir du modèle de tâches du domaine qui est décrit à l'aide de CTT. Les caractéristiques de l'interface utilisateur sont décrites selon deux niveaux d'abstraction (interface abstraite puis interface concrète) qui conduisent à la génération d'une interface finale (Figure 4.8 extraite de `www.w3.org/2005/Incubator/model-based-ui/XGR-mbui/#bib-CCB02`).

Au niveau abstrait, l'interface est donc décrite avec des objets d'interface génériques. En s'appuyant sur le modèle des tâches, le concepteur décrit l'organisation de l'interface sous forme d'espaces de travail et fixe les règles de navigation entre ces espaces. Un espace de travail est un "*lieu d'activité virtuel offrant les éléments nécessaires à la réalisation d'une ou plusieurs tâches*" [Nor92]. A ce niveau, la description de l'interface

reste indépendante de toute plateforme et de toute organisation logique des éléments d'interface.

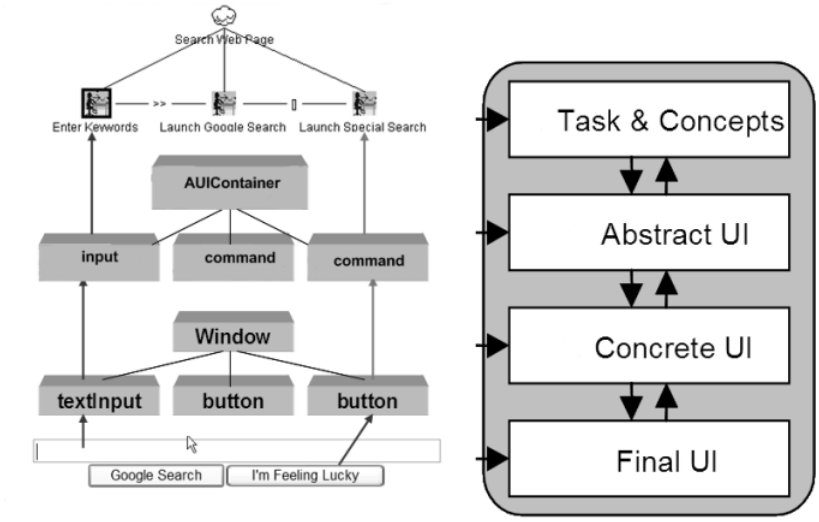


FIGURE 4.8 – Modélisation de l'interface utilisateur dans le projet Cameleon

L'interface concrète est une instance de l'interface abstraite. L'instanciation consiste à donner corps aux espaces de travail sous forme de fenêtres ou canevas et le contenu des espaces sous forme d'objets d'interactions (boutons, menus...).

Communément admise aujourd'hui, la séparation de l'interface abstraite et de l'interface concrète a pour avantage d'enrichir progressivement la spécification d'un système d'information et surtout de pouvoir réutiliser les modèles définis. Une interface abstraite permet, par exemple, de créer plusieurs interfaces concrètes, dans le cadre d'un système d'information multi-plateforme.

4.2.2.4 Modèles architecturaux

Les modèles architecturaux visent à organiser le code d'une application de manière raisonnée de sorte à faciliter son développement, sa maintenance et son évolution. Certains modèles architecturaux dédiés aux interactions homme-machine viennent en support des modèles de conception précédemment présentés. L'objectif est de séparer le code qui implante l'IHM du code "métier" de l'application. Cette séparation évite ainsi de tout modifier si des changements sont à apporter sur la partie fonctionnelle de l'application ou bien sur la partie d'IHM. Elle favorise également la modularité, l'évolutivité et la flexibilité de l'application ainsi que la réutilisabilité de certains de ces composants.

A. Modèle de Seeheim

Dans le cadre de la conception des interactions homme-machine, le premier modèle architectural proposé est sans doute celui de Seeheim [Pfa85]. En plus du module fonctionnel, ce modèle organise l'interface en trois modules :

- l'adaptateur au noyau fonctionnel contient les fonctionnalités permettant d'appliquer les fonctions du noyau fonctionnel à cette interface ;
- la présentation est la partie dédiée à l'interface graphique de l'application ;
- le dialogue permet de lier les deux autres parties en maintenant la cohérence entre les deux.

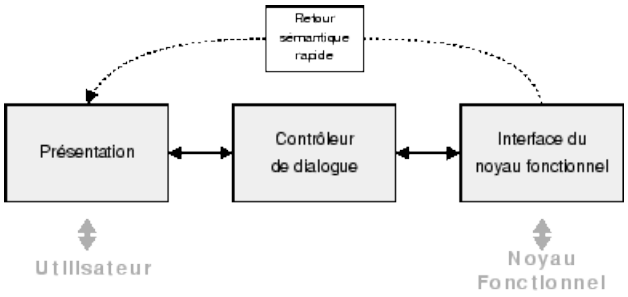


FIGURE 4.9 – Modèle de Seeheim [DF04]

Ce modèle de référence a permis la définition des concepts clés et du vocabulaire qui est utilisé dans l'ensemble des autres modèles architecturaux liés à l'IHM.

B. Modèle Arch

Le modèle Arch [Arc92] fait évoluer le modèle de Seeheim pour tenir compte des évolutions technologiques au sein des IHM et intégrer les possibilités interactives des boîtes à outils graphiques actuelles. Le modèle Arch est organisé autour de cinq composants organisés schématiquement en forme d'arche (Figure 4.10).

Arch repose d'un côté sur le noyau fonctionnel de l'application (non interactif) et d'un autre côté sur le composant d'interaction qui est en contact avec l'utilisateur. Ce composant d'interaction désigne l'ensemble des widgets (objets interactifs) d'une boîte à outils, ainsi que les communications avec les périphériques physiques. La connexion entre les deux piliers est établie par le contrôleur de dialogue qui gère le flux d'information entre les deux composants intermédiaires. Ce contrôleur de dialogue joue donc le même rôle de coordination que celui du modèle de Seeheim. Il gère notamment l'ordonnancement des tâches ainsi que le maintien de la cohérence entre les vues multiples de l'application.

L'adaptateur de domaine joue le rôle de médiateur entre le contrôleur de dialogue et le noyau fonctionnel. Il est responsable des tâches dépendantes du domaine qui ne font pas partie du noyau fonctionnel mais qui sont nécessaires à sa manipulation par l'utilisateur. Lors de la conception d'une application, il est par exemple fréquent qu'un noyau fonctionnel existe déjà (par exemple le système de fichiers d'un système d'exploitation ou une base de données). Dans ce cas, l'adaptateur de domaine va fournir une façade aux objets du noyau fonctionnel préexistant.

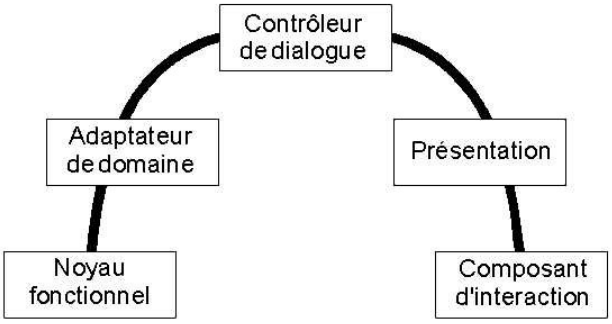


FIGURE 4.10 – Structure du modèle architectural Arch [Arc92]

Dans le même ordre d'idée, le composant de présentation joue le rôle de médiateur entre le composant d'interaction et le contrôleur de dialogue. Il maintient une représentation logique des widgets qui est indépendante de la plate-forme.

C. Modèle MVC

Le modèle MVC (*Modèle Vue Contrôleur*) [KP88] fait partie de la famille des modèles à base d'agents. Dans ce type de modélisation, les applications interactives sont décomposées selon une hiérarchie d'agents. Dans le cas de MVC, un agent est défini par un modèle, une ou plusieurs vues et un ou plusieurs contrôleurs (Figure 4.11).

Le modèle est le noyau fonctionnel de l'agent. Il peut représenter des données brutes ou des objets ayant un comportement complexe. Le modèle notifie les vues qui lui sont associées à chaque fois que son état se trouve modifié par le noyau de l'application ou par ses contrôleurs.

La vue maintient une représentation du modèle perceptible par l'utilisateur. Cette représentation, qui prend la forme d'une IHM, est mise à jour par la couche vue à chaque changement d'état du modèle.

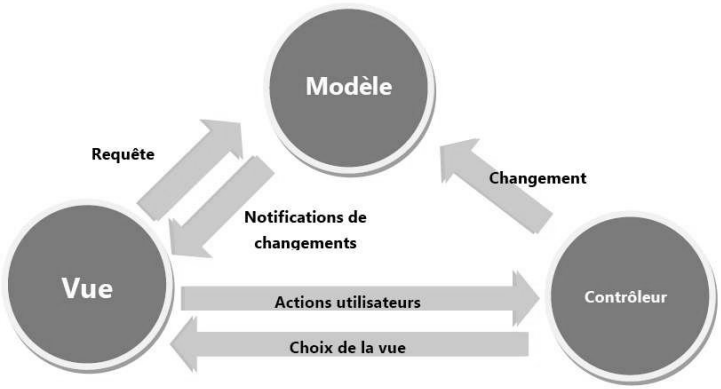


FIGURE 4.11 – Structure du modèle architectural MVC [KP88]

Le contrôleur reçoit et interprète les événements utilisateur, en les répercutant sur le modèle (modification de son état) ou sur la vue (feedback).

Le modèle architectural MVC reste encore aujourd'hui très populaire. La plupart des frameworks de développement Web tels que Zend [41], Symfony [42] ou encore Ruby on Rails [43] reposent sur son architecture.

D. Modèle PAC

Tout comme MVC, le modèle PAC (*Présentation Abstraction Contrôle*) [Cou87] fait partie de la famille des modèles à base d'agents. Ce modèle est également organisé autour de trois couches :

- La couche *Présentation* gère les entrées / sorties avec l'utilisateur ;
- La couche *Abstraction* intègre le cœur de l'application (données et fonctionnalités associées). Il s'agit du noyau fonctionnel ;
- La couche *Contrôle* maintient la cohérence entre les deux couches précédentes.

Ce modèle est parfois confondu avec MVC mais le rôle des composants est sensiblement différent. Dans le cadre du modèle PAC, le contrôle des entrées/sorties utilisateur est géré au niveau de la couche *Présentation*, alors que dans MVC, ce contrôle est assuré par le contrôleur. Le contrôleur du modèle PAC effectue un contrôle différent dans le sens où il s'agit de maintenir en cohérence la couche *Présentation* et la couche *Abstraction*.

41. http://framework.zend.com/
42. http://www.symfony-project.org/
43. http://rubyonrails.org/

4.2.2.5 Modèles UML pour décrire l'interaction

UML (*Unified Modeling Language* - www.uml.org) offre un standard de modélisation pour spécifier les différentes facettes d'une application informatique. Depuis sa version 2, UML propose treize modèles spécialisés permettant au concepteur de décrire les différentes caractéristiques d'une application en considérant plusieurs angles de vue. Aucune méthode n'est imposée, c'est au concepteur de choisir les modèles qui lui semblent les plus appropriés pour spécifier son application.

Ces différents modèles peuvent être organisés selon deux principaux domaines/vues (dont la division est arbitraire) [RJB+04, Aud09] :

- La classification *structurelle* vise à décrire les caractéristiques statiques de l'application. Les diagrammes concernés sont les diagrammes de classes, d'objets, de composants, de déploiement, des paquetages et de structures composites.
- Le comportement *dynamique* est supporté par les autres modèles qui permettent de décrire les propriétés comportementales de l'application. Les diagrammes concernés sont les diagrammes états-transitions, de séquence, de communication (nommé diagramme de collaboration en UML 1.x), global d'interaction, de temps et et le diagramme d'activités.

Le diagramme des cas d'utilisation, qui scinde la fonctionnalité du système en unités cohérentes du point de vue d'un utilisateur extérieur, se retrouve parfois classé dans le premier domaine, parfois dans le deuxième voire même dans un troisième dit *fonctionnel*.

L'objectif ici n'est pas de décrire ces treize modèles mais de s'intéresser à ceux qui présentent des propriétés intéressantes pour décrire l'interaction.

A. Diagramme états-transitions

Le diagramme états-transitions permet de décrire sous forme de machine à états finis le comportement dynamique de tous les objets d'une classe. Son formalisme est similaire aux modèles de dialogue à base d'états (*cf.* Section 4.2.2.2) mais sa finalité est différente puisqu'il s'agit ici de décrire le comportement des différents objets et états d'une classe spécifique du système (Figure 4.12).

B. Diagramme de séquence et Diagramme de communication

Ces deux diagrammes sont des diagrammes d'interaction qui montrent comment les objets collaborent pour réaliser une fonctionnalité donnée. Ce sont deux vues différentes, mais logiquement équivalentes. Les diagrammes de séquence mettent l'accent sur la chronologie de l'envoi des messages (Figure 4.13), tandis que que les diagrammes de communication mettent l'accent sur l'organisation structurelle des objets qui communiquent.

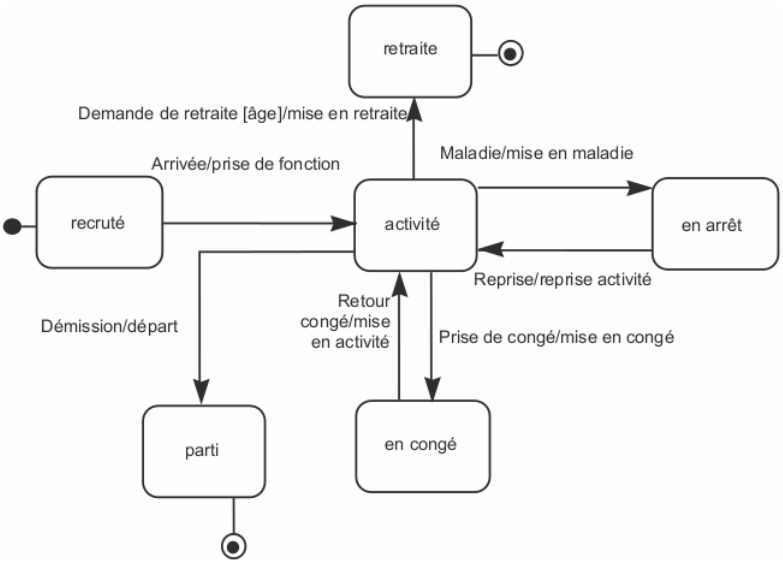


FIGURE 4.12 – Exemple de diagramme états-transitions pour une classe Personne [GG08]

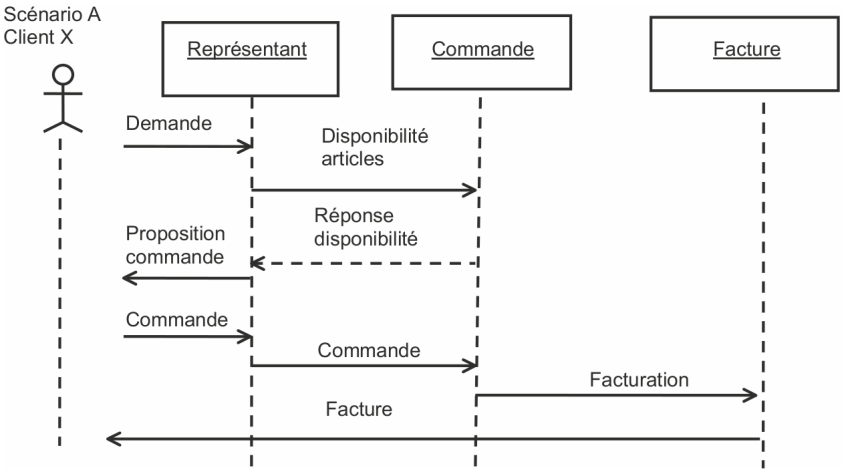


FIGURE 4.13 – Exemple de diagramme de séquence pour l'achat d'articles [GG08]

Dans un diagramme de séquence, la chronologie des échanges est représentée verticalement par une ligne de vie et les échanges entre composants et/ou acteurs sont matérialisés par des messages placés sur des flèches dont l'origine désigne l'objet émetteur et la destination le récepteur.

C. Diagramme global d'interaction

Le diagramme global d'interaction permet de représenter une vue générale des interactions décrites dans le diagramme de séquence et des flots de contrôle décrits dans le diagramme d'activités (Figure 4.14).

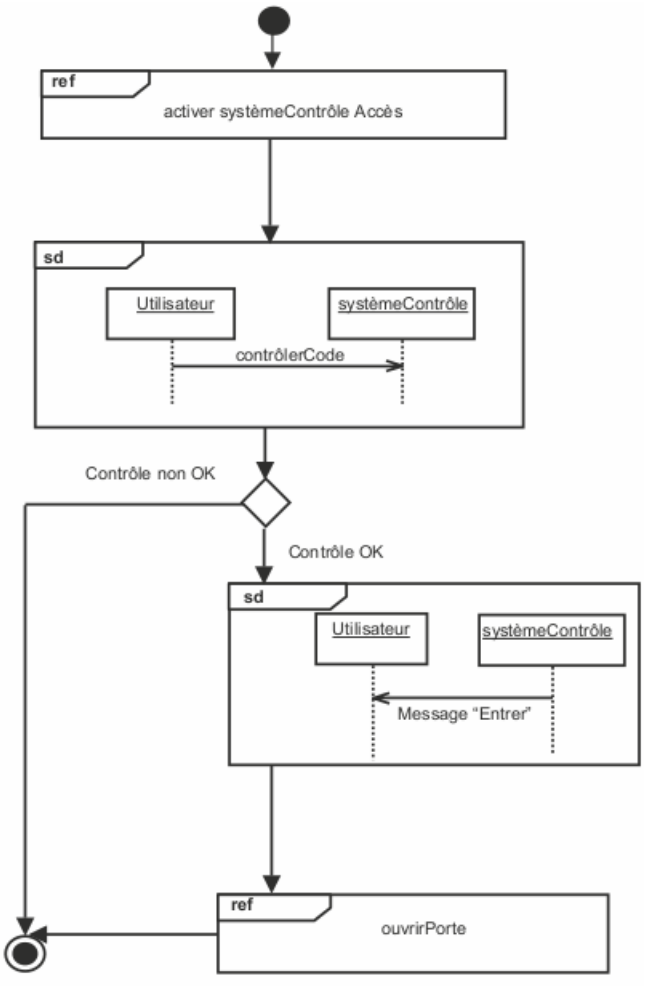


FIGURE 4.14 – Exemple de diagramme global d'interaction [GG08]

D. Diagramme de temps

Le diagramme de temps est une forme de diagramme de séquence sur lequel l'axe du temps est horizontal et gradué, et où les lignes de vie s'affichent dans des compartiments horizontaux. Son usage est limité à la modélisation des systèmes qui s'exécutent sous de fortes contraintes temporelles, comme les systèmes temps réel.

E. Diagramme d'activités

Les diagrammes d'activités permettent de modéliser le cheminement de flots de contrôle et de flots de données. Ces diagrammes permettent de spécifier les activités telles que les voient les acteurs qui collaborent avec le système dans le cadre d'un processus métier (Figure 4.15). Nous nous rapprochons alors fortement de la description de workflows mais aussi de la modélisation de tâches au sens de ce que nous avons décrit dans la section 4.2.2.1.

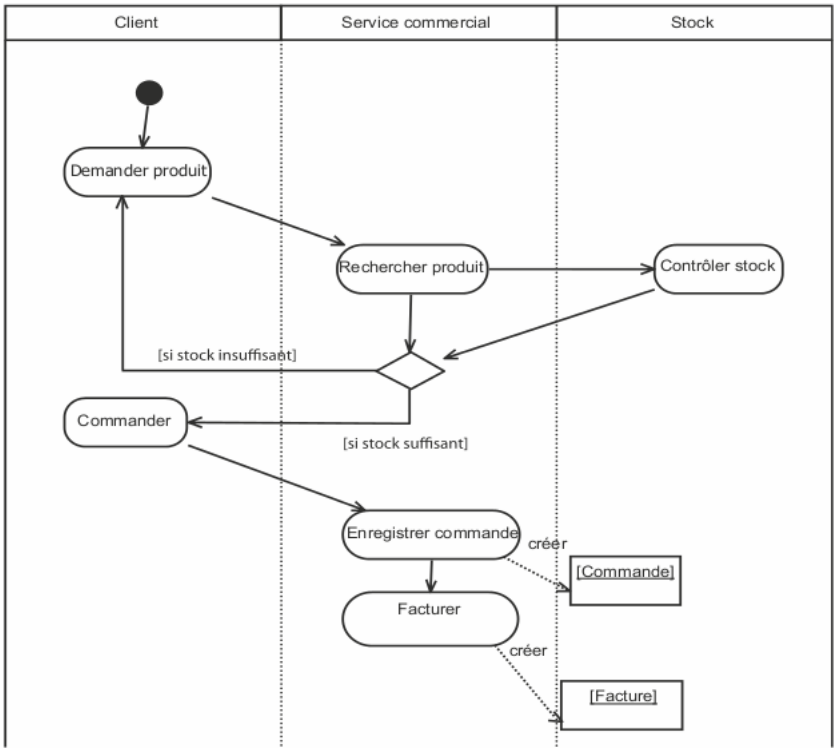


FIGURE 4.15 – Exemple de diagramme d'activités [GG08]

F. Cas particulier d'UML*i*

UML*i* [Sil00, PdSP03] est une extension d'UML proposée pour décrire l'IHM d'une application. Cette extension propose de décrire les liens entre les objets de l'interface et ceux de l'application via deux diagrammes destinés à modéliser la couche présentation de l'application (*cf.* Section 4.2.2.3) et la couche dialogue.

La couche présentation peut être spécifiée à l'aide d'un diagramme de classes ou bien à l'aide d'un diagramme spécifique d'UML*i* qui propose au concepteur cinq éléments nommés *objets d'interaction primaires*[44] (Figure 4.16). Ces éléments représentent des objets d'interaction génériques : conteneurs libres, conteneurs, éléments d'acquisition de données, éléments d'affichage de données et éléments d'édition de données. Ce diagramme vise donc à décrire l'interface abstraite de l'application au sens de la définition donnée dans la section 4.2.2.3.

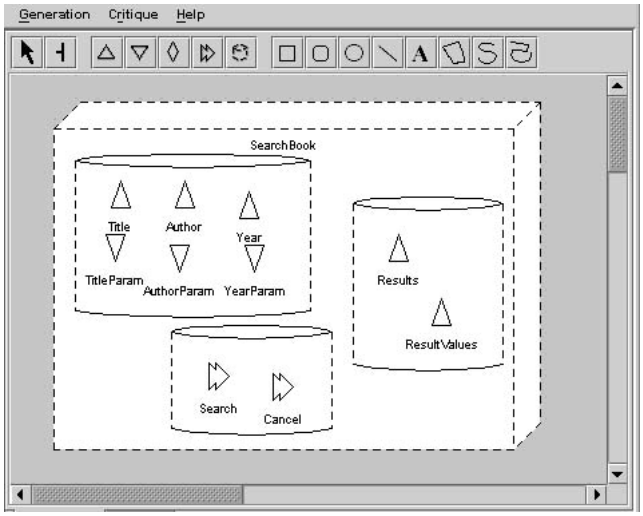


FIGURE 4.16 – Exemple de diagramme de présentation avec UML*i* [PdSP03]

La couche dialogue vise à spécifier les liens entre le noyau fonctionnel de l'application et la couche présentation. Cette spécification est réalisée à partir d'un diagramme nommé diagramme de l'activité d'interaction[45] (Figure 4.17). Ce diagramme introduit de nouveaux items qui relient les éléments d'interaction à une activité primitive, indiquant quel élément de présentation prend en charge quelle activité.

Dans le même esprit que UML*i*, MACAO [Cra02] est une méthode qui repose sur les neufs modèles d'UML 1.5, avec quatre modèles supplémentaires spécifiques dédiés

44. Primitive Interaction Objects
45. Interaction Activity Diagram

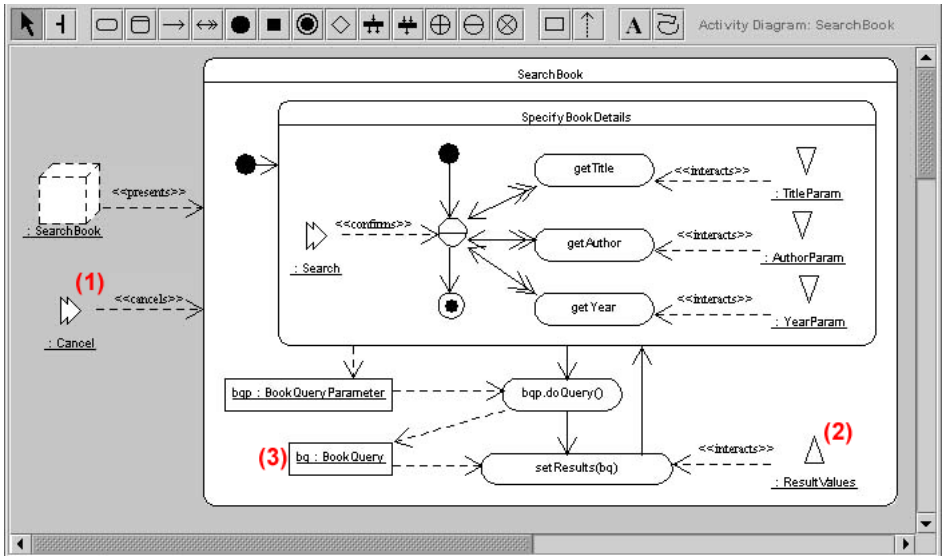


FIGURE 4.17 – Exemple de diagramme de dialogue avec UML*i* [PdSP03]

notamment à la prise en compte des interactions et interfaces dès la phase de conception dans le cycle de vie.

4.2.3 Langages visuels pour décrire l'interaction

Comme présenté dans la section 4.2.1, les modèles offrent un cadre de réflexion en fournissant au concepteur un ensemble de concepts et de propriétés permettant de décrire l'objet à concevoir. Les modèles évitent donc au concepteur de rechercher et d'identifier les propriétés sur lesquelles il doit se pencher pour décrire l'objet qu'il souhaite concevoir.

Si les modèles permettent d'apporter un cadre de réflexion structuré au concepteur, il est souhaitable qu'ils proposent aussi un langage adapté permettant au concepteur d'instancier les concepts du modèle afin de décrire l'objet qu'il élabore. A titre d'exemple, GOMS propose un méta-modèle (*cf.* Figure 4.1) qui liste les caractéristiques permettant de caractériser ce qu'est une tâche. En plus de ce méta-modèle, GOMS fournit également un langage de description textuel (*cf.* Figure 4.2) spécialisé pour décrire des tâches. Le contexte est identique pour CTT qui fournit un méta-modèle (*cf.* Figure 4.3) accompagné d'un langage graphique (*cf.* Figure 4.4) spécialisé pour décrire des tâches selon le point de vue de CTT.

On notera la dimension importante que recouvrent les langages de description dans le sens où ils sont souvent plus connus que les méta-modèles sur lesquels ils reposent. En pratique, la plupart des informaticiens savent par exemple créer des diagrammes de classes ou des diagrammes états-transitions sans forcément connaître complètement le

méta-modèle sous-jacent. Ce dernier est souvent déduit intuitivement à partir des règles et des possibilités de représentation induites par le langage. La plupart des exemples cités autour des modèles de dialogue (section 4.2.2.2) ou de présentation (section 4.2.2.3) présentent d'ailleurs des formalismes graphiques décrivant des instances des modèles et non les modèles eux-mêmes.

Dans cette section, nous nous intéressons aux langages et notations permettant de décrire l'interaction au sein d'une application informatique. Nous focalisons notre étude sur les langages de nature graphique jugés en général, par des non experts, plus simples à lire et à interpréter que des langages textuels ou mathématiques. Le fait que les langages graphiques produisent des diagrammes sur un plan à deux dimensions offre d'ailleurs un pouvoir de représentation non négligeable par rapport à des langages textuels ou mathématiques qui s'écrivent et se lisent de manière linéaire. Une comparaison du langage descriptif de GOMS (Figure 4.1) et de CTT (Figure 4.4) permet, en partie, de comprendre cette idée.

Nous portons donc ici notre attention sur les travaux relatifs aux langages visuels au sens de [Mye90] et plus précisément aux langages de programmation visuels tels que présentés en section 2.1.2 et définis dans [Mye90, Shu99, KH93, NH98] :

- Pour [Mye90], la programmation visuelle se réfère à tout système qui permet à l'utilisateur de spécifier un programme selon deux dimensions ou plus. Bien que cette définition soit large, elle exclut les langages de programmation conventionnels (textuels) car ils ne sont pas considérés comme des langages à deux dimensions dans le sens où les compilateurs / interpréteurs ne les traitent que comme des flux à une dimension.
- Pour [KH93], la programmation visuelle consiste en un langage basé sur des représentations visuelles afin de réaliser des tâches qui, sans ce langage, devraient être écrites dans un langage de programmation traditionnel à une dimension.
- Pour [Shu99], la programmation visuelle se définit comme l'utilisation de représentations graphiques porteuses de sens pour construire des programmes.
- Enfin, pour [NH98], un langage de programmation visuel est un langage de programmation avec un alphabet composé de représentations visuelles.

Dans cette thèse, nous utilisons le terme de *programmation visuelle* pour désigner la "**construction d'un programme à partir de la représentation graphique de son comportement**.

Nous définissons un *langage visuel de programmation* de la manière suivante : "**tout langage graphique qui permet à des programmeurs de créer des programmes en manipulant des éléments du programme représentés graphiquement**". Le fait, qu'il s'agisse d'un langage de programmation sous-entend l'existence d'un système capable de traduire une description graphique sous forme de code exécutable.

4.2.3.1 Langages visuels non spécifiques aux applications géographiques

Nous présentons ici deux langages visuels issus de travaux récents et permettant de décrire des interactions de manière graphique puis d'en générer le code sous-jacent.

A. ICON

La notation ICON [DF04] permet de décrire des interactions dans lesquelles les entrées utilisateur dépassent les techniques standards par le clavier et la souris : interaction gestuelle, interaction multi-digitale, interaction 3D... La notation et l'environnement de conception associé proposent des boîtes à outils qui permettent de décrire des interactions à l'aide de trois familles de dispositifs (Figure 4.18) : les dispositifs d'entrée physique tels que le clavier ou la souris, les dispositifs de traitement pour effectuer des transformations de données et les dispositifs à retour graphique.

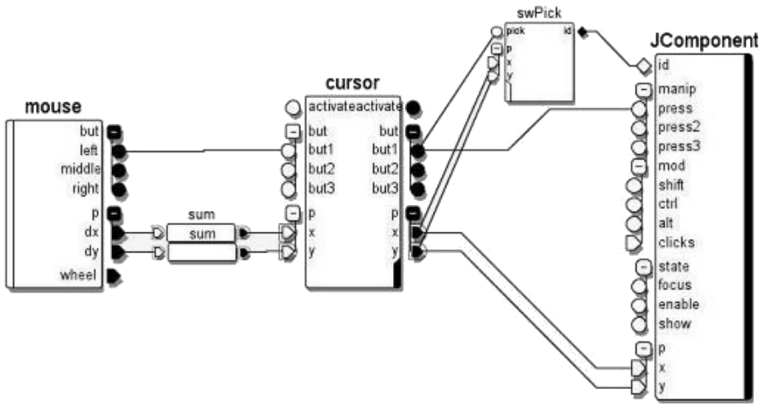


FIGURE 4.18 – Sélection d'un objet graphique modélisée via ICON [DF04]

L'interaction est décrite en plaçant ces dispositifs sur un plan de travail et en les connectant les uns aux autres via des ports spécifiques à chaque dispositif. Nous obtenons alors une interaction décrite selon un flux qui part d'un dispositif d'entrée et se termine sur un ou plusieurs dispositifs à retour graphique (*cursor* et *JComponent* sur la figure 4.18). D'un point de vue technique les interactions décrites ne fonctionnent qu'avec des composant Java / Swing mais le concepteur peut simuler directement l'interaction décrite au sein d'une fenêtre Java.

B. Squidy

Comme ICON, Squidy [KRR10] permet la spécification d'interactions qui vont au delà des interfaces graphiques classiques : reconnaissance vocale, multi-touch, suivi de gestes... Le framework fournit un environnement graphique qui permet au concepteur

de faire abstraction de la complexité technique nécessaire à la mise en œuvre de ce type d'interactions. L'environnement fournit une notation visuelle (Figure 4.19) basée sur la description de flux de données entre dispositifs (à la manière de ICON).

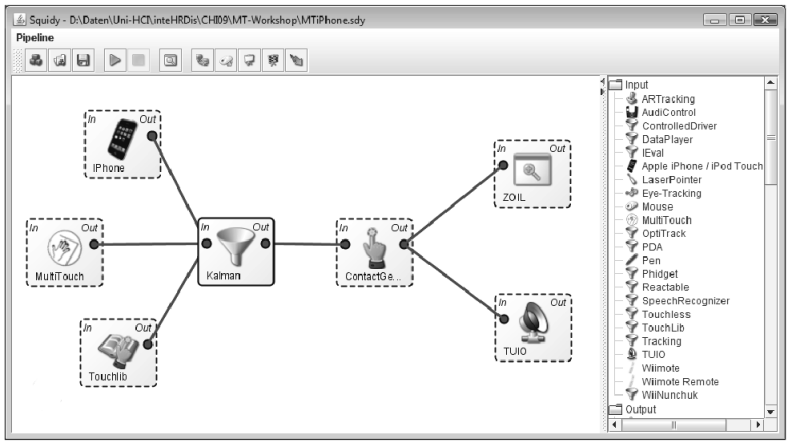


FIGURE 4.19 – Interaction avec un pointeur laser modélisée via Squidy [KRR10]

Squidy peut s'interfacer avec Java, C++, .NET ainsi qu'avec le SDK d'Apple pour l'iPhone. Des détails supplémentaires sont fournis dans [JKR09].

4.2.3.2 Langages visuels spécifiques aux systèmes d'informations géographiques

L'idée d'intégrer des langages visuels de programmation dans les systèmes d'informations géographiques n'est pas nouvelle. Des systèmes tels que ArcGIS [46], Mapinfo [47] ou encore AutoCAD Map 3D [48] restent massivement réservés à des spécialistes du domaine. Dans ce cadre, les langages visuels tentent de rendre accessibles de tels systèmes à des utilisateurs finaux.

Nous décrivons ici uniquement deux systèmes qui restent assez représentatifs des possibilités offertes par les langages visuels au sein des systèmes d'informations géographiques.

A. ArcGIS

ArcGIS est un système d'information géographique permettant d'exploiter des données géographiques pour analyser, traiter et publier des informations spatiales. Les don-

46. www.arcgis.com
47. www.pbinsight.com/welcome/mapinfo
48. www.autodesk.fr/map3d

nées géographiques et les traitements appliqués sur ces données sont décrits sous forme d'un workflow simple d'utilisation (Figure 4.20).

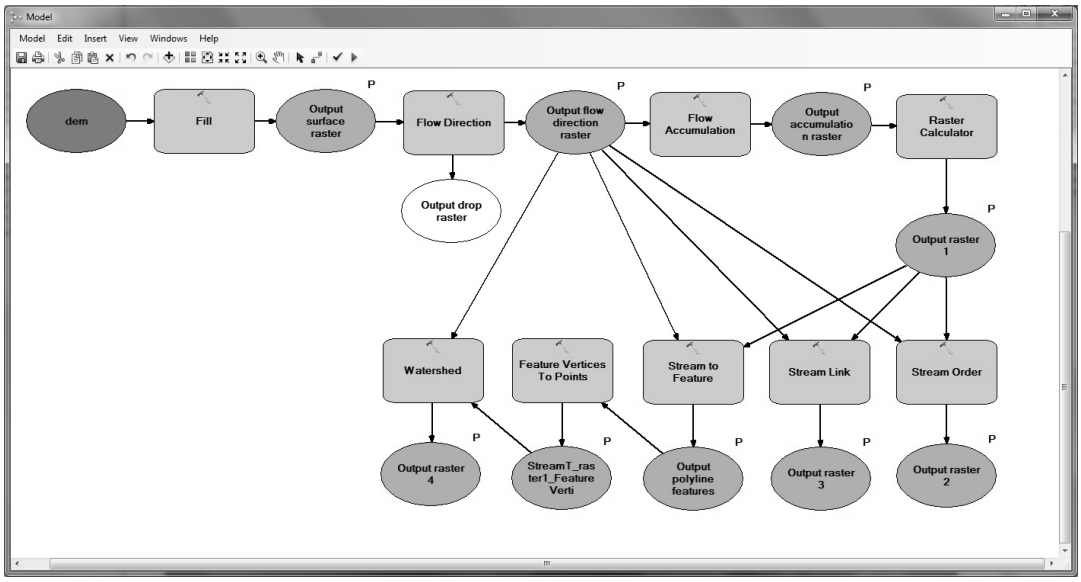


FIGURE 4.20 – Construction et affichage de données géographiques avec le *Model Builder* de ArcGIS

Le workflow est construit à partir de deux entités graphiques de base : des rectangles aux bords arrondis représentant des traitements et des ellipses représentant les données qui sont en entrée et en sortie des traitements. Les données sont connectées aux traitements par des flèches dont le sens précise si les données sont en entrée ou en sortie. Pour donner du sens au workflow élaboré, le concepteur peut ajouter des annotations à divers endroits du diagramme.

B. AutoCAD Map3D

De manière similaire à ArcGIS, AutoCAD Map3D permet de décrire graphiquement un ensemble de traitements à appliquer sur des données géographiques. Cette description est réalisée sous forme d'un workflow où les seules entités graphiques représentées sont les traitements (Figure 4.21).

Les données transformées ne sont pas explicitement représentées mais sont précisées sous forme de paramètres. Les traitements peuvent être enchaînés parallèlement ou séquentiellement et le détail de chaque traitement peut être défini par un ensemble d'icônes.

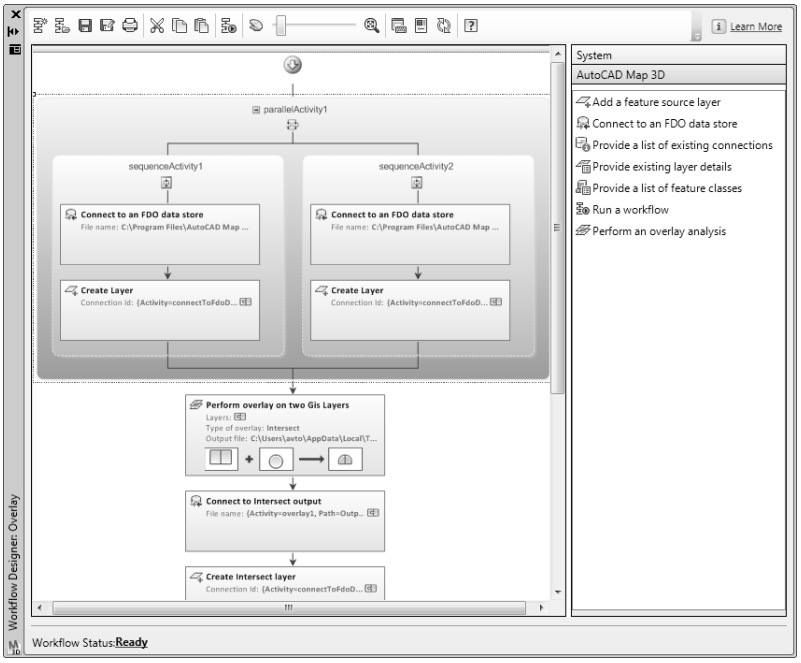


FIGURE 4.21 – Exemple de workflow élaboré sous AutoCAD Map 3D

4.2.4 Synthèse et discussion

4.2.4.1 Notre vision de l'interaction

Comme présenté dans la section 4.2.1, il existe plusieurs définitions de l'interaction mais probablement aucune admise par tous. L'objectif de proposer une définition unifiant les points de vues des uns et des autres n'entre pas dans le cadre de cette thèse. Notre objectif ici est de caractériser l'interaction telle que nous la considérons dans ce travail de thèse et telle que nous voulons la traiter pour concevoir des applications Web à connotation géographique.

Étant donné que nous souhaitons rendre nos travaux à la portée à des non-informaticiens, nous proposons un cadre de conception basé sur du prototypage rapide dans lequel le concepteur peut décrire les caractéristiques de son application, générer le code correspondant et évaluer si le résultat correspond à ses attentes. Dans ce contexte, la génération automatique de code reste une priorité forte pour guider nos choix de conception.

Nous avons donc besoin d'une définition qui soit à la fois simple et opérationnelle. Ces deux critères sont complémentaires l'un de l'autre. Selon nous, l'idée de considérer une définition de trop haut niveau avec des concepts trop abstraits et nombreux se fait au détriment des capacités opérationnelles des outils qui en résulte. La difficulté consiste

donc à trouver le meilleur compromis entre ces deux axes de réflexion et trouver une définition qui soit à la fois :

- simple pour être comprise de tous mais pas trop simple de sorte à ce qu'elle permette de décrire des interactions les plus riches possibles ;
- opérationnelle, de sorte à ce que toute spécification d'une interaction puisse à terme se traduire par un code capable de la mettre en œuvre.

Cette définition de l'interaction doit donc nous permettre d'élaborer des modèles qui puissent être intégrés dans des environnements de conception et qui puissent être instanciés jusqu'à la couche la plus basse, à savoir du code de l'application.

En croisant les définitions données par [Mar99] et [VR09] (*cf.* section 4.2.1), nous considérons qu'une interaction est un échange entre un utilisateur humain et un système informatique. Du fait que nous souhaitons placer l'humain au centre de l'interaction, les travaux de cette thèse se focalisent essentiellement sur les interactions initiées par l'acteur humain. Nous définissons donc une *interaction* comme **une action généralement mise en œuvre par un acteur humain sur un système et qui déclenche une réaction du système qui peut être directe (perceptible par l'acteur humain) ou indirecte (interne au système comme par exemple un calcul)**.

Dans le cadre de cette thèse, la conception d'interaction vise donc à modéliser les actions qu'un utilisateur peut faire sur un système ainsi que les réactions de ce dernier.

4.2.4.2 Synthèse des modèles pour décrire l'interaction

Dans les sections précédentes, nous avons vu que la modélisation de l'interaction débute généralement par la modélisation des tâches que l'utilisateur doit mener. A la vue des tâches à réaliser, il s'agit ensuite de définir les dialogues que le système devra pouvoir mettre en œuvre afin d'assister l'utilisateur dans la réalisation de ses tâches. Les modèles de présentation viennent donner un visage à l'IHM avec laquelle l'utilisateur interagira pour mener ses tâches. Les modèles architecturaux proposent, en bout de chaîne, un cadre logiciel pour produire un code applicatif dont le comportement interactif est conforme aux propriétés établies à l'aide des modèles précédents.

Ces différents modèles offrent ainsi un cadre de modélisation riche qui permet de décrire les différentes facettes de l'interaction. La complémentarité des modèles permet au concepteur de réaliser son travail de modélisation de manière organisée et structurée et ce cadre de réflexion reste particulièrement appréciable lorsqu'il s'agit de concevoir des systèmes informatiques dans lesquels le niveau d'interactivité est à la fois riche et complexe. Des cadres de modélisation tels que UsiXML [LV04], UIML [APB+99] ou encore Teresa [PS03] offrent ainsi au concepteur des "packs" de modèles complémentaires permettant d'organiser sa réflexion afin d'aboutir à des interfaces adaptées à la réalisation des tâches modélisées.

Évidemment, l'usage de ces différents modèles nécessite des capacités d'abstraction et de modélisation qui demeurent l'apanage de spécialistes. Il faut en effet être capable de séparer clairement les différents niveaux d'abstraction qui permettent de décrire les différentes couches interactives d'un système. Certains modèles restent d'ailleurs, même pour des spécialistes, complexes à utiliser. Instancier un modèle de tâches pré-suppose par exemple d'être capable d'identifier les tâches à décrire. Or ce travail d'identification préalable reste lui-même délicat et a d'ailleurs nécessité de mettre au point des méthodes issues des sciences cognitives [PP01].

Dans le cadre de cette thèse, nous focalisons notre intérêt sur la conception d'applications géographiques de taille "raisonnable" telles que définies dans le chapitre 1 et nous souhaitons que la création de ces applications puisse être réalisée en autonomie par des concepteurs non-informaticiens. Ces deux raisons nous poussent donc à écarter tout cadre de modélisation qui s'avérerait trop complexe pour le public de concepteur que nous visons mais aussi pour la famille d'applications ciblée dont la complexité doit restée maîtrisable par un non-informaticien.

Par rapport à nos objectifs, il nous faut toutefois considérer l'apport potentiel des différents modèles décrits dans cet état de l'art :

- Les *modèles de tâches* et les approches cognitives associées nous semblent trop lourds à utiliser vis-à-vis de la taille des applications visées mais aussi des capacités d'analyse et d'abstraction des concepteurs que nous souhaitons toucher. Nous proposons ainsi d'alléger le processus de conception de cette phase d'analyse des tâches tout en restant conscient que les applications générées ne pourront réaliser que des tâches basiques que le concepteur n'aura pas besoin de décrire de manière formelle. Ce choix s'inscrit dans l'idée de pouvoir concevoir des applications dont les caractéristiques interactives ne sont pas forcément clairement identifiées dès le départ.
- Les *modèles de dialogue* nous semblent incontournables dans le sens où ils décrivent les actions possibles d'un utilisateur sur un système ainsi que les réactions de ce dernier. Ce modèle reste au cœur de l'interaction et notre proposition doit donc proposer des moyens permettant exprimer cette dimension de l'interaction.
- Les *modèles de présentation* recouvrent également un aspect important de l'interaction puisqu'ils décrivent la couche visible de l'interaction. Au delà du fait qu'elle est incontournable, cette couche, de par sa nature visuelle, reste particulièrement importante pour des concepteurs non experts qui auront plus de facilité à raisonner sur des éléments graphiques.
- Les *modèles architecturaux* jouent un rôle secondaire pour les concepteurs que nous visons. Étant donné que nous souhaitons mettre en place des outils de génération automatique de code, les concepteurs ne doivent pas avoir besoin de se soucier de la manière d'organiser le code final de leur application. La mise au point de ces outils de génération de code nécessite néanmoins que nos travaux se basent sur un modèle adapté à l'interaction. Bien que la terminologie et certains concepts

différent d'un modèle architectural à l'autre, certains principes incontournables restent récurrents : tous les modèles architecturaux structurent le code de l'application selon plusieurs couches, chacune ayant un rôle spécifique. Les architectures PAC [Cou87] et MVC [KP88] par exemple séparent, chacune avec leur spécificité, la couche données, la couche présentation et la couche contrôle. Cette thèse n'a pas pour objectif d'apporter une contribution sur les modèles architecturaux mais plutôt d'utiliser les principes architecturaux mis en avant par ces modèles de sorte à faciliter la mise au point de nos outils de génération de code par transformation de modèles.

L'objectif de nos travaux vise à proposer des modèles permettant de décrire l'interaction de manière visuelle. Selon les modèles cités précédemment nous nous situons à la croisée des modèles de dialogues et des modèles de présentation. Le but est de permettre au concepteur de décrire les éléments de l'interface (couche présentation) avec lesquels l'utilisateur peut interagir. Dans le cadre des applications géographiques, il s'agira de spécifier avec quelles données géographiques présentées sur l'interface l'utilisateur pourra interagir. Cette spécification conduira le concepteur à décrire les possibilités interactives de son application en fonction des données affichées, sachant que ces possibilités interactives consisteront à spécifier des dialogues possibles entre l'utilisateur et le système (couche dialogue). Il s'agit donc de proposer une approche hybride permettant à des concepteurs non spécialistes de décrire les éléments qui peuvent être rendus interactifs sur une interface et les conséquences d'une action utilisateur sur ces éléments affichés.

4.2.4.3 Synthèse des langages visuels pour décrire l'interaction

Les langages de programmation visuelle visent à faciliter le travail du concepteur en proposant des représentations graphiques riches de sens facilitant la spécification de l'objet à concevoir. La richesse graphique du langage permet de manipuler des descripteurs de haut niveau et donc de décrire plus facilement des objets qui, par nature, peuvent se révéler complexes.

Les langages visuels décrits dans la section 4.2.3.1 offrent une approche intéressante pour décrire l'interaction en termes de flux depuis un dispositif d'entrée jusqu'à un composant de sortie retranscrivant les réactions du système. Ces flux peuvent traverser des composants internes au système pour décrire des traitements visant à modéliser les répercussions de l'action utilisateur au sein du système. L'interaction reste à l'initiative de l'utilisateur et cette initiative est modélisée par l'usage d'un périphérique d'entrée. Les actions de l'utilisateur sont donc décrites par le biais de périphériques de saisie mais les éléments d'interface sur lesquels l'utilisateur agit via ces périphériques ne sont pas explicitement décrits. En ce sens, la partie de l'interaction qui touche à la couche présentation n'est que partiellement décrite et impose au concepteur de décrire le déclenchement de l'interaction à partir d'un périphérique et non pas en fonction des éléments graphiques disponibles sur l'interface.

Les langages visuels utilisés actuellement dans les systèmes d'informations géographiques (*cf.* section 4.2.3.2) ne permettent pas vraiment de concevoir la dimension interactive d'une application géographique. Comme le montre [Dob11], les deux problèmes récurrents traités par les langages visuels au sein de ces systèmes concernent la création, l'agrégation, la sélection de données et la manière de les représenter sur une carte géographique ou de manière textuelle. Les éléments graphiques composant le langage permettent au concepteur de définir ou de calculer les données géographiques qui l'intéresse puis de choisir sous quelle forme les représenter. Aucun élément du langage ne permet ensuite de définir des interactions utilisateur sur ces données affichées, seules les interactions prédéfinies proposées par défaut dans les afficheurs sont possibles. La plupart du temps, ces interactions prédéfinies demeurent d'ailleurs peu ou pas personnalisables par le concepteur.

Au delà des langages visuels présentés dans la section 4.2.3, il nous faut également considérer à nouveau le cas particulier d'UML. La section 4.2.2.5 a mis en avant les diagrammes d'UML permettant de décrire la dimension interactive d'une application. Ces diagrammes peuvent être assimilés à des langages visuels car ils offrent au concepteur des notations graphiques lui permettant de spécifier les caractéristiques de l'application qu'il souhaite concevoir. Dans le cadre de réflexion qui est le nôtre les capacités de génération automatique de code offertes par les modèles et les langages de description associés demeurent un critère de choix central. Il faut d'une part que les modèles et langages soient suffisamment élaborés pour pouvoir décrire des interactions riches mais il faut également que ces modèles manipulent des concepts qui puissent être retranscrits sous forme de code exécutable. La difficulté consiste donc à trouver le meilleur compromis entre richesse du modèle et opérationnalisation du modèle, ces deux critères étant parfois antinomiques.

Ils sont toutefois souvent pris en compte dans les travaux autour d'UML. Le langage est à la fois riche (13 diagrammes sont disponibles) et beaucoup d'Ateliers de Génie Logiciel [49] et travaux de recherche ont montré qu'il était possible de générer du code à partir d'une spécification UML. Comme présenté dans [Bar08, KNNZ99, ZBR09], UML peut être utilisé comme langage de modélisation pour générer du code Java à partir de diagrammes de classes et de diagrammes états-transitions. Les aspects interactifs du système sont décrits par des diagrammes états-transitions qui spécifient les possibilités interactives de l'utilisateur en fonction de l'état du système mais aussi les réactions internes de ce dernier lorsque les éléments qui le composent changent d'état suite à une action de l'utilisateur. Ces diagrammes états-transitions permettent de décrire la dynamique interactive d'une application dans sa globalité en décrivant le comportement de chaque composant du système en réaction des différents événements qui peuvent survenir. Cette spécification globale peut se révéler toutefois complexe à décrire, en particulier

49. IBM Rational Rose (www.ibm.com/software/fr/rational), Modelio (www.modeliosoft.com), BOUML (www.bouml.fr), etc.

si l'application à élaborer doit proposer des interactions nombreuses.

Les travaux présentés dans [SCE07] offrent une méthodologie de modélisation d'une histoire pour une séquence d'animation d'un jeu en utilisant des diagrammes d'activités UML (Figure 4.22).

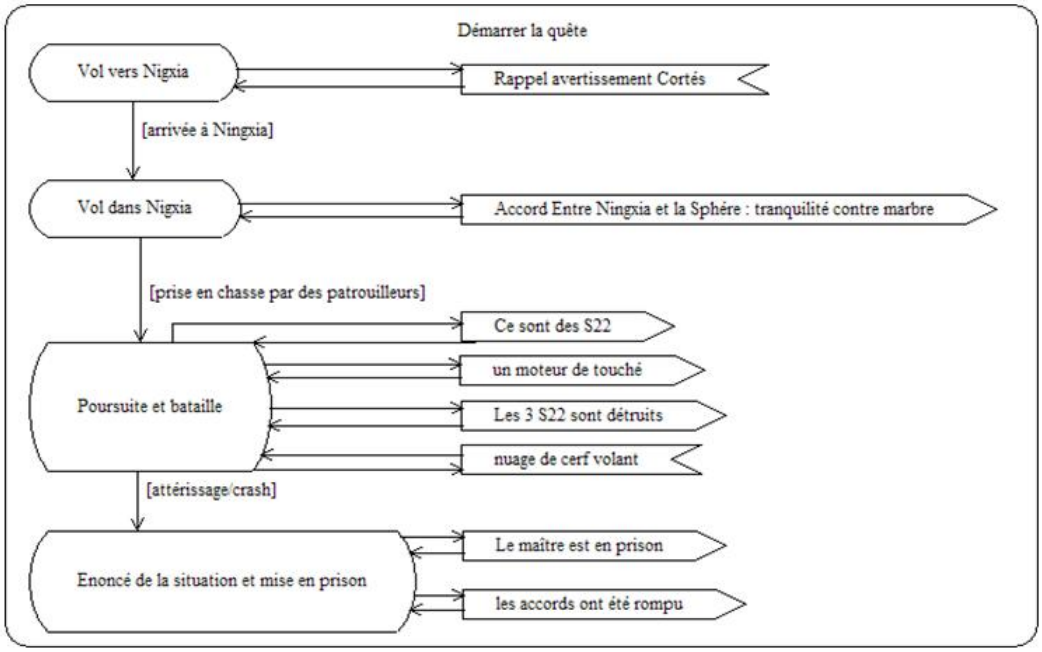


FIGURE 4.22 – Diagramme d'activités d'une histoire [SCE07]

Les travaux présentés dans [HM03] mettent en avant les capacités intéressantes des diagrammes de séquence UML pour décrire en temps réel, à l'exécution, les interactions de l'utilisateur avec son application (Figure 4.23).

A la manière d'ICON ou de Squidy, présentés en section 4.2.3.1, les diagrammes de séquence décrivent l'interaction sous forme de flux matérialisant les échanges entre l'utilisateur et le système mais aussi entre les composants du système. Les diagrammes de séquence permettent, par contre, de faire abstraction des périphériques utilisés par l'utilisateur et soulignent, sous forme de messages, ce que l'utilisateur peut faire sur le système indépendamment du périphérique d'entrée utilisé. Ces diagrammes sont d'ailleurs quasiment les seuls [50] à intégrer une représentation de l'utilisateur et de son rôle dans

50. Les diagrammes de communication sont également intéressants par la simplicité de leur formalisme mais la représentation temporelle non explicite rend leur création moins intuitive pour un non-spécialiste.

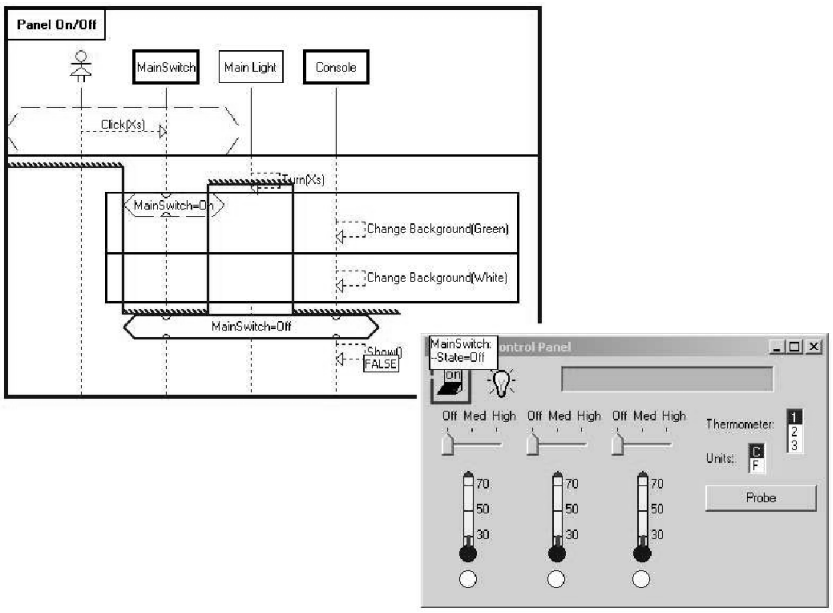


FIGURE 4.23 – Suivi d'une interaction utilisateur sur un diagramme de séquence [HM03]

l'interaction. La représentation de l'interaction sous forme de flux décrivant les échanges entre l'utilisateur et le système nous apparaît comme une manière naturelle pour décrire ce qui se passe entre un utilisateur et un système au cours d'une interaction. De plus, contrairement à des diagrammes états-transitions, ils permettent au concepteur de décrire la dimension interactive du système en plusieurs étapes et non pas dans sa globalité. Cette propriété de décomposition de l'interaction nous semble également intéressante pour des concepteurs non-informaticiens qui peuvent avoir du mal à considérer la dimension interactive d'une application dans sa globalité, en particulier si le système à concevoir offre des possibilités interactives nombreuses.

4.3 Contribution : Phase *Interaction*

La contribution de ce chapitre de thèse porte sur la modélisation de l'interaction et sa spécification via un langage visuel de programmation [LEMN12]. Nous décrivons d'abord le modèle d'interaction que nous proposons, puis un langage visuel permettant au concepteur d'instancier le modèle précédent afin de spécifier les caractéristiques interactives d'une application Web géographique. Nous abordons enfin les capacités de génération de code offertes par ce modèle et le langage visuel associé.

4.3.1 Un modèle d'interaction axé sur les contenus

Dans cette section nous proposons un modèle d'interaction élaboré selon l'hypothèse suivante : une application finale met à disposition de l'utilisateur des contenus (présentés dans la section 3.4) sur une interface (présentée dans la section 3.5) avec lesquels il peut interagir et le résultat d'une interaction se traduit par la valorisation et la création de nouveaux contenus. Dans cette approche, la conception de l'interaction est guidée par des contenus (géographiques) à valoriser. La valorisation peut être matérialisée par un contenu affiché avec lequel l'utilisateur va interagir, un contenu que l'utilisateur va saisir ou bien encore par un nouveau contenu que le système va créer puis présenter à l'utilisateur suite à une interaction.

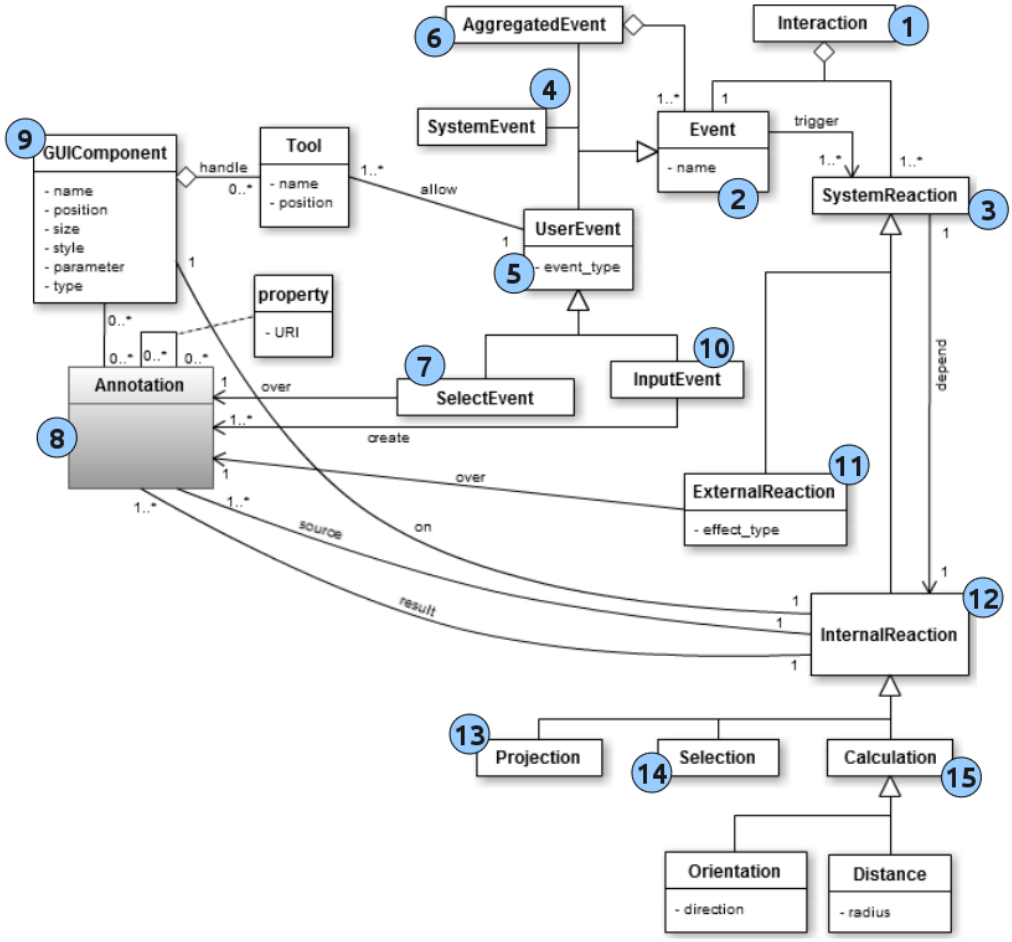


FIGURE 4.24 – Modèle d'interaction centré sur les contenus

Le modèle d'interaction que nous proposons est décrit sur la figure 4.24. Une interaction ① est définie comme un événement particulier ② déclenchant une réaction du système ③. L'événement déclencheur peut être envoyé par le système ④, créé par une action de l'utilisateur ⑤ ou il peut encore s'agir d'une agrégation ⑥ d'événements système et/ou utilisateur ayant eu lieu (par exemple, il est 10h et l'utilisateur a réalisé une action particulière).

Un événement utilisateur ⑤ peut être défini à partir de deux types d'actions possibles : il peut s'agir d'une action de sélection ou d'une action de saisie. Une action de sélection ⑦ est définie par la sélection d'une annotation ⑧ affichée à l'interface ⑨ tandis qu'une action d'entrée ⑩ est caractérisée par la création d'une nouvelle annotation ⑧ dans un composant d'interface ⑨. Dans ce modèle, les actions de l'utilisateur se résument donc à désigner des annotations présentées à l'interface (par exemple, `click`, `mouseover`, `dragdrop`...) ou bien à créer de nouvelles annotations via des éléments d'interface spécifiques (comme par exemple, des boutons d'annotation manuelle dans un composant textuel ou des outils à dessiner des lignes, polygones sur un composant cartographique). Nous rappelons que le concept d'*annotation* a été présenté dans la section 3.4 et qu'il existe au sein des trois sous-modèles de *Contenu*, d'*Interface* et d'*Interaction*.

Les réactions du système ③ peuvent être de deux natures : externes ou internes. Les réactions externes ⑪ sont des réactions du système perceptibles par l'utilisateur. Elles sont définies par un effet visuel (`show`, `hide`, `highlight`, `zoom`...) appliqué sur une annotation ⑧ présentée sur l'interface (`GUIComponent`) ⑨. Les réactions internes ⑫ correspondent à des opérations qui, au cours d'une interaction, vont créer une nouvelle annotation, modifier ou déplacer une annotation existante. Le modèle actuel considère trois types de réactions internes :

- les opérations de projection (`projection`) ⑬ permettent de copier une annotation ⑧ existante d'un composant d'interface ⑨ vers un autre ;
- les opérations de sélection (`sélection`) ⑭ permettent au système de déterminer quelle annotation ⑧ a été sélectionnée par l'utilisateur parmi toutes les annotations affichées sur un composant d'interface ⑨ ;
- les opérations de calcul (`calculation`) ⑮ permettent de calculer de nouvelles annotations à partir d'annotations existantes ou précédemment calculées. Dans le cadre des applications géographiques, ces opérations de calcul seront de nature spatiale ou temporelle (par exemple, déterminer le département auquel appartient une ville, calculer le nombre de jours entre deux dates...)

Quelle que soit l'interaction considérée, une action de l'utilisateur sur le système pourra engendrer une ou plusieurs réactions internes mais l'interaction devra, à terme, se conclure par une ou plusieurs réactions externes appliquant ainsi des effets visibles par l'utilisateur sur une ou plusieurs annotations présentées sur l'interface.

L'interaction que nous proposons via ce modèle reste dirigée par les contenus (annotations) que le concepteur souhaite valoriser au travers de son application. La valorisation est mise en œuvre via l'interaction : les annotations qui entrent en jeu dans une interaction peuvent être :

- prédéfinies par le concepteur via la phase *Contenu* de l'environnement WINDMash et rendues interactives sur un composant d'interface (un afficheur) ;
- sélectionnées par l'utilisateur au cours de l'interaction (l'utilisateur sélectionne alors une annotation parmi l'ensemble des annotations qui lui sont présentées à l'interface) ;
- calculées par le système lors d'une interaction (*cf.* réactions internes ci-dessus) ;
- mises en valeur par le système via une réaction externe.

4.3.2 Un langage visuel pour décrire l'interaction

Le modèle d'interaction présenté dans la section 4.3.1 propose un ensemble de concepts pour décrire l'interaction au sein d'une application. Le modèle représente en lui-même un cadre de réflexion qui permet de guider le travail du concepteur en lui proposant de spécifier l'interaction à partir de contenus interactifs pouvant déclencher des réactions du système.

Pour faciliter la conception de l'interaction nous proposons, dans cette section, un langage visuel permettant au concepteur de décrire les composants de chaque interaction qu'il souhaite mettre en place. Basé sur le modèle d'interaction présenté sur la figure 4.24, le langage visuel proposé permet de caractériser, pour chaque interaction, des actions utilisateur, les annotations sur lesquelles portent ces interactions, les réactions internes du système (projection, sélection, calcul) ainsi que les réactions externes (effets visuels résultants de l'interaction).

4.3.2.1 Principes du langage visuel

Comme annoncé en synthèse (*cf.* section 4.2.4.3), nous retenons les potentialités intéressantes des diagrammes de séquence UML pour décrire des interactions. L'intérêt porté à ces diagrammes est né des observations suivantes :

- Ils sont plutôt faciles à maîtriser car ils se basent sur peu de concepts pour décrire les interactions entre un utilisateur et un système.
- Ils distinguent clairement les actions de l'utilisateur sur le système (messages allant de l'utilisateur vers le système) et les réactions du système vers l'utilisateur (messages allant du système vers l'utilisateur).
- Ils permettent aussi bien de décrire les interactions entre l'utilisateur et le système que les interactions entre les composants du système.
- La chronologie des messages est clairement et simplement exprimée avec la ligne de vie dans sa dimension verticale.

Comme l'ensemble des langages visuels présentés en section 4.2.3, ils permettent de décrire l'interaction sous forme de flux échangés entre l'utilisateur et le système. Cette représentation par flux nous apparaît comme une manière intuitive de décrire les différentes phases d'une interaction depuis l'initialisation de l'interaction via une action utilisateur, en passant par les répercussions de cette action sur les composants internes du système et en allant jusqu'aux retours visuels finaux qui clôturent l'interaction et restituent un résultat à l'utilisateur.

Le mode d'expression de ces diagrammes permet de retranscrire assez naturellement des interactions décrites sous la forme "*lorsque l'utilisateur réalise telle action sur tel contenu affiché sur tel composant de l'application, le système réagit de la manière suivante.*". Cette façon informelle de décrire une interaction nous semble assez proche du schéma de pensée utilisé par un concepteur non-informaticien pour décrire le comportement de l'application qu'il souhaite réaliser.

Chaque diagramme décrit une et une seule interaction : une action de l'utilisateur sur le système provoquant une ou plusieurs réactions de la part de ce dernier. Ces diagrammes permettent donc au concepteur de décrire la couche interactive par étape, interaction après interaction. L'ensemble des diagrammes de séquence décrits spécifieront ainsi l'ensemble des capacités interactives de l'application construite.

Le langage que nous proposons s'inspire du formalisme des diagrammes de séquence UML mais les diagrammes résultants ne peuvent pas être considérés comme des diagrammes de séquence en soit ni une de leurs extensions. Les adaptations majeures que nous proposons sont les suivantes :

- Les seuls composants du système que le langage considère sont des composants définissant l'interface utilisateur (texte, carte...). Cette restriction a été adoptée afin que l'interaction soit décrite selon une approche visuelle dans laquelle le concepteur décrit ce qui se passe pour chaque composant d'interface lorsque l'utilisateur interagit avec l'application. En ce sens nous rejoignons et adhérons aux idées présentées dans [HK01] où l'interaction est décrite à partir des composants visuels constituant l'interface.
- Les annotations manipulées au cours de l'interaction doivent être représentées et liées aux composants d'interface sur lesquels elles apparaissent en début d'interaction ou vont apparaître en fin d'interaction.

4.3.2.2 Composants du langage visuel

Dans cette section nous présentons les blocs de construction du langage visuel que nous proposons pour décrire l'interaction. Chaque interaction est décrite à partir d'un diagramme qui spécifie quelle est l'action utilisateur qui initie l'interaction et quelles sont les réactions du système qui en résultent. Les annotations impliquées dans l'interaction sont représentées et mises en évidence sur les diagrammes.

A. Spécification d'une action utilisateur

Comme défini précédemment, une action utilisateur peut être de deux types : une action de sélection ou bien une action de saisie.

Action de sélection

L'action de sélection permet à l'utilisateur de sélectionner une annotation présentée à l'interface. Elle est matérialisée par un événement utilisateur (`click`, `mouseover`...) appliqué sur une annotation particulière qui est affichée sur un composant d'interface donné. Les événements déclencheurs d'une interaction sont à l'initiative de l'utilisateur et sont représentés par une flèche ayant pour origine l'utilisateur et pour étiquette le nom de l'événement déclenchant l'interaction (Figure 4.25).

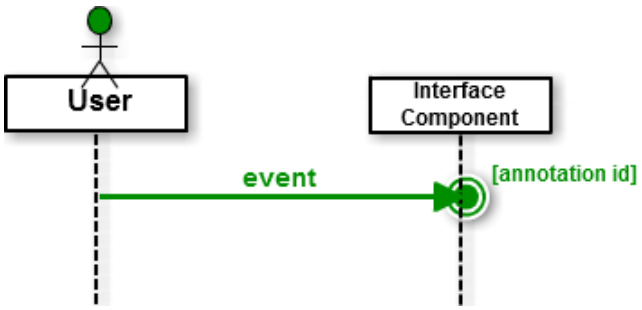


FIGURE 4.25 – Spécification d'une action de sélection de la part de l'utilisateur

La destination de la flèche désigne d'une part l'annotation avec laquelle l'utilisateur souhaite interagir mais aussi le composant d'interface sur lequel cette annotation est affichée. L'annotation, qui dans ce cas devient interactive, est représentée sur le composant d'interface où elle est affichée.

Action de saisie

L'action de saisie permet à l'utilisateur de créer une nouvelle annotation à partir d'un outil disposé dans chaque composant d'interface (par exemple, des boutons d'annotation manuelle dans un composant textuel ou des outils à dessiner des lignes, polygones sur un composant cartographique). Cette action est représentée par une flèche allant de l'utilisateur vers le composant d'interface. La destination de la flèche désigne l'annotation créée par l'utilisateur (Figure 4.26).

Les modalités de la saisie sont définies par le concepteur au niveau du composant d'interface (dans la phase *Interface*). Au niveau interactif, le concepteur ne se soucie plus de la manière dont l'annotation a été saisie par l'utilisateur mais uniquement des réactions du système que cette annotation saisie peut déclencher.

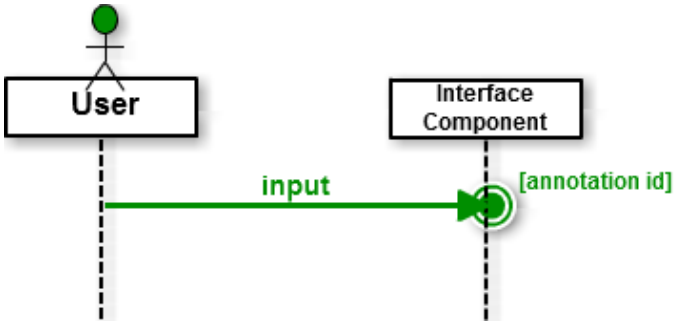


FIGURE 4.26 – Spécification d'une action de saisie de la part de l'utilisateur

B. Spécification d'une réaction externe du système

Les réactions externes du système sont les réactions perceptibles par l'utilisateur. Dans notre cas, elles se traduisent par la modification visuelle d'une annotation présentée sur l'interface. Cette modification est réalisée par le système en appliquant un ou plusieurs effets sur l'annotation à valoriser.

Étant donné qu'une réaction externe provient du système et est perceptible par l'utilisateur, nous la représentons (Figure 4.27) par une flèche ayant pour origine le système (le composant sur lequel se trouve l'annotation) et pour destination l'utilisateur (celui qui perçoit la réaction).

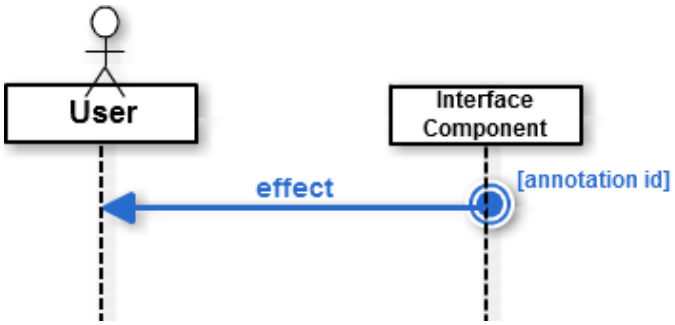


FIGURE 4.27 – Spécification d'une réaction externe du système

De manière plus précise, la flèche prend pour origine l'annotation qui doit être modifiée visuellement et l'étiquette de la flèche précise quel est l'effet qui est appliqué pour mettre en valeur cette annotation.

C. Spécification des réactions internes du système

Le modèle d'interaction proposé dans la section 4.3.1 propose trois types de réactions internes au système : la sélection, la projection et le calcul.

Opération de sélection

L'opération de sélection permet au système de déterminer quelle annotation a été sélectionnée par l'utilisateur parmi toutes les annotations affichées sur un composant d'interface. L'annotation sélectionnée par l'utilisateur devient une annotation à part entière clairement identifiée par le système et qui peut être valorisée dans la suite de l'interaction. L'opération de sélection est représentée graphiquement par une flèche ayant pour origine un ensemble des annotations de départ parmi lesquelles l'utilisateur va faire sa sélection (Figure 4.28). Cet ensemble est rattaché à un composant d'interface donné.

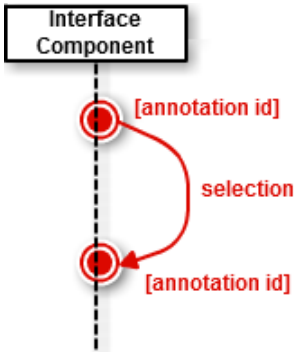


FIGURE 4.28 – Spécification de la sélection (réaction interne du système)

La destination de la flèche désigne l'annotation sélectionnée par l'utilisateur et identifiée par le système. La nouvelle annotation créée appartient par défaut au composant d'interface qui présente l'ensemble des annotations dans lequel la sélection a été réalisée. L'annotation créée peut ensuite être affichée via une réaction externe ou bien transférée sur un autre composant d'interface par une opération de projection par exemple (voir ci-après).

Opération de projection

L'opération de projection consiste à transférer une annotation présente sur un composant d'interface vers un autre composant d'interface. Dans cette opération, le système doit calculer, en cours d'interaction, la représentation de l'annotation transférée vers le composant de destination.

L'origine de la flèche détermine l'annotation à projeter tandis que la destination définit le composant vers lequel l'annotation doit être projetée (Figure 4.29).

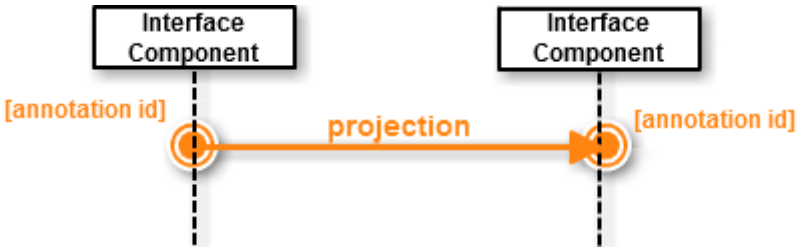


FIGURE 4.29 – Spécification de la projection (réaction interne du système)

Soulignons que l'opération de projection n'affiche pas l'annotation sur le composant d'interface de destination. Elle détermine seulement comment cette annotation doit être représentée : la position ou la valeur de l'annotation dans un composant textuel ou encore les coordonnées géographiques de l'annotation si cette dernière est projetée sur un composant cartographique.

Opération de calcul

Une opération de calcul permet de créer une nouvelle annotation à partir d'une annotation de départ sur laquelle un calcul est appliqué. Du fait que les annotations sont de nature géographique, les opérations autorisées sont elles-mêmes de nature géographique : calcul de distance, d'orientation...

L'opération de calcul (Figure 4.30) est représentée par une flèche portant le nom du service de calcul utilisé.

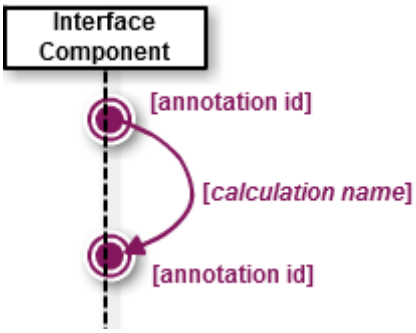


FIGURE 4.30 – Spécification du calcul (réaction interne du système)

Cette flèche a pour origine l'annotation qui sert de point d'entrée au calcul et pour destination l'annotation créée en sortie du calcul. Cette nouvelle annotation est représentée sur le même composant d'interface que celui où se trouve l'annotation de départ.

4.3.2.3 Exemple de mise en œuvre

Le diagramme suivant (Figure 4.31) montre la spécification d'une interaction combinant la plupart des briques du langage visuel proposé.

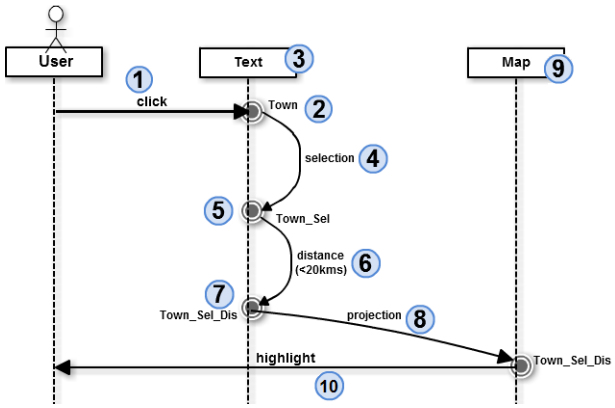


FIGURE 4.31 – Exemple d'un diagramme d'interaction

Le diagramme de la figure 4.31 spécifie l'interaction suivante : "lorsque l'utilisateur clique (1) sur une ville (2) située dans le texte (3) affiché à l'interface, le système identifie (4) la ville sélectionnée (5) puis calcule (6) toutes les villes situées à moins de 20 kms de la ville sélectionnée. Cet ensemble de villes (7) est ensuite transféré (8) sur le composant cartographique (9) puis mis en évidence évidence (10) via un effet de type *highlight*."

Le diagramme présenté sur la figure 4.32 montre un autre exemple d'une interaction déclenchée suite à une saisie de l'utilisateur.

Ce diagramme spécifie l'interaction suivante : "lorsque l'utilisateur saisit un lieu sur la carte (en utilisant un outil mis à disposition sur la carte et défini dans la phase *Interface*), le système calcule le département auquel ce lieu appartient puis il transfère (`projection`) et affiche le nom de ce département sur le composant textuel présent à l'interface."

4.3.2.4 Capacité à décomposer la complexité de l'interaction

Le langage visuel que nous proposons offre une flexibilité d'usage permettant au concepteur de spécifier l'interaction selon différents degrés de complexité. Le langage permet en effet de décomposer une interaction complexe en plusieurs diagrammes simples

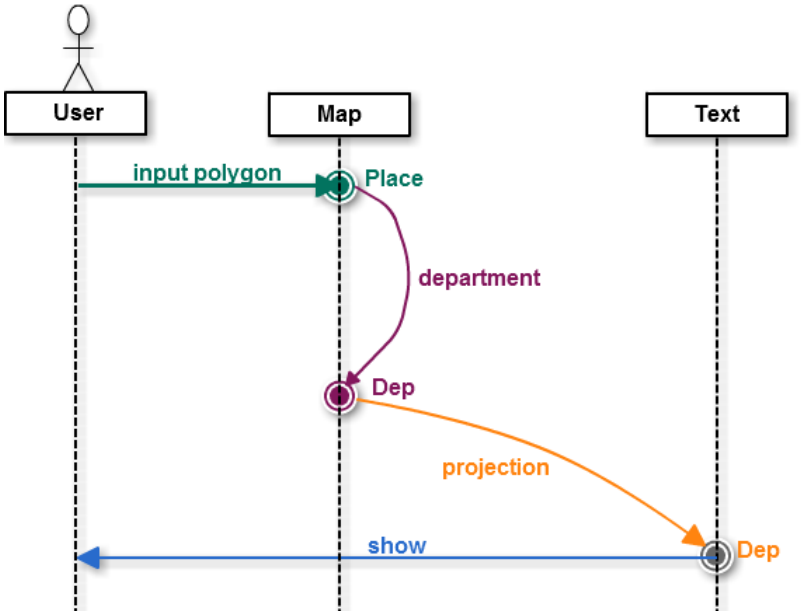


FIGURE 4.32 – Exemple d'un diagramme d'interaction avec une saisie utilisateur

donc le comportement global correspond à celui de l'interaction complexe.

Pour illustrer ce principe de décomposition, nous pouvons considérer un exemple présenté sur la figure 4.33.

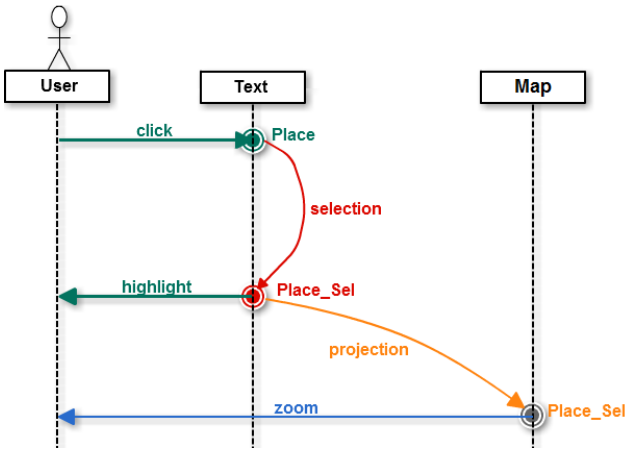


FIGURE 4.33 – Exemple de diagramme d'interaction décomposable

Ce diagramme décrit l'interaction suivante : "lorsque l'utilisateur clique sur un lieu dans le texte, le système identifie le lieu sélectionné, le surligne dans le texte, puis envoie ce lieu sur la carte afin de faire un zoom avant à cet endroit."

Cette interaction peut être décomposée et décrite à l'aide des deux diagrammes présentés sur la figure 4.34 :

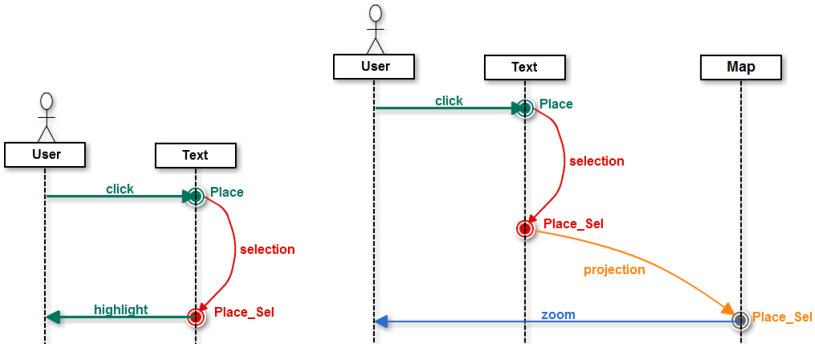


FIGURE 4.34 – Décomposition de l'interaction présentée sur la figure 4.33

- Le diagramme de gauche spécifie que lorsque l'utilisateur clique sur un lieu dans le texte, le système identifie ce lieu et le surligne dans le texte.
- Le diagramme de droite précise que lorsque l'utilisateur clique sur un lieu dans le texte, le système identifie le lieu sélectionné, le projette sur la carte puis effectue un zoom avant sur ce lieu.

Ces deux diagrammes décrivent ensemble un comportement équivalent à celui présenté sur la figure 4.33. La réaction "complexe" du système a simplement été répartie sur deux diagrammes, l'un décrivant le surlignement du lieu dans le texte, l'autre décrivant le zoom avant sur le lieu sur la carte.

Le travail présenté dans [LEMN12] montre, dans le cadre d'une expérimentation, un exemple d'interaction plus complexe décomposée en quatre diagrammes d'interaction différents (*cf.* Figure 5.21 et Figure 5.22 dans la partie 5.2.2.1).

4.3.2.5 Définir des interactions à un niveau "système"

Pour simplifier le travail de conception, nous proposons également de pouvoir définir des interactions au niveau du système dans son ensemble et non pas seulement au niveau de chacun des composants d'interface. Ce niveau d'abstraction doit permettre de créer des interactions sur l'ensemble des composants d'interface du système. Considérons une application affichant une carte et un texte relatant un récit de voyage puis étudions l'interaction suivante : "*lorsque l'utilisateur clique sur une ville, toutes les villes (à la fois*

dans le texte et sur la carte) situées à moins de 10 kms (de la ville sélectionnée) sont mises en évidence."

Étant donné que des villes peuvent être sélectionnées à la fois sur le composant texte et sur le composant carte, nous proposons de "factoriser" cette propriété en associant l'annotation cliquable à un niveau "système" et non plus au niveau de chaque composant d'interface (Figure 4.35).

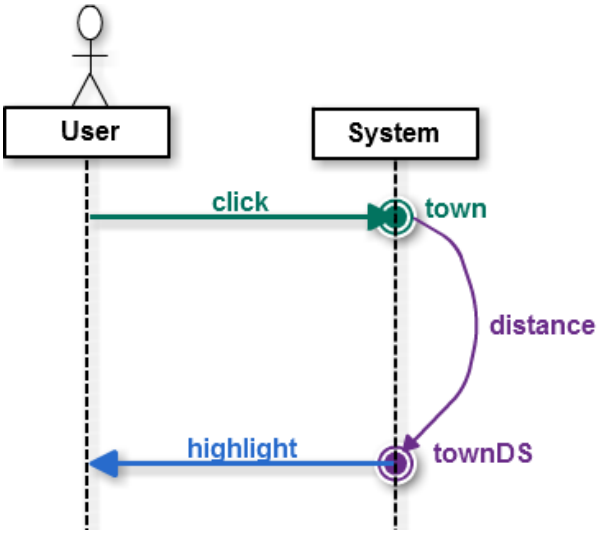


FIGURE 4.35 – Description d'interactions à un niveau "système"

Définir des interactions à un niveau système suppose aussi de pouvoir décomposer ce diagramme pour revenir à un diagramme de niveau composant, afin d'être en mesure de générer le code sous-jacent. La figure 4.36 montre une manière de décomposer le diagramme d'interaction précédent en deux diagrammes.

Ce type d'interaction suppose que :
- l'annotation déclenchant l'interaction est représentable sur au moins un composant d'interface ;
- l'annotation présentée à l'utilisateur, à l'issue de l'interaction, est représentable sur au moins un composant d'interface ;
- l'effet appliqué sur l'annotation renvoyée à l'utilisateur est un effet applicable sur au moins un composant d'interface qui est capable de le réaliser. Par exemple, le "surlignage" ou le zoom sont des effets qui peuvent être appliqués à la fois sur une annotation affichée dans un composant textuel ou cartographique. Il y a donc un sens à vouloir manipuler ces effets à un niveau système. A l'inverse, une mise en italique ne pourra être appliquée que sur une annotation présentée dans un com-

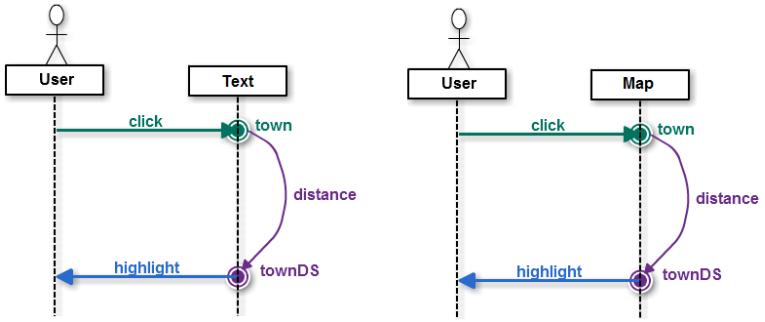


FIGURE 4.36 – Décomposition d'interactions définies à un niveau "système"

posant textuel tandis qu'un affichage en mode satellite ne pourra être appliqué que sur une annotation présentée dans un composant cartographique. Manipuler ces effets à un niveau système ne présente donc pas d'intérêt dans ces deux cas.

Nous proposons également de décrire une interaction à la fois à un niveau système et à un niveau composant d'interface. L'idée est de pouvoir définir les annotations :

- déclenchant une interaction au niveau composant ou bien au niveau système dans sa globalité ;
- présentées suite à une interaction au niveau composant ou bien au niveau système dans sa globalité.

La figure 4.37 illustre un premier exemple de comportement suivant : "*Lorsque l'utilisateur clique sur une ville (que ce soit sur le texte ou la carte), le système surligne sur la carte les villes situées à moins de 10 kms*".

Enfin la figure 4.38 illustre un second exemple de comportement : "*Lorsque l'utilisateur clique sur une ville de la carte, le système surligne les villes situées à moins de 10 kms (à la fois sur le texte et sur la carte)*".

4.3.2.6 Dépendance temporelle entre interactions

Un diagramme permet de décrire les réactions du système suite à une action de l'utilisateur. L'ensemble des diagrammes d'interaction décrivent ainsi que les possibilités interactives offertes à l'utilisateur et les réactions qu'elles engendrent sur le système. Finalement, l'application créée présente à l'utilisateur l'ensemble des annotations avec lesquelles il peut interagir et ces interactions peuvent engendrer l'apparition de nouvelles annotations. Les annotations ne sont donc pas toutes présentées en même temps sur l'interface de l'utilisateur.

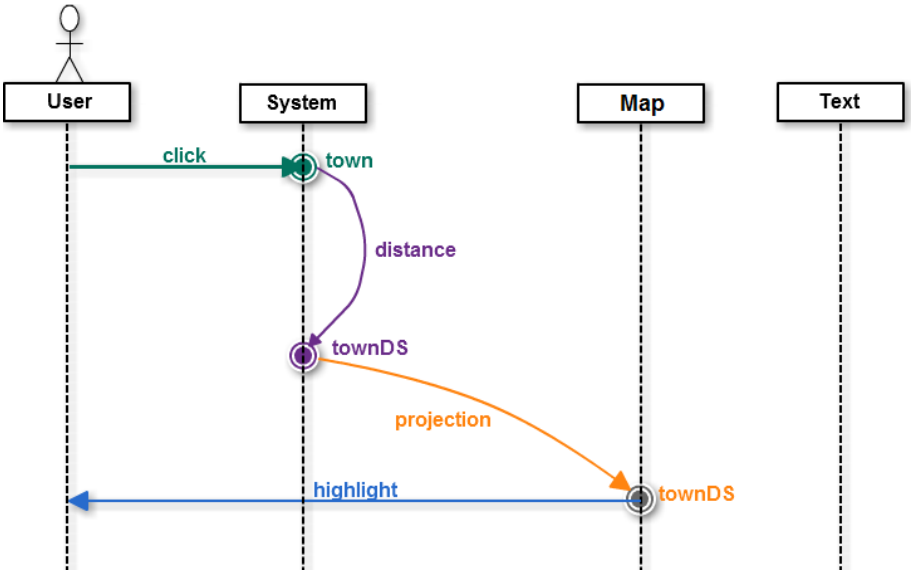


FIGURE 4.37 – Un exemple de spécification d'interaction multi-niveaux

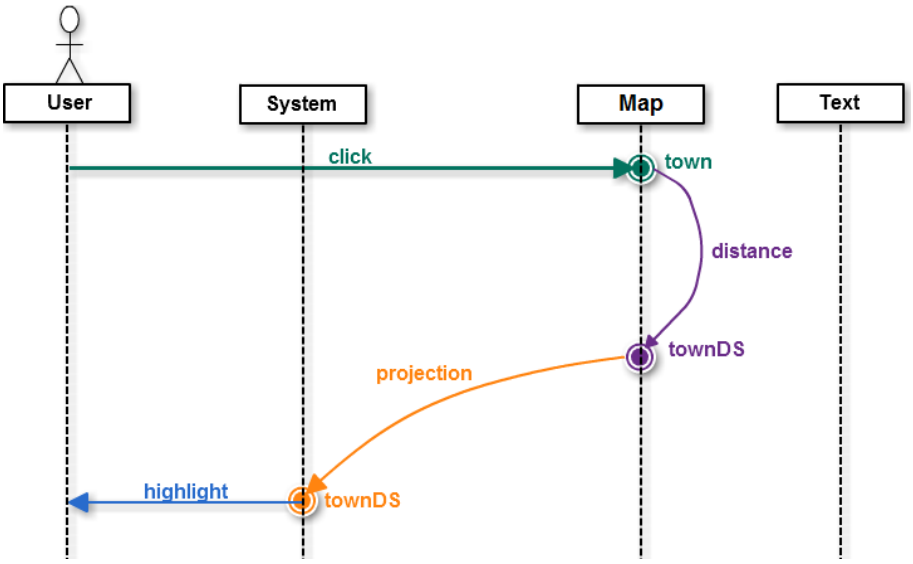


FIGURE 4.38 – Un autre exemple de spécification d'interaction multi-niveaux

Nous ne pouvons pas prédire, *a priori*, l'ordre dans lequel les interactions seront déclenchées car cet ordre dépend des actions de l'utilisateur. Il est cependant possible d'exprimer des contraintes d'enchaînement entre certaines interactions car une interaction n'est déclenchable que si l'annotation interactive permettant de déclencher (celle qui permet d'initier l'interaction) est présentée à l'utilisateur.

Cette règle est *a priori* vraie pour l'ensemble des applications interactives. Dans un traitement de texte par exemple, l'utilisateur ne pourra pas paramétrer l'impression d'un document s'il n'a pas précédemment demandé à imprimer ce document. En fait la possibilité de paramétrage de l'impression n'est possible que lorsque la boîte de dialogue de paramétrage est présentée à l'utilisateur.

Ces contraintes d'enchaînements existent également dans les interactions que nous décrivons à l'aide de nos diagrammes : une interaction ne sera déclenchable par l'utilisateur que si l'annotation permettant de déclencher cette interaction est présentée dans un composant d'interface. Les diagrammes suivants (Figure 4.39) illustrent ce cas.

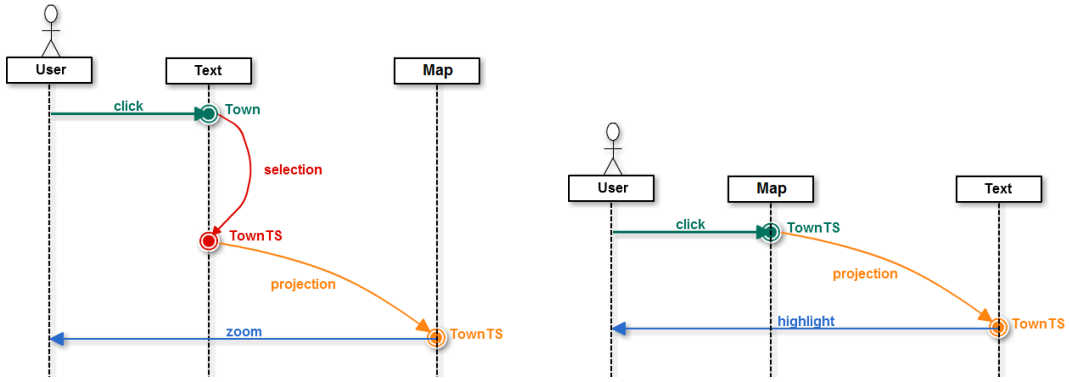


FIGURE 4.39 – Exemple de dépendance temporelle entre interactions

Le diagramme de gauche (Figure 4.39) décrit l'interaction suivante : "lorsque l'utilisateur clique sur une ville dans le texte, la carte fait un zoom avant sur la ville sélectionnée". La ville zoomée correspond à une annotation (notée `TownTS` sur le diagramme) qui apparaît à l'écran en fin d'interaction.

Cette même annotation (`TownTS`) sert d'élément déclencheur pour une seconde interaction décrite sur le diagramme à droite (Figure 4.39). Ceci signifie que cette deuxième `interaction` ne pourra avoir lieu que si l'interaction précédente a déjà eu lieu et que l'annotation déclencheur `TownTS` a déjà été créée et présentée à l'utilisateur.

Ainsi, en jouant sur la mise à disposition ou sur le retrait d'annotations interactives, il devient possible de définir certaines contraintes d'enchaînements sur l'ensemble des interactions intégrées dans l'application construite. Cependant, la mise en place de ces contraintes n'est pas forcément une tâche facile car elle impose au concepteur d'avoir une vue d'ensemble sur les interactions décrites.

4.4 Exécutabilité de la phase *Interaction*

4.4.1 Opérationnalisation des interactions via l'API WIND

L'API WIND présentée en section 3.6.1 intègre un ensemble d'objets et de méthodes spécialisés permettant de programmer des interactions sur des annotations affichées au sein d'une application géographique. La programmation de ces interactions est réalisée en considérant qu'une application interactive présente une interface composée d'annotations pouvant déclencher des réactions de la part du système. D'un point de vue "bas niveau", une interaction est définie par une zone d'écran qui, sous une action spécifique de l'utilisateur, va provoquer une réaction du système.

Comme précisé précédemment, une réaction du système s'achève toujours par un effet perceptible par l'utilisateur. Au niveau de l'API WIND, cet effet se traduit par la modification d'une zone d'écran qui mettra en valeur une annotation. Conformément à ce que nous avons décrit dans la section 4.3.2.6, cette annotation mise en valeur peut à son tour devenir sensible et initier de nouvelles possibilités interactives pour l'utilisateur.

L'API WIND propose donc des objets spécialisés permettant au programmeur de définir des annotations, d'y associer des événements déclencheurs et des réactions du système. Chacun de ces objets dispose de méthodes spécifiques permettant de le manipuler. Le principe consiste à créer des objets `GUIComponent` (composants d'interface), `Annotation`, des objets `Reaction` puis des objets `Interaction` qui associeront un objet `Annotation`, `UserEvent` et un ou plusieurs objets `Reaction`.

Pour illustrer les possibilités de cette API, nous proposons de reprendre l'exemple illustré sur la figure 3.26. Pour rappel, le programmeur avait créé, en utilisant l'API WIND, plusieurs composants d'interface et plusieurs annotations via le code présenté ci-après (Figure 4.40).

```
1  // Création de l'interface de l'application
2  var mydoc = new WIND.GUI("main", {"title": "Mon Application",
       "description": "..."});
3
4  // Création du composant Texte
5  var t = mydoc.createDisplayer('text', {'top': 100, 'left': 10,
       'width': 500, 'height': 200, 'draggable': false, 'resizable'
       : false, 'color': '#0033CC', 'border': '#0033CC 2px solid',
       'header': false, 'removable': false,'configurable': false});
6  var p = t.createParagraph();
7  p.setContent("Durant l'été 2011, je suis parti pour
       Mauléon-Licharre le 14 juillet 2011. Je suis rentré à
       Bayonne deux jours après.");
8
9  // Création du composant Carte
10 var m = mydoc.createDisplayer('map', {'top': 100, 'left': 520,
       'width': 600, 'height': 450, 'name': "toto", 'type': 'Google
       Street', 'longitude': -0.32, 'latitude': 43.45, 'zoom': 8,
       'draggable': false, 'resizable' : false, 'color': '#FF9900',
       'border': '#FF9900 2px solid', 'header': false, 'removable':
       false, 'configurable': false});
11
12 // Création du composant Frise chronologique
13 var tl = mydoc.createDisplayer('timeline', {top:310, left:10,
       width:500, height:240, color: "#FF6600", border: "#FF6600
       1px solid", 'header': false, 'removable':
       false,'configurable': false});
14
15 // Création des annotations
16 var annot1 = t.createAnnotation("Town", "Mauléon-Licharre", 1,
       10, 10);
17 var annot2 = t.createAnnotation("Town", "Bayonne", 1, 20, 20);
18
19 var annot3 = m.createAnnotation("Town", "Mauléon-Licharre",
       MULTIPOLYGON(((...))));
20 var annot4 = m.createAnnotation("Town", "Bayonne",
       MULTIPOLYGON(((...))));
21
22 var annot5 = tl.createAnnotation("Voyage", "parti pour
       Mauléon-Licharre", "14/07/2011", "14/07/2011");
```

FIGURE 4.40 – Création des composants d'interface et des annotations de l'application

Pour rendre interactive son application, le programmeur peut créer des objets `Reaction` en associant un effet visuel à appliquer sur des annotations précédemment créées (Figure 4.41).

```
23 var r1 = new WIND.Reaction(annot1, 'highlight');
24 var r2 = new WIND.Reaction(annot3, 'zoom');
```

FIGURE 4.41 – Création de réactions via l'API WIND

Les réactions codées sur la figure 4.41 sont les suivantes :
- L'objet réaction `r1` correspond à un surlignage de l'annotation `annot1` (ligne 16) dans le texte (le mot *Mauléon-Licharre*) ;
- L'objet réaction `r2` correspond à un zoom sur l'annotation `annot3` (ligne 19) située sur la carte qui correspond la ville de Mauléon-Licharre.

Pour définir une interaction, le programmeur doit ensuite créer un objet interaction et l'activer (Figure 4.42). La création d'un objet interaction consiste à définir l'annotation servant de déclencheur, l'événement auquel cette annotation doit réagir et les réactions à déclencher. La ligne de code 25 crée une nouvelle interaction qui sera déclenchée au clic sur l'annotation `annot1` précédemment créée, c'est-à-dire lorsque l'utilisateur clique sur le mot "*Mauléon-Licharre*" dans le texte. La ligne suivante ajoute à l'objet interaction précédemment créé, les réactions à déclencher (réactions `r1` *puis* `r2` définies précédemment sur la figure 4.41). La dernière ligne de code active l'interaction et la rend effective.

```
25 var i1 = new WIND.Interaction(annot1, 'click', null);
26 i1.addReaction(r1);
27 i1.addReaction(r2);
28 i1.activate();
```

FIGURE 4.42 – Création d'interactions via l'API WIND

L'API WIND propose donc au programmeur des objets et méthodes spécialisés pour implanter des interactions sous forme d'annotations déclenchant des réactions suite à un événement utilisateur. L'intérêt de l'API est d'offrir un premier niveau d'abstraction permettant de décrire des interactions de manière uniforme, quelque soit le composant d'interface sur lequel elles sont implantées.

L'API reste une contribution utile pour des programmeurs et elle devient accessible pour des non-informaticiens via l'environnement de conception visuel WINDMash qui supporte notre langage visuel.

4.4.2 Opérationnalisation des interactions au sein de WINDMash

Pour faciliter la conception des interactions, nous avons intégré dans l'environnement WINDMash (*cf.* Annexe F pour plus de détails) le langage visuel proposé dans la section 4.3.2.2 (Figure 4.43). L'exemple traité est celui de la section 4.3.2.4 (Figure 4.33) que nous rappelons : "*lorsque l'utilisateur clique sur un lieu dans le texte, le système identifie le lieu sélectionné, le surligne dans le texte, puis envoie ce lieu sur la carte afin de faire un zoom avant à cet endroit.*"

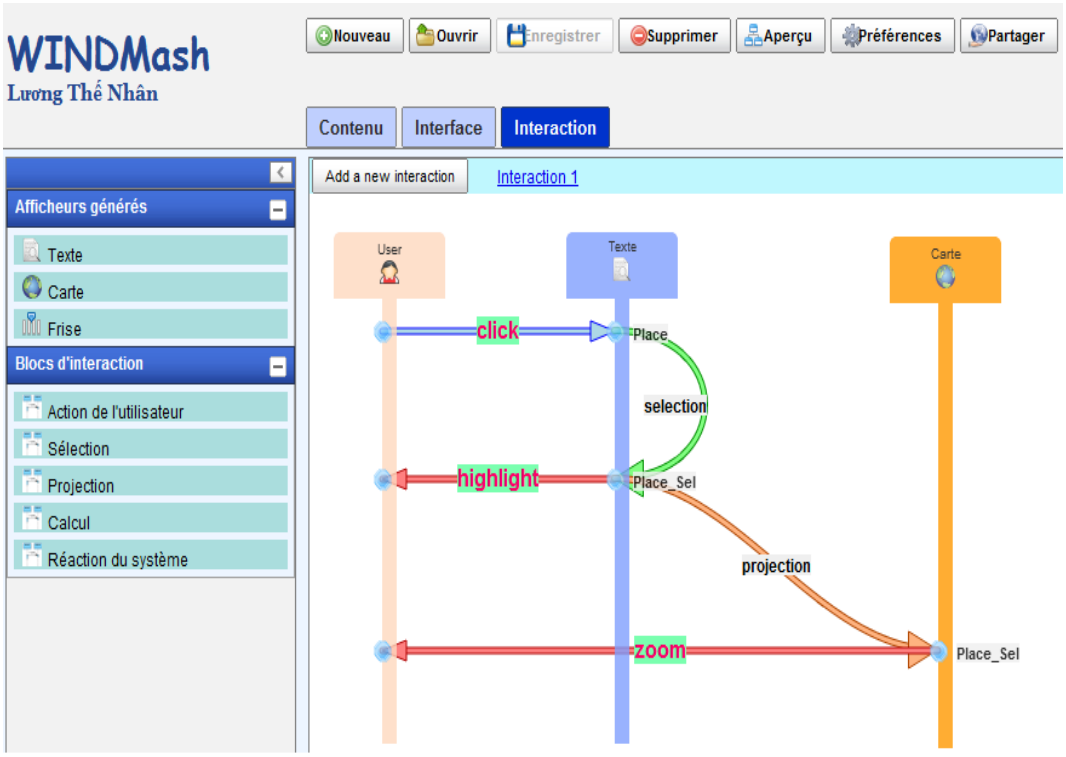


FIGURE 4.43 – Spécification graphique d'une interaction dans WINDMash

L'espace de conception est divisé en deux zones :

- une zone à gauche qui présente dans sa partie basse l'ensemble des composants d'interface sur lesquels le concepteur va pouvoir définir des interactions à partir de cinq blocs disponibles (ceux présentés en section 4.3.2.2). Cette zone présente aussi dans sa partie haute les différents composant d'interface (les afficheurs) qui ont été créés/paramétrés dans la phase Interface ;
- une zone de travail centrale dans laquelle le concepteur va spécifier chacune de ses interactions via le langage visuel que nous avons proposé.

Comme pour les autres espaces de travail de WINDMash, la spécification d'une interaction est réalisée graphiquement par des opérations de glisser-déposer visant à assembler des briques de notre langage visuel. Lors de la spécification d'une action utilisateur, d'une réaction interne ou externe sur un composant d'interface, l'environnement présente au concepteur la liste des annotations présentes sur ce composant [51] et demande à l'utilisateur quelles sont les annotations qu'il souhaite impliquer dans l'interaction en cours de définition (Figure 4.44). Selon les spécifications de l'interaction qu'il veut décrire avec le langage visuel, le concepteur fait successivement glisser-déposer des blocs disponibles dans le menu de gauche et renseigne leurs paramètres.

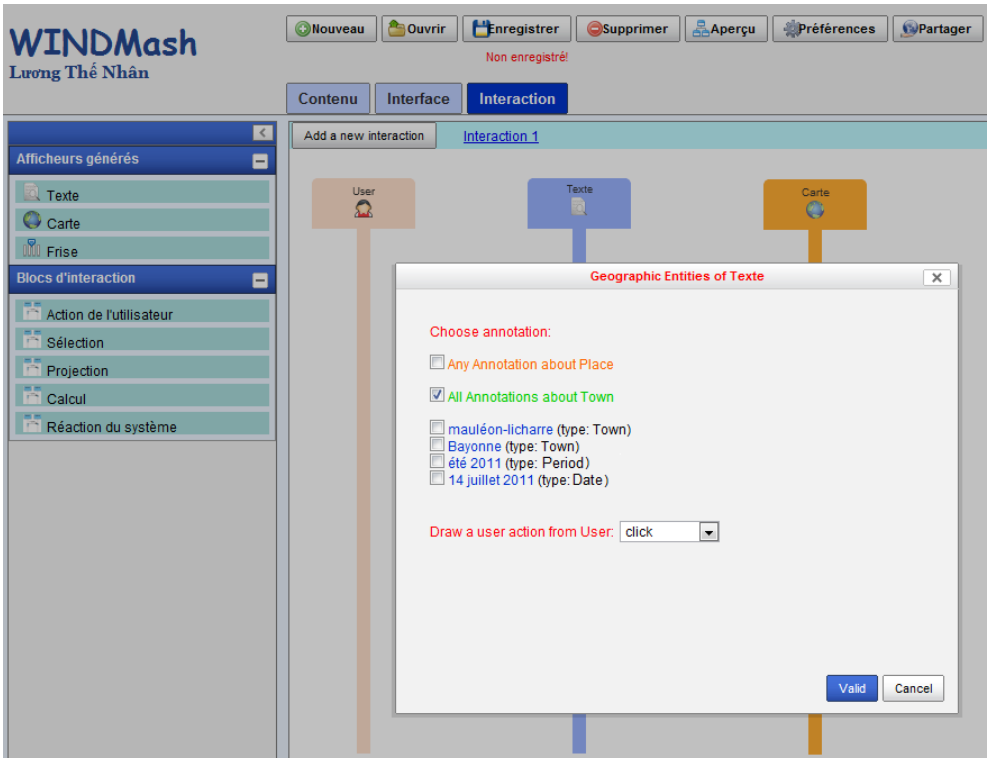


FIGURE 4.44 – Choix de l'annotation sur laquelle porte l'action utilisateur

Par exemple, la figure 4.44 illustre une spécification de l'action d'utilisateur concernant un clic (gauche) sur n'importe quelle ville affichée dans le composant `Texte`. Lorsque le concepteur valide cette spécification (avec le bouton "Valid", une flèche (avec le label "Click") est automatiquement ajoutée entre les lignes de vie `User` et `Texte` décrivant cette action utilisateur. Ensuite, le concepteur choisit deux réactions internes :

– une "*Sélection*" (pour préciser que le clic concerne la ville qui a été sélectionnée (par le clic) et une "*Projection*" sur le composant `Texte` (afin que la ville cliquée

51. Ceci constitue un premier niveau d'assistance au concepteur.

sur le composant `Texte` puisse être manipulée sur le composant cartographique), et
- deux "*Réaction du système*" : une mise en surbrillance dans le composant `Texte` de la ville cliquée (`highlight`) ainsi qu'un zoom avant dans le composant `Carte` (`zoom`).

Ceci permet ainsi de construire le diagramme de la figure 4.43 qui représente le comportement décrit en début de section. Notons toutefois que l'autre choix possible concernant l'action utilisateur est le survol sur un contenu. De plus, il est possible de préciser que l'interaction concerne non plus un clic (ou survol) sur n'importe quelle ville, mais sur n'importe quel lieu (détecté/identifié auparavant dans la phase *Contenu*) ou alors un contenu précis parmi les quatre détectés/identifiés auparavant dans la phase *Contenu* (la ville de "mauléon-licharre", la ville de "Bayonne", la période "été 2011" ou la date "14 juillet 2011").

Comme dans les étapes de conception précédentes (section 3.6.2), la spécification graphique d'une interaction est ensuite traduite au format RDF (Figure 4.45).

La description RDF de la figure 4.45 décrit une interaction (`I1` - ligne 3). Cette interaction se compose d'une action de sélection (lignes 5 - 10) : lorsque l'utilisateur clique sur une annotation dans `Displayer1` (ligne 7), le système va déclencher une séquence de réactions (lignes 8 et 13). Cette séquence contient quatre réactions :
- la sélection (lignes 15 - 21) permet de récupérer l'annotation sur laquelle l'utilisateur vient de cliquer ;
- la projection (lignes 24 - 32) effectue une copie de l'annotation cliquée depuis un afficheur (`Displayer1`) vers un autre (`Displayer2`) ;
- la réaction (lignes 35 - 39) permet de mettre en évidence (`highlight`) de l'annotation cliquée sur l'afficheur `Displayer1` ;
- la réaction (lignes 42 - 46) permet à l'afficheur `Displayer2` d'effectuer un zoom avant (`zoom`) sur l'annotation copiée ;

Rappelons qu'il existe également par ailleurs, suite au résultat de la phase *Interface* une description RDF précisant que le (`Displayer1`) est un afficheur textuel et que le (`Displayer2`) est un afficheur cartographique.

De même, les contenus géographiques manipulés ont été définis dans la phase *Contenu* et ont donné lieu à une autre description RDF (qu'il est possible d'utiliser en phase *Interface* pour lier ces contenus et afficheurs (*cf.* section 3.6.2).

```
<rdf:RDF xmlns:wind="http://erozate.iutbayonne.univ-pau.fr/wind#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xml:base="http://erozate.iutbayonne.univ-pau.fr/Nhan/windmash/uBWjWZcKau">
  <wind:Interaction rdf:about="#I1">
    <wind:event>
      <wind:SelectEvent rdf:about="#I1-event">
        <wind:event_type>click</wind:event_type>
        <wind:over rdf:resource="#displayer1-annotation" />
        <wind:trigger rdf:resource="#I1-reaction" />
        <wind:via rdf:resource="#displayer1" />
      </wind:SelectEvent>
    </wind:event>
    <wind:reaction>
      <rdf:Seq rdf:about="#I1-reaction">
        <rdf:li>
          <wind:Selection rdf:about="#I1-R2">
            <wind:on rdf:resource="#displayer1" />
            <wind:result>
              <wind:Annotation rdf:about="#displayer1-annotation-selection"
                  />
            </wind:result>
            <wind:source rdf:resource="#displayer1-annotation" />
          </wind:Selection>
        </rdf:li>
        <rdf:li>
          <wind:Projection rdf:about="#I1-R4">
            <wind:depend rdf:resource="#I1-R2" />
            <wind:on rdf:resource="#displayer1" />
            <wind:result>
              <wind:Annotation
                  rdf:about="#displayer1-annotation-selection-projection2"
                  />
            </wind:result>
            <wind:source rdf:resource="#displayer1-annotation-selection" />
            <wind:target rdf:resource="#displayer2" />
          </wind:Projection>
        </rdf:li>
        <rdf:li>
          <wind:ExternalReaction rdf:about="#I1-R3">
            <wind:depend rdf:resource="#I1-R2" />
            <wind:effect_type>highlight</wind:effect_type>
            <wind:over rdf:resource="#displayer1-annotation-selection" />
          </wind:ExternalReaction>
        </rdf:li>
        <rdf:li>
          <wind:ExternalReaction rdf:about="#I1-R5">
            <wind:depend rdf:resource="#I1-R4" />
            <wind:effect_type>zoom</wind:effect_type>
            <wind:over
                rdf:resource="#displayer1-annotation-selection-projection2"
                />
          </wind:ExternalReaction>
        </rdf:li>
      </rdf:Seq>
    </wind:reaction>
  </wind:Interaction>
</rdf:RDF>
```

FIGURE 4.45 – Extrait RDF/XML correspondant à l'interaction décrite sur la figure 4.43

Par conséquent, la description RDF de l'interaction correspond à un cahier des charges structuré de l'interaction que le système doit rendre exécutable. L'opérationnalisation de chaque interaction est ensuite réalisée en parcourant les différents fichiers RDF générés puis en générant les objets JavaScript correspondant via l'API WIND précédemment présentée. Les concepts du modèle d'interaction, qui ont été décrits dans la description de la figure 4.45 sont créés comme des objets (instances) des classes implémentées dans l'API WIND (`WIND.SelectEvent`, `WIND.Interaction`, `WIND.Selection`, `WIND.Projection`, `WIND.ExternalReaction`). Le code JavaScript généré qui est interprété par un navigateur Web devient alors très synthétique (Figure 4.46).

```
1  var I1_event = new WIND.SelectEvent("click", new
       Array(annot1,annot2));
2  I1_event.trigger(function(evt){
3    var I1_R1 = new WIND.Selection(new Array(annot1,annot2));
4    I1_R1.result = evt.annotationSelected;
5    var I1_R2 = new WIND.Projection(I1_R1.result, displayer2);
6    I1_R2.setDependency(I1_R1);
7    var I1_R3 = new WIND.ExternalReaction(I1_R1.result,
       "highlight");
8    I1_R3.setDependency(I1_R1);
9    var I1_R4 = new WIND.ExternalReaction(I1_R2.result, "zoom");
10   I1_R4.setDependency(I1_R2);
11   var I1= new WIND.Interaction(evt, new
       Array(I1_R1,I1_R2,I1_R3,I1_R4));
12   I1.activate();
13 });
```

FIGURE 4.46 – Extrait du code JavaScript correspondant à l'interaction décrite sur la figure 4.43

Cette génération automatique de code permet ainsi au concepteur d'exécuter son application, d'avoir un retour immédiat sur les interactions spécifiées et, en cas d'insatisfaction, de pouvoir revenir en phase de conception pour modifier ses diagrammes d'interaction.

4.5 Bilan, limites

Dans ce chapitre, nous avons présenté un état des l'art des modèles et langages visuels permettant de prendre en compte l'interaction dans la phase de conception d'une application. Nous avons mis en évidence la richesse de ces modèles mais aussi leur complexité dès lors que la conception était menée par des non-informaticiens qui veulent créer des applications simples mais avec un degré d'interactivité non trivial.

Nous avons proposé un modèle permettant de décrire l'interaction à un niveau dialogue afin que le concepteur puisse spécifier ce que l'utilisateur peut faire ainsi que les réactions engendrées par le système. Cette spécification s'appuie sur les modèles de contenu et d'interface (présentés dans le chapitre 3) pour permettre au concepteur de décrire l'interaction en s'appuyant sur des éléments concrets situés à un niveau interfacique. La spécification de l'interaction consiste alors à décrire ce que l'utilisateur peut faire par rapport aux contenus et aux zones d'interface qui lui sont présentés puis à décrire la manière dont ces contenus et ces composants d'interface sont modifiés lorsque le système réagit. Les actions utilisateur considérées restent simples et n'imposent pas au concepteur de raisonner à un niveau tâche.

Pour faciliter l'usage de ce modèle d'interaction, nous avons proposé un langage visuel composé de briques simples. Ces briques peuvent être combinées pour spécifier ce que l'utilisateur peut faire par rapport à l'interface affichée et par rapport aux contenus présentés mais aussi pour décrire l'enchaînement des réactions possibles du système. Le langage permet de décomposer les comportements interactifs d'une application en plusieurs interactions dont la complexité dépend du niveau de maîtrise du concepteur. La conception de l'interaction est ainsi facilitée dans le sens où chaque diagramme décrit un comportement interactif de l'application.

Nous avons outillé le modèle et le langage d'interaction avec une API JavaScript (WIND) intégrant des classes et des méthodes permettant d'implanter des interactions en termes d'actions de l'utilisateur et de réactions du système. L'API WIND intègre une couche d'abstraction permettant au programmeur d'implanter des interactions de manière uniforme, quelques soient les contenus mis en jeu et quelques soient les composants d'interface où ces contenus sont affichés. Implantée selon le paradigme orienté objet, l'API WIND permet au programmeur de créer des objets `interaction` disposant de méthodes permettant d'y associer des actions de l'utilisateur, des réactions du système... Cette couche objet spécialisée permet d'implanter des interactions selon un code à la fois concis, lisible et uniforme.

Le modèle et le langage visuel d'interaction ont été intégrés dans l'environnement de conception WINDMash pour permettre au concepteur d'implanter des interactions de manière graphique. La spécification des interactions est réalisée en combinant graphiquement les cinq éléments de base du langage définissant des actions de l'utilisateur, des réactions externes du système et diverses réactions internes possibles. La spécification graphique de chaque interaction est ensuite traduite selon le formalisme RDF pour servir de point d'entrée à un générateur de code qui génère le code exécutable correspondant en s'appuyant sur les classes et méthodes spécialisées disponibles dans l'API WIND.

Nos quatre contributions peuvent être illustrées par la figure 4.47. Elles se situent à différents niveaux d'abstraction pour lesquels les concepts sont pris en compte à des degrés divers précisés dans les prochains paragraphes.

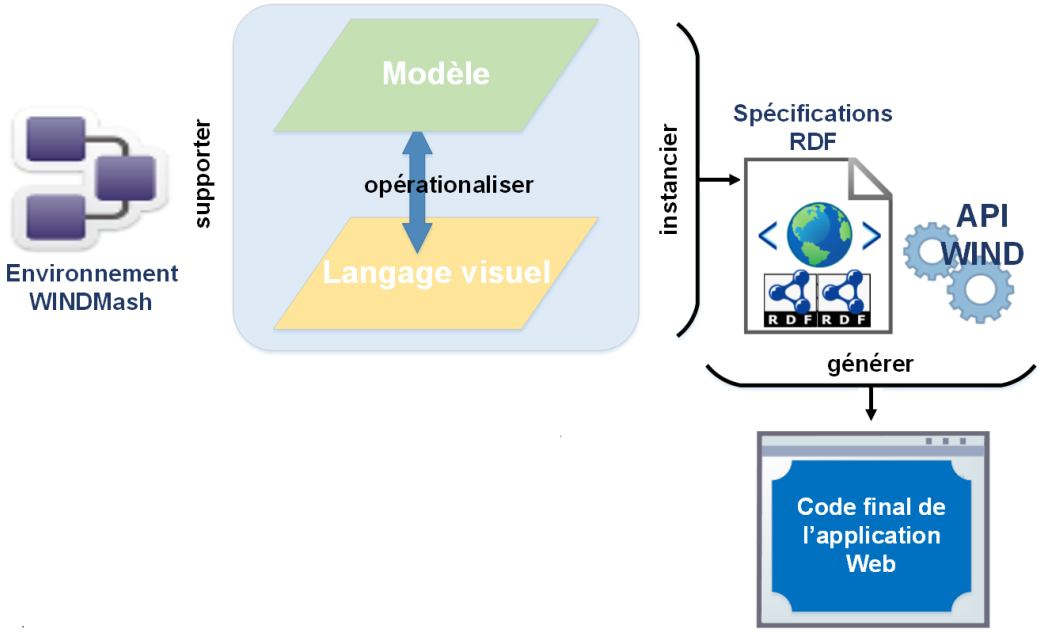


FIGURE 4.47 – Schéma des contributions relatives aux interactions

Parmi nos propositions, le modèle d'interaction reste, par son niveau d'abstraction, la contribution qui propose les concepts les plus nombreux pour décrire l'interaction. Ces concepts offrent un cadre de réflexion pour spécifier les interactions d'une application mais, par manque de temps, tous ces concepts n'ont pas pu être opérationnalisés / intégrés / outillés au niveau du langage visuel, de l'API WIND ou de l'environnement WINDMash. La figure 4.48 résume les concepts pris en compte dans les différentes contributions. La légende de la figure est la suivante :

- La lettre **L** signifie que le concept est pris en compte au niveau du langage visuel ;
- La lettre **A** signifie que le concept est intégré au niveau de l'API WIND (il existe une classe ou une méthode spécialisée pour gérer le concept) ;
- La lettre **W** signifie que le concept est pris en compte au niveau de l'environnement WINDMash (le concepteur peut manipuler graphiquement le concept); la lettre **w** signifie que le concept est en cours d'implémentation au niveau de WINDMash.

Comme présenté dans la figure 4.48, la plupart des concepts du modèle d'interaction ont été opérationnalisés au niveau du langage visuel, de l'API WIND et de l'environnement WINDMash.

Le langage visuel permet au concepteur de manipuler la plupart des concepts proposés dans le modèle. Les cinq briques de base du langage permettent de manipuler les concepts primordiaux du modèle d'interaction. L'API WIND dispose d'un ensemble de

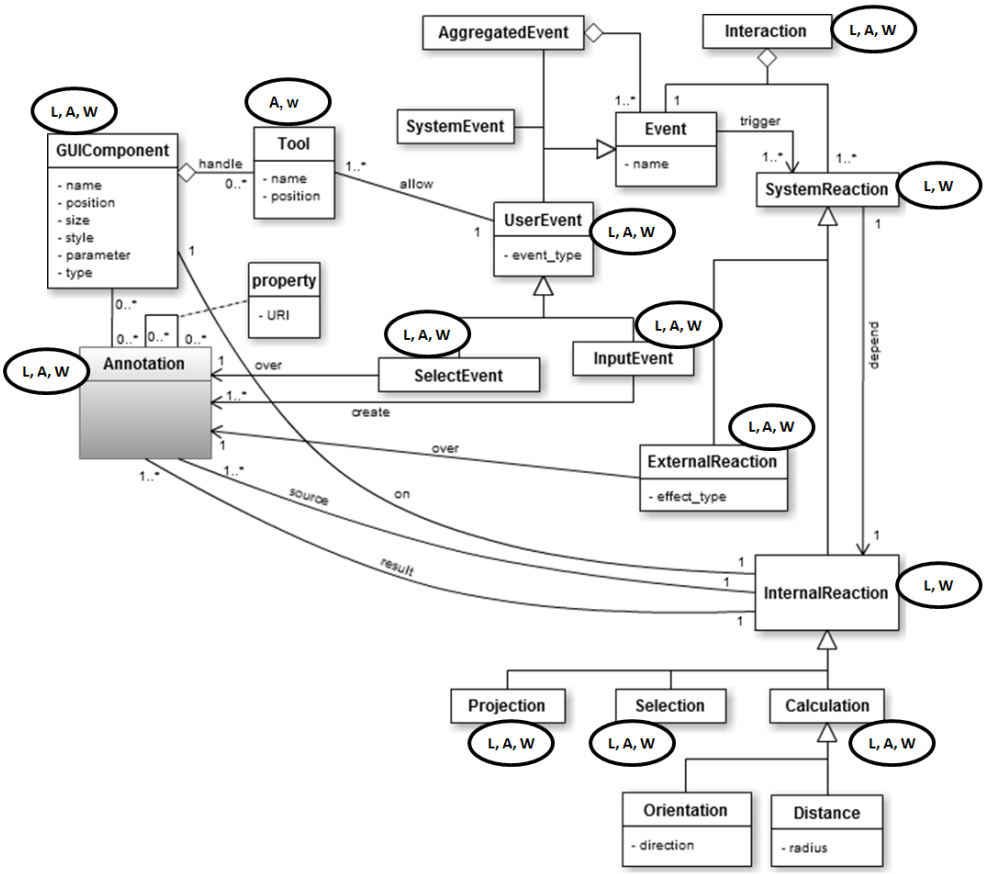


FIGURE 4.48 – Degré d'opérationnalisation du modèle d'interaction

classes et de méthodes permettant d'implanter les principaux éléments d'une interaction. La distinction entre les réactions externes et les réactions internes du système a été implantée au niveau de l'API. Dans WINDMash, les concepts du modèle d'interaction ne sont manipulables qu'au travers du langage visuel que nous proposons. Toutes les briques du langage visuel ont été implantées dans WINDMash ; de ce fait les concepts d'interaction manipulables dans WINDMash sont identiques à ceux manipulables via le langage visuel.

Dans le cadre de cette thèse, l'accent a été mis sur les interactions centrées utilisateur. Comme présenté dans la figure 4.48, les interactions déclenchées via des événements système n'ont pas été opérationnalisées et n'ont dont pas été prises en compte au niveau de l'API et du langage visuel. De même, le concept d'évènements agrégés capables de déclencher des réactions suite à une succession d'événements (utilisateur et/ou système) reste un travail à réaliser et à opérationnaliser.

La capacité du langage visuel à décrire les comportements interactifs d'une application à l'aide de plusieurs diagrammes reste un atout lorsque la dimension interactive d'une application est complexe. L'effet de bord de cette capacité de décomposition est que le concepteur ne dispose pas d'une vue globale des capacités interactives de son application et, de ce fait, il devient difficile de vérifier la consistance de l'ensemble des interactions décrites. Nous pensons qu'il est possible d'obtenir une vue globale des comportements interactifs de l'application en fusionnant l'ensemble des diagrammes produits pour obtenir un diagramme similaire aux diagrammes états-transitions UML. Cette possibilité n'a pas été étudiée dans le cadre de cette thèse mais nous pensons qu'elle représente une piste intéressante pour mettre en place des méthodes et des outils de contrôle. Dans l'état actuel, le concepteur conçoit son application par un processus itératif d'essais - erreurs jusqu'à obtenir un comportement interactif répondant à ses attentes. En cas d'erreur ou en cas de comportement inattendu, le modèle, le langage visuel ou l'environnement WINDMash ne proposent, à ce jour, aucune assistance approfondie au concepteur. Notons toutefois que l'environnement-auteur n'autorise pas les expressions atomiques inconsistances à l'instar d'un atelier de génie logiciel interdisant de spécifier qu'une classe hérite d'elle-même. En ce sens, WINDMash offre un certain guidage par l'outil.

Le chapitre 5 de cette thèse présente quelques perspectives pour répondre aux limites que nous venons de citer dans cette section.

Chapitre 5

Bilan et perspectives

Sommaire

Les contributions scientifiques présentées dans ce manuscrit ont été décrites en plusieurs parties. Suite à un travail mené avec des enseignants de terrain de l'école élémentaire, nous avons identifié les besoins pour une plateforme de conception dont les

fondements et les phases ont été décrites dans les chapitres 2, 3 et 4. Ici, nous rappelons tout d'abord les contributions de la thèse (Section 5.1) avant de présenter les premiers résultats des évaluations qui ont été lancées (Section 5.2) et les perspectives de ces travaux opérationnalisés dans la plateforme WINDMash (Section 5.3).

5.1 Rappel des contributions de la thèse

Dans cette section, nous revenons sur notre problématique initiale, à savoir : "*permettre à un concepteur non-informaticien de construire en autonomie des applications Web géographiques*". Nous rappelons en quoi nos contributions apportent des solutions à cette problématique, et quelles en sont les limites.

Pour répondre à la problématique énoncée, nous avons proposé différents éléments de solution complémentaires. Il s'agissait :

- de proposer un processus souple et rapide à mettre en œuvre dédié à la conception et à l'évaluation des applications Web géographiques (*cf.* Chapitre 2) ;
- de définir un modèle unifié pour la conception et l'exploitation des applications Web géographiques, depuis l'expression des contenus à valoriser (*cf.* Chapitre 3) jusqu'à la spécification des interactions offertes sur ces contenus (*cf.* Chapitre 4) ;
- de concevoir et opérationnaliser un langage de programmation visuel pour spécifier l'interaction de façon simple et riche (*cf.* Chapitre 4) ;
- d'implémenter le modèle unifié en une API exécutable (*cf.* Chapitres 3 et 4) ;
- sur la base des fondements précédents, de développer un environnement-auteur Web permettant aux utilisateurs finaux de concevoir par eux-mêmes des applications Web géographiques (*cf.* Chapitres 3 et 4).

5.1.1 Contribution autour du processus de conception

Le processus de conception proposé est composé de trois phases complémentaires :

1. Phase *Contenu* : Identifier les données (géographiques) à valoriser qui doivent être manipulées par l'application. Les données peuvent se référer à des textes bruts, à des données structurées (issues de bases de données spécifiques type BD IGN ou du Web de données) ou à toute combinaison de ces données grâce aux services et opérateurs logiques offerts par la plateforme.
2. Phase *Interface* : Spécifier la mise en page graphique de l'interface de l'application. Cette interface peut être composée de plusieurs afficheurs, tels que des afficheurs textuels, des afficheurs cartographiques ou des afficheurs chronologiques. Les données qui ont été définies lors de l'étape précédente, peuvent être mises en évidence via ces afficheurs.
3. Phase *Interaction* : Définir des interactions potentielles de l'utilisateur avec l'application. Plusieurs interactions peuvent être spécifiées entre l'utilisateur et les contenus présentés dans les afficheurs, chaque interaction pouvant donner lieu à

des traitements tenant compte de l'afficheur sur lequel ces contenus doivent être valorisés.

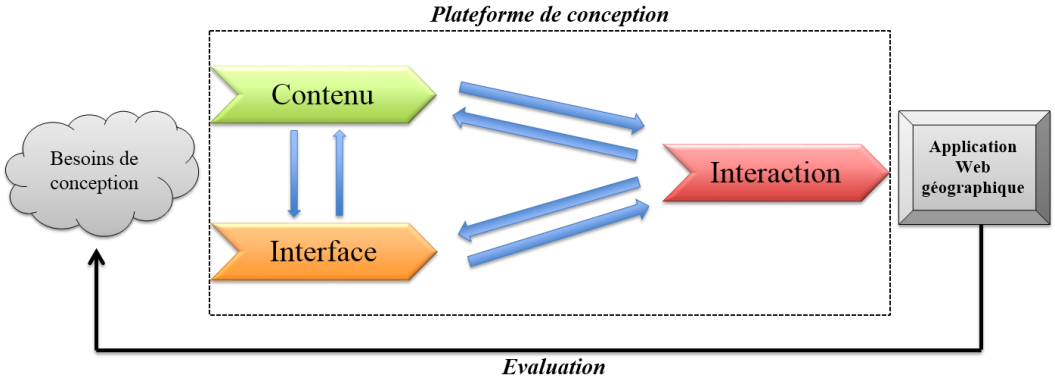


FIGURE 5.1 – Trois phases du processus de conception

La figure 5.1 qui reprend la figure 2.8, rappelle le processus de conception proposé : les phases sont ordonnées dans la figure mais il est possible de revenir en arrière tout au long du processus de conception/évaluation. Autrement dit, à tout moment, le concepteur de l'application peut ajouter, modifier et/ou supprimer certaines données, certains afficheurs ou certaines interactions. En outre, il est également possible de concevoir une application géographique statique (sans interaction) car dès que le concepteur a défini les afficheurs de phase *Interface*, il/elle est en mesure de générer un aperçu de l'application.

Lors de nos expérimentations, nous avons constaté que certains concepteurs démarraient le processus par la phase *Interface* quand d'autres commençaient par la phase *Contenu*. Ces différents types de comportements des concepteurs n'ont pas fait l'objet d'études spécifiques qui auraient sans doute permis d'identifier des profils de concepteurs.

5.1.2 Contribution autour du modèle unifié permettant de décrire des applications Web géographiques interactives

Nous avons spécifié un modèle unifié (Figure 5.2) permettant de stocker les informations relatives aux trois phases mentionnées ci-dessus. Nous avons montré à travers notre modèle unifié que les annotations sont centrales dans le processus de conception pour décrire des applications Web géographiques. Ce modèle peut être utilisé pour décrire les contenus, pour afficher des contenus à l'intérieur des afficheurs et pour spécifier le comportement des applications.

Notre modèle est facilement évolutif et il a d'ailleurs évolué suite aux évaluations présentées en Section 5.2. Nous verrons également en section 5.3 que ce modèle peut être étendu afin de tenir compte des perspectives de cette thèse.

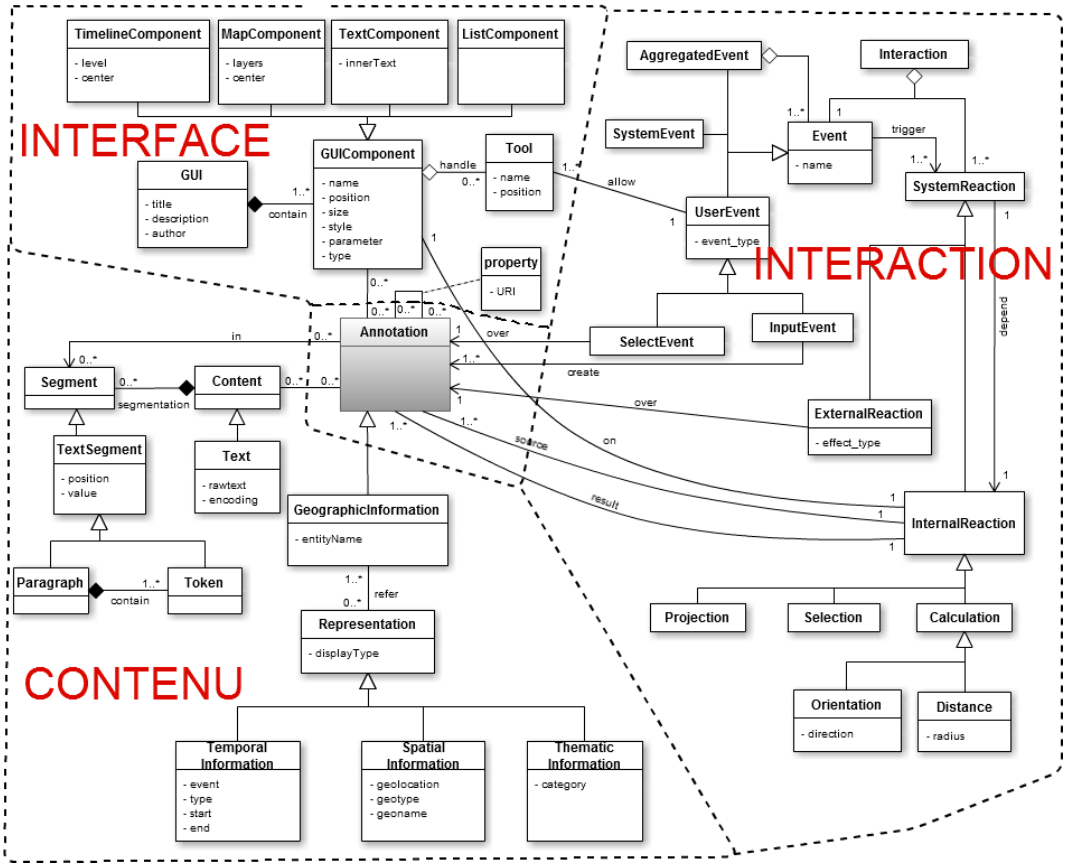


FIGURE 5.2 – Modèle générique contenant de trois parties *Contenu*, *Interface* et *Interaction*

5.1.3 Contribution autour du langage visuel pour la conception d'interactions

Nous avons proposé un langage visuel permettant de spécifier graphiquement et de mettre en œuvre les interactions dans des applications Web géographiques. Nous nous sommes basés sur un formalisme inspiré du diagramme de séquence UML pour concevoir ce langage. Cinq notions de base ont été identifiées : action de l'utilisateur, réaction externe du système, sélection, projection et calcul. Elles étaient mises en correspondance avec le modèle d'*Interaction* comme illustré dans la figure 5.3.

Les diagrammes d'interaction conçus avec notre langage visuel mettent en évidence les flux échangés entre l'utilisateur et le système, mais aussi entre les composants d'interface (afficheurs) qui composent l'application. Les diagrammes d'interaction sont en général simples car ils sont toujours décrits avec les éléments à haut niveau d'abstraction qui augmentent la puissance expressive du diagramme.

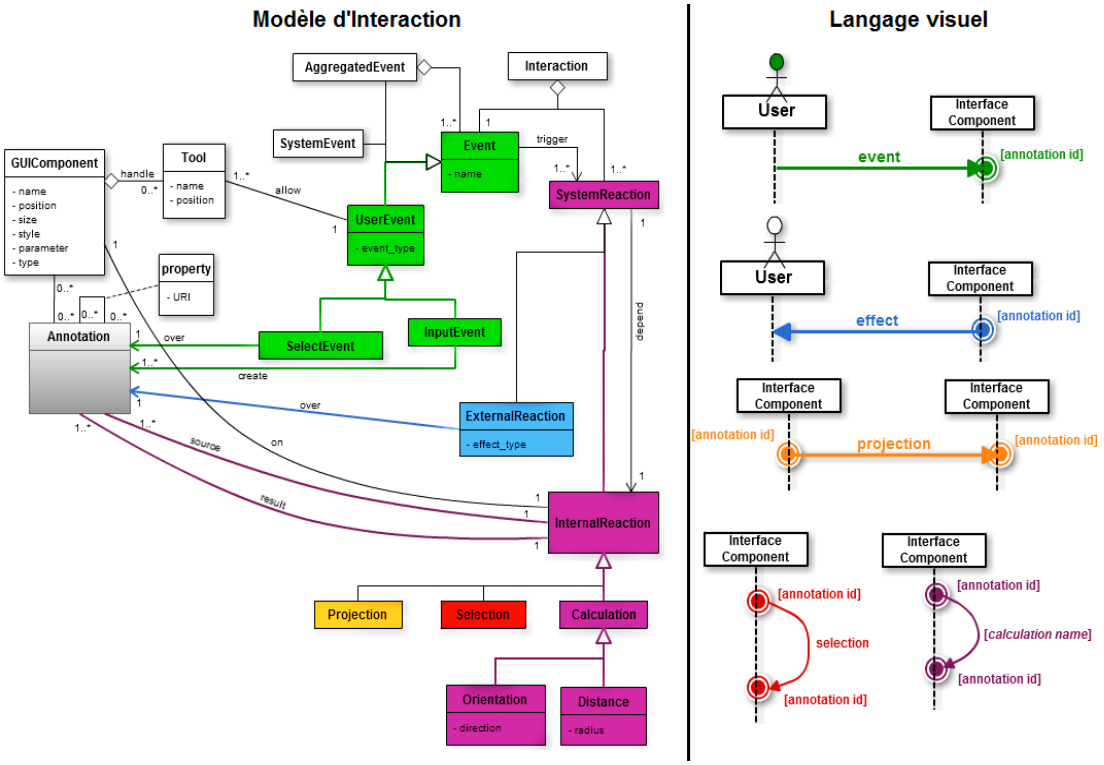


FIGURE 5.3 – Les cinq notions de base du langage de programmation visuel

Notre langage de programmation visuel offre suffisamment de souplesse pour rendre compte de la complexité et de la diversité des interactions à caractériser. Selon son niveau d'expertise, le concepteur peut choisir de produire une interaction complexe (composée de multiples réactions du système) avec un seul diagramme ou avec plusieurs diagrammes simples, chacun spécifiant une partie des réactions du système de l'interaction complexe.

Notons enfin que ce langage qui est basé sur le concept d'Objet-Evénement-Réaction est suffisamment général pour rendre compte d'interactions basées sur des informations autres que les informations géographiques qui ont fait l'objet de ces travaux de thèse.

5.1.4 Contribution autour de l'API WIND

Nous avons montré que l'API WIND [LENM09] permet de programmer à un haut niveau d'abstraction des applications Web interactives diverses (applications WIND) intégrant textes, cartes, calendriers... Cette API est une implémentation quasi complète des classes et des méthodes du modèle unifié (*Contenu*, *Interface* et *Interaction*) en JavaScript.

WIND dispose des cinq caractéristiques principales suivantes :

- **WIND est facile à intégrer**
  WIND fournit aux développeurs des composants d'interface (afficheurs) différents. La carte n'est pas un élément central, pas plus qu'un afficheur textuel, ou un afficheur temporel. Ces afficheurs jouent un rôle équivalent. Le concepteur peut d'ailleurs inclure dans une application Web autant d'afficheurs qu'il le souhaite, en mélangeant par exemple très facilement des cartes provenant de fournisseurs cartographiques différents (par exemple, Google Maps, Bing Maps, Géoportail IGN) sans que cela ne nécessite de compétences de programmation particulière.
- **WIND est orientée objet**
  WIND a été conçu pour profiter des avantages (en particulier, l'encapsulation et l'héritage) des langages de programmation orientée objet. Les objets WIND sont tout simplement créés par leur constructeur de classes. Les méthodes permettent de mettre en œuvre des relations entre les classes. Par exemple, la méthode `createAnnotation` de chaque classe (`Map`, `Text`, `Timeline`...) met en œuvre la relation d'agrégation entre cette classe et la classe `Annotation` ; la méthode `addReaction` met en œuvre la relation d'association entre la classe `Interaction` et la classe `Reaction`.
  Le point principal de WIND est la programmation d'interactions. Dans cet objectif, la classe `Annotation` joue un rôle important. Ce n'est pas seulement une zone source de l'interaction, mais aussi une zone réactive cible de la réaction. Un objet `Annotation` peut avoir plusieurs représentations qui sont des objets `SensiblePart`. Grâce au polymorphisme de la classe `SensiblePart` (par exemple, les classes `MapPart`, `Textpart`, `TimelinePart`, `ListPart`... héritent de la classe `SensiblePart`), WIND permet donc de programmer des interactions quelles que soient les zones réactives. En outre, la programmation d'interactions suit toujours la même structure.
- **WIND est exécutable**
  Le modèle unifié n'est pas un modèle contemplatif de conception [Boc03, MB02]. Grâce à l'API WIND, les développeurs peuvent concevoir rapidement une interaction et immédiatement l'évaluer. Toutes les instances du modèle peuvent être transformées en code exécutable. Il est très simple à exécuter parce que l'API WIND permet de cacher les méthodes complexes. Par exemple, le constructeur de la classe `Reaction` a un attribut `calledFunction` qui est de type `String`. Si l'attribut `calledFunction` est "`hightlight`", la zone réactive de l'objet `Reaction` sera appliquée par la méthode `hightlight()` qui est définie dans l'API.
  Une interaction devient exécutable quand on lui applique la méthode `activate()`. Les réactions définies pour l'interaction sont enregistrées dans le système et seront exécutées quand l'événement d'interaction se produira sur la zone réactive définie pour cette interaction.
- **WIND est déclarative**
  Les développeurs remarquent qu'une application WIND (*cf.* Figure 4.40) a une structure très simple : le code JavaScript d'une application Web interactive ne

comporte aucune déclaration conditionnelle et aucune boucle. Tous ces éléments sont encapsulés dans l'API WIND, et le développeur / concepteur a uniquement besoin de déclarer des interactions, des événements et des réactions. Notons qu'en exploitant le modèle unifié, l'environnement WINDMash est capable de générer des interactions que l'API WIND peut exécuter.

– **WIND favorise la programmation légère**
L'API WIND a été programmée avec le langage JavaScript. WIND est exécutée côté client (et est compatible avec les navigateurs Web récents tels que Firefox >4.x et Google Chrome). Cette caractéristique est très importante car nous voulons que les utilisateurs / concepteurs n'aient pas à installer et paramétrer un serveur Web pour exécuter les applications WIND. Ceci est un gros avantage comparé à des solutions type GeoDjango [Geo09], notamment pour des usages dans des écoles où les moyens techniques (capacité des machines) et les compétences des enseignants ne vont pas jusqu'à la mise en œuvre d'architectures n-tiers.

5.1.5 Contribution autour de l'environnement-auteur WINDMash

Nous avons implémenté notre processus de conception dans un environnement nommé WINDMash. Il propose trois modules correspondant aux trois phases du processus de conception qui permettent aux concepteurs de spécifier graphiquement les comportements de l'application Web géographique (Figure 5.4). La conception avec WINDMash repose sur le modèle unifié que nous avons proposé : chaque module crée des instanciations d'une partie du modèle. A l'issue de la conception, les instances du modèle sont finalement combinées pour générer le code exécutable de l'application finale. Pour cette partie nous profitons largement de la puissance des opérateurs sur le langage RDF dans lequel tout le modèle unifié est codé (Figure 5.5).

WINDMash est donc un environnement Web capable de générer des applications WIND. Il permet la programmation visuelle et la génération automatique du code des applications WIND. Cet environnement permet non seulement une réduction du temps de développement des applications Web géographiques, mais aussi une simplification de la tâche de programmation pour des utilisateurs finaux non-informaticiens grâce au langage visuel utilisé qui en facilite l'appropriation. L'environnement peut être à la fois utilisé pour la conception et pour l'évaluation de l'application grâce à la capacité de génération automatique de code qui se fait par un simple clic sur un bouton. Les applications validées peuvent ensuite être déployées en dehors de l'environnement, car ce sont des simples applications Web comportant un seul fichier HTML incorporant du code JavaScript.

Dans la section 5.2, nous proposons d'évaluer les usages de nos outils (l'environnement-auteur WINDMash et l'API WIND). Les perspectives de la thèse sont ensuite présentées dans les sections 5.3 et 5.4.

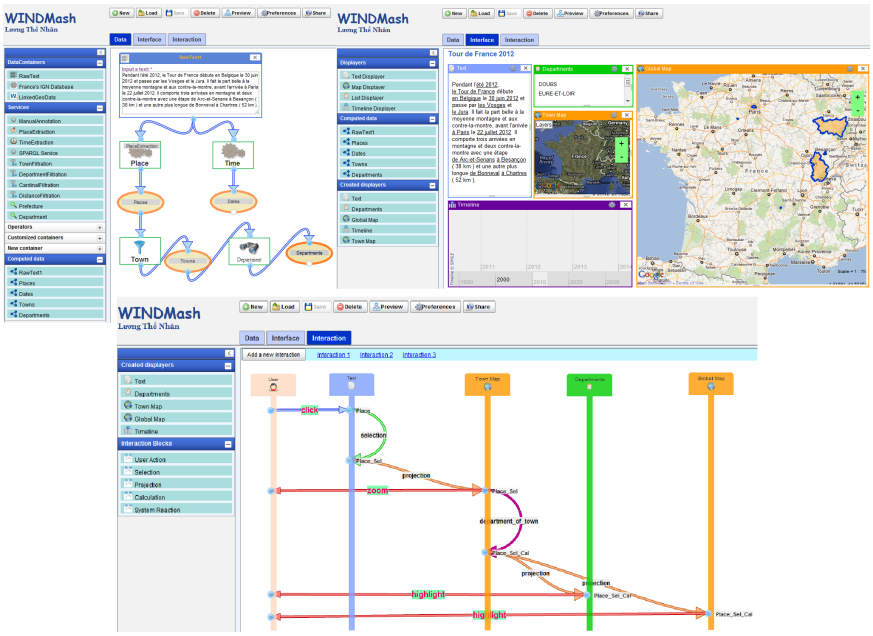


FIGURE 5.4 – Manipulation de WINDMash

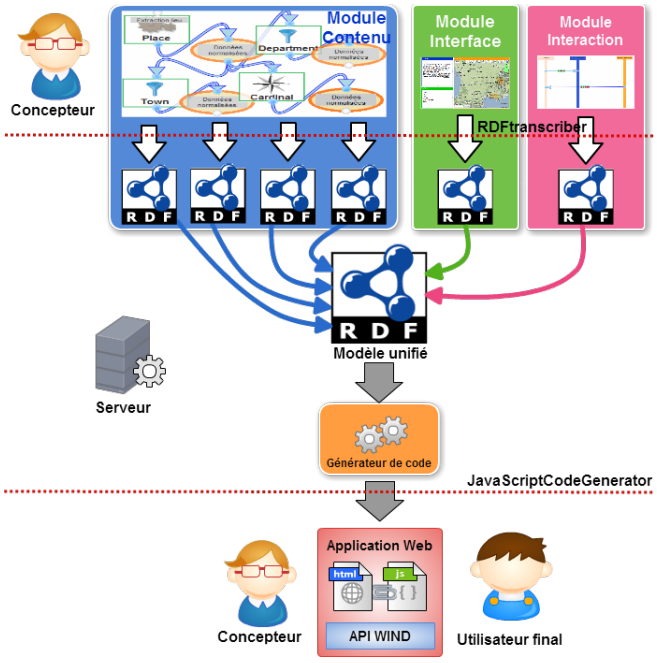


FIGURE 5.5 – Architecture de WINDMash

5.2 Évaluation générale de la plateforme de conception proposée

La plateforme WINDMash et son API support WIND ont été imaginées sur la base d'échanges avec des enseignants. Faire évaluer cette plateforme par des enseignants est bien sûr l'objectif sur lequel travaille l'équipe T2i mais cet objectif n'était pas atteignable sur la durée de cette thèse. Nous avons cependant conduit un ensemble d'évaluations préparatoires portant sur les points suivants :

- Capacité de concevoir et mettre en œuvre facilement avec WINDMash des applications géographiques répondant aux besoins d'enseignants tels que décrits dans le chapitre 1.
- Capacité pour des non-informaticiens de concevoir visuellement avec WINDMash des interactions portant sur des contenus géographiques affichés sur des composants d'interface.
- Couverture des besoins et robustesse de l'API WIND pour des projets de développements informatiques.

Les prochains paragraphes relatent chacune des évaluations effectuées.

5.2.1 Évaluation du processus global de conception pour des usages pertinents à l'école élémentaire

Dans son mémoire de Master2, [Pai11] avait proposé un scénario complexe (*cf.* Section 1.2.2.2) mêlant interactions simples et annotations de corrélation. Nous reprenons ici ce scénario pour montrer le processus complet d'implémentation tel que WINDMash le permet.

Intention pédagogique du scénario : l'enseignant souhaite que les élèves améliorent leurs stratégies pour réfléchir sur un texte (mieux le comprendre) en s'appuyant sur des activités de schématisations et d'annotation de corrélations.

Intention pédagogique spécifique : l'enseignant souhaite que le groupe classe repère des informations explicites et implicites permettant d'établir des liens entre le texte, une représentation du temps, une représentation de l'espace.

Interface associée : La maquette d'interface (Figure 5.6) se compose de trois composants (Texte, Carte, Frise). L'utilisateur sélectionne l'outil et intervient directement sur le composant pour l'annoter. Chaque composant a ses propres caractéristiques.

- Le composant-Texte (zone haute gauche) dispose de deux outils d'annotation : le rouge pour les lieux et le bleu pour les dates. Il a aussi une barre de défilement verticale pour permettre de lire en intégralité le texte affiché.

– Le composant-Carte (zone droite) dispose d'un outil permettant de tracer des itinéraires. La carte est en mode relief et centrée sur la zone des Pyrénées. Les interactions par défaut sont le zoom et le déplacement.
– Le composant-Frise (zone basse gauche) peut glisser horizontalement.

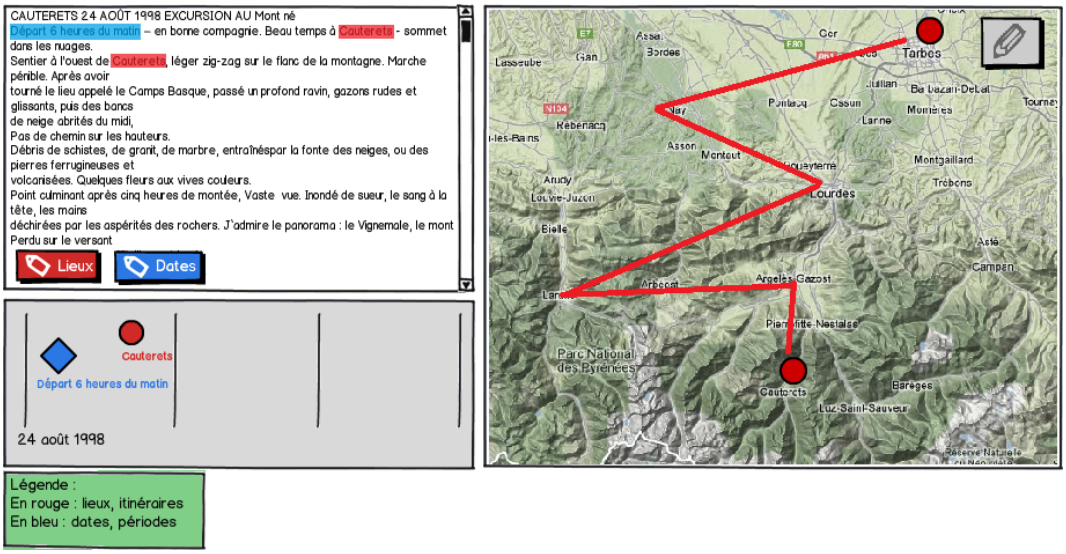


FIGURE 5.6 – Maquette de l'interface

Des outils permettent d'intervenir de façon globale sur les composants sous la forme d'activité d'annotations et d'ajout de contenus associés aux composants. La synchronisation entre les trois composants permet d'associer sur différents composants la même information issue des annotations. L'annotation de corrélation automatique entre le composant cartographique et le composant textuel permet de placer automatiquement des lieux sur le composant cartographique.

En résumé, les comportements de l'application souhaitée sont les suivants :
– Sur la carte, l'utilisateur peut tracer des itinéraires comme il veut.
– Sur le texte, il y a deux outils (boutons) pour l'annotation :
  – Quand l'utilisateur annote un texte avec le bouton rouge ("Lieux") : si le texte concerne un lieu, la carte met un point rouge automatiquement sur ce lieu et la frise ne fait rien. Cependant, l'utilisateur peut cliquer sur le texte pour le rendre déplaçable et il peut glisser-déposer ce texte vers la frise ; la frise met un point rouge (avec le nom de lieu) là où il le dépose.
  – Quand l'utilisateur annote un texte avec le bouton bleu ("Dates") : si le texte concerne une date, la frise met un losange bleu avec le nom de l'événement à cette date sur la frise et celle-ci se centre à cette date. La carte ne fait rien dans ce cas.

En tant que concepteur, nous suivons les caractéristiques du scénario ci-dessus et utilisons WINDMash pour concevoir les trois phases du processus, puis générons une application Web (Figure 5.14) dont le comportement doit être équivalent à celui décrit dans le scénario.

Phase Contenu

Lors de cette phase, le concepteur a utilisé un conteneur "*Texte brut*" dans lequel il a saisi un texte extrait d'un récit de voyage. Ce texte a été traité par les deux services "*Extraction lieu*" et "*Extraction temps*" pour extraire deux ensembles de contenus géographiques "*lieux*" et "*temps*" correspondant respectivement aux lieux (Cauterets...) et dates (24 août 1998...) cités. (Figure 5.7).

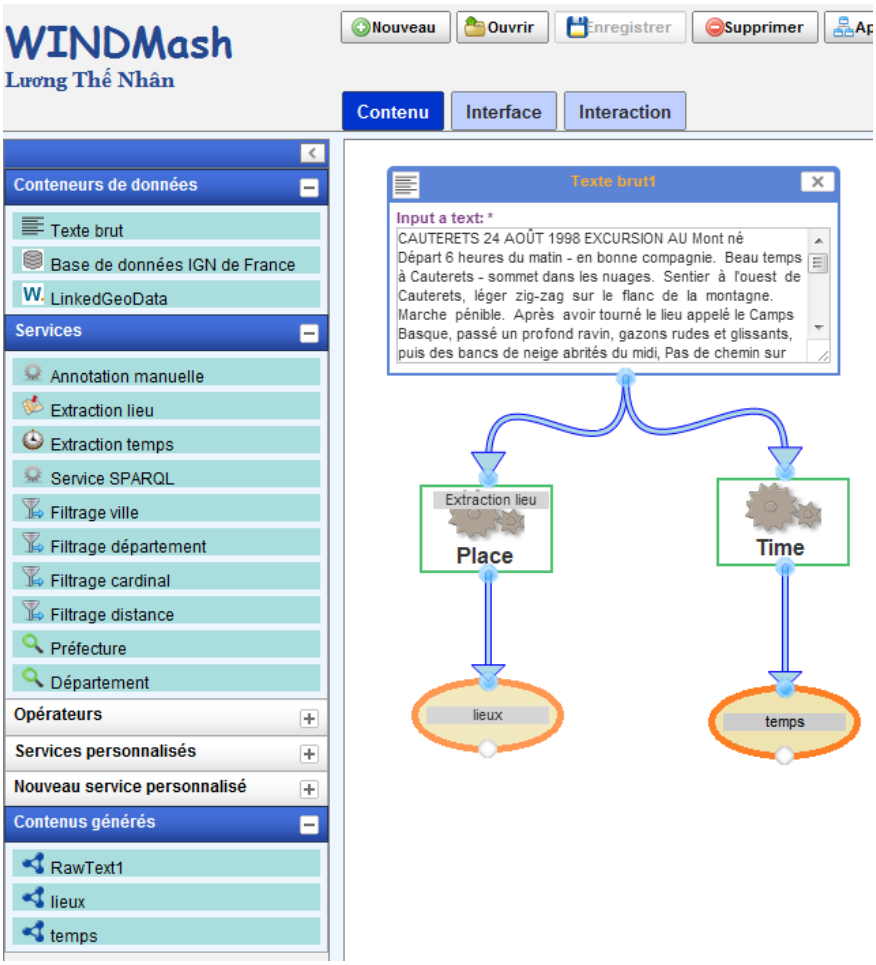


FIGURE 5.7 – Manipulation de la phase *Contenu* pour le scénario

Phase Interface

Cette phase permet au concepteur de configurer l'interface de l'application finale. Le concepteur a successivement pris et placé les composants d'interface "*Afficheur texte*", "*Afficheur carte*", "*Afficheur frise*" dans la zone de travail. Afin d'afficher le contenu de l'extrait du récit de voyage, il a également glissé-déposé le contenu généré "*RawText1*" contenant le récit de voyage dans l'afficheur Texte (Figure 5.8).

Il est important que les afficheurs soient configurables. En général, le concepteur a pu définir leur taille, leur position, leur nom et leur couleur. Pour configurer chaque afficheur, il faut cliquer sur sa petite icône "pignon" se trouvant en haut à droite de l'afficheur.

Dans ce scénario, le concepteur a paramétré l'afficheur-Texte en mode éditable pour pourvoir faire saisir / modifier un texte aux élèves ultérieurement. Sur l'afficheur-Texte, il faut créer deux boutons d'annotation "*Lieux*" et "*Dates*" en remplissant leurs paramètres dans la zone de configuration et cliquant sur "Ajouter un bouton d'annotation" (Figure 5.9).

FIGURE 5.8 – Manipulation de la phase *Interface* pour le scénario

Le concepteur a configuré l'afficheur-Carte pour que les élèves puissent faire un zoom et glisser sur la carte. Il a choisi deux fonds cartographiques (Google Street et Google Terrain). Il a également mis à disposition un outil permettant de faire des tracés (Figure 5.10).

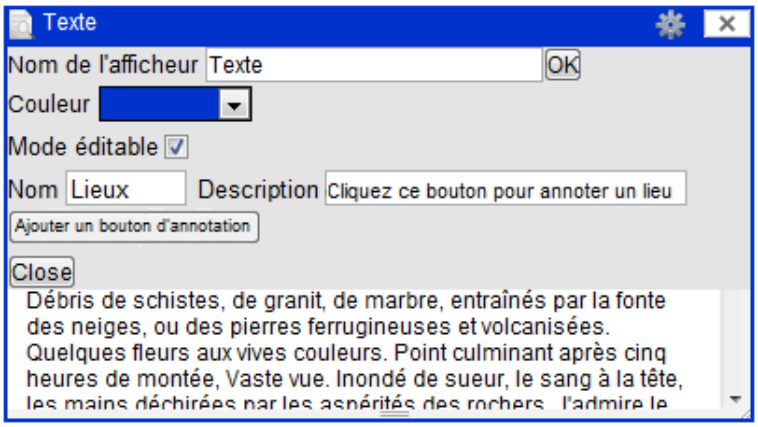


FIGURE 5.9 – Configuration de l'afficheur-Texte

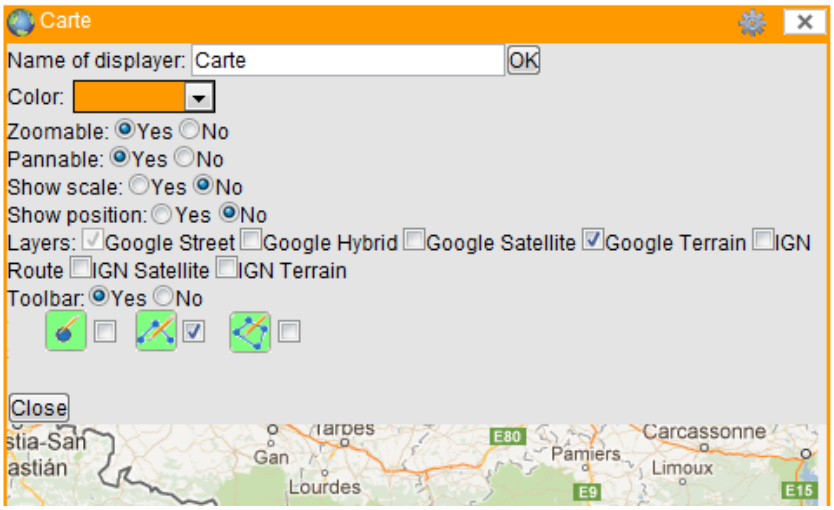


FIGURE 5.10 – Configuration de l'afficheur-Carte

Phase Interaction

Le concepteur a eu besoin de trois diagrammes d'interaction conformes à notre langage visuel pour mettre en œuvre les comportements suivants de l'application finale :

Quand l'utilisateur annote un segment de texte avec le bouton rouge "Lieux", il y a deux possibilités. Si le texte concerne réellement un lieu, alors le texte annoté est surligné de la couleur du bouton et la carte positionne un point rouge (`highlight`) automatiquement sur ce lieu ; sinon (le texte annoté n'est pas reconnu comme un lieu), le texte annoté est surligné et rien ne se produit sur la carte (Figure 5.11).

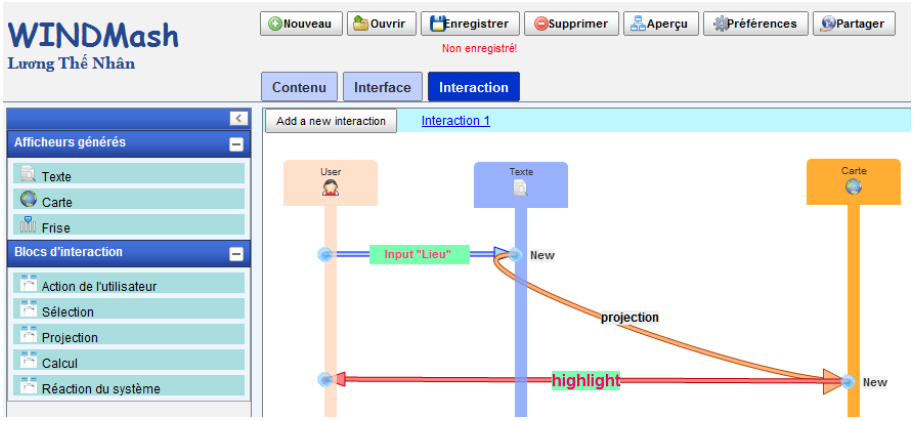


FIGURE 5.11 – Spécification du comportement 1 dans la phase *Interaction* pour le scénario

Dans la figure 5.11, la flèche **Input "Lieu"** décrit une action de saisie présentée dans la section 4.3.2.2 et illustrée dans la figure 4.26. Ici, le concepteur indique que le type du résultat de saisie est un lieu, autrement dit, l'action de l'utilisateur **Input "Lieu"** va créer une annotation dont la sémantique est un lieu.

Quand l'utilisateur annote un segment de texte avec le bouton rouge "Lieux" (`Input "Lieu"`), alors le texte annoté est surligné de la couleur du bouton. L'utilisateur peut alors cliquer sur le texte pour le rendre déplaçable par glisser-déposer vers la frise. Dans ce cas, la frise met un point rouge (avec le nom de lieu) à l'endroit déposé (Figure 5.12).

Dans la figure 5.12, il y a deux actions consécutives déclenchées par l'utilisateur : l'action de saisie **Input "Lieu"** et l'action de sélection **click**. Ce paradigme n'est pas abordé dans la partie des concepts de notre langage de programmation visuel (*cf.* Section 4.3.2.2), mais il peut être considéré comme une piste pour notre perspective sur l'événement agrégé (*cf.* Figure 5.38(c)).

La réaction externe avec le label "**draggable**" permet de modéliser une réaction externe pour laquelle le contenu annoté devient potentiellement manipulable par une action de de glisser-déposer (*drag-n-drop*). Ceci est modélisé comme une modification du style CSS des mots annotés dans le texte par l'utilisateur. Dans notre environnement, pour implémenter cette réaction externe, nous avons choisi un style CSS avec une couleur blanche du texte sur un arrière-plan rouge et le curseur prend la forme de quatre flèches en croix.

L'action de glisser-déposer (*drag-n-drop*) est représentée par deux flèches dans la figure 5.12. Une première flèche qui comporte l'étiquette "*drag*" peut être considérée comme équivalente à une action "*click*" de l'utilisateur et celle avec l'étiquette "*drop*" peut

être considérée comme équivalente à une réaction interne de type "*projection*". Toutefois l'action "*drop*" est faite par l'utilisateur en déposant l'objet. L'action glisser-déposer a donc un comportement quasi équivalent à celui de la séquence "*click*" et "*projection*".

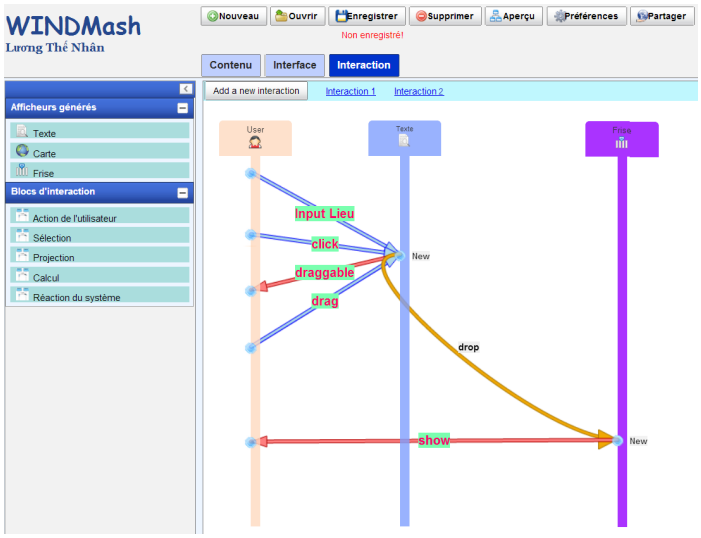


FIGURE 5.12 – Spécification du comportement 2 dans la phase *Interaction* pour le scénario

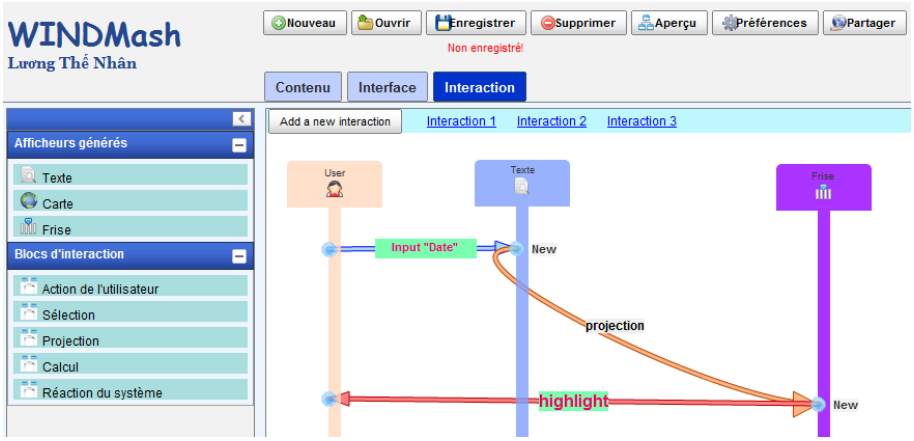


FIGURE 5.13 – Spécification du comportement 3 dans la phase *Interaction* pour le scénario

Quand l'utilisateur annote un segment de texte avec le bouton bleu "Dates", il y a deux possibilités. Si le texte concerne réellement une date, alors le texte annoté est surli-

gné de la couleur du bouton et la frise affiche un losange bleu avec le nom de l'événement à cette date ; la frise se centre automatiquement à cette date. Sinon (le texte annoté n'est pas reconnu comme une date), le texte annoté est surligné et rien ne se produit sur la frise (Figure 5.13).

De la même façon, dans la figure 5.13, la flèche **Input "Date"** décrit une action de saisie présentée dans la section 4.3.2.2 et illustrée dans la figure 4.26. Ici, le concepteur indique que le type du résultat de saisie est une date, autrement dit, l'action de l'utilisateur **Input "Date"** va créer une annotation dont la sémantique est une date.

Après d'avoir complété les trois phases, le concepteur peut générer l'application Web finale (Figure 5.14). En comparant celle-ci avec la Figure 5.6, nous constatons que les figures sont quasiment similaires. Chacune se compose de quatre composants d'interface (afficheurs) : le texte en haut à gauche, la carte à droite et la frise chronologique en bas à gauche. Le texte dispose deux boutons/outils : celui en rouge pour annoter un lieu et celui en bleu pour annoter une date. La carte est aussi dans le même style que la maquette avec le fond cartographique de terrain avec un bouton/outil pour tracer des lignes sur la carte. Pour la frise chronologique, les interfaces de l'application générée et de la maquette ne sont pas parfaitement identiques mais leurs fonctionnalités restent similaires. Les légendes de la maquette ne sont pas implémentées exactement identiques mais les utilisateurs peuvent trouver les sémantiques des couleurs lors du survol sur les boutons/outils dans le composant textuel.

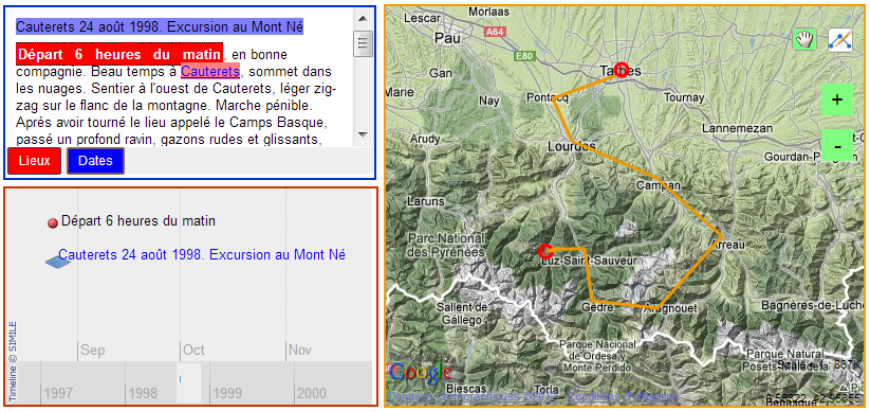


FIGURE 5.14 – Application générée par WINDMash correspondant au scénario

5.2.2 Retours d'expériences relatifs à l'utilisabilité de la plateforme WINDMash

WINDMash offre un environnement complet de conception à ses utilisateurs. Dans les prochains paragraphes nous relatons deux évaluations successives qui ont été menées au cours des derniers mois.

5.2.2.1 Évaluation 1 : Évaluation de l'utilisabilité du langage visuel et du modèle d'Interaction

Nous avons conçu un protocole [LEMN12] afin d'évaluer si notre langage visuel et son modèle sous-jacent (présentés en Section 4.3.2) permettent aux concepteurs de concevoir facilement le comportement interactif d'une application Web géographique.

Pour évaluer les aspects cognitifs des notations offertes par notre langage visuel, nous nous sommes basés sur les résultats des travaux de trois chercheurs. [Ray91] a établi que les densités syntaxique et sémantique sont les caractéristiques principales de langages visuels. [BG00] a proposé des nouvelles dimensions avec différents niveaux d'adoption et de raffinement telles que la cohérence, la visibilité, la viscosité, la dépendance cachée, l'ambiguïté créative ou la gestion d'abstraction. Plus récemment, [Moo09] a proposé un ensemble de neuf principes pour l'évaluation cognitive des notations offertes par un langage visuel. Son approche utilise une combinaison de l'artisanat et des connaissances scientifiques. De plus, [Moo09] a proposé une structure modulaire permettant au concepteur de facilement ajouter ou supprimer des principes car chaque principe est défini par un nom, une définition sémantique (théorique), une définition opérationnelle (empirique), certaines stratégies de conception et des exemples.

A. Description du protocole expérimental

En octobre 2011, nous avons invité trente deux étudiants en tout début de deuxième année DUT Informatique à participer à cette évaluation. Aucun des étudiants n'avait une expérience académique significative du diagramme de séquence UML. La procédure d'évaluation a été organisée en six étapes.

Étape 1 : Présentation d'un exemple

Nous avons donné une présentation en 35 minutes sur un exemple simple mais complet pour montrer la conception d'une application Web géographique. Dans cet exemple, nous avons spécifié une application permettant à l'utilisateur de connaître la "*préfecture*" d'une liste de villes données.

L'exemple (Figure 5.15) se compose d'une liste des villes et d'une carte affichant un point pour chaque ville. Le comportement de l'application se présente comme suit : lorsque l'utilisateur clique sur un nom de lieu dans la liste des villes, l'application calcule

la préfecture de cet endroit et la carte fait un zoom avant sur cette préfecture (voir http://bit.ly/uwAZne).

La séquence des éléments d'interaction peut être réalisée de plusieurs façons équivalentes. Par exemple, une séquence enchaînant une projection et un calcul ainsi que la même séquence dans l'ordre inverse produisent deux diagrammes différents, mais le comportement final est identique (Figure 5.16).

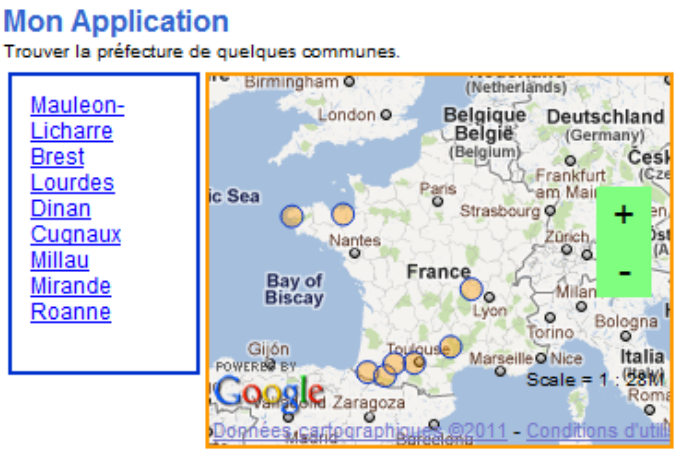


FIGURE 5.15 – Un exemple d'application géographique montré lors de l'évaluation

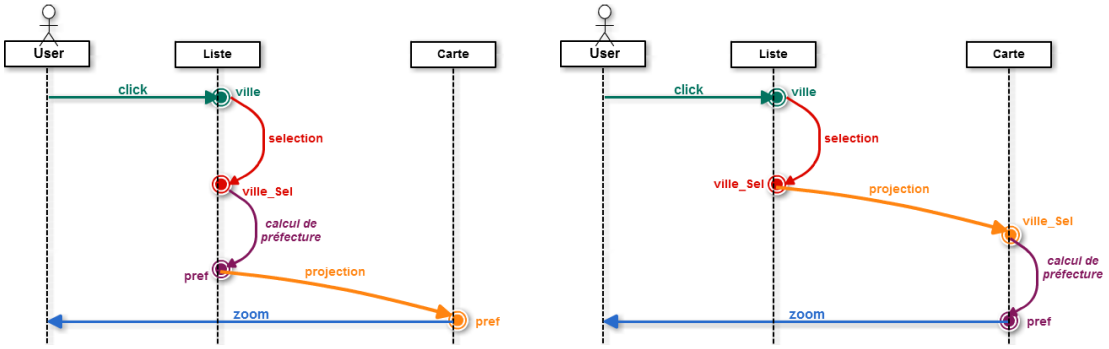


FIGURE 5.16 – Deux différents diagrammes d'interaction avec le même comportement

Pendant cette présentation, nous nous sommes concentrés sur la phase *Interaction* pour évaluer si les participants parvenaient à comprendre le rôle des éléments de notre langage visuel.

Étape 2 : Présentation orale de l'application de test

Pendant dix minutes, nous avons montré aux participants comment concevoir avec WINDMash l'application présentée dans la figure 5.17.

A partir d'un texte et une carte présentant quelques villes situées sur la côte atlantique française et méditerranéenne, l'objectif principal de l'application à concevoir est d'aborder le concept de département. Le comportement de l'application (Figure 5.17) est le suivant : lorsque l'utilisateur clique sur une ville écrite sur le texte (situé sur la partie gauche de l'interface) ou affichée sur la carte (située sur la partie droite de l'interface), le nom du département correspondant est automatiquement affiché dans la zone au centre en haut de l'interface et la frontière administrative du département est mise en évidence dans la zone au centre en bas de l'interface. Le nom de la ville sélectionnée dans le texte principal et la frontière administrative de cette ville dans la carte principale seront également mis en évidence.

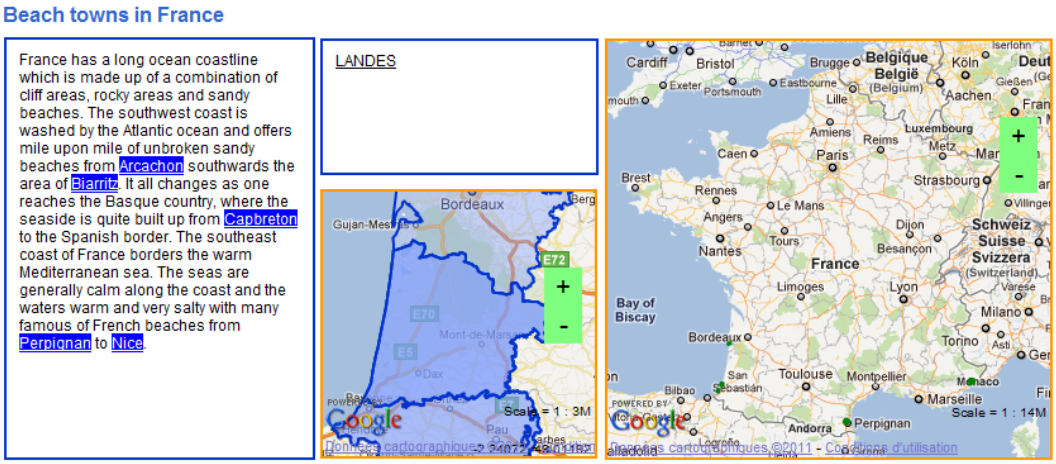


FIGURE 5.17 – Application de test demandée à concevoir / réaliser par les étudiants

Étape 3 : Création guidée des phases *Contenu* et *Interface*

Pendant dix minutes, nous avons aidé les étudiants-testeurs à concevoir les deux phases correspondant à la conception des données géographiques (phase *Contenu*) et la conception de l'interface utilisateur (phase *Interface*).

Étape 4 : Production à la main de la phase *Interaction*

Ensuite, les participants se sont concentrés sur la conception des capacités interactives de l'application. Pendant quinze minutes, ils ont produit, sans aucune aide, ni

réponse de notre part à leurs questions, certains diagrammes d'interaction sur papier (Figure 5.18) en utilisant notre langage visuel. Ce travail préparatoire a permis de se concentrer sur le langage visuel indépendamment de l'environnement WINDMash.

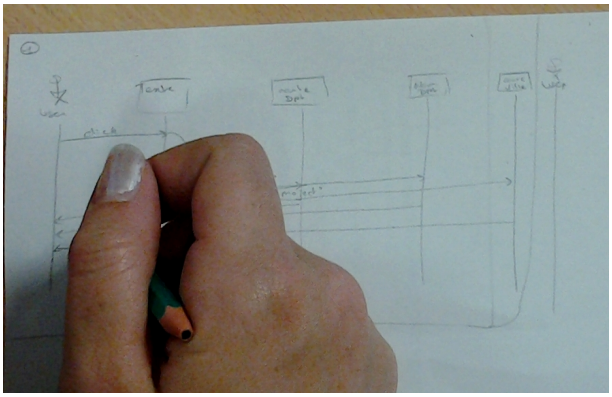


FIGURE 5.18 – Conception d'un diagramme d'interaction sur papier en utilisant notre langage visuel

Étape 5 : Mise en œuvre de la phase *Interaction*

Pendant quinze minutes, les participants ont dû créer leurs diagrammes d'interaction (Figure 5.19) en utilisant WINDMash, puis générer le code exécutable afin d'évaluer l'application finale.

FIGURE 5.19 – Conception d'un diagramme d'interaction avec WINDMash

Étape 6 : Évaluation

Enfin, nous avons demandé aux participants de nous donner leurs spécifications d'interaction rédigées sur papier en leur demandant de les compléter par des suggestions de changements relatifs à l'environnement WINDMash.

Pour conclure, nous avons demandé aux participants de remplir un formulaire d'évaluation composé de onze questions par rapport aux neuf dimensions cognitives [Moo09] utilisées pour évaluer notre langage de visuel :

- **Clarté sémiotique (Semiotic Clarity - SC)** :
  - **SC1** : Trouvez-vous qu'il y a des éléments du langage (graphiques + légendes) redondants ?
  - **SC2** : Trouvez-vous qu'il y a des éléments du langage (graphiques + légendes) manquants ?
- **Discriminabilité perceptuelle (Perceptual Discriminability - PD)** : Trouvez-vous que les éléments du langage (graphiques + légendes) sont faciles à distinguer les uns des autres ?
- **Transparence sémantique (Semantic Transparency - ST)** : Trouvez-vous que la forme des éléments du langage (graphiques + légendes) suggère leur signification ?
- **Gestion de la complexité (Complexity Management - CM)** : Pensez-vous que ce langage permet de décrire facilement (de façon modulaire, hiérarchisée...) des interactions complexes ?
- **Intégration cognitive (Cognitive Integration - CI)** : Selon vous, ce langage permet-il de décrire une interaction en important des informations provenant d'autres diagrammes (interactions) ?
- **Expressivité visuelle (Visual Expressiveness - VE)** : Pensez-vous que les concepteurs du langage ont utilisé toutes les possibilités de formes, couleurs, taille, texte pour que les éléments du langage (graphiques + légendes) soient les plus parlants possibles ?
- **Double codage (Dual Coding - DC)** : Le langage vous propose-t-il de compléter la description d'une interaction avec des annotations textuelles ?
- **Économie graphique (Graphic Economy - GE)** : Pensez-vous que le nombre d'éléments du langage (graphiques + légendes) mis à votre disposition pour décrire une interaction est suffisant ?
- **Ajustement cognitif (Cognitive Fit - CF)** :
  - **CF1** : Pensez-vous que les diagrammes que vous avez manipulés peuvent être réalisés de manière identique sur papier ?
  - **CF2** : Pensez-vous que ce langage pourrait être utilisé par des personnes non familières des diagrammes de séquence ?

Pour chaque question, quatre réponses possibles étaient disponibles (oui, plutôt oui, plutôt non, non). En outre, les participants ont eu la possibilité de justifier leurs réponses avec un bref commentaire ouvert.

B. Résultats et discussion

La figure 5.20 résume les réponses des participants. D'un point de vue global, les résultats de l'évaluation sont bons comme présentés par les smileys ajoutés dans le tableau.

	Oui	Plutôt Oui	Plutôt Non	Non		Notre interprétation
SC1	0%	0%	13%	88%		☺
SC2	0%	0%	63%	38%		☺
PD	38%	50%	0%	13% ☹		☺
ST	50%	38%	13%	0%		☺
CM	38%	63%	0%	0%		☺
CI	13% ☹	13% ☹	25% ☺	50% ☺		: −
VE	13%	50%	38%	0%		: −
DC	0%	13%	38%	50%		☺
GE	25%	75%	0%	0%		☺
CF1	75%	25%	0%	0%		☺
CF2	0%	63%	13%	25%		: −

FIGURE 5.20 – Résultats de l'expérimentation

Toutefois, certains points spécifiques doivent encore être améliorés. Concernant l'intégration cognitive (*Cognitive Integration*), il devrait être intéressant de définir des mécanismes permettant aux concepteurs de réutiliser des parties d'interaction d'un diagramme dans d'autres diagrammes. Cette approche traditionnelle d'ingénierie des logiciels (boîte noire vs boîte de verre) augmenterait les capacités de réutilisation. Une autre perspective intéressante concerne l'expressivité visuelle (*Visual Expressiveness*). Nous devons améliorer la facilité d'utilisation de chaque élément de l'interaction et spécialement sa représentation visuelle. L'environnement de conception doit aussi mieux contrôler les possibilités de connexion entre les éléments graphiques afin de réduire certaines erreurs possibles pendant la conception.

Par ailleurs, pour l'application de test, les participants ont produit deux livrables différents. Ils ont d'abord réalisé une production à la main de la phase *Interaction*. Au cours de cette étape, 81% des participants ont élaboré une proposition "entièrement correcte".

Les autres 19% ont rencontré des problèmes syntaxiques qui peuvent être surmontés par un outil-assistant tel que WINDMash. Nous retenons aussi que 6% des participants ont élaboré une proposition avec quatre diagrammes (Figure 5.22) alors que 94% ont utilisé deux diagrammes.

En outre, l'étude des diagrammes d'interaction met en évidence la simplicité, la flexibilité et la puissance de notre langage visuel qui permet aux concepteurs de spécifier les interactions complexes : il est possible de décomposer la description d'une interaction complexe en plusieurs interactions simples dont le comportement est équivalent à celle de l'interaction complexe.

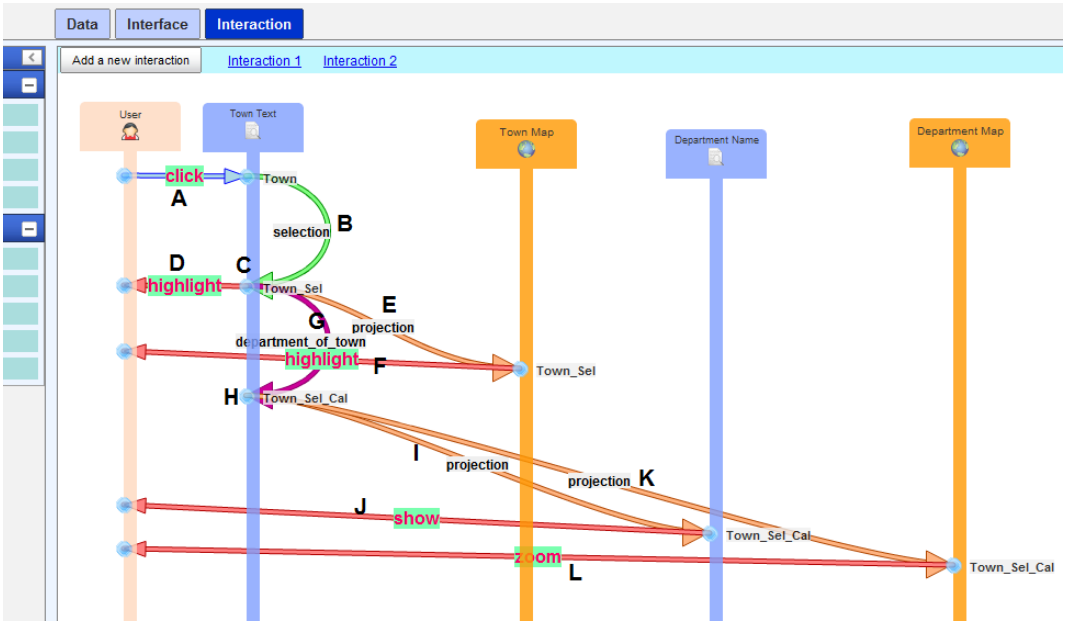


FIGURE 5.21 – Interaction conçue avec WINDMash

Par exemple, la figure 5.21 illustre une interaction complexe : lorsque l'utilisateur sélectionne une ville dans l'afficheur textuel nommé "`Town Text`" (A, B), cette ville (C) est mise en évidence dans cet afficheur (D) ainsi que dans l'afficheur cartographique nommé "`Town Map`" (E, F). En outre, l'afficheur textuel nommé "`Department Name`" va afficher le nom du département de la ville sélectionnée (G, H, I, J) et l'afficheur cartographique nommé "`Department Map`" fera un zoom avant sur le département (G, H, K, L). Un participant a proposé les quatre schémas suivants (Figure 5.22) qui sont équivalents au schéma de la figure 5.21.

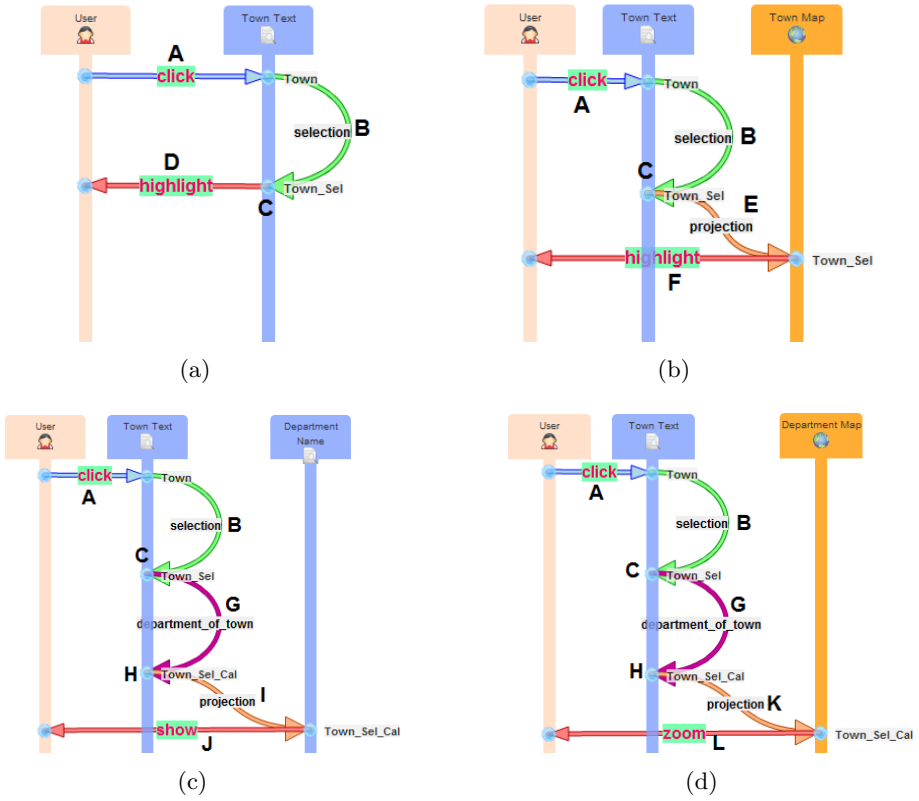


FIGURE 5.22 – Diagramme d'interaction dans la figure 5.21 divisé en quatre diagrammes

Le premier diagramme (Figure 5.22(a)) décrit la mise en évidence de la ville ("`Town`") cliquée dans l'afficheur nommé "`Town Text`".

Le deuxième diagramme (Figure 5.22(b)) décrit la mise en évidence dans l'afficheur nommé "`Town Map`" de la ville ("`Town`") cliquée dans l'afficheur nommé '`Town Text`".

Le troisième diagramme (Figure 5.22(c)) calcule le département de la ville sélectionnée, puis envoie et montre le résultat de ce calcul dans l'afficheur nommé "`Department Name`".

Le quatrième diagramme (Figure 5.22(d)) calcule le département de la ville sélectionnée, puis envoie et montre le résultat de ce calcul dans l'afficheur nommé "`Department Map`" .

Comme WINDMash est encore un prototype, la plupart des participants a rencontré des problèmes dus à notre environnement et surtout quand ils ont voulu corriger ou

modifier un diagramme d'interaction. Ces problèmes relevaient de bogues présents dans le code de l'application WINDMash qui ont été corrigées ensuite.

5.2.2.2 Enquête 2 : Évaluation du processus complet de conception

Compte tenu de l'objectif limité au langage visuel et à la phase *Interaction* de l'évaluation précédente, nous avons mené en juillet 2012 une autre évaluation sur le processus complet de conception avec treize étudiants en reconversion professionnelle préparant un DU (Diplôme d'Université) dans le domaine des Technologies de l'Internet (DU TIC - http://www.iutbayonne.univ-pau.fr/diplomes-universitaires/tic/objectifs.html). Aucun d'eux n'était informaticien mais tous étaient des utilisateurs réguliers de l'outil informatique ayant un profil initial en communication, infographie ou audiovisuel. L'évaluation consistait à évaluer les usages des trois phases (*Contenu*, *Interface* et *Interaction*) de WINDMash.

A. Description du protocole expérimental

La tâche consistait à construire une application Web géographique en utilisant notre environnement WINDMash. La procédure d'évaluation a été organisée en trois étapes. Durant 45 minutes, nous avons effectué une présentation générale de l'environnement WINDMash et montré une vidéo guidant la conception d'application (étape 1). Les étudiants ont travaillé ensuite en autonomie et à distance pour construire une application Web géographique décrite ci-après (étape 2). Ils ont enfin rempli un questionnaire d'enquête (étape 3).

Supposons que l'entraineur d'une école de cyclisme organise un stage estival mêlant cyclisme et activités pédagogiques pour des enfants de cycle 3 (CM1-CM2) de l'école élémentaire (9-10 ans). Dans le cadre d'une activité pédagogique, il utilise des références au Tour de France 2012. Il veut construire une application Web géographique utilisant un texte décrivant une partie de la course et affichant les étapes sur une carte et les dates correspondantes sur une frise chronologique. Cette application a deux buts principaux. Elle doit introduire la notion de département et améliorer leurs capacités en géographie à propos des principales villes et départements visités durant la course.

Le texte qui a servi à cette évaluation est un extrait sur le Tour de France 2012 : "*Pendant l'été 2012, le Tour de France débute en Belgique le 30 juin 2012 et passe par les Vosges et le Jura. Il fait la part belle à la moyenne montagne et aux contre-la-montre, avant l'arrivée à Paris le 22 juillet 2012. Il comporte trois arrivées en montagne et deux contre-la-montre avec une étape de Arc-et-Senans à Besançon (38 km) et une autre plus longue de Bonneval à Chartres (52 km).*". Ce texte est intéressant car il comporte à la fois des références spatiales et temporelles. De plus les références spatiales concernent non seulement des villes mais aussi des départements, certains explicitement et d'autres devant être calculés à partir des villes citées.

L'application souhaitée (Figure 5.23, accessible en ligne à `http://erozate.iutbayonne.univ-pau.fr/Nhan/windmash/demo/tourdefrance/`) doit se présenter selon l'interface utilisateur graphique comportant cinq afficheurs :

1. Une zone en haut à gauche contenant le texte initial ;
2. Un afficheur cartographique à droite pour présenter non seulement les départements (mis en surbrillance) mais aussi pour zoomer en avant sur les villes étapes du département ;
3. Un afficheur de type texte situé en haut au centre pour écrire le nom des départements qui sont soit cités dans le texte soit calculés à partir des villes citées ;
4. Un petit afficheur cartographique central pour zoomer sur les villes citées dans le texte ;
5. Une frise chronologique située en bas à gauche pour afficher les dates et périodes citées dans le texte.

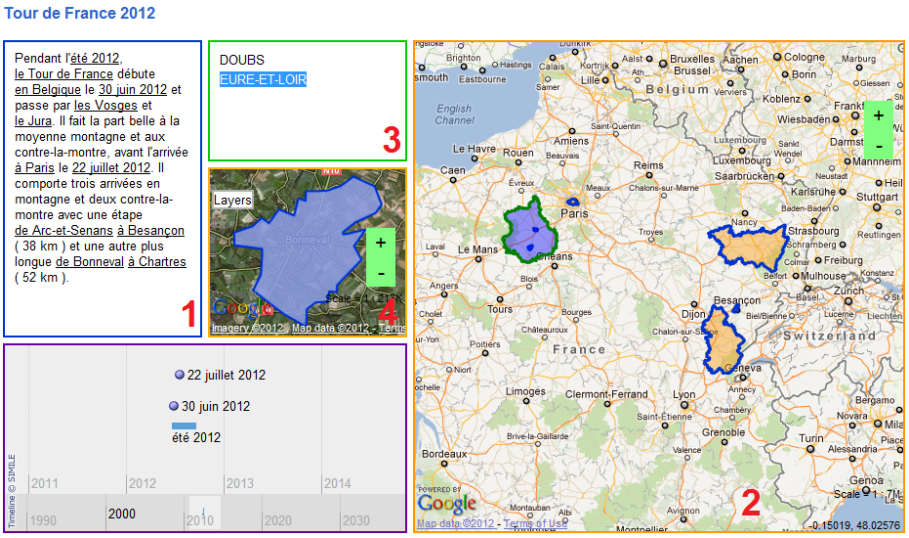


FIGURE 5.23 – Une application Web géographique sur le Tour de France 2012

Le comportement de l'application est le suivant :

- Quand un utilisateur clique sur une ville dans le texte (afficheur 1), le système doit automatiquement mettre en surbrillance le nom du département dans l'afficheur textuel 3 et dans la carte principale (afficheur 2). Le système doit aussi zoomer sur la ville dans l'afficheur cartographique 4.
- Quand un utilisateur clique sur un nom de département dans le texte principal (afficheur 1), ce nom doit s'afficher dans le petit afficheur textuel 3.

– Quand un utilisateur clique sur un nom de département de cet afficheur 3, la carte principale (afficheur 2) doit zoomer sur ce département.

Les participants ont disposé de 20 minutes pour la phase *Contenu*, de 15 minutes pour la phase *Interface*, de 30 minutes pour la phase *Interaction*. A la fin de la conception / réalisation, nous avons posé aux participants 35 questions dont 28 QCM à réponse unique et 7 questions à réponse ouverte courte. Les questionnaires que les étudiants ont du remplir étaient accessibles sur le WebCampus (https://webcampus.univ-pau.fr/courses/EVALUATIONWINDMASH/). Nous avons divisé l'expérimentation en deux parties : réaliser une application Web géographique statique (phases *Contenu* et *Interface*) et dynamique (phase *Interaction*). Une application statique contient des composants d'interface affichant seulement des contenus sans aucune possibilité d'interaction ; une application dynamique ajoute des comportements interactifs entre les composants d'interface. Pour chaque partie, nous présentons ci-dessous un tableau des questions ainsi que la synthèse des réponses des participants.

B. Résultats et discussion

Partie 1 : Réaliser une application Web géographique statique (sans interaction)

Phase *Contenu* : La figure 5.24 montre une façon de prendre en compte la phase *Contenu* pour créer les données géographiques nécessaires aux phases suivantes.

Les questions du tableau 5.25 portent sur l'évaluation de la phase *Contenu*.
Synthèse des réponses :

1. Pour le temps nécessaire à finaliser la phase *Contenu*, 60% des participants ont eu besoin de plus de 15 minutes et le temps moyen était aussi 15 minutes. Certains ont eu un problème de blocage de WINDMash mais en recommençant, ils ont refait beaucoup plus vite.
2. 100% ont été satisfaits par le résultat de la phase *Contenu*.
3. Tous les participants ont utilisé moins de 5 cycles d'essais dans la phase *Contenu*.
4. Peu de participants ont eu besoin d'aide pour finaliser la phase *Contenu* ; mais comme ils étaient dans la même salle, ils ont pu entendre la réponse des autres.
5. Un problème rencontré durant la phase *Contenu* était le blocage d'écran (perte du curseur) lors de l'affichage la fenêtre de configuration (Figure 5.26). Ce problème a eu lieu sur le navigateur Firefox.
6. 77% des participants ont été satisfaits par le positionnement des différents blocs et outils dans l'interface de la phase *Contenu* alors que 23% ne les ont pas trouvés assez bien organisés.

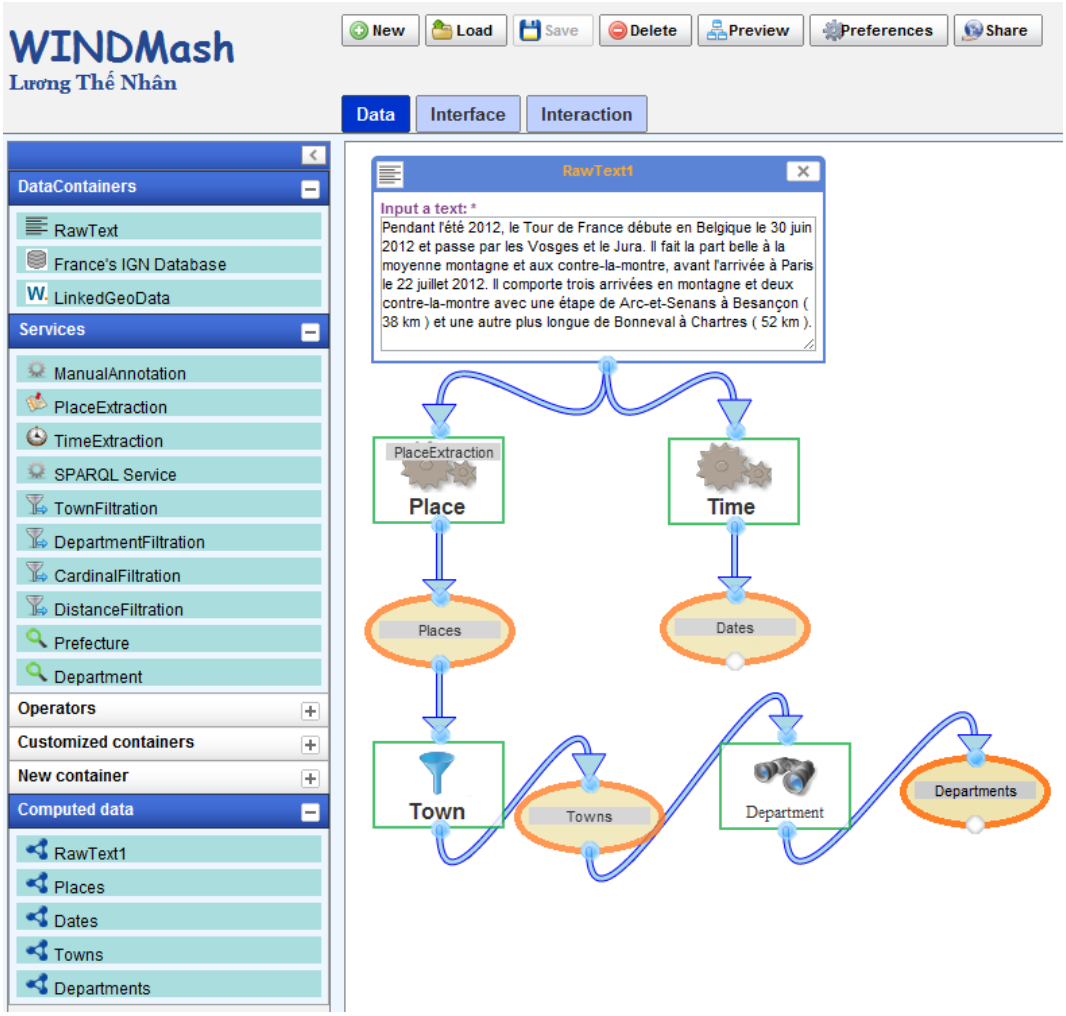


FIGURE 5.24 – Une solution envisageable de la phase *Contenu* lors de l'évaluation

7. Ces 23% des participants ont également cité quelques problèmes d'ergonomie de l'interface de la phase *Contenu* sur les termes utilisés et sur les descriptions des différents modules (aide contextuelle).
8. La note moyenne obtenue était 14 sur 20 ; une seule note était inférieure à 10.

	Questions	Type
1	Combien de temps (minutes) avez-vous passé jusqu'à finaliser la phase *Contenu* ?	Score entre 1 et 20
2	Est-ce que le résultat de la phase *Contenu* est satisfaisant (correspond au besoin) ?	Choix multiple : Tout à fait, Plutôt Oui, Plutôt Non, Pas du tout
3	Combien de cycles essais/erreurs avez-vous réalisé dans la phase *Contenu* ?	Score
4	Combien de fois avez-vous eu besoin d'aide pour finaliser la phase *Contenu* ?	Score
5	Listez les problèmes logiciels rencontrés durant la phase *Contenu*.	Ouverte
6	Le positionnement des différents blocs et outils dans l'interface de l'outil (phase *Contenu*) est-il satisfaisant ?	Choix multiple : Tout à fait, Plutôt Oui, Plutôt Non, Pas du tout
7	Qu'est-ce qui vous a manqué pour être plus efficace dans la spécification de vos besoins pour la phase *Contenu* ?	Ouverte
8	Donnez une note /20 à l'outil (en terme d'utilisabilité) permettant de concevoir la phase *Contenu*.	Score entre 1 et 20

FIGURE 5.25 – Questionnaire d'évaluation de la phase *Contenu*

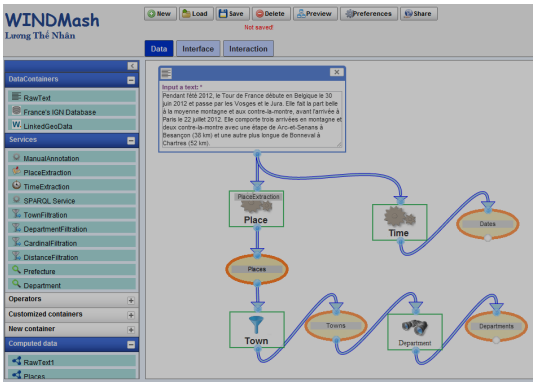


FIGURE 5.26 – Perte du curseur lors de la phase *Contenu*

Phase *Interface* : La figure 5.27 montre une façon de prendre en compte la phase *Interface* pour spécifier l'interface graphique de l'application finale et afficher les données géographiques créées depuis la phase *Contenu*.

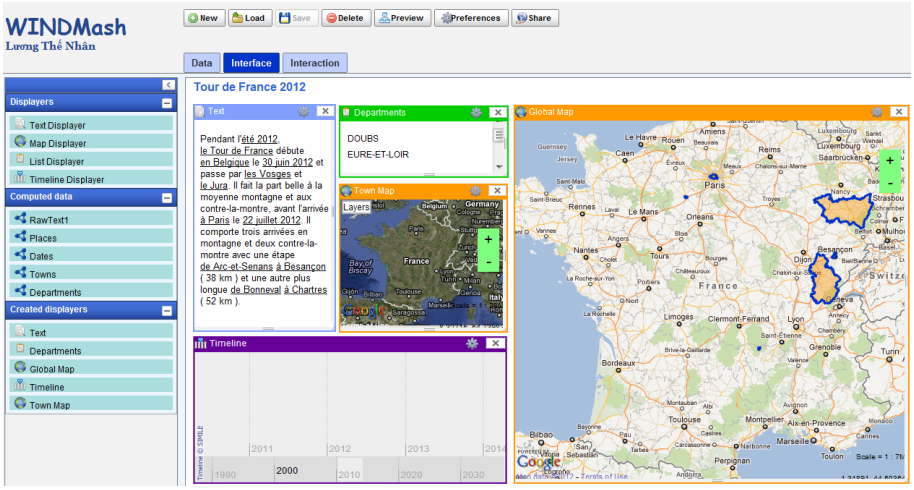


FIGURE 5.27 – Une solution envisageable de la phase *Interface* lors de l'évaluation

Les questions du tableau 5.28 portent sur la phase *Interface*.

Synthèse des réponses :

1. Lors de la phase *Interface*, 85% des participants ont facilement identifié et placé les contenus géographiques (créés depuis la phase *Contenu*) dans les composants d'interface (afficheurs).
2. Le temps moyen nécessaire était 10 minutes pour finaliser la phase *Interface*.
3. 92% ont été satisfaits par le résultat de la phase *Interface*.
4. Tous les participants ont utilisé moins de 3 cycles d'essais dans la phase *Interface*.
5. Peu de participants ont eu besoin d'aide pour finaliser la phase *Interface*.
6. Pour la phase *Interface*, il y a moins de bogues que la phase *Contenu* car elle est plus simple et demande moins de travail à faire pour le concepteur.
7. 77% des participants ont été satisfaits par le positionnement des différents blocs et outils dans l'interface de la phase *Interface* alors que 23% ne les ont pas trouvés assez bien organisés.
8. Le problème de configuration des afficheurs (changement de nom, de style...) devrait être corrigé pour que la phase *Interface* soit plus efficace.
9. Toutes les notes étaient supérieures à 10 ; la note moyenne obtenue était 15 sur 20. Au final, la phase *Interface* de WINDMash est jugée facile à manipuler.

	Questions	Type
1	Lors de la phase *Interface*, avez-vous facilement pu identifier et placer les contenus créés dans la phase *Contenu* dans les composants d'interface (afficheurs) ?	Oui / Non
2	Combien de temps (minutes) avez-vous passé jusqu'à finaliser la phase *Interface* ?	Score entre 1 et 15
3	Est-ce que le résultat de la phase *Interface* est satisfaisant (correspond au besoin) ?	Choix multiple : Tout à fait, Plutôt Oui, Plutôt Non, Pas du tout
4	Combien de cycles essais/erreurs avez-vous réalisé dans la phase *Interface* ?	Score
5	Combien de fois avez-vous eu besoin d'aide pour finaliser la phase *Interface* ?	Score
6	Listez les problèmes logiciels rencontrés durant la phase *Interface*.	Ouverte
7	Le positionnement des différents blocs et outils dans l'interface de l'outil (phase *Interface*) est-il satisfaisant ?	Choix multiple : Tout à fait, Plutôt Oui, Plutôt Non, Pas du tout
8	Qu'est-ce qui vous a manqué pour être plus efficace dans la spécification de vos besoins pour la phase *Interface* ?	Ouverte
9	Donnez une note /20 à l'outil (en terme d'utilisabilité) permettant de concevoir la phase *Interface*.	Score entre 1 et 20

FIGURE 5.28 – Questionnaire d'évaluation de la phase *Interface*

Synthèse de la Partie 1 (*Contenu* et *Interface*) (Figure 5.29) :

	Questions	Type
1	Si vous aviez à développer une autre application Web géographique statique, souhaitez-vous réutiliser cet outil ?	Oui / Non
2	Connaissez-vous d'autres outils permettant de développer des applications Web géographiques ?	Oui / Non
2bis	Si oui à la question précédente, précisez	Ouverte
3	Donnez une note /20 à l'outil (en terme d'utilisabilité) de manière générale (pour développer une application Web géographique statique).	Score entre 1 et 20

FIGURE 5.29 – Questionnaire d'évaluation de la synthèse des phases *Contenu* et *Interface*

Synthèse des réponses :

1. Tous les participants souhaitent réutiliser l'environnement WINDMash pour développer une application Web géographique statique. L'outil est donc fortement apprécié.
2. 92% des participants ont répondu qu'ils ne connaissent pas d'autres outils permettant de développer des applications Web géographiques statiques (un participant n'a pas répondu). Pour eux, WINDMash est original.
3. La note moyenne obtenue était environ de 14 sur 20. Nous en concluons que WINDMash est utilisable par des utilisateurs non-informaticiens pour créer des applications Web géographiques statiques (sans interaction).

Partie 2 : Réaliser une application Web géographique avec de l'interactivité

Phase *Interaction* : La figure 5.30 montre une façon de prendre en compte la phase *Interaction* pour mettre en œuvre les comportements de l'application finale. Plusieurs scénarios (et donc diagrammes) peuvent toutefois convenir pour décrire les comportements souhaités. De plus, pour un même scénario, la richesse du langage visuel permet plusieurs variantes possibles comme par exemple de découpage d'un scénario avec plusieurs réactions externes en autant de scénarios avec une seule réaction externe.

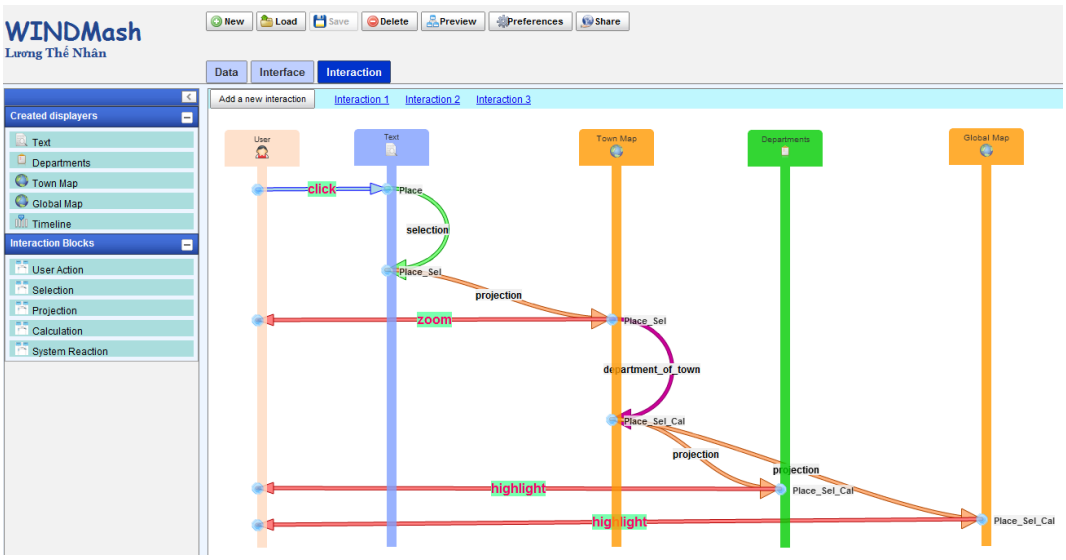


FIGURE 5.30 – Une solution envisageable de la phase *Interaction* lors de l'évaluation

Les questions du tableau 5.31 portent sur la phase *Interaction*.

N	Questions	Type
1	Lors de la phase *Interaction*, avez-vous facilement pu identifier et placer les afficheurs créés dans la phase *Interface* ?	Oui / Non
2	Combien de temps (minutes) avez-vous passé jusqu'à finaliser la phase *Interaction* ?	Score entre 1 et 30
3	Pour modéliser le comportement de l'application, combien de schémas d'interaction avez-vous défini ?	Score
4	Est-ce que le résultat de la phase *Interaction* est satisfaisant (correspond au besoin) ?	Choix multiple : Tout à fait, Plutôt Oui, Plutôt Non, Pas du tout
5	Combien de cycles essais/erreurs avez-vous réalisé dans la phase *Interaction* ?	Score
6	Combien de fois avez-vous eu besoin d'aide pour finaliser la phase *Interaction* ?	Score
7	Listez les problèmes logiciels rencontrés durant la phase *Interaction*.	Ouverte
8	Le positionnement des différents blocs et outils dans l'interface de l'outil (phase *Interaction*) est-il satisfaisant ?	Choix multiple : Tout à fait, Plutôt Oui, Plutôt Non, Pas du tout
9	Qu'est-ce qui vous a manqué pour être plus efficace dans la spécification de vos besoins pour la phase *Interaction* ?	Ouverte
10	Donnez une note /20 à l'outil (en terme d'utilisabilité) permettant de concevoir la phase *Interaction*.	Score entre 1 et 20

FIGURE 5.31 – Questionnaire d'évaluation de la phase *Interaction*

Synthèse des réponses :

1. Lors de la phase *Interaction*, 55% des participants ont facilement identifié et placé les afficheurs créés depuis la phase *Interface*. Le taux était très mitigé à cause de problème de changements de nom des afficheurs (configuration) lors de la phase *Interface*.
2. Les temps nécessaires pour finaliser la phase *Interaction* étaient disparates et supérieurs à 10 minutes.
3. Les réponses n'étaient pas significatives à cause des bogues.
4. La plupart des participants n'ont pas été satisfaits par le résultat de la phase *Interaction*.

5. Le nombre de cycles d'essais dans la phase *Interaction* était bien supérieur à celui des autres phases.
6. Il n'y avait que 7 réponses mais elles n'étaient pas significatives.
7. Pour la phase *Interaction*, il y a plus de bogues que les autres phases à cause des plusieurs problèmes (enregistrement, effets de bord sur les autres phases...). De plus, le diagramme de séquence UML était un concept inconnu pour les participants. Cela signifie que l'auto-formation de la phase *Interaction* était insuffisante et qu'il est nécessaire de fournir une bonne aide contextuelle (par exemple : une vidéo de bonne qualité).
8. 63% des participants ont été satisfaits par le positionnement des différents blocs et outils dans l'interface de la phase *Interaction* alors que 37% ne les ont pas trouvés assez bien organisés.
9. Il y avait plusieurs améliorations listées pour que la phase *Interaction* soit plus efficace.
10. Les participants ont noté et la note d'utilisabilité de l'outil moyenne obtenue était environ de 10 sur 20. Pour nous, cela veut dire que la phase *Interaction* était plus difficile à utiliser pour les participants non-informaticiens par rapport aux phases *Contenu* et *Interface*.

Synthèse finale de l'évaluation du prototype WINDMash (Figure 5.32) :

N	Questions	Type
1	Le processus actuel en 3 phases (*Contenu* <-> *Interface* <-> *Interaction*) facilite-t-il l'amélioration progressive de la conception effectuée de l'application ?	Oui / Non
2	Ce processus pour lier les objets d'une phase à l'autre vous a-t-il paru naturel ou pas ?	Choix multiple : Tout à fait, Plutôt Oui, Plutôt Non, Pas du tout
3	Imagineriez-vous d'autres moyens de lier les différentes phases et d'autres chronologies d'enchainement des phases ?	Ouverte
4	Connaissez-vous d'autres outils permettant de développer des applications Web géographiques dynamiques ?	Oui / Non
4bis	Si oui à la question précédente, précisez	Ouverte
5	Donnez une note /20 (en terme d'utilisabilité) au prototype WINDMash de manière générale.	Score entre 1 et 20

FIGURE 5.32 – Questionnaire d'évaluation de la synthèse finale

Synthèse des réponses :

1. 75% des participants ont répondu que le processus actuel en 3 phases (*Contenu* <-> *Interface* <-> *Interaction*) facilite l'amélioration progressive de la conception effectuée de l'application.
2. 50% ont trouvé le processus naturel pour lier les objets d'une phase à l'autre.
3. Un seul participant a proposé de combiner les phases *Interface* et *Interaction*.
4. 100% des participants ont répondu qu'ils ne connaissent pas d'autres outils permettant de développer des applications Web géographiques dynamiques.
5. Seuls 7 ont fourni une note qui était comprise entre 10 et 16 sur 20. En général, cette expérimentation nous a permis d'évaluer complètement notre environnement-auteur WINDMash. Pour nous, il sera utilisable pour les utilisateurs non-informaticiens mais il faut au préalable corriger les bogues identifiés.

5.2.3 Évaluation de la couverture et de la robustesse de l'API WIND

L'API WIND opérationnalise l'ensemble (et au-delà) des interactions que la plateforme WINDMash est capable de décrire. Depuis 2009, au fil des projets pour lesquels elle a été mise en œuvre, cette API a beaucoup évolué dans ses fonctionnalités et dans ses usages. Les prochains paragraphes présentent les nombreux projets basés sur cette API, le contexte d'usage de l'API et les principales fonctionnalités de l'API mises en œuvre dans le projet.

5.2.3.1 Projet PIND (Photo INteraction Design)

Le projet PIND porte sur la conception d'une galerie de photos interactives dans le cadre de la collaboration pour un projet du Conseil Général des Pyrénées-Atlantiques avec la Ligue de l'Enseignement du 64. Ce projet s'est déroulé en 2009 - 2010.

Le projet a pour but de créer des applications Web qui servent à des classes de découverte dans les Pyrénées-Atlantiques.

La figure 5.33 montre une application développée dans le projet PIND. L'interface de l'application se compose de quatre composants : trois (textuel, cartographique, calendaire) issus d'une ancienne version de l'API WIND et un nouveau composant photographique conçu pendant le projet PIND. Cette application [52] permet de synchroniser les composants textuel, cartographique, calendrier et photographique.

Son comportement est le suivant :
– Quand l'utilisateur clique sur un nom de lieu cité dans le composant textuel, la carte effectue un zoom avant sur ce lieu et le calendrier se centre sur la date correspondant à la visite du lieu (si la date est connue).

52. http://erozate.iutbayonne.univ-pau.fr/Nhan/pind/demo/DemoWIND_PIND.html

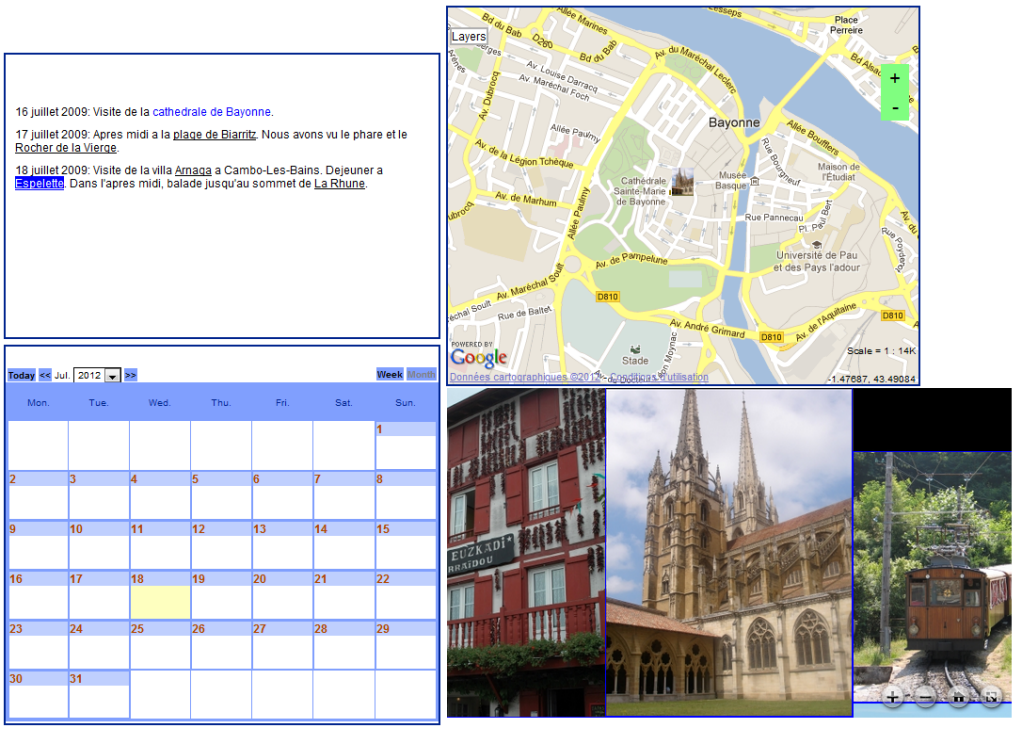


FIGURE 5.33 – Projet PIND

– Quand l'utilisateur clique sur une petite image située sur la carte, le composant photographique montre cette photo en entier. De plus, ce composant photographique dispose des fonctionnalités de manipulation : zoom avant et zoom arrière sur la photo, déplacement sur la photo par "cliquer-glisser", annotation sur une partie de la photo avec une forme géométrique précise.

A l'issue du projet PIND, nous avons ajouté la partie de gestion photographique dans notre modèle unifié et l'API WIND (partie en bleu dans la figure 5.34). Nous n'avons pas mentionné la partie photographique dans les chapitres précédents parce que cette partie ne relève pas (selon notre définition) d'une information géographique (espace, temps, thématique). Toutefois, l'intégration du module PIND à l'API WIND a permis d'élargir le champ des fonctionnalités qu'auront à leur disposition les utilisateurs pour créer des applications Web. L'utilisation d'un module photographique dans une application rend celle-ci plus interactive. De plus, la réussite de cette intégration permet de prouver la qualité existant dans l'API WIND et qu'il est possible de la faire évoluer.

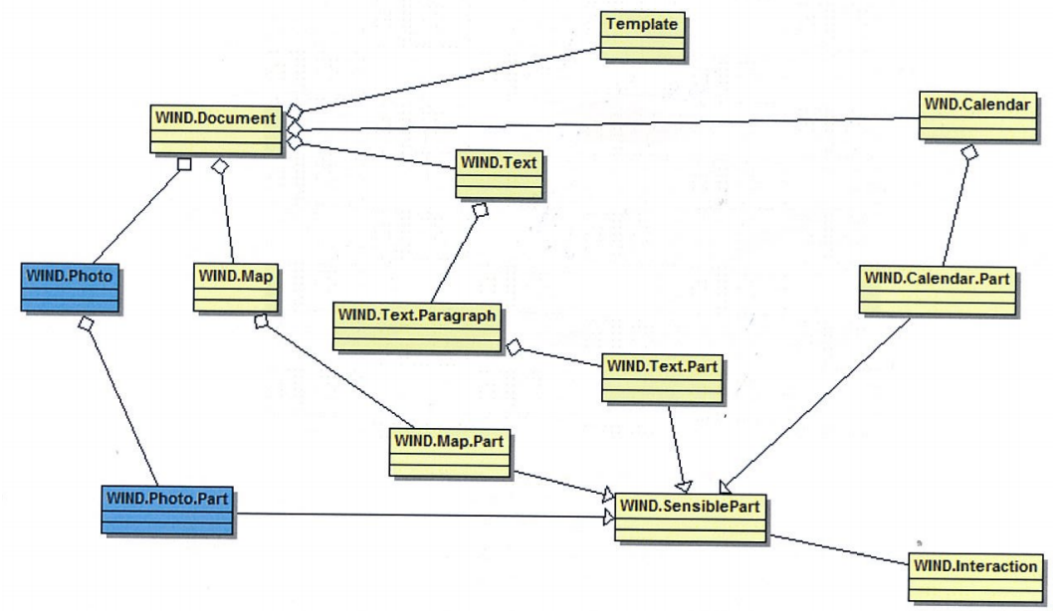


FIGURE 5.34 – Évolution de l'API WIND après le projet PIND

5.2.3.2 Projet GeoText2Map

En 2010-2011, le projet GeoText2Map [53] a permis de mettre en place un outil Web d'annotation de lieux cités dans un texte sur un fond cartographique (Figure 5.35). Ce projet a été effectué en collaboration avec la société Geocime [54].

Dans ce projet, les lieux contenus dans le texte sont automatiquement reconnus (via l'appel à un service Web dédié) et l'utilisateur a la possibilité de confirmer ou de modifier la proposition via des outils cartographiques.

La figure 5.35 illustre l'interface principale de l'outil GeoText2Map. Elle se divise en deux parties : textuelle (à gauche) et cartographique (à droite). Le texte a été traité pour marquer automatiquement les noms des lieux et la carte affiche les punaises sur ces lieux. Pour un lieu, un couple constitué d'un nom de lieu sur le texte, d'une ou plusieurs punaises relatives à ce lieu et positionnées sur la carte peut être considéré comme une annotation sur ce lieu. Grâce à cette interface interactive, l'utilisateur peut valider, modifier, supprimer ou ajouter les annotations.

Les parties textuelle et cartographique de l'API WIND ont été utilisées pour mettre en œuvre ce projet. Grâce à l'API WIND, le temps de codage a été réduit. À partir du projet GeoText2Map, nous avons poursuivi le travail pour focaliser les annotations

53. http://erozate.iutbayonne.univ-pau.fr/geotext2map/
54. http://geocime.fr/

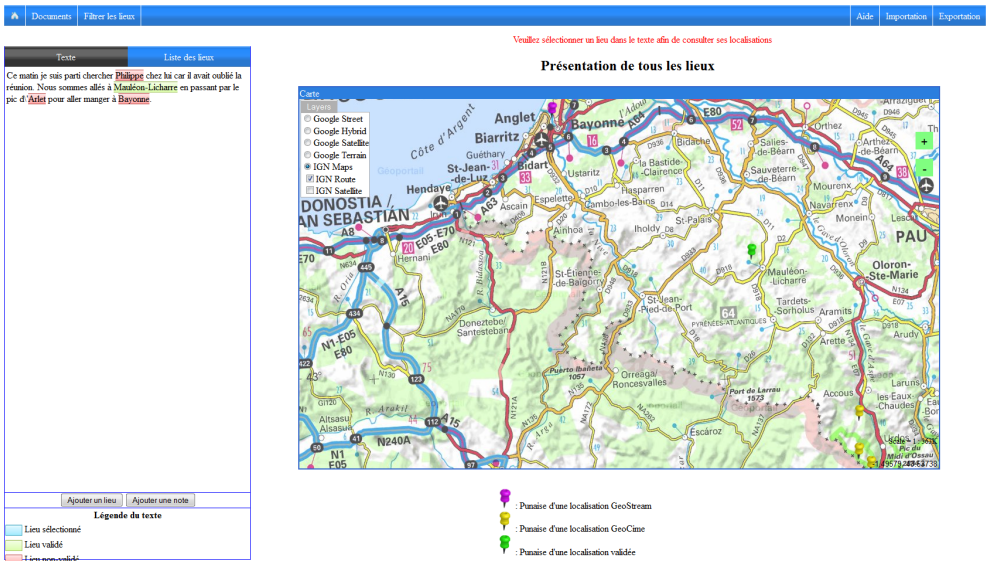


FIGURE 5.35 – Projet GeoText2Map

selon les deux autres facettes (temps et thème) et pour les visualiser aussi sur une frise temporelle.

5.2.3.3 Projet Frise Chronologique

Durant l'année universitaire 2009-2010, l'équipe T2i a travaillé avec l'association "La Boîte à fabriques" [55] pour définir le cahier des charges d'une application permettant :

- aux enseignants de créer des frises chronologiques intégrant des données multimédias ;
- aux élèves de consulter cette frise soit avec des repères chronologiques soit en effectuant une recherche thématique.

Le projet [56] a permis de définir les fonctionnalités à mettre en œuvre tant du point de vue de l'enseignant-concepteur que de celui des élèves.

Durant l'année universitaire 2010-2011, le projet a été repris avec pour ambition la création d'une application Web de conception / déploiement de frises multimédias qui soit totalement opérationnelle.

A la suite de ce projet, l'API WIND a évolué en complétant la partie chronologique qui se base sur l'API Timeline [57].

55. http://www.laboiteafabriques.com
56. http://erozate.iutbayonne.univ-pau.fr/frises/
57. http://www.simile-widgets.org/timeline/

FIGURE 5.36 – Projet Frise Chronologique

5.2.3.4 Projet Saisie Cartographique

En 2011 - 2012, notre équipe T2i a développé un outil Web [58] "Saisie Cartographique" qui serait exploitable dans le cadre d'un projet nommé "Le Dauphin" déposé à l'ANR avec le partenaire IKER [59]. L'objectif du projet consiste à imaginer un module Web permettant à un utilisateur de faire des saisies assistées sur une carte interactive.

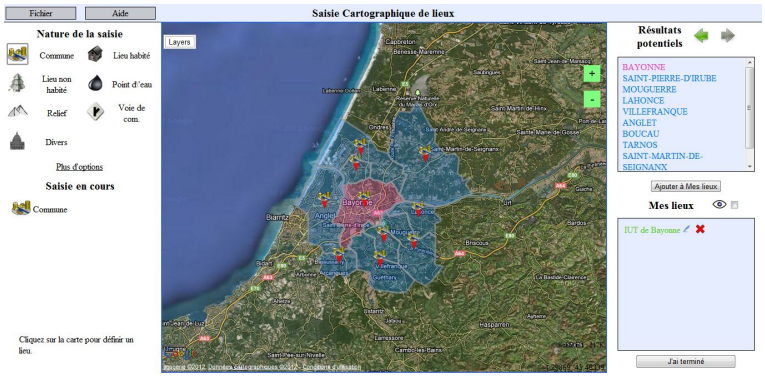


FIGURE 5.37 – Projet Saisie Cartographique

58. http://erozate.iutbayonne.univ-pau.fr/owsc/
59. http://www.iker.cnrs.fr/

L'interface de l'outil (Figure 5.37) se compose de trois parties :
- La partie à gauche contient les natures géographiques à choisir par l'utilisateur.
- La carte au milieu permet à l'utilisateur de chercher toutes les entités géographiques correspondant aux natures choisies autour d'un endroit. Par exemple, si l'utilisateur a choisi "*Commune*" comme la nature à chercher, lors d'un clic sur Bayonne, la carte affiche les communes autour de Bayonne.
- Les noms des communes trouvées sont listés dans la partie à droite. Ces propositions peuvent être ensuite validées (ou pas) par l'utilisateur.

L'enjeu consiste à proposer des modalités permettant de réaliser des saisies précises via la souris : désigner une ville, une rivière, un chemin, un itinéraire... La saisie à la souris devra produire en résultat le nom de l'entité géographique désignée par l'utilisateur, par exemple : Ville de Mauléon-Licharre, Forêt de Sare, Aquitaine, rivière de la Nive, plage des dunes à Anglet...

L'API a été utilisée pour implémenter la partie cartographique de cet outil. Elle a permis de définir la taille de la carte, le type de fonds cartographiques utilisés (plan, satellite...). L'API WIND a aussi permis de proposer les différentes représentations possibles (point, ligne, polygone, marqueur...) d'une entité géographique à l'utilisateur. Grâce à l'API WIND, toutes les possibilités ont été mises en place dans un temps de codage très limité.

5.3 Limite des travaux et perspectives

L'analyse que nous venons de faire des solutions proposées (Section 5.1) et les évaluations de WINDMash et de l'API WIND conduites par ailleurs auprès des utilisateurs finaux (Section 5.2) nous amènent à rendre compte des limites de ce travail. Nous identifions ici certains des verrous scientifiques non résolus, des voies d'amélioration qui constituent autant de pistes de recherche et développement pour des futurs travaux.

5.3.1 Du point de vue de l'Ingénierie Dirigée par les Modèles

Le travail de thèse que nous avons conduit trouve ses fondements dans l'Ingénierie Dirigée par les Modèles puisqu'il s'agit de permettre à des concepteurs de produire des modèles (*cf.* les trois phases de WINDMash) précisant la logique métier d'une application Web géographique puis de projeter ces modèles sur une plateforme technologique via des transformations successives. Dans le cas de WINDMash, l'expression des modèles par le concepteur se fait par des métaphores visuelles ; les modèles sont stockés en RDF, fusionnés, puis transformés en code HTML+JavaScript exploitant le potentiel de l'API WIND qu'un navigateur Web est capable d'interpréter.

L'originalité et le point fort de notre approche est d'avoir choisi de modéliser la logique de l'application dans un vocabulaire RDF et non dans des langages classiquement

utilisés en Génie Logiciel tels que XMI [60] [Gro11b] ou EMF [61] [Ecl11]. Ce choix a clairement facilité la fusion des modèles RDF mais aussi la génération de code pour nos applications.

En revanche, ce choix ne nous a pas incités à nous rapprocher des standards qu'utilisent les acteurs du Génie Logiciel pour transformer des modèles en code : notre approche est *ad hoc*, elle ne s'appuie sur aucun langage de transformation de modèles tel que ATL [62] [BEGP03] ou QVT [63] [Gro11a] qui, s'ils étaient utilisés, offriraient des garanties supplémentaires sur deux points particuliers :

- Capacité à vérifier la cohérence de plusieurs interactions spécifiées successivement par un concepteur. A ce jour, WINDMash n'est pas capable de détecter que le concepteur a décrit des comportements incohérents. Une piste à explorer consiste à transformer nos pseudos diagrammes de séquence, chacun correspondant à un scénario/chemin d'exécution particulier, en pseudos diagrammes états-transitions. Ainsi le caractère déterministe du diagramme généré pourrait être vérifié et, dans l'affirmative, généré sous forme de code exécutable.
- Capacité à garantir que le code généré est conforme à la spécification effectuée par un concepteur grâce au langage visuel.

Ces deux limites ont été identifiées rapidement au cours de la thèse mais, pour les dépasser, nous aurions dû conduire un travail important pour nous les approprier. Nous avons préféré centrer les développements sur des actions visant à rendre l'environnement aussi utilisable que possible (richesse des fonctionnalités pour chaque phase et couverture du processus complet de modélisation). En revanche, compte tenu de l'avancement actuel du prototype, WINDMash nous paraît maintenant pouvoir faire l'objet d'un travail important en Génie Logiciel (plus particulièrement en IDM), les modèles que WINDMash est capable de produire allant bien au-delà des jeux d'essais triviaux qui sont utilisés classiquement pour démontrer la puissance des langages de transformation et de vérification de modèles.

5.3.2 Du point de vue de l'utilisabilité de l'environnement WINDMash

Malgré l'importance des efforts conduits pour faire de WINDMash bien plus qu'un démonstrateur des idées élaborées à l'occasion d'une thèse, les différentes évaluations que nous avons conduites montrent que l'environnement WINDMash nécessite encore plusieurs types de travaux pour qu'il réponde à l'objectif à long terme que nous nous sommes fixés. Il devrait être un environnement de conception d'applications Web pédagogiques que des enseignants pourraient utiliser dans leurs classes pour des expérimentations d'applications en situation réelle avec leurs élèves.

60. XML Metadata Interchange
61. Eclipse Modeling Framework
62. Atlas Transformation Language
63. Query/Views and Tranformations

Nos expériences diverses ont montré que les bogues résiduels du prototype doivent être totalement supprimés afin de pouvoir offrir l'environnement à des non-informaticiens. Bien qu'ayant fait l'objet d'un travail de fond assez systématique, la richesse des fonctionnalités de l'environnement WINDMash (et de son API support) rend ce travail long et difficile. WINDMash n'est cependant déjà plus un démonstrateur mais un prototype faisant l'objet de versions successives (de la version 0.1 qui faisait 500 lignes de code à la version 5.x qui représente 7000 lignes de code de plusieurs centaines de fonctions et 10000 lignes de code pour l'API WIND) que nous envisageons d'industrialiser. Ce travail devait se faire dans le cadre d'un projet européen nommé "eGeo" qui a été déposé en juin dernier mais n'a malheureusement pas été accepté. L'acceptation de ce type de projet étant très difficile à obtenir vu le taux de sélectivité, nous poursuivons malgré tout nos travaux à une plus petit échelle avec des écoles locales ayant participé au projet. Nous sommes également en relation avec la cellule de valorisation de la Recherche de notre Université pour un dépôt de l'API WIND sous forme de licence libre par exemple.

Au-delà de cette correction de bogues, il est nécessaire d'aller plus avant dans la spécialisation de l'environnement WINDMash pour en faire un outil réellement dédié à la conception pédagogique d'applications Web en géographie. Trois pistes sont déjà étudiées, comme en témoignent les prochains paragraphes.

La première piste est déjà partiellement implémentée. L'idée est d'ajouter une phase à WINDMash permettant à un enseignant-concepteur de choisir dans une liste voire de définir les objectifs pédagogiques qu'il vise à travers l'application WINDMash qu'il est en train de concevoir. Une base de données spécifique a été implémentée lors du stage de M2R de P. Nodenot [Nod11]. Elle indexe l'ensemble du socle de compétences, depuis l'école maternelle jusqu'à la fin du secondaire ; elle offre aussi des fonctionnalités de recherche pour accéder à des éléments particuliers de ce socle. Enfin, elle permet d'accéder à des fiches de préparation voire des applications WINDMash sur la base d'une sélection de critères de recherche. Il reste à intégrer cette fonctionnalité dans une nouvelle version de l'environnement WINDMash. Cette piste pourrait être menée plus à fond en donnant la possibilité à l'enseignant d'associer à chaque interaction décrite des justifications sur le pourquoi de telle réaction particulière de l'application produite. Techniquement, ceci ne pose pas de problème mais nos échanges avec les enseignants ont montré qu'ils avaient du mal à formaliser le pourquoi de tel ou tel comportement d'une application qu'il jugeait *a priori* utile pour atteindre un objectif d'apprentissage donné.

La deuxième piste fait actuellement l'objet d'un travail de thèse au laboratoire. Il s'agit de doter WINDMash d'une phase supplémentaire centrée sur le diagnostic du comportement de l'apprenant en train d'utiliser l'application WINDMash conçue. Partant du principe que les seuls moyens d'interaction dont disposera l'apprenant sont ceux qui sont décrits via la phase *Interaction* de WINDMash, il devient possible de décrire des patrons d'interactions typiques de tel ou tel comportement plausible de l'apprenant : des comportements correspondant à des séquences d'interactions jugées correctes ou pas,

optimales ou pas, erronées mais intéressantes à détecter... Le travail de thèse de Jean-François Boullier vise à proposer un langage visuel pour décrire ces patrons d'interaction, pour qualifier chacun de ces patrons d'un point de vue pédagogique, pour les simuler et les valider. Chaque application pédagogique générée sera alors capable de produire des traces d'interactions qualifiées d'un point de vue pédagogique. Cette étape constitue le premier pas vers la mise en place d'un système de tutorat automatisé tenant compte du comportement réel de l'apprenant.

Enfin, la troisième piste de travail vise à étendre le modèle événementiel pour pouvoir enregistrer des événements significatifs, sans qu'il y ait pour autant de réaction immédiate de la part de l'application utilisée par l'apprenant. En effet, dans la version actuelle du modèle d'interaction, nous ne considérons que les interactions dans lesquelles le système réagit immédiatement à l'action de l'utilisateur. Une réaction du système peut être simple (Figure 5.38(a)) ou composée de plusieurs réactions (Figure 5.38(b)) mais elle intervient dès qu'une action attendue de l'utilisateur se produit. Afin de pouvoir mettre en œuvre d'autres types de réactions, l'idée serait de permettre aux concepteurs de définir des interactions pour lesquelles le système réagit après une séquence de plusieurs actions spécifiques de l'utilisateur (Figure 5.38(c)). Pour prendre en compte ce type d'interaction, il nous faut améliorer le modèle d'interaction en intégrant le concept d'"*événement agrégé*". Un événement agrégé serait déclenché lorsque l'utilisateur aurait produit un ensemble d'événements (ensemble ordonné ou non-ordonné, cycle d'événements...). A ce jour nous n'avons pas encore examiné la difficulté de mise en œuvre de tels mécanismes bien que les travaux de David Harel autour de l'environnement Play-Engine [HM03] nous incitent à être optimistes en la matière.

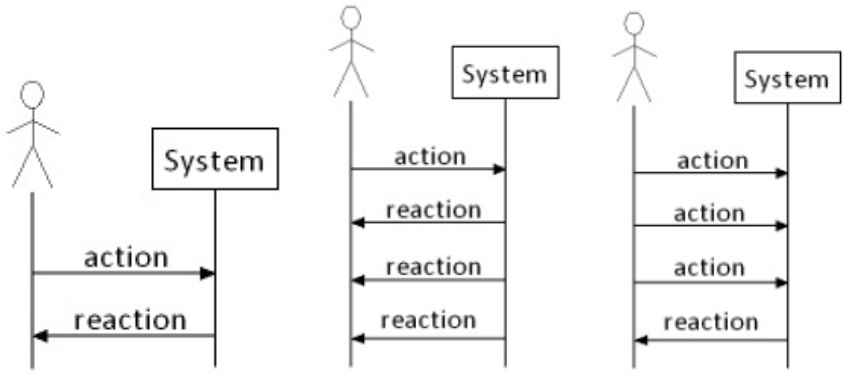


FIGURE 5.38 – Différents types d'interactions

Au final, nous pensons que les trois pistes de travail identifiées dans cette section pourraient contribuer à nous rapprocher des objectifs ambitieux énoncés au début de ce manuscrit, à savoir amener des enseignants à utiliser WINDMash de manière réellement efficiente pour concevoir des applications Web géographiques à finalité pédagogique [64]. Un grand pas a été fait pour démontrer la faisabilité de ce projet bien qu'il reste plusieurs verrous scientifiques et techniques à lever pour prouver le caractère opérationnel de l'approche proposée.

5.3.3 Généricité de l'approche et des contributions

Les perspectives que nous venons de proposer ajoutées aux résultats obtenus durant la thèse permettent de déterminer assez précisément le potentiel de cette approche qui a été mise en œuvre pour répondre avant tout aux objectifs de l'équipe T2i. L'approche et les technologies développées nous paraissent cependant couvrir un spectre plus large. Cette section se propose donc de montrer en quoi cette approche nous paraît générique mais aussi quels sont les pas restant à franchir pour étendre la couverture des besoins.

5.3.3.1 Besoins non identifiés initialement mais couverts par l'approche proposée

Tout au long de la thèse, nous avons principalement exploité les contenus géographiques rendus disponibles par les services d'extraction développés par l'équipe T2i. Le modèle que nous avons développé est cependant capable d'exploiter bien d'autres contenus géographiques, ce que nous avons démontré en exploitant les données issues de l'Open Data, des données issues des bases de données IGN...

Par ailleurs, même si nous ne l'avions pas anticipé, nous avons été amenés à manipuler via l'API WIND :

- des contenus ne contenant pas forcément des informations géographiques, par exemple des photos issues d'espaces partagés tels Flickr,
- des composants d'interface non dédiés à la visualisation d'informations géographiques tels des visualisateurs d'images en très haute définition, des éditeurs d'annotation manuelle...

Sur la base du modèle proposé en section 4.3, nous avons réussi à coder des applications WIND offrant des interactions qui mêlaient ces divers composants et contenus, ce qui démontre bien le caractère générique de ce modèle et de son API support. Le modèle d'interaction s'est lui aussi enrichi au fur et à mesure de l'avancement de la thèse, couvrant toujours plus de possibilités : drag and drop, édition/modification de textes, édition/modification de cartes géographiques...

64. Les applications pédagogiques visées doivent permettre à l'apprenant de réaliser des interactions finalisées pour conduire une tâche non-collaborative de résolution de problèmes basée sur la lecture active de documents géographiques.

5.3.3.2 Besoins identifiés mais non couverts actuellement par l'approche proposée

Nous n'avons pas encore démontré que notre modèle est capable de supporter l'exécution d'applications exploitant des fonctionnalités souvent offertes par des périphériques mobiles : positionnement GPS de l'utilisateur, orientation de l'utilisateur... Développer avec WINDMash des d'applications exploitant ce type de données ne nous paraît pas pouvoir mettre notre modèle en défaut ; déployer des applications sur périphérique mobile est par contre un tout autre chantier qui nécessitera de revoir le modèle d'interface (templates spécifiques pour appareils de type Smatphone, PDA...) mais surtout le moteur de génération de code.

A ce jour, nous n'avons pas non plus traité certains problèmes techniques et scientifiques dont la résolution permettrait de couvrir un spectre plus large d'applications géographiques. L'API WIND n'est par exemple pas dotée des fonctionnalités permettant de déterminer, lorsque deux contenus sont superposés dans un composant d'interface, lequel des deux doit être activé prioritairement. Le langage visuel proposé ne traite donc pas de la problématique d'interactions qui s'appuieraient sur plusieurs couches de contenus superposés susceptibles de donner lieu à des interactions avec l'utilisateur bien qu'un nombre non négligeable d'applications géographiques exploitent ce principe. Plutôt que de développer nous-mêmes les mécanismes nécessaires, nous avons identifié une SSII locale qui a déjà développé des solutions à ces problèmes compatibles avec la technologie choisie pour l'API WIND. Nous examinons aussi la possibilité de monter des coopérations avec cette société de service sur ce sujet particulier.

Enfin, nous devons montrer que les applications géographiques générées par l'environnement WINDMash peuvent s'insérer dans un cadre applicatif plus large. En effet, nous avons volontairement limité le spectre de ces travaux à la description des interactions offertes pour une activité donnée, refusant par là même de considérer dans cette thèse la problématique de la chronologie d'activités s'enchaînant selon un scénario donné :

- Pour ce qui est de l'enchainement d'activités selon un scénario prédéfini, les travaux que nous avions conduits auparavant dans le cadre de l'approche CPM [Laf04] nous incitent à être très optimistes sur les chances de succès à court terme.
- Pour des scénarios dont les activités doivent s'enchaîner en fonction de conditions portant sur des activités précédentes (en fonction de critères cognitifs notamment), rien n'existe actuellement dans les applications générées par WINDMash pour tenir compte, dans la mise en œuvre d'une interaction, d'éléments issus d'activités précédentes. Par ailleurs, même si l'un de nos doctorants actuels fait porter ses travaux sur le diagnostic du comportement d'un utilisateur manipulant des applications générées par WINDMash, les travaux sont insuffisamment avancés pour affirmer qu'il sera possible de spécifier les comportements à observer puis de les détecter à l'exécution.

La section suivante liste d'autres perspectives pour élargir le champ d'utilisation de nos travaux à d'autres acteurs et d'autres usages.

5.4 Autres pistes de travail : vers une plateforme WINDMash ouverte et modulaire

A ce jour, le prototype WINDMash peut évoluer vers différentes directions. Comme nous l'avons vu dans la section précédente, nous souhaitons spécialiser cet environnement vers des usages pédagogiques, mais il nous semble aussi pertinent d'étendre ses champs d'applications en facilitant l'installation d'extensions grâce à de modules spécifiques. C'est d'ailleurs ce qui a été déjà réalisé lorsque nos collègues de l'équipe T2i ont souhaité pouvoir disposer d'un outil de test des briques logicielles qu'ils avaient construites (briques d'extraction automatique des itinéraires d'un texte, briques d'annotation manuelle de textes...). Ces briques ont été rendues disponibles et paramétrables via WINDMash, cette intégration permettant à nos collègues de facilement tester et présenter les résultats de leurs travaux. L'idée est d'établir les spécifications que devront respecter les extensions à intégrer dans WINDMash et d'ajouter une fonctionnalité d'import de ces extensions à la manière des plugins de Firefox par exemple. Bien que cette fonctionnalité ne soit pas mise en place à ce jour, nous examinons ci-dessous quelques extensions nous paraissant particulièrement intéressantes.

5.4.1 De nouveaux supports de déploiement pour les applications géographiques décrites

Dans nos travaux actuels, nous visons le déploiement d'applications Web utilisables sur un ordinateur. Compte tenu de la multiplication des dispositifs mobiles types tablettes et smartphones, il serait pertinent de proposer une phase *Interface* spécifique pour ce type de périphérique afin de tenir compte des caractéristiques de leurs écrans et de leurs fonctionnalités. Dans le cadre du projet ANR MOANO coordonné par l'équipe T2i du LIUPPA, le besoin s'est fait sentir de doter le public visitant un parc naturel (le parc MOSAIC à Lille) mais aussi les jardiniers chargés d'applications dédiées de l'entretien de ce parc de 33 hectares. Si la proposition d'une extension offrant au concepteur une phase *Interface* spécifique ne pose guère de problème, il est plus complexe de garantir que les interactions possibles tiendront compte des caractéristiques du smartphone utilisé. Cette capacité nécessitant d'étendre WINDMash pour pouvoir gérer des interactions du type "secouer", "glisser" ou d'états du système relevant par exemple de la position GPS du smartphone. Pour tenir compte de ces fonctionnalités système, il n'est pas sûr qu'une génération de code en HTML+JavaScript soit adaptée.

5.4.2 De nouveaux services d'indexation

Jusqu'à présent, les services d'indexation automatique de textes qui ont été intégrés dans WINDMash sont des services Web développés par l'équipe T2i, ce qui nous

permet d'en maîtriser totalement les mécanismes. Nous souhaiterions que des développeurs externes puissent utiliser WINDMash pour exploiter leurs propres services Web d'indexation de textes et ainsi valoriser leurs services en créant des applications Web manipulant leurs contenus indexés. Cet objectif soulève deux problèmes :

- Par expérience, nous savons que les services d'indexation sont construits avec des règles de tokenisation des textes qui leur sont propres, ces règles étant *a priori* différentes de celles utilisées par WINDMash pour segmenter les textes à valoriser.
- Par ailleurs, si l'on veut mêler (via les opérateurs offerts par WINDMash) les résultats de services d'indexation issus de fournisseurs différents, il est nécessaire de les gérer sur toute la durée du processus de conception d'une application.

Un travail a été entrepris pour trouver des solutions à ce problème sur la base d'extensions à réaliser sur les résultats des travaux de Pierre Loustau [LNG09].

Par ailleurs, actuellement, notre prototype ne traite que des contenus textuels. Considérant que le modèle de *Contenu* présenté dans le chapitre 3 est suffisamment générique pour être étendu afin de traiter des contenus multimédias (vidéos, audios et images). Nous avons entrepris quelques expérimentations d'indexation d'images Flickr.

Si nous parvenons à proposer des interactions sur des contenus multimédias, cela permettra à plusieurs collègues de l'équipe T2i de s'appuyer sur les capacités de WINDMash pour valoriser les contenus et services sur lesquels ils font porter leur recherche.

5.4.3 Vers des interactions à plus fort contenu sémantique

Afin de pouvoir décrire des interactions plus riches, il nous paraît important d'introduire dans le modèle d'interaction une couche sémantique permettant de décrire des interactions avec un plus haut niveau d'abstraction. Nous avons initié de premiers travaux pour pouvoir utiliser la sémantique spatiale contenue dans des textes pour décrire les événements et réactions.

Pour étendre les éléments sémantiques pouvant être pris en compte par le système, WINDMash pourrait exploiter une ontologie géographique (Figure 5.39). Cette ontologie sert de référence pour étendre par déduction la sémantique d'un élément spatial. Par exemple, le système serait en mesure d'exploiter les liens d'héritage présents dans l'ontologie pour déduire que si Mauléon-Licharre est une commune, alors Mauléon-Licharre est aussi un lieu-dit habité ou, plus généralement, un lieu.

Les concepts "Lieu-dit habité" et "Lieu" deviennent ainsi des concepts accessibles au concepteur pour décrire des interactions : "*lors d'un clic sur un lieu dans le texte, afficher une infobulle qui présente les coordonnées géographiques du lieu*".

L'ontologie intégrée dans WINDMash est donc un premier moyen d'accroître la richesse sémantique de notre langage visuel permettant au concepteur de décrire ses inter-

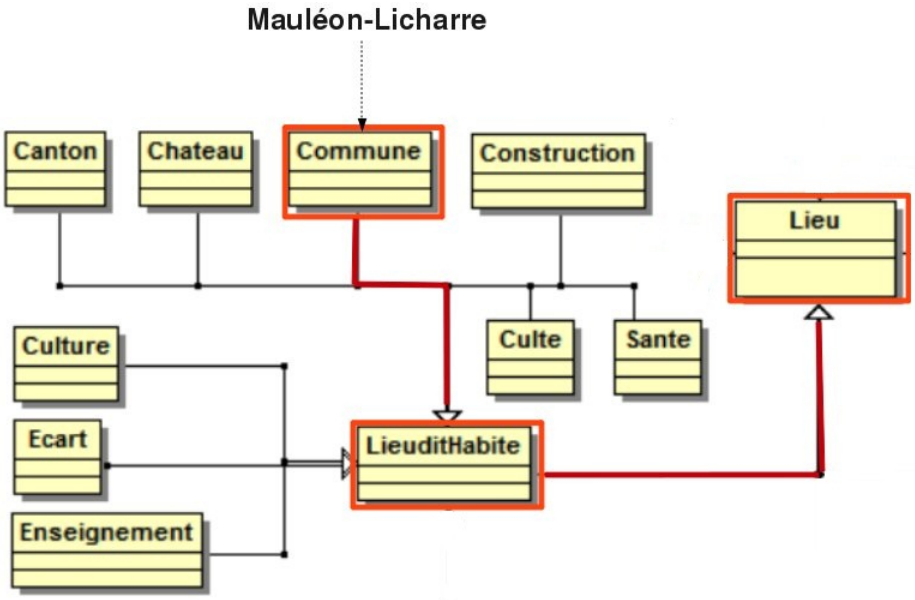


FIGURE 5.39 – Extrait de l'ontologie

actions. Cet enrichissement sémantique est essentiellement basé sur les liens d'héritage présents dans l'ontologie.

Les informations géographiques manipulées par notre environnement WINDMash peuvent être importées depuis d'autres ressources, par exemple du LinkedGeoData [65] qui exploite les informations spatiales collectées par OpenStreetMap [66]. Ainsi, un travail futur consistera à développer un *transformateur* entre les ensembles de données LinkedGeoData importés et notre modèle de *Contenu*.

Nous prévoyons aussi d'exporter les applications Web géographiques générées en XHTML+RDFa [ABMP08] au lieu de XHTML. L'exportation de XHTML+RDFa permettra aux applications WINDMash générées de mieux profiter du Web sémantique et, en particulier mettre en valeur les liens des données sur le Web (Linked Data) [HB11].

5.5 Conclusion

Il n'est pas simple de mettre un terme à un travail de thèse, les objectifs initiaux du sujet nous amenant à des contributions prévisibles et d'autres qui se dessinent au fur et à mesure de l'avancement. Nous avons essayé de rendre compte dans ce chapitre de la plupart des pistes d'investigation qui ont été évoquées au cours des très nombreuses

65. http://linkedgeodata.org
66. www.openstreetmap.org

réunions de travail avec mes encadrants de thèse. Nul doute que d'autres étudiants viendront prolonger ces travaux et doter notre modèle, notre API WIND, notre langage visuel et notre environnement-auteur WINDMash de fonctionnalités répondant à la fois :

- au besoin de démonstrateurs qui permettent aux chercheurs d'évaluer et si possible de valider des contributions scientifiques ;
- au besoin des utilisateurs potentiels des fonctionnalités et outils résultant de ces travaux.

Bibliographie

[ABC+07] Mehmet Altinel, Paul Brown, Susan Cline, Rajesh Kartha, Eric Louie, Volker Markl, Louis Mau, Yip-Hing Ng, David Simmen, and Ashutosh Singh. Damia - a data mashup fabric for intranet applications. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 1370 – 1373, 2007.

[ABCG09] Matteo Albinola, Luciano Baresi, Matteo Carcano, and Sam Guinea. Mashlight: a lightweight mashup framework for everyone. In *2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web*, 2009.

[ABMP08] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and Processing. Recommendation, W3C, October 2008. http://www.w3.org/TR/rdfa-syntax/.

[AM10] Nicole Allieu-Mary. Penser l'espace et le temps en cours d'histoire et géographie, 2010. Didactiques, apprentissages, enseignements.

[AMSK09] Vincent Aleven, Bruce M. McLaren, Jonathan Sewall, and Kenneth R. Koedinger. A new paradigm for intelligent tutoring systems: Example-tracing tutors. *I. J. Artificial Intelligence in Education*, 19(2):105–154, 2009.

[APB+99] Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster. UIML: an appliance-independent XML user interface language. 31:1695–1708, 1999.

[Arc92] A metamodel for the runtime architecture of an interactive system: the uims tool developers workshop. *SIGCHI Bull.*, 24(1):32–37, January 1992.

[ATW08] F. Abdallah, C. Toffolon, and B. Warin. Models transformation to implement a project-based collaborative learning (pbcl) scenario : Moodle case study. In *8th IEEE International Conference on Advanced Learning Technologies (ICALT 08)*, July 1-5 2008.

[Aud09] L. Audibert. *UML 2: De l'apprentissage à la pratique*. Info +. Ellipses Marketing, 2009.

[AVD72] Mortimer J. Adler and Charles Van Doren. *How to Read a Book*. Touchstone, 1972.

[Bar08] Franck Barbier. Supporting the uml state machine diagrams at runtime. In Ina Schieferdecker and Alan Hartman, editors, *Model Driven Architecture Foundations and Applications*, volume 5095 of *Lecture Notes in Computer Science*, pages 338–348. Springer Berlin / Heidelberg, 2008.

[BBC03] Sandra Bringay, Catherine Barry, and Jean Charlet. Les documents et les annotations du dossier patient hospitalier. *Information - Interaction - Intelligence*, 4(1):191 – 211, 2003.

[BCGP00] Michelle Baldonado, Steve Cousins, Jacek Gwizdka, and Andreas Paepcke. Notable: At the intersection of annotations and handheld technology. In *Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing*, pages 100 – 113, 2000.

[BCR04] Margaret Burnett, Curtis Cook, and Gregg Rothermel. End-user software engineering. *Communications of the ACM*, 47:53–58, 2004.

[BEGP03] Jean Bézevin, Breton Erwan, Dupé Grégoire, and Valduriez Patricx. The atl transformation-based model management framework. Report de recherche, INRIA, Septembre 2003.

[Bes96] Daniel Bessonnat. Apprendre à écrire une fiche de lecture, Juin 1996.

[BG00] A. Blackwell and Thomas R. Green. A Cognitive Dimensions Questionnaire Optimised for Users. In *Proceedings of 12th Workshop of the Psychology of Programming Interest Group*, pages 137–154, Corigliano Calabro, Cosenza, Italy, 2000.

[BG01] Jean Bézivin and Olivier Gerbé. Towards a precise definition of the omg/mda framework. In *Proceedings of the 16th IEEE international conference on Automated software engineering*, ASE '01, pages 273–. IEEE, 2001.

[BG10] Luciano Baresi and Sam Guinea. Mashups with mashlight. In Paul P. Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato, editors, *Service-Oriented Computing - 8th International Conference, ICSOC 2010, San Francisco, CA, USA, December 7-10, 2010. Proceedings*, volume 6470 of *Lecture Notes in Computer Science*, pages 711–712, 2010.

[Bil91] E. Bilange. An approach to oral dialogue modelling. In Acquafredda di Maratea, editor, *Proceeding of Second Venaco Workshop on the structure of multimodal dialogue*, pages 1–12, 1991.

[BL00] Claudine Boulanger and Karine Lotigie. Frise d'une journée en cp. *Grand N*, (66):23–33, 2000.

[BL06] Tim Berners-Lee. Notation 3 (n3) a readable rdf syntax, 2006. http://www.w3.org/DesignIssues/Notation3.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34 – 43, 2001.

[BM04] Dave Beckett and Brian McBride. RDF/XML Syntax Specification (Revised). Recommendation, W3C, February 2004. http://www.w3.org/TR/rdf-syntax-grammar/.

[Boc03] C. Bock. Uml without pictures. *IEEE Computer Society Press*, 20:33–35, 2003.

[Boe86] B Boehm. A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes*, 11(4):14–24, August 1986.

[Bot03] Luca Botturi. E2ml - educational environment modeling language. In David Lassner and Carmel McNaught, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003*, pages 304–311, 2003.

[Bou07] Jean-François Boullier. Conception de l'application forbes, 2007. Rapport de recherche Master 2.

[Bou08] Jean-François Boullier. Et si les enseignants étaient mis en situation de pouvoir assurer tout le processus de scénarisation des applications éducatives ? problématique et contexte de mise en œuvre. In *Conférence RJC-EIAH 2008*, pages 51 – 56, 2008.

[BP09] Biörn Biörnstad and Cesare Pautasso. Service-oriented computing - icsoc 2007 workshops. chapter Let It Flow: Building Mashups with Data Processing Pipelines, pages 15–28. Springer-Verlag, 2009.

[Bra08] Myriam Bras. Entre relations temporelles et relations de discours, 2008. Habilitation à diriger les recherches en Informatique de l'Université de Toulouse.

[Bri87] Stefan Britts. Dialog management in interactive systems: a comparative survey. *SIGCHI Bull.*, 18(3):30–42, January 1987.

[But08] Max Butlen. Les voies de la littérature au cycle 2, 2008. Argos demarches.

[BV07] Julien Broisin and Philippe Vidal. Une approche conduite par les modèles pour le traçage des activités des utilisateurs dans des EIAH hétérogènes. *Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation, Analyses des traces d'utilisation dans les EIAH*, 14:(support électronique), 2007.

[BWR+05] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. Automation and customization of rendered web pages. In *Proceedings of the ACM Conference on User Interface Software and Technology (UIST)*, pages 163 – 172, 2005.

[Car11] Thibault Carron. *Observation dans les environnements informatiques pour l'apprentissage humain (HDR)*. PhD thesis, Université de Savoie, October 2011.

[CAS97] Gavin E. Churcher, Eric S. Atwell, and Clive Souter. Dialogue management systems: a survey and overview. Technical report, School of Computer Science, University of Leeds, 1997.

[CCB+02] G. Calvary, Coutaz, L. Bouillon, M. Florins, Q. Limbourg, L. Marucci, F. Paternò, C. Santoro, N. Souchon, D. Thevenin, and J. Vanderdonckt. The cameleon reference framework. Technical Report Deliverable 1.1, 2002.

[CCT+03] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308, 2003.

[CFMP06] Maria F. Costabile, Daniela Fogli, Piero Mussio, and Antonio Piccinno. End-User Development: The Software Shaping Workshop Approach. In *End User Development*, volume 9, chapter 9, pages 183–205. Kluwer Academic Publishers, 2006.

[CGGS09] S. Caffiau, P. Girard, L. Guittet, and D.L. Scapin. Hierarchical structure: A step for jointly designing interactive software dialog and task model. In *Proceedings of the 13th International Conference on HCI*, pages 664–673. Springer-Verlag, 2009.

[Che92] R. Chevallier. *Studia: mise en oeuvre d'un modèle dynamique de dialogue dans un tuteur intelligent.* PhD thesis, 1992.

[CHK+93] Allen Cypher, Daniel C. Halbert, David Kurlander, Henry Lieberman, David Maulsby, Brad A. Myers, and Alan Turransky, editors. *Watch what I do: programming by demonstration.* MIT Press, Cambridge, MA, USA, 1993.

[Cla07] Maryse Clary. Les paysages français au cycle 3, 2007. Guide pédagogique.

[CMN83] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction.* CRC Press, 1st edition, 1983.

[Com08] Benoît Combemale. Ingénierie Dirigée par les Modèles (IDM) – État de l'art. http://hal.archives-ouvertes.fr/hal-00371565/PDF/mde-stateoftheart.pdf, 2008.

[Cos04] Lotta De Coster. L'acquisition et la construction de la notion de temps chez les enfants de 5 à 9 ans. *La Feuille d'IF*, décembre 2004.

[Cou87] Joëlle Coutaz. PAC, on object oriented model for dialog design. In *Interact'87*, pages 431–436, 1987.

[CP10] Maryline Coquidé and Michèle Prieur. Espace et temps dans la scolarité obligatoire, 2010. Didactiques, apprentissages, enseignements.

[Cra02] Jean-Bernard Crampes. *Méthode orientée-objet intégrale MACAO : démarche participative pour l'analyse, la conception et la réalisation*

de logiciels. Technosup. Ellipses, Paris, 2002. MACAO : Méthode d'Analyse et de Conception d'Applications Orientées-Objet.

[Cur92] ACM SIGCHI curricula for human-computer interaction. Technical report, 1992.

[Dal06] James Roland Dalziel. Lessons from lams for ims learning design. In *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, ICALT '06, pages 1101–1102. IEEE Computer Society, 2006.

[Den00] Laurent Denoue. *De la création à la capitalisation des annotations dans un espace personnel d'informations*. PhD thesis, Université de Savoie, 2000.

[DF04] Pierre Dragicevic and Jean-Daniel Fekete. Support for input adaptability in the icon toolkit. In *Proceedings of the 6th international conference on Multimodal interfaces*, ICMI '04, pages 212–219, New York, NY, USA, 2004. ACM.

[DG10] Jean-Pierre Desclés and Zlatka Guentcheva. Référentiels aspecto-temporels : une approche formelle et cognitive appliquée au français. *2ème Congrès Mondial de Linguistique Française*, pages 1675 – 1696, 2010.

[DLG+08] Bruno Dumas, Denis Lalanne, Dominique Guinard, Reto Koenig, and Rolf Ingold. Strengths and weaknesses of software architectures for the rapid creation of tangible and multimodal interfaces. In *Proceedings of 2nd international conference on Tangible and Embedded Interaction (TEI 2008)*, page 47–54, Bonn (Germany), 2008.

[Dob11] Zdena Dobesova. Visual programming language in geographic information systems. In *Proceedings of the 2nd international conference on Applied informatics and computing theory*, AICT'11, pages 276–280. World Scientific and Engineering Academy and Society (WSEAS), 2011.

[DS04] Jean Denègre and François Salgé. *Les Systèmes d'information géographique*. Presses Universitaires de France, 2004.

[EBG+06] Patrick Enhabert, Andrée Borillo, Mauro Gaio, Christian Sallaberry, and Patrick Etcheverry. Traitements sémantiques pour l'information géographique : Textes et cartes. Technical report, Université de Caen, 2006. Rapport du projet GéoSem.

[Ecl11] Fondation Eclipse. Eclipse modeling framework, version 2.7, 2011. http://www.eclipse.org/modeling/emf/.

[Eer06] Matthew Eernisse. Build your own ajax web applications, 2006. http://articles.sitepoint.com/article/build-your-own-ajax-web-apps.

[EG07] Rob Ennals and Minos Garofalakis. Mashmaker: Mashups for the masses. In *Proceedings of the 27th ACM SIGMOD International Conference on Management of Data*, pages 1116 – 1118, 2007.

[ELW98] Robert Eckstein, Marc Loy, and Dave Wood. *Java Swing*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1998.

[EM10] Valérie Emin-Martinez. *Modélisation dirigée par les intentions pour la conception, le partage et la réutilisation de scénarios pédagogiques*. PhD thesis, Université Joseph-Fourier - Grenoble I, 2010.

[Etc99] Olivier Etcheverria. *Les chemins ruraux et leur revalorisation touristico-culturelle. L'exemple du Pays basque*. PhD thesis, Université de Paris 1 Panthéon Sorbonne, 1999.

[FMV07] Christine Ferraris, Christian Martel, and Laurence Vignollet. Helping teachers in designing cscl scenarios: a methodology based on the ldl language. In *CSCL*, pages 193–195, 2007.

[For01] James D. Forbes. *Le voyage aux Pyrénées de James D. Forbes en 1835*. Cairn Eds, 2001.

[Gai01] Mauro Gaio. Traitements de l'information géographique : Représentations et structures, 2001. Mémoire d'HDR. Université de Caen.

[Gar05] Jesse James Garrett. Ajax: A new approach to web applications, 2005. http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[Geo09] Geodjango. A world-class geographic web framework, 2009. http://geodjango.org/.

[GG08] J. Gabay and D. Gabay. *UML 2 Analyse et conception -: Mise en oeuvre guidée avec études de cas*. Etudes et développement. Dunod, 2008.

[GGGCVMA09] Josefina Guerrero-Garcia, Juan Manuel Gonzalez-Calleros, Jean Vanderdonckt, and Jaime Munoz-Arteaga. A theoretical survey of user interface description languages: Preliminary results. In *Proceedings of the 2009 Latin American Web Congress*, LA-WEB '09, pages 36–43. IEEE Computer Society, 2009.

[GGVGC08] J. Guerrero-Garcia, J. Vanderdonckt, and J.M. Gonzalez Calleros. Towards a multi-user interaction meta-model. *IAG Working Paper*, (08/28), 2008.

[GMP+08] Matthias Giese, Tomasz Mistrzyk, Andreas Pfau, Gerd Szwillus, and Michael Detten. Amboss: A task modeling approach for safety-critical systems. In *Proceedings of the 2nd Conference on Human-Centered Software Engineering and 7th International Workshop on Task Models and Diagrams*, pages 98–109. Springer-Verlag, 2008.

[GP04] G. Granier and F. Picot. La place des documents dans l'enseignement de l'histoire et de la géographie. In *Colloque "Apprendre l'histoire et la géographie à l'école"*, Versailles, 2004.

[Gre86] M. Green. A survey of three dialogue models. *ACM Trans. Graph.*, 5:244–275, July 1986.

[Gri99] Arthur Griffith. *Linux Gnome/Gtk+ Programming Bible*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[GRMD07] Agnès Guerraz, Cécile Roisin, Jan Mikác, and Romain Deltour. Multimedia authoring for communities of teachers. *International Journal of Web-Based Learning and Teaching Technologies*, 2(3):1–18, Juillet 2007.

[Gro11a] Object Management Group. Meta object facility (mof) 2.0 query/-view/transformation specification, version 1.1, Janvier 2011. http://www.omg.org/spec/QVT/1.1/.

[Gro11b] Object Management Group. Omg mof 2 xmi mapping specification, version 2.4.1, Août 2011. http://www.omg.org/spec/XMI/2.4.1/.

[GS97] F. Gamboa and D.L. Scapin. Editing MAD* task descriptions for specifying user interfaces, at both semantic and presentation levels. In *Proceedings of Fourth Int. Workshop on Design, Specification, and Verification of Interactive Systems*, pages 193–208, 1997.

[GSE+08] Mauro Gaio, Christian Sallaberry, Patrick Etcheverry, Christophe Marquesuzaà, and Julien Lesbegueries. A global process to access documents'contents from a geographical point of view. *Journal of Visual Languages and Computing*, pages 3 – 23, Février 2008.

[Guy01] L. Guyot. Évaluer au collège : Mobiliser les repères spatio-chronologiques et les documents patrimoniaux, 2001. Echelles. Bulletin de liaison des professeurs n12.

[HB11] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, 2011.

[HG92] H. Rex Hartson and Philip D. Gray. Temporal aspects of tasks in the user action notation. *Hum.-Comput. Interact.*, 7(1):1–45, March 1992.

[HH93] Deborah Hix and H Rex Hartson. *Developing User Interfaces: Ensuring Usability Through Product & Process*. John Wiley Sons, 1993.

[HK01] Rolf Hennicker and Nora Koch. Modeling the user interface of web applications with UML. In *Workshop of the pUML-Group held together with the "UML"2001 on Practical UML-Based Rigorous Development Methods - Countering or Integrating the eXtremists*, pages 158–172, 2001.

[HKM07] David F. Huynh, David R. Karger, and Robert C. Miller. Exhibit: Lightweight structured data publishing. In *Proceedings of the 16th International World Wide Web Conference*, pages 737 – 746, 2007.

[HM03] David Harel and Rami Marelly. *Come, Let's Play: Scenario-Based Programming Using LSC's and the Play-Engine*. Springer-Verlag New York, Inc., 2003.

[HMK05] David F. Huynh, Stefano Mazzocchi, and David R. Karger. Piggy bank: Experience the semantic web inside your web browser. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 413 – 430, 2005.

[Hsi01] Patricia Yee Hsieh. Intelligent tutoring system authoring tool for manufacturing engineering education. *International Journal of Engineering Education*, 17:569–579, 2001.

[Hua96] P. Huart. Définition d'un poste de lecture active de documents électroniques, 1996. Rapport de DEA, IRIT.

[Iks12] Sébastien Iksal. *Ingénierie de l'observation basée sur la prescription en EIAH (HDR)*. PhD thesis, Université du Maine, 2012.

[IMS03] IMS. Ims learning design information model, 2003. IMS Global Learning Consortium.

[Int01] Business Interactif. Méthodes agiles : l'état des lieux, 2001. http://www.dsi.cnrs.fr/methodes/gestion-projet/methodologie/bi-methodes-agiles.pdf.

[JBB+05] Bézivin J., M. Blay, M. Bouzeghoub, J. Estublier, and J.-M. Favre. Rapport de synthèse de l'action spécifique cnrs sur l'ingénierie dirigée par les modèles, 2005. Action Spécifique MDA du CNRS.

[JCV12] Jean-Marc Jézéquel, Benoit Combemale, and Didier Vojtisek. *Ingénierie Dirigée par les Modèles : des concepts à la pratique...* Références sciences. Ellipses, February 2012.

[Jhi06] Anant Jhingran. Enterprise information mashups: Integrating information, simply. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 3 – 4, 2006.

[JK96] Bonnie E. John and David E. Kieras. The goms family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3:320–351, 1996.

[JKR09] Hans-Christian Jetter, Werner A. König, and Harald Reiterer. Understanding and designing surface computing with zoil and squidy. In *CHI 2009 Workshop - Multitouch and Surface Computing*. ACM Press, Apr 2009.

[JPR+02] Christopher B. Jones, R. Purves, A. Ruas, M. Sanderson, M. Sester, M. van Kreveld, and R. Weibel. Spatial information retrieval and geographical ontologies an overview of the spirit project. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 387–388. ACM, 2002.

[JWMP93] Peter Johnson, Stephanie Wilson, Panos Markopoulos, and James Pycock. Adept: Advanced design environment for prototyping with task

models. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, pages 56–, New York, NY, USA, 1993. ACM.

[KAB+11] Andrew J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Mary Shaw, and Susan Wiedenbeck. The state of the art in end-user software engineering. *ACM Comput. Surv.*, 43(3):21:1–21:44, April 2011.

[Ker11] Eric Kergosien. *Point de vue ontologique de fonds documentaire territorialisés indexés.* PhD thesis, Université de Pau et des Pays de l'Adour, 2011.

[KH93] John F. Koegel and Jesse M. Heines. Improving visual programming languages for multimedia authoring. In *ED-MEDIA '93, World Conference on Educational Multimedia and Hypermedia*, pages 286–293, 1993.

[KK01] José Kahan and Marja-Ritta Koivunen. Annotea: an open rdf infrastructure for shared web annotations. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 623–632. ACM, 2001.

[KNNZ99] Hans-Josef Kohler, Ulrich Nickel, Jorg Niereand, and Albert Zandorf. Using UML as visual programming language. *Technical Report tr-ri-99-205*, 1999.

[KP88] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1:26–49, August 1988.

[Kri07] Michel Krieger. From scrapbook to mashup: A review of end-user web programming tools and supporting toolkits, 2007.

[KRR10] Werner A. König, Roman Rädle, and Harald Reiterer. Interactive design of multimodal user interfaces - reducing technical and visual complexity. *Journal on Multimodal User Interfaces*, 3(3):197–213, Feb 2010.

[Laf04] Pierre Laforcade. *Méta-modélisation UML pour la mise en œuvre de situations problèmes coopératives.* PhD thesis, Université de Pau et des Pays de l'Adour, 2004.

[LC94] Colette Laborde and Bernard Capponi. Cabri géomètre, constituant d'un milieu pour l'apprentissage de la notion de figure géométrique. *Recherche en didactique des mathématiques*, 14(1.2):165 – 210, 1994.

[Leh97] Jérôme Lehuen. *Un modèle de dialogue dynamique et générique intégrant l'acquisition de sa compétence linguistique - le système COALA.* PhD thesis, Université de Caen, 19/05/1997 1997.

[LEM11] The Nhan Luong, Patrick Etcheverry, and Christophe Marquesuzaà. An interaction model and a framework dedicated to web-based geographic applications. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 235–242, New York, NY, USA, 2011. ACM.

[LEMN12] The Nhan Luong, Patrick Etcheverry, Christophe Marquesuzaà, and Thierry Nodenot. A visual programming language for designing interactions embedded in web-based geographic applications. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, IUI '12, pages 207–216. ACM, 2012.

[LENM09] The Nhan Luong, Patrick Etcheverry, Thierry Nodenot, and Christophe Marquesuzaà. Wind: an interaction lightweight programming model for geographical web applications. In Erwan Bocher and Markus Neteler, editors, *Geospatial Free and Open Source Software in the 21st Century*, Lecture Notes in Geoinformation and Cartography, pages 211–225. Springer Berlin Heidelberg, 2009.

[Les07] Julien Lesbegueries. *Plate-forme pour l'indexation spatiale multi-niveaux d'un corpus territorialisé*. PhD thesis, Université de Pau et des Pays de l'Adour, 2007.

[Lev96] W. J. M. Levelt. Perspective taking and ellipsis in spatial descriptions. In *Language and space*, pages 77 – 107. MIT Press, 1996.

[LGL06] Julien Lesbegueries, Mauro Gaio, and Pierre Loustau. Geographical information access for non-structured data. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 83–89. ACM, 2006.

[LGMR05] Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind. *Geographic Information Systems and Science*. John Wiley & Sons, 2005.

[LHML08] Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. Coscripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 1719–1728, New York, NY, USA, 2008. ACM.

[LJ93] B. Landau and R. Jackendoff. "What" and "Where" in Spatial Language and Spatial Cognition. *Behavioral and Brain Sciences*, 16(2):217–238, 1993.

[LL05] Hugo Liu and Henry Lieberman. Metafor: visualizing stories as code. In *Proceedings of the 10th international conference on Intelligent user interfaces*, IUI '05, pages 305–307, New York, NY, USA, 2005. ACM.

[LL06] Julien Lesbegueries and Pierre Loustau. Extraction et interprétation d'information géographique dans des données non structurées. In

COnférence en Recherche d'Infomations et Applications, pages 347–352, 2006.

[LLN11] The Nhan Luong, Sébastien Laborie, and Thierry Nodenot. A framework with tools for designing web-based geographic applications. In *Proceedings of the 11th ACM symposium on Document engineering*, DocEng '11, pages 33–42. ACM, 2011.

[LNCC07] Pierre Laforcade, Thierry Nodenot, Christophe Choquet, and Pierre-André Caron. *Model-Driven Engineering (MDE) and Model-Driven Architecture (MDA) applied to the Modeling and Deployment of Technology Enhanced Learning (TEL) Systems: promises, challenges and issues*, chapter 7, pages 116 – 136. Hershey New York, 2007.

[LNG09] Pierre Loustau, Thierry Nodenot, and Mauro Gaio. Design principles and first educational experiments of piir, a platform to infer geo-referenced itineraries from travel stories. *International Journal of Interactive Technology and Smart Education*, pages 23 – 29, 2009.

[LNLM09] The Nhan Luong, Thierry Nodenot, Philippe Lopistéguy, and Christophe Marquesuzaà. A framework to author educational interactions for geographical web applications. In *Proceedings of the 4th European Conference on Technology Enhanced Learning: Learning in the Synergy of Multiple Disciplines*, EC-TEL '09, pages 769–775, Berlin, Heidelberg, 2009. Springer-Verlag.

[Lou08] Pierre Loustau. *Interprétation automatique d'itinéraires dans des récits de voyages*. PhD thesis, Université de Pau et des Pays de l'Adour, 2008.

[LPKW06] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. End-User Development: An Emerging Paradigm. In Henry Lieberman, Fabio Paternò, and Volker Wulf, editors, *End User Development*, volume 9 of *Human-Computer Interaction Series*, chapter 1, pages 1–8. Springer Netherlands, 2006.

[LPLGS07] Annig Le Parc-Lacayrelle, Mauro Gaio, and Christian Sallaberry. La composante temps dans l'information géographique textuelle. *Revue Document Numérique*, 10(2):129–148, 2007.

[LPV01] Quentin Limbourg, Costin Pribeanu, and Jean Vanderdonckt. Towards uniformed task models in a Model-Based approach. In *Proceedings of the 8th International Workshop on Interactive Systems: Design, Specification, and Verification-Revised Papers*, DSV-IS '01, pages 164–182, London, UK, 2001. Springer-Verlag.

[LSG06] Julien Lesbegueries, Christian Sallaberry, and Mauro Gaio. Associating spatial patterns to text-units for summarizing geographic information. In ACM, editor, *Proceedings of ACM SIGIR 2006. GIR, Geographic Information Retrieval, Workshop*, pages 40–43, 2006.

[LV04] Quentin Limbourg and Jean Vanderdonckt. Usixml: A user interface description language supporting multiple levels of independence. In *ICWE Workshops*, pages 325–338, 2004.

[LVFAP+06] Davinia Hernández Leo, Eloy D. Villasclaras-Fernández, Juan I. Asensio-Pérez, Yannis A. Dimitriadis, Iván M. Jorrín-Abellán, Inés Ruiz-Requies, and Bartolomé Rubia-Avi. Collage: A collaborative learning design editor based on patterns. *Educational Technology & Society*, 9(1):58–71, 2006. http://www.ifets.info/journals/9_1/6.pdf.

[Mar97] Catherine C. Marshall. Annotation: from paper books to the digital library. In *Proceedings of the second ACM international conference on Digital libraries*, DL '97, pages 131–140. ACM, 1997.

[Mar99] Craig Marion. What is interaction design and what does it mean to information designers?, 1999. http://mysite.verizon.net/resnx4g7/PCD/WhatIsInteractionDesign.html.

[MAT01] A. Mahfoudhi, M. Abed, and D. Tabary. *From the formal specifications of user tasks to the automatic generation of the HCI specifications*, pages 331–347. Springer, London, 2001.

[MB02] S. Mellor and M. Balcer. *Executable UML - A Foundation for Model-Driven Architecture*. Addison-Wesley, 2002.

[MFB09] Pierre-Alain Muller, Frédéric Fondement, and Benoît Baudry. Modeling modeling. In *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*, MODELS '09, pages 2–16. Springer-Verlag, 2009.

[Mia05] Yongwu Miao. Cosmos: Facilitating learning designers to author units of learning using ims ld. In *Proceedings of the 2005 conference on Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences: Sharing Good Practices of Research, Experimentation and Innovation*, pages 275–282. IOS Press, 2005.

[Mil05] Dominique Mille. *Modèles et outils logiciels pour l'annotation sémantique de documents pédagogiques*. PhD thesis, Université Joseph Fourier - Grenoble I, 2005.

[MM99] Richard G. McDaniel and Brad A. Myers. Getting more out of programming-by-demonstration. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 442–449, New York, NY, USA, 1999. ACM.

[MMS00] F. Martin, B. Mikhak, and B. Silverman. Metacricket: a designer's kit for making computational devices. *IBM Syst. J.*, 39(3-4):795–815, 2000.

[Mog07] Bill Moggridge. *Designing Interactions*. MIT Press, 1st edition, October 2007.

[Moo09] Daniel Moody. The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Softw. Eng.*, 35:756–779, November 2009.

[MPS04] G. Mori, F. Paterno, and C. Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Trans. Softw. Eng.*, 30:507–520, August 2004.

[MVF+09] Christian Martel, Laurence Vignollet, Christine Ferraris, Emmanuelle Villiot-Leclercq, and Salim Ouari. Ppdesigner: An editor for pedagogical procedures. In *Proceedings of the 4th European Conference on Technology Enhanced Learning: Learning in the Synergy of Multiple Disciplines*, EC-TEL '09, pages 379–384, 2009.

[Mye90] Brad A. Myers. Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing*, 1:97–123, March 1990.

[Nar96] Bonnie A. Nardi. Some reflections on the application of activity theory. In Bonnie A. Nardi, editor, *Context and consciousness: Activity theory and human-computer interaction*, pages 235–246. Massachusetts Institute of Technology, 1996.

[Ngu05] N.H. Nguyen. *Dialogue homme-machine : Modélisation de multisession.* PhD thesis, Université Joseph Fourier - Grenoble I, 2005.

[NH98] N. Hari Narayanan and Roland Hübscher. *Visual language theory: towards a human computer interaction perspective*, pages 87–128. Springer-Verlag New York, Inc., New York, NY, USA, 1998.

[Nic87] J.-F. Nicaud. *APLUSIX : un système expert de résolution pédagogique d'exercices d'algèbre.* PhD thesis, Université de Paris-Sud, 1987.

[NL06] Thierry Nodenot and Pierre Laforcade. Learning from a planets game: Elements of a didactical transposition described with the cpm language. In *6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)*, pages 1164–1165, 2006.

[NLP07] Thierry Nodenot, Pierre Laforcade, and Xavier Le Pallec. *Visual Design of coherent Technology-Enhanced Learning Systems: a few lessons learned from CPM language*, pages 254 – 280. IGI Global, 2007.

[NLSM03] Thierry Nodenot, Pierre Laforcade, Christian Sallaberry, and Christophe Marquesuzaà. A uml profile incorporating separate viewpoints when modeling co-operative learning situations. In *IEEE International Conference on Information Technology : Research and Education*, pages 605 – 609, 2003.

[NMLS04] Thierry Nodenot, Christophe Marquesuzaá, Pierre Laforcade, and Christian Sallaberry. Model based engineering of learning situations for adaptive web based educational systems. In *Proceedings of the 13th*

international World Wide Web conference on Alternate track papers & posters, WWW Alt. '04, pages 94–103. ACM, 2004.

[Nod05] Thierry Nodenot. Contribution à l'ingénierie dirigée par les modèles en eiah : le cas des situations-problèmes coopératives, 2005. Habilitation à diriger les recherches en Informatique de l'Université de Pau et des Pays de l'Adour.

[Nod11] Pascal Nodenot. Etude du potentiel de l'api wind pour la production d'applications pédagogiques dans le domaine du spatio-temporel, 2011. Master EFE – 2I2N – FEN de l'Université Toulouse 2 (IUFM Toulouse).

[Nor88] Donald A. Norman. *The Psychology Of Everyday Things*. Harper & Collins, 1988.

[Nor92] Véronique Normand. *Le Modèle Siroco : de la spécification conceptuelle des interfaces utilisateur à leur réalisation*. PhD thesis, 1992. Thèse de doctorat Informatique préparée au Laboratoire de Génie Informatique (IMAG), Université Joseph Fourier 258 pages.

[Ogb02] Uche Ogbuji. The languages of the semantic web, 2002. New Architect.

[OS04] Thomas C Ormerod and A Shepherd. *Using task analysis for information requirements specification: The SGT method.*, pages 1–24. Lawrence Erlbaum Associates, 2004.

[Pai11] Virginie Paillas. Intérêt des applications "wind" pour l'exploitation pédagogique de textes décrivant des itinéraires : les pratiques d'annotation au service du lire et interpréter différents langages, 2011. Master EFE – 2I2N – FEN de l'Université Toulouse 2 (IUFM Toulouse).

[Pal01] Sean B. Palmer. The semantic web: An introduction, 2001. http://infomesh.net/2001/swintro/.

[Pal03] V. Pallota. Computational dialogue models. Technical Report Report IM2.MDM-02, Faculty of Computer and Communication Sciences, Swiss Federal Institute of Technology - Lausanne, 2003.

[Pal10] Damien Palacio. *Combinaison de critères par contraintes pour la Recherche d'Information Géographique*. PhD thesis, Université de Pau et des Pays de l'Adour, 2010.

[Pap80] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Perseus Books, 1980.

[Pat99a] Fabio Paternò. Concurtasktrees : An engineered approach to model-based design of interactive systems. *The Handbook of Task Analysis for HumanComputer Interaction*, 25(2):1–18, 1999.

[Pat99b] Fabio Paterno. *Model-Based Design and Evaluation of Interactive Applications*. Springer-Verlag, London, UK, UK, 1st edition, 1999.

[PdSP03] Paulo Pinheiro da Silva and Norman W. Paton. User interface modeling in umli. *IEEE Software*, 20(4):62–69, 2003.

[Pfa85] G. E. Pfaff, editor. *User Interface Management Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1985.

[PG07] K. Lundgren-Cayrol et M. Léonard Paquette G. *The MOT+ Visual Language for Knowledge-Based Instructional Design*, chapter VIII. IGI Global, 2007.

[PMM97] Fabio Paternò, Cristiano Mancini, and Silvia Meniconi. Concurtasktrees: A diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, INTERACT '97, pages 362–369, London, UK, UK, 1997. Chapman & Hall, Ltd.

[PP01] Philippe Palanque and Fabio Paternò. Design, specification, and verification of interactive systems. *SIGSOFT Softw. Eng. Notes*, 26(1):74–75, January 2001.

[PS03] Fabio Paternò and Carmen Santoro. A unified method for designing interactive systems adaptable to mobile and stationary platforms. *Interacting with Computers*, 15(3):349–366, 2003.

[Rab95] Pierre Rabardel. *Les hommes et les technologies : Approche cognitive des instruments contemporains*. Armand Colin, 1995.

[RAM+85] E. Roulet, A. Auchlin, J. Moeschler, C. Rubattel, and M. Schelling. *L'articulation du discours en français contemporain*. Université de Genève - Faculté des lettres, 1985.

[Ray91] Darrell R. Raymond. Characterizing visual languages. In *Proc. 1991 IEEE Workshop on Visual Languages. (Kobe*, pages 176–182. Society Press, 1991.

[RJB+04] James Rumbaugh, Ivar Jacobson, Grady Booch, Emmanuelle Burr, Véronique Campillo, and Véronique Warion. *UML 2 : guide de référence*. CampusPress Référence. CampusPress, Paris, 2004. La couv. porte en plus : Un ouvrage de référence complet sur le langage UML 2.0 et ses versions antérieures.

[RMMH+09] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. Scratch: programming for all. *Commun. ACM*, 52(11):60–67, 2009.

[RMSS96] M. Resnick, F. Martin, R. Sargent, and B. Silverman. Programmable bricks: toys to think with. *IBM Syst. J.*, 35(3-4):443–452, 1996.

[Rob01] Laurent Robert. *Annotation et visualisation interactives de documents hypermédias*. PhD thesis, Télécom ParisTech, 2001.

[RP00] Alexander Repenning and Corrina Perrone. Programming by example: programming by analogous examples. *Commun. ACM*, 43(3):90–97, March 2000.

[RPC07] Bertrand Richard, Yannick Prié, and Sylvie Calabretto. Lecture active de documents audiovisuels : organisation de connaissances personnelles par la structuration d'annotations, 2007. Rapport de recherche RR-LIRIS-2007-007.

[RSP11] Yvonne Rogers, Helen Sharp, and Jenny Preece. *Interaction Design: Beyond Human - Computer Interaction*. 3rd edition, April 2011.

[Sca10] Christopher Scaffidi. Sharing, finding and reusing end-user code for reformatting and validating data. *J. Vis. Lang. Comput.*, 21(4):230–245, August 2010.

[SCE07] David Sarramia, Ronan Champagnat, and Pascal Estraillier. Méthodologie de conception d'une application interactive à exécution adaptative - Conception d'un AIEA. In *Hypermedias, Hypertests, Products, Tools and Methods (H2PTM'07)*, pages 289–294, 2007.

[Sei03a] Ed Seidewitz. What models mean. *IEEE Software*, pages 26 – 32, 2003.

[Sei03b] Ed Seidewitz. What models mean. *IEEE Softw.*, 20(5):26–32, September 2003.

[SGP98] Bill N. Schilit, Gene Golovchinsky, and Morgan N. Price. Beyond paper: supporting active reading with free form digital ink annotations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '98, pages 249–256. ACM, 1998.

[SH01] Aaron Swartz and James Hendler. The semantic web: A network of content for the digital city. In *Proceedings Second Annual Digital Cities Workshop*, 2001.

[Shu99] Nan C. Shu. Visual programming: Perspectives and approaches. *IBM Systems Journal*, 38(2/3):199–221, 1999.

[Sil00] P. Pinheiro Da Silva. User interface declarative models and development environments: A survey. In LNCS, editor, *Proceedings of DSV-IS2000*, volume 1946, pages 207–226. Springer-Verlag, 2000.

[SJ07] Andrew Sears and Julie A. Jacko, editors. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. 2nd edition, September 2007.

[SPG89] D. Scapin and C. Pierret-Goldbreich. Towards a method for task description : MAD. In L. Berlinguet and D. Berthelette, editors, *Proceedings of Work with Display Units (WWU '89)*, pages 27–34. North-Holland: Elsevier Science, 1989.

[SRL+09] Christian Sallaberry, Albert Royer, Pierre Loustau, Mauro Gaio, and Thierry Joliveau. GeoStream: Spatial information indexing within textual documents supported by a dynamically parameterized web service. In *Proceedings of the International Opensource Geospatial Research Symposium*, OGRS'09, 2009. Available online.

[ST07] Arno Scharl and Klaus Tochtermann. *The Geospatial Web: How Geobrowsers, Social Software and the Web 2.0 are Shaping the Network Society (Advanced Information and Knowledge Processing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[Sti95] Bernard Stiegler. Annotation, navigation, édition électronique : Vers une géographie de la connaissance, 1995. Actes du séminaire "Hypermédias, Education et Formation".

[TB96] Jean-Claude Tarby and Marie-France Barthet. The diane+ method. In Jean Vanderdonckt, editor, *CADUI*, pages 95–120. Presses Universitaires de Namur, 1996.

[tre92] Le trésor de la langue française, 1992. http://atilf.atilf.fr.

[Use96] E. Lynn Usery. A feature-based geographic information system model. *Journal of The American Society for Photogrammetry and Remote Sensing*, 62(7):833–838, July 1996.

[Vé97] Mathieu Véron. Modélisation de la composante annotative dans les documents électroniques, 1997. Master de recherche.

[VKV+06] Max Völkel, Markus Krötzsch, Denny Vrandecic, Heiko Haller, and Rudi Studer. Semantic wikipedia. In *Proceedings of the 15th International World Wide Web Conference*, pages 585 – 594, 2006.

[VL07] Emmanuelle Villiot-Leclercq. *Modèle de soutien à l'élaboration et à la réutilisation de scénarios pédagogiques*. PhD thesis, Université Joseph-Fourier - Grenoble I, 2007.

[VLB96] Gerrit C. Van Der Veer, Bert F. Lenting, and Bas A. J. Bergevoet. Gta: Groupware task analysis - modeling complexity. *Acta Psychologica*, 91:297–322, 1996.

[VM03] H. Vogten and H. Martens. Coppercore – the ims learning design engine, 2003. http://www.coppercore.org.

[Voj12] Didier Vojtisek. Comment allons-nous développer d'ici 5 à 10 ans ?, mars 2012. Programmez ! Le magazine du développement.

[VR09] Romain Vuillemot and Béatrice Rumpler. A web-based interface to design information visualization. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '09, pages 26:172–26:179. ACM, 2009.

[WH07] Jeffrey Wong and Jason I. Hong. Making mashups with marmite: Towards end-user programming for the web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1435 – 1444, 2007.

[Wika] Wikipédia. Méthode agile. http://fr.wikipedia.org/wiki/Méthode_agile.

[Wikb] Wikipédia. Pédagogie active. http://fr.wikipedia.org/wiki/Pédagogie_active.

[WP94] Allison Gyle Woodruff and Christian Plaunt. Gipsy: automated geographic indexing of text documents. *J. Am. Soc. Inf. Sci.*, 45(9):645–655, October 1994.

[YBDZ97] Sherry Yang, Margaret M. Burnett, Elyon DeKoven, and Moshé M Zloof. Representation design benchmarks: A design-time aid for vpl navigable static representations. *Journal of Visual Languages amp; Computing*, 8(5–6):563–599, 1997.

[ZBR09] Tewfik Ziadi, Xavier Blanc, and Amine Raji. From requirements to code revisited. In *Proceedings of the 2009 IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, pages 228–235. IEEE Computer Society, 2009.

Annexe A

Travaux menés par l'équipe T2I

Sommaire

A.1 Travaux menés par l'équipe T2i autour de l'information géographique

Depuis 2004, l'équipe T2i du laboratoire LIUPPA a effectué de nombreux travaux de recherche portant sur la modélisation de l'information géographique présente dans des textes et les techniques d'extraction / indexation automatiques associées. Ces techniques sont mises en œuvre à partir d'un texte en entrée pour passer de manière automatisée des modèles linguistiques à des modèles informatiques de cette information géographique.

La thèse de Julien Lesbégueries [Les07] a permis de proposer un modèle opérationnel pour l'extraction et l'interprétation des points de repères spatiaux exprimés textuellement afin que cette information géographique soit indexée et qu'un système de Recherche d'Information spécialisé dans le spatial puisse retourner des documents spatialement pertinents. L'indexation proposée est multi-grains, c'est-à-dire que l'on peut modéliser les noms de lieux à différentes échelles du corpus (dans une phrase, dans un paragraphe, dans un chapitre...). Cette proposition a donné lieu à l'implémentation et à l'évaluation d'un prototype d'indexation et de recherche d'information spatiale, nommé PIV pour "*Pyrénées Itinéraires Virtuels*" (Figure A.1).

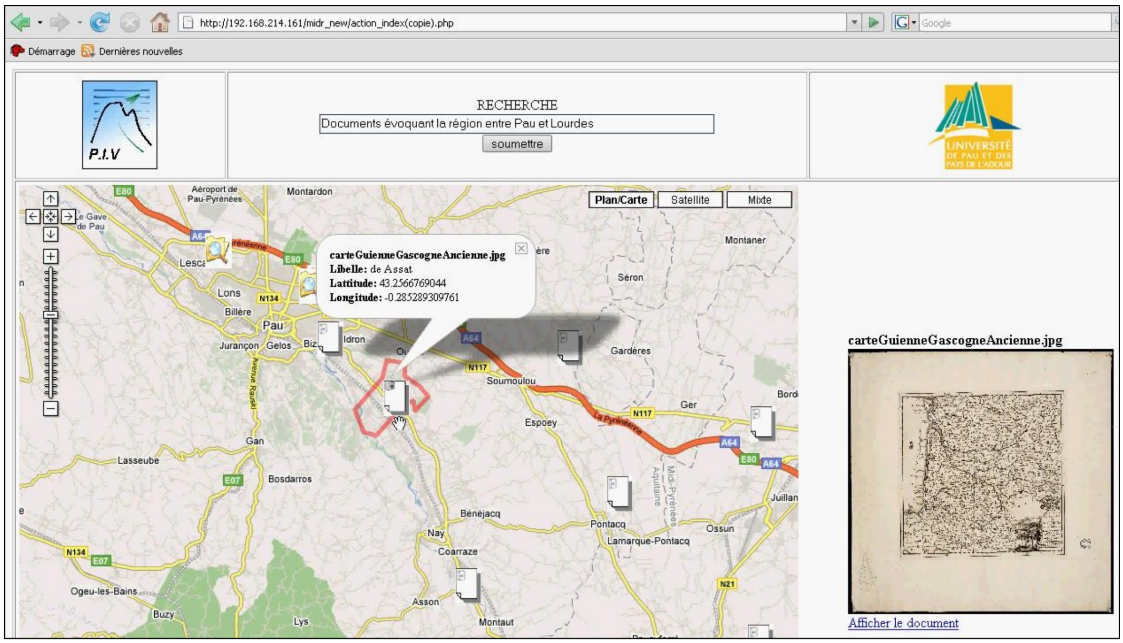


FIGURE A.1 – Visualisation des résultats de PIV sur une carte [Les07]

La figure A.1 illustre un exemple de PIV dont les résultats de la recherche parmi les documents patrimoniaux numérisés fournis par la Médiathèque InterCommunale à Dimension Régionale (MIDR) de la Communauté d'Agglomération de Pau Pyrénées (CAPP) des "*Documents évoquant la région entre Pau et Lourdes*" s'affichent sur une carte.

Grâce aux caractéristiques géographiques des informations qui sont modélisées et exploitées dans le cadre de cette thèse, le système est capable de mettre en valeur des documents relatifs au village d'Assat par exemple (qui se situe effectivement entre Pau et Lourdes) alors qu'il n'a pas été explicitement nommé dans la requête.

La thèse de Pierre Loustau [Lou08] a permis de proposer un mode opératoire pour interpréter automatiquement l'information géographique de textes de type "récit de voyage". La plateforme opérationnalise un modèle d'extraction qui permet de capter l'expression du déplacement dans des textes de type "récit de voyage" et d'en produire une représentation informatique. La figure A.2 présente un exemple dans lequel il est facile de constater avec [LJ93] que les prépositions sont des éléments du langage vecteurs de sens tout autant que les entités spatiales/temporelles et les verbes d'état ou de mouvement.

Nous voyons par exemple dans la figure A.2 que le jeu de couleurs utilisé permet de discerner les lieux reconnus et les verbes de mouvement.

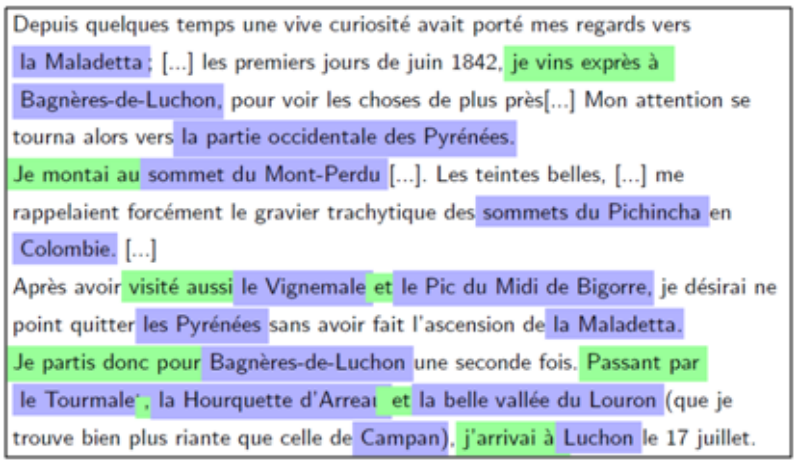

Depuis quelques temps une vive curiosité avait porté mes regards vers la Maladetta ; [...] les premiers jours de juin 1842, je vins exprès à Bagnères-de-Luchon, pour voir les choses de plus près[...] Mon attention se tourna alors vers la partie occidentale des Pyrénées.
Je montai au sommet du Mont-Perdu [...]. Les teintes belles, [...] me rappelaient forcément le gravier trachytique des sommets du Pichincha en Colombie. [...]
Après avoir visité aussi le Vignemale et le Pic du Midi de Bigorre, je désirai ne point quitter les Pyrénées sans avoir fait l'ascension de la Maladetta.
Je partis donc pour Bagnères-de-Luchon une seconde fois. Passant par le Tourmale[illegible], la Hourquette d'Arreau et la belle vallée du Louron (que je trouve bien plus riante que celle de Campan), j'arrivai à Luchon le 17 juillet.

FIGURE A.2 – Extrait du journal de James David Forbes dans les Pyrénées [Lou08]

La thèse d'Eric Kergosien [Ker11] s'est focalisée sur l'exploitation de la facette thématique de l'information géographique. Le travail a permis d'extraire, à partir d'un corpus de textes, des représentations sémantiques mettant en avant un territoire. Il s'appuie sur le travail d'indexation fait par les bibliothécaires via des index normalisés (Figure A.3).

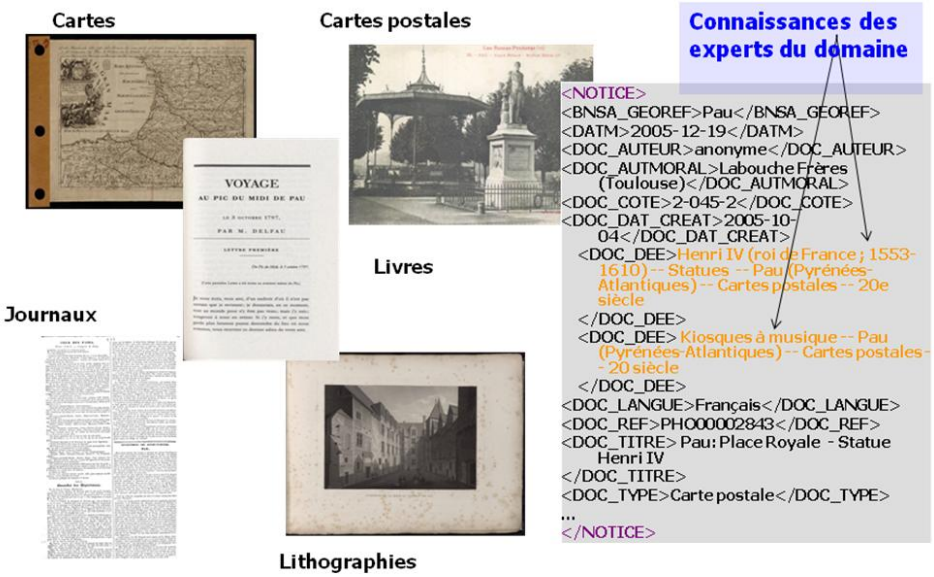


FIGURE A.3 – Extraction thématique des documents traités [Ker11]

La thèse de Damien Palacio [Pal10] a permis de proposer des solutions de recherche d'informations pour combiner, via des méthodes statistiques, les critères spatiaux, temporels et thématiques en permettant de plus à l'utilisateur de paramétrer l'importance de ces critères. La figure A.4 illustre l'interface de PIVasse permettant de justifier les résultats de recherche associés à chaque requête.

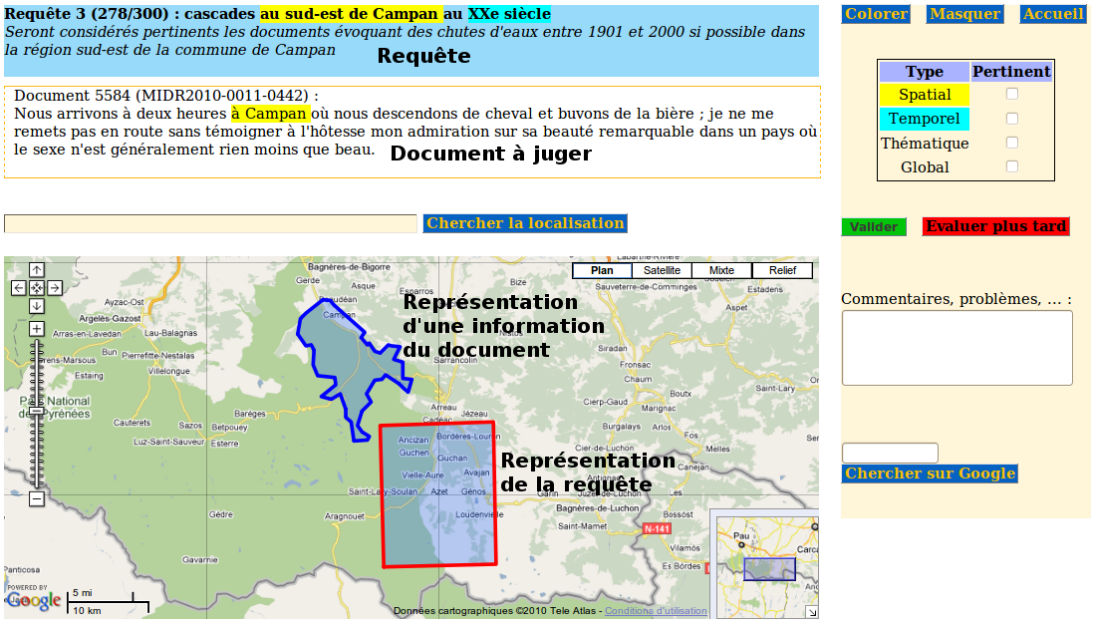


FIGURE A.4 – PIVasse : évaluation d'un document [Pal10]

A.2 Travaux menés par l'équipe T2i en matière de scénarisation pédagogique

Les travaux menés par l'équipe T2i se situent aussi dans le domaine des Environnements Informatiques pour l'Apprentissage Humain (EIAH), et en particulier concernent la pédagogie active qui "*a pour objectif de rendre l'apprenant acteur de ses apprentissages afin qu'il construise ses savoirs à travers des situations de recherche*" [Wikb]. Nos travaux s'appuient sur la théorie de l'activité [Nar96] et le concept d'action médiatisée pour aborder les relations entre un individu (ici un concepteur d'applications à vocation pédagogique) et son environnement. En considérant les travaux de Pierre Rabardel [Rab95], nous allons étudier l'activité de conception pédagogique au travers des actions effectuées sur des objets grâce à un certain nombre d'instruments mis à disposition des concepteurs (Figure A.5).

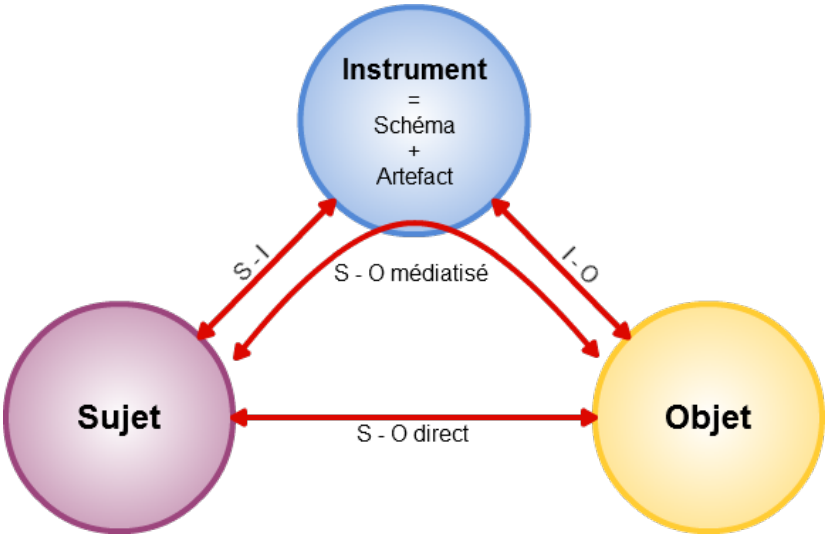


FIGURE A.5 – Modèle des situations d'activités instrumentées de Rabardel

Contrairement à la notion d'artefact classiquement utilisée dans la théorie de l'activité, un instrument d'après Rabardel présuppose l'utilisation de schémas permettant de prendre en compte et organiser les expériences passées pour mieux agir sur de nouveaux objets. Les objets manipulés sont créés lors de l'activité de conception de scénarios d'interaction. Chaque acteur manipule ces objets au travers d'un certain nombre de points de vue qui conduisent à masquer certaines caractéristiques des objets pour en mettre en valeur d'autres.

A.2.1 Travaux menés autour du langage CPM

Dans les travaux précédents de l'équipe en matière de scénarisation pédagogique, nous avons proposé aux concepteurs des éditeurs spécialisés basés sur le langage CPM ("*Cooperative Problem-Based Learning Metamodel*") [Laf04]. Ce langage, basé sur une spécialisation d'UML via les profils, permet l'élaboration de modèles de conception pédagogique [NLSM03, NMLS04] en amont des langages plus formels actuels tels que la spécification standard IMS-LD ("*IMS Learning Design*"). En effet, le langage CPM est un langage visuel qui a été implanté dans un Atelier de Génie Logiciel (AGL) d'UML existant, en l'occurrence Objecteering. Un prototype d'environnement-auteur (Figure A.6) exploitant le langage CPM a pu être expérimenté afin d'envisager l'aide à la création, au suivi et à la maintenance des modèles.

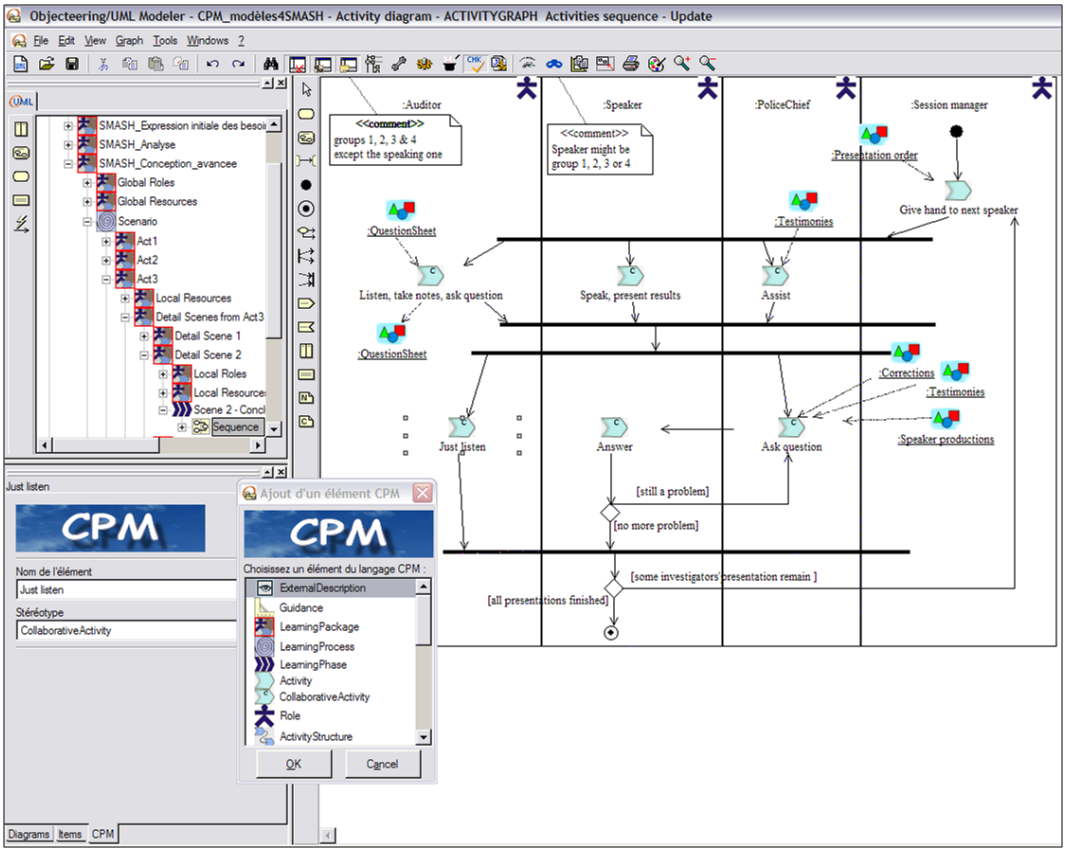


FIGURE A.6 – Modélisation d'une séquence d'activités pédagogiques avec CPM [LNCC07]

Lors des nombreuses expérimentations menées [Nod05, NLP07], nous avons pu noter que les caractéristiques du langage CPM répondaient à un certain nombre d'attentes des concepteurs :

- CPM offre des métaphores visuelles pour manipuler les concepts décrivant une situation d'apprentissage. Ceci a facilité grandement l'activité que devait mener des non-informaticiens pour décrire des situations d'apprentissage coopératives.
- Les retours ont également démontré que la complexité de l'activité de modélisation a pu être maîtrisée par les concepteurs grâce à deux caractéristiques du langage. D'une part, CPM offre la possibilité de décrire de manière séparée différentes perspectives d'un scénario pédagogique complet (*cf.* les facettes cognitives, structurelles et sociales du langage), c'est à dire un ensemble structuré d'activités pédagogiques. D'autre part, une même perspective d'un scénario peut se décrire selon différents niveaux d'abstraction, les modèles raffinés tendant à se rapprocher des services offerts par les composants d'une plateforme de FOAD (Figure A.7).

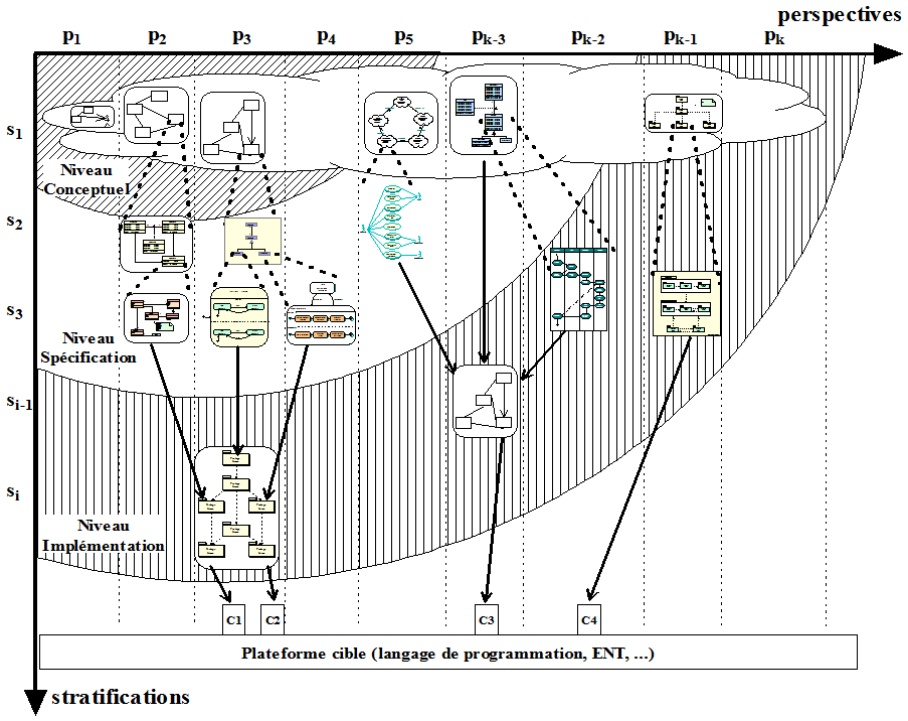


FIGURE A.7 – Une plateforme de FOAD [Nod05]

Au cours des diverses expérimentations menées avec CPM autour de la situation d'apprentissage *Smash* [Nod05], et d'un jeu visant l'apprentissage des *Planètes du système solaire* [NL06], nous avons cependant pu constater que l'environnement d'outils développé autour du langage CPM posait des problèmes d'utilisabilité qui empêchaient les concepteurs de considérer cet environnement comme un vrai instrument de conception (d'après Rabardel). En effet, deux problèmes empêchaient les concepteurs de produire et d'évaluer rapidement des scénarios pédagogiques grâce aux éditeurs mis à leur disposition :

- d'une part, les éditeurs ne fournissaient pas suffisamment de support méthodologique aux concepteurs mais surtout les concepteurs n'étaient pas en mesure de pouvoir tester rapidement les modèles de scénarios produits ;
- d'autre part, dans les objectifs assignés au langage CPM, ce passage par une phase de codage pour obtenir des modèles exécutables sur une plateforme de FOAD était assumé. Cependant au cours des expérimentations menées, cette phase de codage s'est révélée longue et parfois plus compliquée à mettre en œuvre que prévu, ce qui a conduit à des frustrations chez les concepteurs ayant investi du temps dans l'utilisation de l'environnement visuel construit autour du langage CPM.

Clairement, l'environnement CPM a donc été perçu par ses utilisateurs comme un environnement visuel pertinent pour la modélisation de scénarios pédagogiques mais il ne s'est pas comporté comme un instrument au service du prototypage d'applications éducatives, ce que les concepteurs auraient souhaité.

A.2.2 Orientations pour dépasser les limites constatées du langage CPM

Ces éléments nous ont conduits à approfondir l'une des études de cas menées afin de comprendre l'origine des problèmes constatés : la situation-problème *Smash* (Figure A.8). Cette situation-problème qui amenait les élèves à enquêter sur les raisons d'un accident de la route était basée sur l'analyse de témoignages écrits des acteurs et passants ayant assisté à cet accident ainsi que sur l'analyse d'une carte des lieux de l'accident.

La modélisation avec le langage CPM de cette situation d'apprentissage avait conduit [Nod05] à identifier le rôle particulièrement important de l'activité de lecture active de ces documents et l'obligation de faire référence aux informations présentes dans les témoignages pour construire les scénarios pédagogiques. Nous rappelons ici que la lecture active [SGP98, Sti95] est définie comme une technique qui permet de rendre efficace et profitable la lecture d'un texte. Elle est dite active car le lecteur doit annoter sans cesse le texte en même temps qu'il assimile les connaissances dans le texte. D'autres définitions de la lecture active se trouvent dans la section 1.2.2.1. Pour ce type de situation d'apprentissage, l'environnement d'outils CPM n'apportait pas d'aide au concepteur (*cf.* la notion d'instrument de Rabardel), le caractère générique de l'environnement ne permettant pas d'exploiter notamment la sémantique géographique contenue dans les témoignages pour guider l'activité de conception.

Témoignage n°1

Transcription de l'interview

Témoin interrogé sur les lieux de l'accident par l'inspecteur Geoffrey Morante et son assistante Carine Gregor.

Nom: Sophie Lebrun
Adresse: rue des figuiers, les Marais
Age: 32 ans

Sophie Lebrun : J'attendais à l'arrêt de bus. J'ai vu une voiture blanche arriver vers le village. Elle venait des Marais. C'est vrai qu'elle allait à plus de 50 km/h mais cette route n'est pas dangereuse. Et puis ce vélo est arrivé comme un fou, il a fait un écart sur la route, sans prévenir avec le bras. En tous cas, la voiture blanche n'a pas pu le voir : c'est allé trop vite. Elle l'a percuté... le conducteur était décomposé...

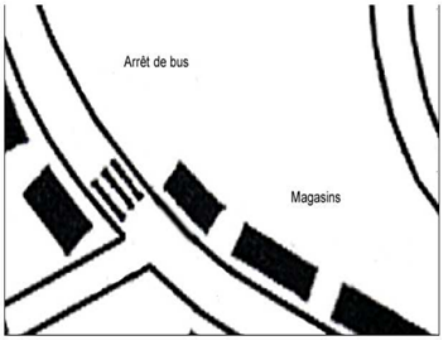


FIGURE A.8 – Un témoignage extrait de Smash et un plan de la zone de l'accident [Nod05]

Afin de nous approcher de la notion d'instrument de Rabardel, il nous a semblé nécessaire de nous orienter vers des outils de conception de scénarios moins génériques que ne l'était CPM. Ceci nous a conduit à étudier de manière plus approfondie le domaine de la lecture active de documents à caractère géographique.

Ainsi, notre équipe a développé une application [Bou07] en 2007 ayant pour objectif de faciliter la lecture et la compréhension de récits de voyages (Figure A.9). L'interface (voir http://erozate.iutbayonne.univ-pau.fr/forbes2007/exp/) était divisée en plusieurs zones, chaque zone étant chargée de rendre compte d'un aspect particulier du récit de voyage : une carte géographique pour présenter la dimension spatiale (zone droite), un calendrier (zone basse gauche) pour rendre compte des éléments temporels et un index d'activités (zone basse droite) pour mettre en évidence les activités décrites par le narrateur dans son récit (zone basse centrale).

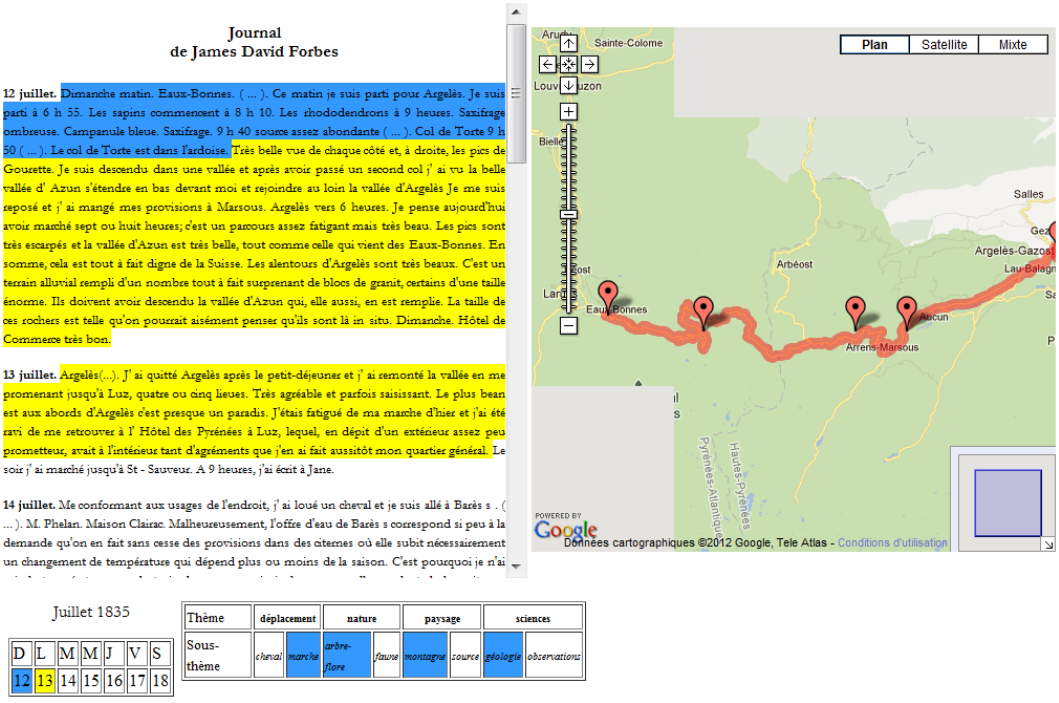


FIGURE A.9 – Aperçu du prototype de lecture de récits de voyages

Les expérimentations menées à l'époque nous ont permis d'identifier les opportunités pédagogiques de ce type d'application [Bou08]. Ceci nous a ensuite conduit à étudier ce que pourrait être la valeur ajoutée, comparée à CPM, d'un environnement de conception de situations d'apprentissage valorisant spécifiquement ces contenus géographiques.

Il semble, tout d'abord, que ce domaine d'apprentissage particulier permette d'approfondir la partie d'un scénario d'interaction consacrée à l'expression des interactions entre l'utilisateur et le système. L'exemple de l'application basée sur l'exploitation du récit de voyages de J.D. Forbes [For01] le montre bien : l'activité de scénarisation doit amener les concepteurs à préciser les effets qu'auront les actions de l'utilisateur sur les objets visibles dans l'un ou l'autre des composants constituant l'interface-utilisateur. Dans le cadre d'activités de lecture active, le centre de l'activité de scénarisation ne doit pas être placé sur la simple présentation d'informations comme dans les applications classiques de type Google Maps, mais plus sur la mise en relation de contenus présentés sous différentes formes (textes, cartes, frises chronologiques...). Cette dimension relative à la "conception détaillée d'interactions exploitant des informations géographiques" se révèle particulièrement intéressante. En effet, l'information géographique apparait comme un domaine relativement stable mais cependant moins bien formalisé que d'autres domaines tels l'algèbre [Nic87], la géométrie [LC94], l'arithmétique [AMSK09], ou l'ingénierie [Hsi01] pour lesquels des environnements de conception spécifiques ont pu être mis au point.

Il semble par ailleurs que la focalisation sur l'information géographique est susceptible de faciliter le passage des modèles de haut niveau (pris en compte par le langage CPM) à des modèles exécutables sur une plateforme cible (Figure A.7), permettant ainsi un prototypage rapide des scénarios exprimés par les concepteurs. Cette avancée repose sur les éléments suivants observés lors du développement de l'application Forbes (Figure A.9) :

- La possibilité de capter de manière automatique une partie de la sémantique géographique contenue dans des textes afin de l'exploiter lors de la scénarisation des applications éducatives.
- La capacité, pour ce domaine particulier, à développer et spécialiser assez facilement des composants d'interface (textuels, cartographiques, photographiques, calendaires...) qui semblent pouvoir être paramétrés et être reliés les uns aux autres au service d'un scénario d'interaction donné.

Ces deux orientations nous paraissent susceptibles de donner lieu à la conception d'un environnement de scénarisation pédagogique centré sur les usages de l'information géographique. Cet environnement aurait des caractéristiques que nous n'avons pas pu atteindre pour l'environnement conçu autour du langage CPM :

1. Un environnement offrant à des non-informaticiens un ensemble d'outils dédiés réellement acceptables dans des usages quotidiens,
2. Un environnement permettant aux concepteurs de s'approprier les outils mis à leur disposition, d'en étendre les usages au-delà des seules fonctionnalités initialement prévues.

Annexe B

Géographie, Espace, Temps et Histoire à l'école primaire et au collège

Sommaire

B.1 Découverte de l'espace et de la Géographie à l'école primaire

Avant de considérer plus particulièrement la question de l'usage pédagogique des textes riches en informations spatio-temporelles, considérons l'enseignement de la géographie.

Extraits des programmes de cycle 3 - Bulletin Officiel, hors-série numéro 3 du 19 juin 2008 : "*L'histoire et la géographie donnent des repères communs, temporels et spatiaux, pour commencer à comprendre l'unité et la complexité du monde. Elles développent chez les élèves curiosité sens de l'observation et esprit critique. Les travaux des élèves font l'objet d'écrits divers, par exemple des résumés et frises chronologiques, des cartes et croquis.*

Le programme de géographie a pour objectifs de décrire et de comprendre comment les hommes vivent et aménagent leurs territoires. Les sujets étudiés se situent en premier

lieu à l'échelle locale et nationale ; ils visent à identifier, et connaître les principales caractéristiques de la géographie de la France dans un cadre européen et mondial. La fréquentation régulière du globe, de cartes, de paysages est nécessaire."

L'étude de cartes ainsi que la lecture de paysage sont des supports incontournables pour les enseignants car elles contribuent à situer et localiser les espaces étudiés. Pour ce faire les élèves devront notamment apprendre à changer d'échelle et à envisager les différents aspects d'un espace à des échelles différentes.

Les élèves en fin de cycle 3 devront notamment savoir : "*Connaître les principaux caractères géographiques physiques et humains de la région où vit l'élève, de la France et de l'Union européenne, les repérer sur des cartes à différentes échelles.*"

La lecture des grilles de références pour l'évaluation et la validation des compétences du socle commun au palier 2 (CM2) - Janvier 2011 permet d'identifier des situations précises d'enseignement et d'évaluation (Figure B.1).

Avoir des repères relevant du temps et de l'espace	
Item	Connaître les principaux caractères géographiques physiques et humains de la région où vit l'élève, de la France et de l'Union européenne, les repérer sur des cartes à différentes échelles.
Explication des items	– Identifier sur des cartes, à différentes échelles, des espaces et des lieux. – Savoir utiliser des cartes à différentes échelles pour situer les repères géographiques étudiés. – Savoir utiliser les principaux termes du vocabulaire spécifique lié aux notions et aux espaces étudiés.
Indications pour l'évaluation	La notion d'espace géographique se construit progressivement, à partir des paysages puis des cartes. Il s'évalue ainsi régulièrement, tout au long du cycle 3. Les exercices d'apprentissage, comme d'évaluation, pourront prendre différentes formes : – réaliser un croquis légendé à partir de photographies paysagères ; – localiser des lieux, des zones, des axes et des réseaux ; – mémoriser des repères géographiques et du vocabulaire spécifique pour être capable de construire un court texte descriptif et explicatif à partir d'un document géographique. L'item est évalué positivement lorsque l'élève est capable de restituer les principales localisations correspondant aux thèmes étudiés en en identifiant l'échelle, lorsque l'élève sait simplifier le contenu d'une carte à l'échelle de la France et sait produire la description d'un paysage en lien avec un thème étudié.

FIGURE B.1 – Extrait des grilles des références

Concernant plus particulièrement la lecture de paysages, "*la notion de paysage est au cœur des programmes du cycle 3 mais ce n'est que rarement que le groupe classe se rendra sur le terrain pour observer le paysage. De plus, il serait totalement impossible d'appréhender ainsi directement tous les types de paysages français. C'est donc*

par l'intermédiaire des photographies que les élèves vont étudier la plupart des paysages . . ." [Cla07].

B.2 La Géographie au collège

Extrait de l'introduction du Bulletin Officiel spécial numéro 6 du 28 août 2008 fixant le programme d'histoire et géographie au collège : "*L'enseignement de la géographie à l'école primaire est centré essentiellement sur la connaissance du milieu local et de la France, qui est resituée en Europe et dans le monde. Dans le souci d'assurer la transition entre primaire et collège, c'est par le territoire proche de l'élève, replacé dans le contexte national puis mondial que débute le programme de sixième. Il est ainsi résolument fondé sur les acquis de l'école primaire qu'il s'agit de mobiliser, d'enrichir et de dépasser pour analyser progressivement d'autres territoires en les situant dans le contexte mondial. L'élève approfondit au collège la capacité à lire et à utiliser des cartes ainsi qu'à réaliser des croquis.*"

Les programmes mentionnent une continuité dans les supports utilisés comme la cartographie, mais n'ignorent pas les nouveaux outils à disposition de cette discipline : "*Si la géographie, comme l'histoire, doit prendre toute sa place dans la maîtrise progressive des langages, elle accorde bien évidemment une place particulière au langage cartographique. Les programmes prévoient que les élèves soient, de la sixième à la troisième, régulièrement et progressivement initiés à la lecture de cartes (de tous types et à toutes échelles) et à la réalisation de croquis. Au-delà de l'acquisition d'un langage spécifique, le croquis doit être, pour l'élève, le moyen de développer un discours argumenté sur l'espace. Pour localiser (placer un lieu) et situer (un lieu par rapport à un autre) dans l'espace, l'élève doit apprendre à lire et à utiliser des documents cartographiques de tous types : différentes échelles, différentes projections, différents points de vue, mais également des globes virtuels, des SIG, des outils du quotidien (géolocalisation), etc.*

Il s'agit enfin de donner aux élèves la pratique des outils que la géographie met, quotidiennement, au service d'une meilleure compréhension de l'espace : cartes et croquis, mais aussi schémas, images, documents statistiques. . . Les systèmes d'information géographiques (SIG) sont désormais d'une utilisation courante (carte routière en ligne, systèmes de positionnement, images satellites. . .). Les élèves doivent en apprendre l'usage et en acquérir l'intelligence. Ils constituent des outils privilégiés tant au service de l'acquisition de connaissances que de la pratique de l'approche géographique."

Ces nouveaux usages mettent en œuvre les Technologies de l'Information et de la Communication, les enseignants sont encouragés à en faire usage dans le cadre notamment du travail critique sur les documents : "*. . . Les techniques de l'information et de la communication doivent, chaque fois que possible, être mises à contribution pour conduire la recherche, l'exploitation et le travail critique sur les documents. Il convient non seulement de varier les modalités d'utilisation des documents mais aussi d'accorder une place*

au récit par le professeur : sa parole est indispensable pour capter l'attention des élèves grâce à un récit incarné et pour dégager l'essentiel de ce qu'ils doivent retenir."

B.3 Découverte du temps et de l'Histoire

A l'école primaire, le programme d'Histoire vise l'acquisition de repères permettant "*d'identifier et de caractériser simplement les grandes périodes qui seront étudiées au collège. Elle s'effectue dans l'ordre chronologique par l'usage du récit et l'observation de quelques documents patrimoniaux*". Les repères étudiés permettront de "*s'assurer que les élèves connaîtront les personnages ou événements représentatifs de chacune de ces périodes*" .

Au collège l'enseignement se poursuit en lieu avec l'enseignement de l'Histoire. "*De nouveaux repères sont introduits en tenant compte d'un élargissement des perspectives. Si l'histoire nationale reste essentielle, elle ne constitue plus un passage obligé pour une ouverture sur l'histoire de l'Europe et du monde. La recherche du sens des repères, événements, hommes et œuvres, est devenue essentielle. Les collégiens apprennent également, de façon progressive, à identifier et à analyser les différentes sources de l'histoire. Enfin, comme à l'école primaire, l'enseignement de l'histoire s'articule avec celui de l'histoire des arts qui passe toujours par l'étude d'œuvres. Cependant, sur ce plan également, les perspectives sont élargies : certaines œuvres fondamentales déjà abordés à l'école primaire peuvent se retrouver, leur étude permettant de consolider et d'approfondir leur connaissance.*" [Extrait de l'introduction du Bulletin Officiel spécial numéro 6 du 28 août 2008 fixant le programme d'histoire et géographie au collège.]

Les enseignements liés aux savoirs ne sont pas les seuls mentionnés dans les programmes, les élèves acquièrent tout au long de leur scolarité des capacités qui nous l'avons vu contribuent à l'acquisition d'une culture humaniste. En Histoire, comme en Géographie, le travail mené devra contribuer à l'acquisition de la capacité "Lire différents langages". Pour l'acquisition du palier 2 (CM2) du socle commun les élèves devront "Lire et utiliser textes, cartes, croquis, graphiques" (Figure B.2).

On retrouve les mêmes considérations "méthodologiques" au collège : "*L'appréhension du temps et de la durée est une capacité qui se forge par la pratique de l'histoire. Elle se construit, de manière progressive, autour de quatre composantes : se repérer dans le temps, représenter le temps, utiliser les représentations du temps, se représenter le temps.*

Afin de faciliter les repérages élémentaires dans le temps mais aussi l'acquisition progressive du sens de la profondeur historique (en restant modeste dans les attentes avec des collégiens), il est utile de conduire un travail régulier sur le temps et sur ses représentations. Il s'agit d'habituer l'élève à utiliser des unités de temps variées (mois, année, siècle, millénaire), à ordonner des événements par ordre chronologique en utilisant

Lire et pratiquer différents langages		
Item	Explication des items	Indications pour l'évaluation
Lire et utiliser textes, cartes, croquis, graphiques.	L'explication du monde demande la maîtrise de différents langages articulés entre eux. "Lire et utiliser" recouvrent à la fois la compréhension de ces différents langages et la capacité à les mobiliser dans des productions schématiques, orales ou écrites.	Les situations d'évaluation impliquent que les élèves soient susceptibles d'utiliser les différents langages de la culture humaniste, elle prennent donc des dormes diverses : – dégager les idées essentielles d'un texte (rapports texte / image dans des documents, des albums...) ; – articuler un texte, un document avec d'autres textes ou documents ; – localiser sur une carte des lieux et des zones ; – compléter une frise chronologique, compléter un schéma ; – légender un document (carte, affiche, paysage) ; – élaborer un croquis à partir d'une carte, d'un paysage, d'un texte ; – construire un graphique à partir de données chiffrées ; – réaliser un exposé et le présenter oralement. Ces situations peuvent être mises en œuvre aussi bien dans le cadre du temps consacré à la culture humaniste que dans celui mobilisé pour d'autres domaines disciplinaires (français, mathématiques, sciences...). L'item est évalué positivement lorsque l'élève est capable : – d'identifier les informations d'un document en prélevant l'essentiel (titre, époque ou date, légende, contenu) ; – de restituer à l'oral ou à l'écrit des informations simples ; – de reformuler à l'oral ou l'écrit le contenu (ou une partie) d'un document en choisissant un vocabulaire approprié ; – de rapprocher deux documents de nature différente et portant sur un même thème ; – d'élaborer un codage simple et de l'exploiter.

FIGURE B.2 – Extrait des grilles de références pour l'évaluation et la validation des compétences du socle commun au palier 2 - 2011

des repères, à relier une date à une grande période de l'histoire, à placer des dates sur une frise chronologique, à inscrire un événement dans un "avant" et un "après", à placer les causes avant les conséquences.

En lien avec le professeur de français, un travail peut être engagé avec profit sur les temps de la langue, les connecteurs logiques, les marqueurs du temps et de la durée, c'est-à-dire tous les indices du langage qui permettent de mieux se repérer dans le temps, de distinguer l'avant et l'après, la cause de la conséquence.

Le travail de localisation doit devenir une habitude intellectuelle pour l'élève, en géographie comme en histoire. La maîtrise des capacités par les élèves doit s'inscrire dans la durée et n'est possible que par l'exercice régulier et répété." Extraits des Grilles de références pour l'évaluation et la validation des compétences du socle commun au palier 3 - Janvier 2011.

B.4 Quelle place pour les récits de voyage dans les enseignements ?

La lecture de textes et plus précisément de récits est bien considérée dans les instructions officielles : le document de validation du palier 2 du socle commun (cycle 3) mentionne l'item suivant à évaluer "*Lire seul des textes du patrimoine et des œuvres intégrales de la littérature de jeunesse, adapté à son âge.*" Cet item est considéré acquis lorsque l'élève est "*en mesure de lire au moins 5 œuvres intégrales de littératures de jeunesse, d'en présenter les grandes lignes, d'émettre un jugement et de faire des liens entre les œuvres rencontrées*".

Pour organiser les enseignements relatifs à la littérature les enseignants ont à disposition des listes de références d'œuvres de littérature. Le rôle de ces listes est explicité par [But08] auprès de la commission ministérielle nationale chargée de la sélection des livres de référence pour l'école. Les récits de voyage n'apparaissent pas dans les catégories, mais la lecture par exemple des notices de la liste pour le cycle 3 permet de relever une série d'ouvrages qui abordent les récits de voyage :

- BEGAG Azouz et LOUIS Catherine : Un train pour chez nous (Éditeur : Thierry Magnier)
- ERRERA Eglal et SATRAPI Marjane : Les Premiers Jours (Éditeur : Actes Sud Junior)
- LAGERLOF Selma : Le Merveilleux Voyage de Nils Holgerson à travers la Suède (Éditeur : Flammarion)
- PLACE François : Les Derniers Géants (Éditeur : Casterman)
- BERGAME Ferdinand et THERS Nicolas : Voyages en plusieurs régions éloignées du monde par Lemuel Gulliver (Éditeur : Soleil)
- Le Livre des merveilles de Marco Polo, qui "relate des voyages réels en Orient mais les présente comme des contrées merveilleuses" est également mentionné.

Selon les programmes de cycle 3, "*la lecture et l'écriture sont systématiquement liées : elles font l'objet d'exercices quotidiens, non seulement en français, mais aussi dans le cadre de tous les enseignements. L'étude des textes, et en particulier des textes littéraires, vise à développer les capacités de compréhension, et à soutenir l'apprentissage de la rédaction autonome.*"

Extraits (du Bulletin Officiel, numéro 3 du 19 juin 2008) de la progression du programme de Français en lecture au cycle 3 :

- CE2 : Dans un récit, s'appuyer sur le repérage des différents termes désignant un personnage ; sur les temps des verbes et sur les mots de liaison exprimant les relations temporelles pour comprendre avec précision la chronologie des évènements ; sur les deux-points et guillemets pour repérer les paroles des personnages pour lire un texte documentaire, descriptif ou narratif, et restituer à l'oral ou par écrit

l'essentiel du texte (sujet du texte, objet de la description, trame de l'histoire, relations entre les personnages...).
- CM1 : Dans un récit ou une description, s'appuyer sur les mots de liaison qui marquent les relations spatiales et sur les compléments de lieu pour comprendre avec précision la configuration du lieu de l'action ou du lieu décrit.
- CM2 : S'appuyer sur les mots de liaison et les expressions qui marquent les relations logiques pour comprendre avec précision l'enchaînement d'une action ou d'un raisonnement.

Les programmes du cycle 3 mentionnent également que "*les textes ou ouvrages donnés à lire aux élèves sont adaptés à leur âge et à leur maturité, du point de vue de la complexité linguistique, des thèmes traités et des connaissances à mobiliser. Du CE2 au CM2, ils sont de plus en plus longs et difficiles. Les textes lus par l'enseignant sont plus complexes que ceux que les élèves peuvent lire seuls.*"

Le palier 2 du socle commun met clairement en avant des items précis au sujet du lire : repérer informations explicites, et inférer informations implicites (Figure B.3).

Lire		
Item	Explication des items	Indications pour l'évaluation
Repérer dans un texte des informations explicites.	Repérer dans un texte des informations explicites.	L'évaluation repose sur des activités de lecture, en français et dans les autres enseignements, et sur des exercices spécifiquement conçus pour l'évaluation. Elle porte sur la capacité à : – prélever des informations directement accessibles dans un document écrit (texte littéraire, texte documentaire, compte rendu d'expérience, fiche technique...) ; – repérer les structures spécifiques de textes littéraires ou documentaires (mode d'emploi, fiches techniques...). L'observation porte sur les stratégies de repérage utilisées (balayage du texte, recherche de mots clés) et sur les informations apportées à l'oral ou à l'écrit par l'élève. L'item est évalué positivement lorsque l'élève parvient à apporter les informations avec exactitude.
Inférer des informations nouvelles (implicites).	Inférer des informations nouvelles (implicites).	L'évaluation repose sur des activités de lecture littéraire ou documentaire, en français et dans les autres enseignements ainsi que sur des activités spécifiquement conçues pour l'évaluation. Elle porte sur la capacité à mettre en relation les informations du texte entre elles et à mobiliser des connaissances culturelles ou appartenant à des domaines disciplinaires. L'observation porte sur la capacité à : – mettre en relation plusieurs informations explicitement contenues dans le texte pour en déduire une information nouvelle ; – questionner le texte : organisation syntaxique, orthographique et grammaticale (reprises anaphoriques, connecteurs logiques, mots de liaison, marques des relations spatiales et temporelles) ; – apporter des références, des connaissances et des informations complémentaires permettant une interprétation cohérente. L'item est évalué positivement lorsque l'élève parvient à élaborer une interprétation cohérente et pertinente par rapport aux éléments du texte.

FIGURE B.3 – Extrait des grilles de références pour l'évaluation et la validation des compétences du socle commun au palier 2 - 2011

B.5 Dimensions temporelles et spatiales au cycle 3

Le document de synthèse, ci-dessous réalisé dans le cadre du Master 2 Recherche de P. Nodenot [Nod11], décrit de manière la plus exhaustive possible tout qui concerne les dimensions temporelles et spatiales au cycle 3. En effet, par la nature des activités demandées aux élèves, les applications développées dans cette thèse se prêtent plus à un public d'élèves de cycle 3.

A la fin du cycle 3, l'enfant est notamment capable de nombreuses compétences en français, en mathématiques ou encore en culture humaniste. Ces compétences sont listées ci-dessous.

Compétence en français :
- dans un récit, s'appuyer sur le repérage des différents termes désignant un personnage, sur les temps des verbes et sur les mots de liaison exprimant les relations temporelles pour comprendre avec précision la chronologie des événements, la configuration du lieu de l'action ou du lieu décrit.
- comprendre le sens d'un texte en reformulant l'essentiel et en répondant à des questions le concernant. Cette compréhension s'appuie sur le repérage des principaux éléments du texte (par exemple, le sujet d'un texte documentaire, les personnages et les événements d'un récit), mais aussi sur son analyse précise.
- comprendre l'usage de l'imparfait et du passé simple dans un récit, du présent dans un texte scientifique ou documentaire.
- s'appuyer sur les mots de liaison et les expression qui marquent les relations logiques pour comprendre avec précision l'enchaînement d'une action ou d'un raisonnement.
- rendre compte de sa lecture, exprimer ses réactions ou ses points de vue et échanger avec d'autres sur ses sujets, mettre en relation des textes entre eux (auteurs, thèmes, sentiments exprimés, personnages, événements, situation spatiale ou temporelle, tonalité comique ou tragique...). Les interprétations diverses sont toujours rapportées aux éléments du texte qui les autorisent ou, au contraire, les rendent impossibles.
- raconter de mémoire une œuvre lue.
- utiliser à bon escient des termes appartenant aux lexiques des repères temporels, de la vie quotidienne et du travail scolaire.
- savoir répondre oralement aux questions où, quand, comment, pourquoi.
- comprendre les notions d'action passée, présente, future.
- comprendre la notion d'antériorité relative d'un fait passé par rapport à un autre, d'un fait futur par rapport à un autre.

Compétence en mathématiques :
- trier des données et les classer
- lire ou à produire des tableaux, des graphiques et les analyser (la proportionnalité est abordée à partir des situations faisant intervenir les notions de pourcentage, d'échelle...).

Compétence en culture humaniste :
- ouvrir son esprit à la diversité et à l'évolution des civilisations, des sociétés, des territoires, des faits religieux et des arts (dimensions historiques, géographiques, artistiques et civiques).
- acquérir des repères temporels, spatiaux, culturels et civiques.

– identifier sur une carte et connaître quelques caractères principaux des grands ensembles physiques et humains de l'échelle locale à celle du monde.
– lire et utiliser différents langages : carte, croquis, graphiques, chronologie, iconographie.

"*La culture humaniste des élèves dans ses dimensions historiques, géographiques, artistiques et civiques se nourrit aussi des premiers éléments d'une initiation à l'histoire des arts*" (Bulletin Officiel, numéro 0 du 20 février 2008, programme du CE2, du CM1 et du CM2).

L'histoire et la géographie donnent des repères communs, temporels et spatiaux, pour commencer à comprendre l'unité et la complexité du monde. Elles développent chez les élèves curiosité, sens de l'observation et esprit critique. Les travaux des élèves font l'objet d'écrits divers, par exemple des résumés et frises chronologiques, des cartes et croquis.

L'histoire des arts porte à la connaissance des élèves des œuvres de référence qui appartiennent au patrimoine ou à l'art contemporain ; ces œuvres leur sont présentées en relation avec une époque, une aire géographique (sur la base des repères chronologiques et spatiaux acquis en histoire et en géographie).

Avec la fréquentation des œuvres littéraires, la culture humaniste contribue à la formation de la personne et du citoyen.

Annexe C

Cycles de vie du logiciel et méthodes de programmmation centrée utilisateur

Sommaire

C.1 Cycles de vie du logiciel utiles pour l'utilisateur

C.1.1 Cycle en V

Pour définir un modèle de conception adapté à la prise en compte des besoins utilisateur, Coutaz [Cou87] a proposé une adaptation du modèle en V pour intégrer des points d'entrée du point de vue utilisateur. Nous présentons ce modèle en V dans la figure C.1 qui propose un processus de conception linéaire permettant d'avancer progressivement vers l'application souhaitée.

Toutefois, l'étude de l'utilisation de processus de conception correspondant aux modèles séquentiels et linéaires (comme les modèles en cascade ou en V) a révélé que les

concepteurs préféraient réaliser simultanément plusieurs étapes, choisissant leurs tâches de conception de manière opportuniste.

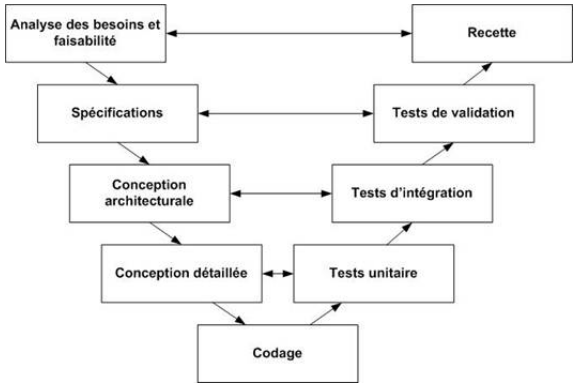


FIGURE C.1 – Cycle en V

C.1.2 Développement incrémental

Le développement itératif ou incrémental est au cœur d'un processus de développement logiciel cyclique développé en réponse aux faiblesses du modèle en cascade. Il commence par une planification initiale et se termine par le déploiement avec des interactions cycliques entre les deux.

L'idée de base est de développer un système grâce à des cycles répétés (itératifs) et avec de plus petits nombres de portions à la fois (incrémental), permettant aux développeurs de logiciels de profiter de ce qu'ils ont appris pendant le développement des parties ou des versions précédentes (Figure C.2). Apprendre provient à la fois du développement et de l'utilisation du système où les étapes clés possibles du processus démarrent avec une simple mise en œuvre d'un sous-ensemble de la configuration logicielle et renforcent itérativement les versions jusqu'au système mis en œuvre. A chaque itération, les modifications de conception sont prises et les nouvelles fonctionnalités sont ajoutées.

Ce processus se compose d'une étape d'initialisation, d'une étape d'itération, et de la liste de contrôle du projet. L'étape d'initialisation crée une version de base du système. Le but de cette mise en œuvre initiale est de créer un produit pour lequel l'utilisateur peut réagir. Elle offre un échantillon des principaux aspects du problème et fournit une solution qui est assez simple à comprendre et à mettre en œuvre facilement. Pour guider le processus d'itération, une liste de contrôle de projet est créée, et contient un enregistrement de toutes les tâches qui doivent être effectuées. Cette liste comprend des éléments tels que les nouvelles fonctionnalités à mettre en œuvre. La liste de contrôle est révisée à la suite de la phase d'analyse.

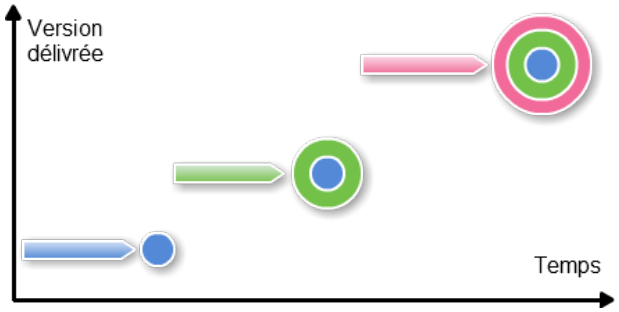


FIGURE C.2 – Modèle incrémental

L'application est délivrée en plusieurs fois, de manière incrémentale, en la complétant au fur et à mesure et en profitant de l'expérimentation opérationnelle des incréments précédents. Chaque incrément peut impliquer un cycle de vie classique plus ou moins complet et peut donc relever un coût important de temps.

C.1.3 Cycle en spirale

De nombreuses études ont montré que les applications étaient développées par une succession d'itérations des étapes de conception, suivant donc un processus de conception itératif. Le modèle en spirale (Figure C.3) propose un processus de conception adapté à la conception itérative.

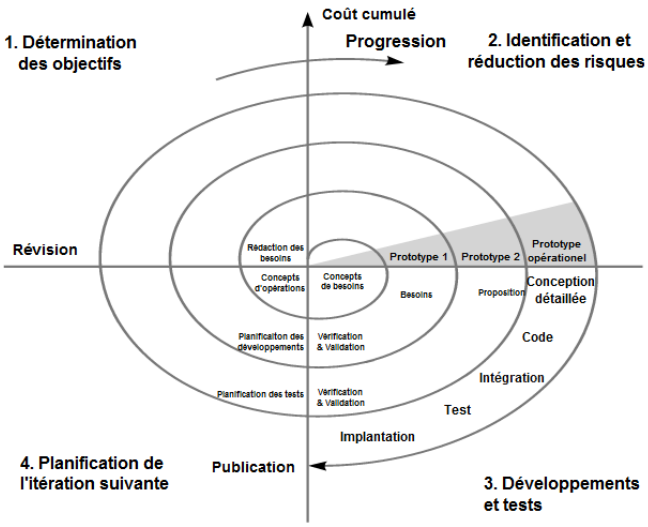


FIGURE C.3 – Modèle en spirale [Boe86]

Le modèle en spirale propose un processus de conception dans lequel chacune des étapes est répétée jusqu'à l'obtention du système. Ce processus est basé sur l'analyse de risque. Chaque itération vise à supprimer les risques relevés préalablement. Le processus est réalisé tant que des risques sont relevés, il en résulte un nombre d'itérations pouvant être important et donc un processus potentiellement coûteux. Le coût du modèle en spirale, le fait que le point de départ de chaque itération soit l'analyse des risques sont des éléments primordiaux.

C.1.4 Développement rapide d'applications (RAD)

La méthode RAD ("*Rapid Application Development*") est une méthode de développement de logiciels où le cycle de développement est plus court que celui du modèle en cascade et reprend les principes du développement incrémental et du cycle en spirale. L'objectif de la méthode RAD est d'obtenir un applicatif adéquat à partir d'un prototype impliquant l'utilisateur final.

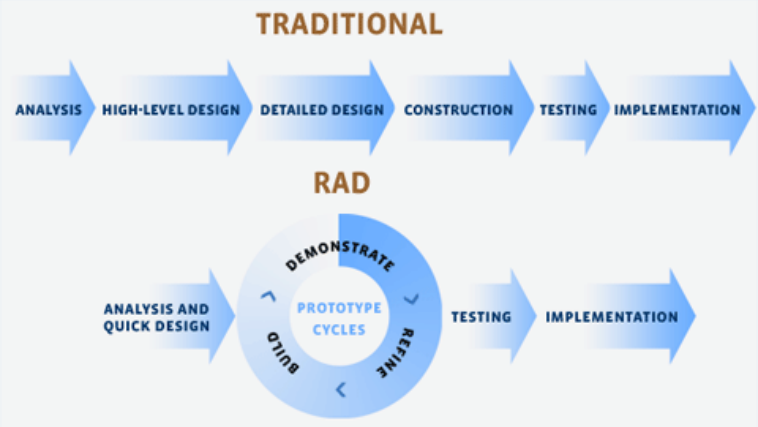


FIGURE C.4 – Développement rapide d'applications (http://software-document.blogspot.fr)

Le principe fondamental du RAD (Figure C.4) est le suivant : il s'agit de fixer, dès l'initialisation du projet, une enveloppe temps/argent dans laquelle le projet doit impérativement s'inscrire. Le projet étant construit intégralement avec les utilisateurs, c'est à eux, avec l'aide d'un animateur RAD, qu'incombe la tâche de faire cadrer le projet avec le budget défini. L'ensemble des phases du projet est couvert par le RAD et réalisé avec les futurs utilisateurs du système : de la phase de conception du système informatisé jusqu'à la mise au point des interfaces et des états produits, par itérations successives de prototypes. Il en résulte alors l'obligation de ne développer que des fonctionnalités "utiles", en éliminant les développements particuliers n'emportant pas l'adhésion générale. Il

est souvent demandé par des utilisateurs des versions multiples d'une restitution (qu'elle soit imprimable ou consultable à l'écran) : dans le cas d'un projet RAD, on cherche à produire une restitution unique, rassemblant l'ensemble des informations et permettant d'emporter les suffrages de chacun des participants au projet.

Les outils RAD permettent facilement de créer des programmes à l'aide d'une interface graphique dédiée ainsi que de nombreux outils et modules. Des tels outils sont par exemple WinDev, Delphi, JBuilder...

Cette approche nous intéresse particulièrement pour un processus de conception rapide des applications et c'est un des aspects importants qui a été repris dans le cadre des méthodes agiles présentées dans la partie suivante.

C.1.5 Développement agile

Les méthodes de développement agile ont pour but de réduire le cycle de vie du logiciel et ainsi d'accélérer son développement en développant une version minimale, puis en intégrant les fonctionnalités par un processus itératif. La mise en œuvre de ce processus se réalise par l'adaptation du logiciel aux changements des besoins détectés par l'utilisateur et aux tests tout au long du cycle de développement.

Les méthodes agiles ont pour origine du fait que les clients ne peuvent pas définir leurs besoins de manière exhaustive dès le début du projet. Le concept "agile" signifie ainsi la capacité d'adaptation aux changements de contexte et aux modifications de spécifications pendant le processus de développement.

Les méthodes de développement agiles sont de plus en plus utilisées dans les développements logiciels et il existe plusieurs méthodes répondant à des besoins différents [Wika]:

- Développement rapide d'applications (RAD) est un cycle de développement court que nous avons présenté dans la partie précédente.
- DSDM (*Dynamic Software Development Method*) est une méthode basée sur RAD qui le complète en offrant les principes prenant en compte l'ensemble du cycle de développement.
- Scrum est une méthode agile dédiée à la gestion de projets. La méthode s'appuie sur le découpage d'un projet en incréments, appelés "*sprint*" qui peuvent durer dans un temps varié entre quelques heures et un mois. Chaque *sprint* commence par une estimation suivie d'une planification opérationnelle et se termine par une démonstration prouvant qu'il a été achevé.
- XP (*eXtreme Programming*) est une méthode agile de développement d'un logiciel plus particulièrement concentrée sur l'aspect réalisation d'une application. Elle repose sur des cycles rapides de développement et permet de placer les clients au centre du processus de conception.
- Le "*lean*" est un terme inventé à la fin des années 1980 par une équipe de chercheurs du MIT pour décrire le système de production de Toyota (TPS). Le *lean*

software developpement est l'application du développement logiciel pour obtenir des résultats équivalents à ceux obtenus par les diverses applications du *lean*.
- Kanban est une méthode pour développer des produits logiciels en mettant l'accent sur le juste-à-temps de livraison afin de ne pas surcharger les développements de logiciels. Les développeurs prennent en charge leur travail à partir d'une file d'attente ; et le processus de conception, à partir de la définition d'une tâche jusqu'à sa livraison au client, est affiché pour tous les participants.

Toutes les méthodes agiles ne sont pas applicables dans toutes les situations, mais elles seront plus ou moins efficaces en fonction de la taille du projet [Int01]. Par exemple, *eXtreme Programming*, Scrum ne sont applicables que pour des "petits" projets, tandis que DSDM est facilement adaptable à des projets plus importants. Quant au RAD, il peut s'appliquer aussi bien à des petits projets qu'à de plus importants en raison de la possibilité de paralléliser les tâches. De plus, ces méthodes sont plus ou moins faciles à mettre en œuvre. Par exemple, bien que *eXtreme Programming* soit une méthode la plus connue, elle présente les pratiques les plus exigeantes. Au contraire, RAD présente les tâches élémentaires à réaliser facilement.

C.2 Méthodes de programmation utilisables par l'utilisateur

Nous présentons ci-dessous les méthodes de programmation utilisables par un utilisateur et facilitant le développement d'applications en toute autonomie.

C.2.1 Programmation visuelle

Il existe plusieurs environnements supportant la programmation utilisant des attributs visuels. Dans ces environnements, nombre de sémantiques d'un programme sont exprimées à travers la présentation visuelle du programme.

Dans un langage visuel, la sémantique peut hypothétiquement être encodée avec des attributs d'une représentation visuelle tels que la position, la couleur, la taille, et l'intersection avec d'autres formes.

Scratch est un langage de programmation permettant de créer des histoires, des jeux interactifs... Il est développé par l'équipe Lifelong Kindergarden au sein du laboratoire Media du MIT (`http://llk.media.mit.edu`). La figure C.5 montre l'interface visuelle pour éditer trois scénarios d'une balle dans une animation du jeu *Pong* en utilisant Scratch. Le concepteur pose les formes à droite ; cliquer sur une forme fait apparaître son scénario à éditer au centre. Les primitives peuvent être glissées-déposées à partir de la boîte d'outils à gauche. Cette interface met en œuvre un langage visuel [RMMH+09] où chaque instruction est un bloc ("brique programmable" [RMSS96, MMS00]) dont la couleur correspond au type d'instruction, et dont la forme indique d'autres blocs pouvant

apparaître à côté ou devant ce bloc. Ce principe a été aussi utilisé pour développer les briques LEGO MindStorms [Pap80].

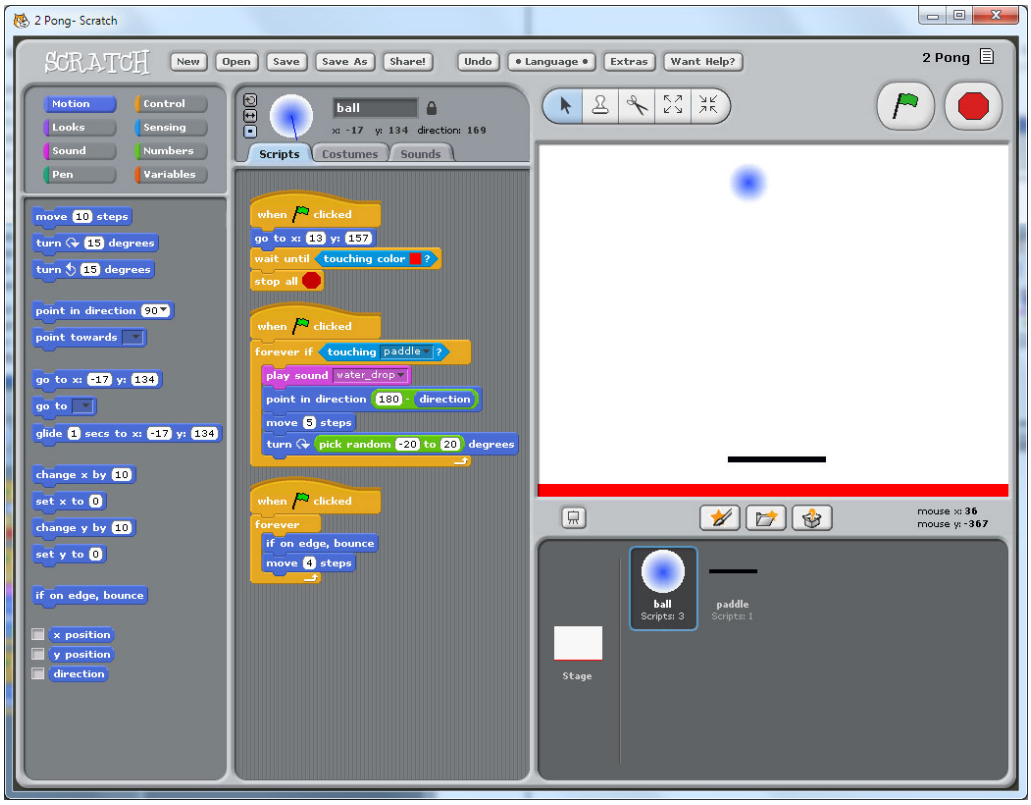


FIGURE C.5 – Interface visuelle du jeu *Pong Scratch*

La figure C.6 montre un autre exemple de langage visuel. Comme de nombreux langages visuels, l'outil de programmation *LabVIEW* permet aux utilisateurs de glisser et de déposer ces formes en utilisant l'interface graphique. Le langage de programmation *LabVIEW* permet de créer des simulations de circuits. Chaque boîte représente un composant de calcul, tandis que les lignes indiquent des flux de données.

Une autre manière pour interagir à travers un langage visuel peut résider en l'utilisation d'un formulaire (Figure C.7). Dans une telle interface, le concepteur ne peut pas librement faire glisser et déposer des formes, mais doit plutôt faire des sélections à partir des champs prédéfinis. La figure C.7 montre une interface utilisateur de Microsoft Word pour créer un style, qui est un ensemble d'instructions de formatage appliquées à plusieurs régions étiquetées dans le document.

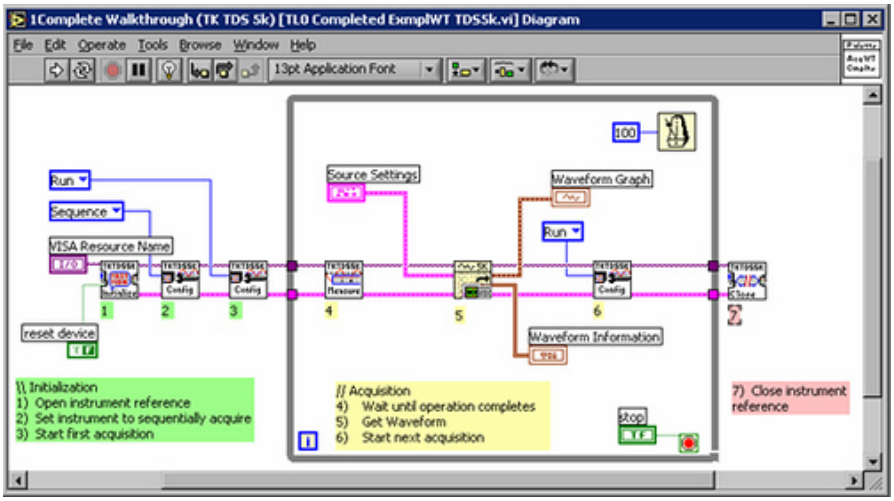


FIGURE C.6 – Langage de programmation *LabVIEW*

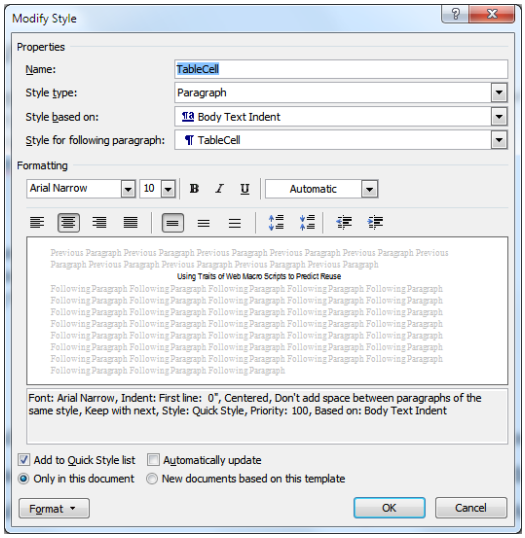


FIGURE C.7 – Interface utilisateur de Microsoft Word pour créer un style

C.2.2 Programmation par démonstration

Parfois appelée *programmation par exemple*, c'est une technique de programmation permettant à l'utilisateur de montrer la logique du nouveau programme, à partir de l'environnement de programmation en déduisant un programme qui représente cette logique. Plusieurs outils basés sur la programmation par démonstration sont disponibles pour créer par exemple des animations [RP00, MM99] et de nombreux autres types de programmes [CHK+93]. Un problème de la programmation par démonstration réside

dans la représentation du programme final sous une forme utile à l'utilisateur [CHK+93, YBDZ97] afin de permettre à l'utilisateur de vérifier, tester et déboguer le programme. Ainsi, la programmation par démonstration est souvent utilisée en combinaison avec des langages visuels ou textuels.

```
Sub PasteSpecial()
Selection.PasteSpecial DataType:=wdPasteText
End Sub
Sub SmallCaps()
'
' SmallCaps Macro
'
'
    With Selection.Font

        .SmallCaps = True

    End With
End Sub
Sub Subscript()
'
' Subscript Macro
'
'
    With Selection.Font
        If (.Superscript) Then
        .Superscript = False
        .Subscript = False
        ElseIf (.Subscript) Then
        .Subscript = False
        .Superscript = True
        Else
        .Subscript = True
        .Superscript = False
        End If


    End With
End Sub
```

FIGURE C.8 – Une macro Microsoft Word

Par exemple, un utilisateur peut créer une macro de Microsoft Word (comme illustré sur la figure C.8) en utilisant la programmation par démonstration. L'utilisateur clique d'abord sur un bouton ou un élément de menu indiquant que l'application doit commencer à regarder / enregistrer les actions de l'utilisateur. L'utilisateur peut utiliser l'interface graphique pour montrer le comportement désiré pour la macro ; par exemple, l'utilisateur peut utiliser une série d'éléments de menu et les fenêtres de dialogue pour coller les presse-papiers du système sous forme de texte. L'utilisateur peut cliquer sur un autre bouton pour "*Arrêter l'enregistrement*" afin que Microsoft Word arrête de regarder / enregistrer les actions de l'utilisateur. À ce moment, l'application génère une macro contenant des instructions textuelles en langage *VBScript* pour répéter les actions démontrées. L'utilisateur peut donner à la macro un raccourci clavier et un nom afin de l'éditer plus tard. En outre, l'utilisateur peut vouloir modifier les instructions de la macro afin d'effectuer une tâche légèrement différente que celle enregistrée, notamment si l'utilisateur veut affecter à la macro une tâche qui est impossible à décrire avec l'interface graphique actuelle.

C.2.3 Programmation en langage naturel

La programmation par spécification est un style de programmation avec lequel l'utilisateur décrit un programme souhaité et un outil génère ensuite le programme pour l'utilisateur. Comme dans la programmation par démonstration, le programme à générer peut alors être représenté pour faciliter la vérification et la personnalisation par l'utilisateur. Par exemple, [LL05] a mis en place un système qui acceptait une spécification en langage naturel et générait un programme correspondant écrit en Python.

Une limitation majeure de cette approche est qu'il est difficile pour un utilisateur de prédire quel programme va être généré à partir de n'importe quelle entrée. Par exemple, si le système prend en entrée de l'anglais, mais que la sortie est un langage de programmation tel que Python, l'utilisateur final doit être à l'aise dans les deux langages. Une autre limitation est que l'outil de programmation ne peut souvent traiter correctement qu'un nombre restreint d'entrées. Cela limite l'utilité de l'outil et rend également difficile la prédiction par l'utilisateur de certaines entrées particulières (et comment elles seront "comprises" par l'outil). Afin de rendre les limites de la langue d'entrée d'un outil plus abordables pour les utilisateurs, certains systèmes offrent une interface basée sur des formulaires visuels (Figure C.9).

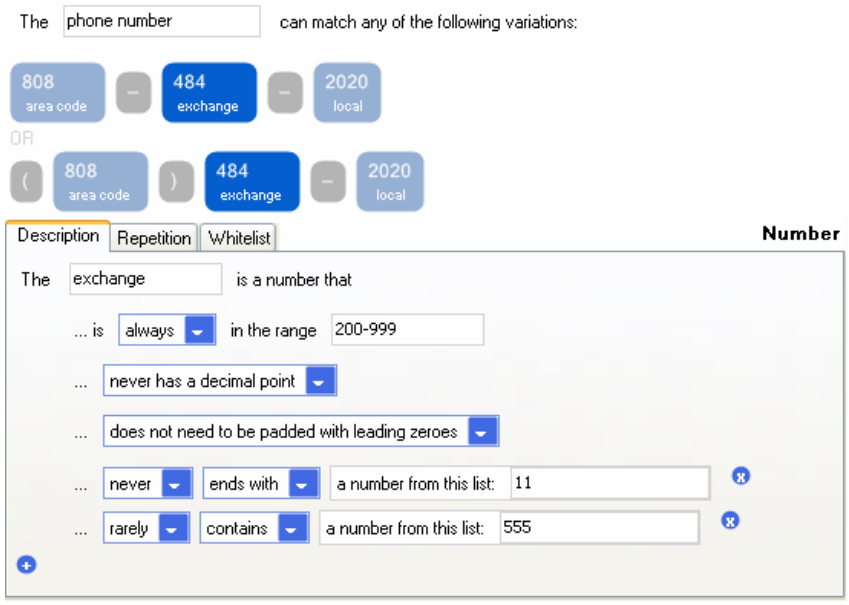


FIGURE C.9 – Spécification visuelle d'un numéro de téléphone

C.2.4 Programmation par texte

La programmation par texte est la technique traditionnelle de programmation. La figure C.10 illustre l'utilisation de l'outil de programmation *CoScripter* pour éditer une macro Web qui indique au navigateur de rechercher des informations sur le site Web d'*American Airlines* [LHML08]. L'exécution de la macro a été suspendue à la seconde instruction (à gauche), et demande à *CoScripter* de surligner le numéro de vol sur la page Web (à droite) et de le remplir à partir du fichier de configuration de l'utilisateur "Base de données personnelles" (en bas à gauche) [Sca10]. Une telle macro Web peut être généralement créée avec une approche de programmation par démonstration et personnalisée sous une forme textuelle.

Malgré la prolifération des styles de programmations alternatives, la programmation par texte reste largement utilisée en raison de sa concision et de son efficacité pour communiquer des concepts abstraits.

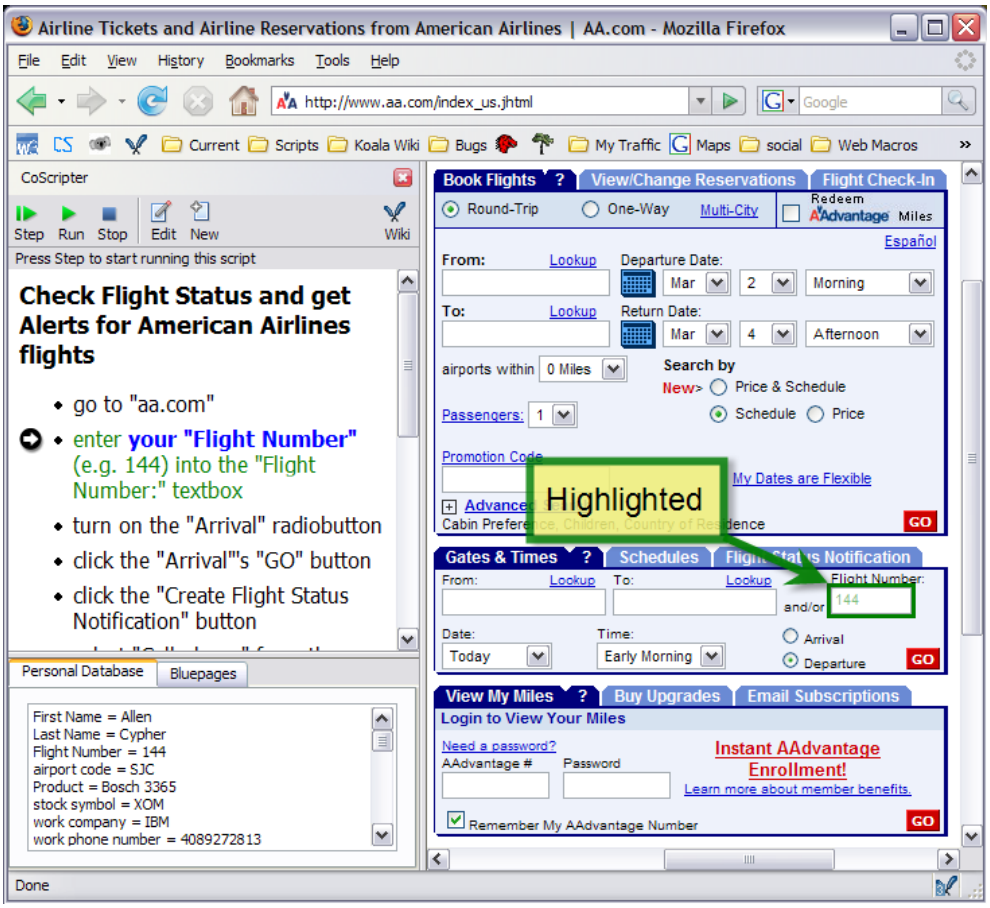


FIGURE C.10 – Interface de *CoScripter*

Annexe D

Web 2.0

Sommaire

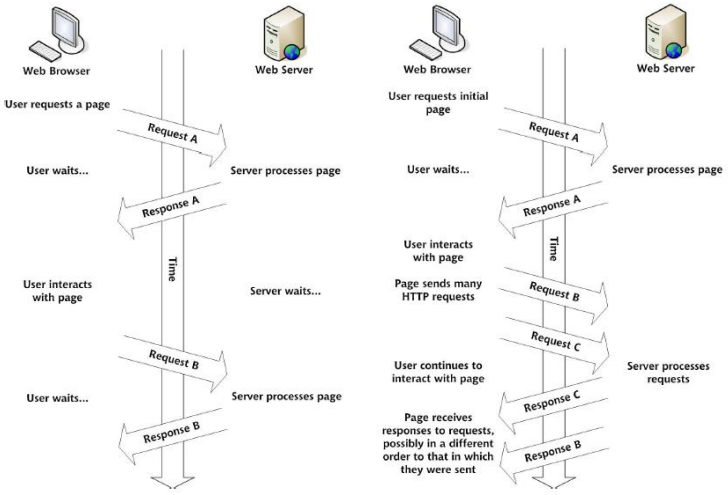
L'objectif de cette annexe n'est pas de lister exhaustivement toutes les technologies Web 2.0, mais elle a pour but de présenter les concepts du Web 2.0 utilisés dans les travaux de thèse.

AJAX (*Asynchronous JavaScript And XML*) [Gar05] n'est pas une nouvelle technologie. Elle consiste en plusieurs technologies qui se combinent dans de nouvelles approches puissantes. C'est un terme qui évoque l'utilisation conjointe d'un ensemble de technologies libres couramment utilisées sur le Web. AJAX peut améliorer l'interactivité et la rapidité d'une application Web pour qu'elle soit plus facile et plus intuitive à utiliser. Les applications AJAX sont parfois décrites comme "application de bureau dans le navigateur" [Eer06].

Ainsi le développement d'AJAX représente un bond énorme pour le développement du Web. Au lieu d'avoir à envoyer au serveur une seule et énorme masse puis attendre que le serveur renvoie une nouvelle page rendue, les développeurs Web peuvent communiquer avec le serveur en petits morceaux, et de manière sélective par des mises à jour de zones spécifiques de la page. La communication est donc *asynchrone*.

C'est une technique qui utilise JavaScript pour rafraîchir le contenu d'une page à partir d'un serveur Web sans recharger la page entière. Ceci est différent de la méthode traditionnelle de mise à jour de page Web qui nécessite du navigateur Web l'actualisation

de la page entière afin d'afficher toutes les modifications du contenu (Figure D.1). AJAX fait aussi un usage plus étendu des techniques d'interaction telles que l'*édition textuelle à la volée*, *glisser-déposer*, et les animations ou des transitions en CSS afin d'effectuer des changements dans l'interface utilisateur. Le résultat final est une application qui répond bien plus rapidement, que les utilisateurs passent beaucoup moins de temps d'attente pour les demandes à traiter.



(a) Web traditionnelle (b) AJAX

FIGURE D.1 – Comparaison entre une application Web traditionnelle (a) et AJAX (b)

Les technologies utilisées pour créer des applications Web AJAX englobent un certain nombre de langages différents : HTML, XML, JSON, DOM, CSS, XMLHttpRequest, JavaScript.

Dans les sections suivantes, nous allons détailler chaque technologie et discuter son rôle pour une application Web AJAX.

D.1 HTML

Le HTML (*HyperText Markup Language*) est un format de présentation de données permettant de créer des pages Web pouvant être lues dans des navigateurs. C'est un langage de description de données, et non un langage de programmation. Il est figé c'est-à-dire qu'une fois le document chargé dans le navigateur, il ne répond à aucune action de l'utilisateur sur le contenu de la page. Ce langage est pourvu d'un système de balisage qui va permettre de structurer notre document.

Une balise HTML (ou *tag* en anglais) est un élément que l'on va ajouter au texte de départ pour dire au navigateur de quelle manière l'afficher. Elle n'est pas affichée telle quelle dans le navigateur, elle est interprétée par celui-ci. Elle est toujours délimitée par les signes chevrons ouvrant et fermant (`<` et `>`) entourant le nom de chaque balise. Par exemple : `<html>`, `<body>`, `<div>`... Une balise peut aussi comporter de zéro à plusieurs attributs qui caractérisent des informations complémentaires. Ils se présentent sous la forme `nomattribut="valeur"`. Par exemple : `<html lang="fr">`.

Chaque balise ouverte doit être fermée, mais il existe des exceptions. Nous distinguons deux types de balises :

- Les balises simples. Ce sont des balises qui sont dites "vides" c'est-à-dire qu'elles ne vont contenir aucune autre balise HTML. Ces balises n'ont pas besoin d'être fermées. Par exemple : `<img src="adresse/de/l/image"/>`
- Les balises doubles. Les balises doubles sont dites ouvrantes/fermantes, c'est-à-dire qu'elles nécessitent deux balises, une ouvrante et une fermante dans lesquelles on va pouvoir mettre d'autres balises ou du texte. La balise fermante est identique à la balise ouvrante, à la différence qu'elle contient un `"/"` pour indiquer à quel endroit on la ferme. Par exemple : `<p>Ici du texte ou tout autre balise.</p>`.

Il est important de connaître la structure d'une page HTML. Commençons donc par voir celle-ci avec la structure minimale obligatoire d'une page Web (Figure D.2).

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
      "http://www.w3.org/TR/html4/loose.dtd">
2 <html>
3   <head><title>Titre de la page</title></head>
4   <body>
5     <!-- Ici le contenu du site web -->
6   </body>
7 </html>
```

FIGURE D.2 – Document HTML de base

La première ligne du document s'appelle le `DOCTYPE`. C'est une balise indispensable pour conserver la compatibilité du rendu d'une page sur les différents navigateurs modernes. Après le `DOCTYPE`, vient la balise `<html>`. Elle encadre l'ensemble des autres balises. La balise `<html>` contient deux grandes balises : `<head>`, qui va contenir tout l'en-tête de la page et `<body>` qui va contenir tout le corps du document.

XHTML (*eXtensible HyperText Markup Language*) est le successeur d'HTML sur le Web et se base sur la syntaxe définie par XML. La première version du XHTML est devenue une recommandation du W3C en 2000 et la version 1.1 est apparue en 2001. Pendant les années 2000, HTML 4 et XHTML sont les deux langages de programmation Web utilisés par les développeurs et interprétés par les navigateurs Web. Le XHTML 2.0 a été

ensuite développé. Cependant, le W3C (*World Wide Web Consortium* - www.w3.org) a officiellement annoncé l'abandon du développement du XHTML 2 en juillet 2009 afin de se consacrer entièrement à sa nouvelle star, le HTML 5.

Les pages Web dans les applications AJAX se composent de balises XHTML, qui sont facilement utilisées par tout développeur familier avec la technologie HTML, et qui possèdent de nombreux avantages du XML. Nous pouvons citer les avantages suivants :

- Nombre d'outils standards et de bibliothèques de scripts pour la visualisation, l'édition et la validation du XML ;
- Compatibilité avec plusieurs navigateurs ;
- Fonctionnement avec HTML DOM ainsi que XML DOM.

D.2 XML

XML (*eXtensible Markup Language*) est un langage de balisage permettant de stocker/transférer des données structurées. Contrairement à HTML, qui présente un nombre limité de balises (e.g. `html, body, p, a, div, table`, etc.), XML est un *métalangage*, qui va permettre d'inventer à volonté de nouvelles balises. Dans une application AJAX, XML est un format d'échange des données utilisé dans les requêtes HTTP asynchrones communiquant entre le navigateur et le serveur. Il y a évidemment de nombreux autres formats des données pour la communication entre le navigateur et le serveur, mais XML est l'un des plus utilisés.

La syntaxe XML est reconnaissable par son usage des chevrons (`<` et `>`) pour les balises d'éléments et de la chaîne `<!-- -->` pour les commentaires du code. L'utilisation du XML est recommandée car un document XML a toujours une structure définie et peut être transformé dans un autre document XML. XML permet à l'utilisateur de définir le contenu d'un document indépendamment de sa mise en forme, ce qui rend facile la réutilisation et l'intégration au sein d'autres applications et dans les environnements de présentation. Le code ci-dessous (Figure D.3) est un exemple du XML présentant le nom de ville, le code postal, les coordonnées d'une ville française.

```
<city>
  <postalcode>64130</postalcode>
  <name>Mauléon-Licharre</name>
  <countryCode>FR</countryCode>
  <lat>43.2253</lat>
  <lng>-0.885833</lng>
</city>
```

FIGURE D.3 – Un exemple XML d'information sur la ville de Mauléon-Licharre

D.3 JSON

JSON (JavaScript Object Notation) est un format léger d'échange de données. Il est facile à lire ou à écrire pour des humains et est aisément analysable ou générable par des machines. Il est basé sur un sous-ensemble du langage de programmation JavaScript. JSON est un format texte complètement indépendant de tout langage, mais les conventions qu'il utilise seront familières à tout programmeur habitué aux langages descendant du C. Ces propriétés font de JSON un langage d'échange de données idéal.

JSON se base sur deux structures :
- Une collection de couples nom/valeur. Divers langages la réifient par un *objet*, un enregistrement, une structure, un dictionnaire, une table de hachage, une liste typée ou un tableau associatif.
- Une liste de valeurs ordonnées. La plupart des langages la réifient par un *tableau*, un vecteur, une liste ou une suite.

Ces structures de données sont universelles. Pratiquement tous les langages de programmation modernes les proposent sous une forme ou une autre. Il est raisonnable qu'un format de données interchangeable avec des langages de programmation se base aussi sur ces structures.

En JSON, elles prennent les formes suivantes (Figure D.4) :
- Un *objet* est un ensemble de couples nom/valeur non ordonnés. Un objet commence par `{` (accolade gauche) et se termine par `}` (accolade droite). Chaque nom est suivi de `:` (deux-points) et les couples nom/valeur sont séparés par `,` (virgule).
- Un *tableau* est une collection de valeurs ordonnées. Un tableau commence par `[` (crochet gauche) et se termine par `]` (crochet droit). Les valeurs sont séparées par `,` (virgule).
- Une *valeur* peut être soit une *chaîne de caractères* entre guillemets, soit un *nombre*, soit `true` ou `false` ou `null`, soit un *objet* soit un *tableau*. Ces structures peuvent être imbriquées.
- Une *chaîne de caractères* est une suite de zéro ou plusieurs caractères Unicode, entre guillemets, et utilisant les échappements avec antislash. Un caractère est représenté par une chaîne d'un seul caractère. Une *chaîne de caractères* est très proche de ses équivalents en C ou en Java.
- Un *nombre* est très proche de ceux qu'on peut rencontrer en C ou en Java, sauf que les formats octal et hexadécimal ne sont pas utilisés.

Nous reprenons ensuite le même exemple que dans le paragraphe consacré à XML (Figure D.3) et le représentons en format JSON (Figure D.5).

```
object
    {} | { members }
members
    pair | pair , members
pair
    string : value
array
    [] | [ elements ]
elements
    value | value , elements
value
    string | number | object | array | true | false | null
string
    "" | " chars "
chars
    char | char chars
char
    any-Unicode-character-except-"-or-\-or-control-character
    | \" | \\ | \/ | \b | \f | \n | \r | \t | \ufour-hex-digits
number
    int | int frac | int exp | int frac exp
int
    digit | digit1-9 digits | - digit | - digit1-9 digits
frac
    . digits
exp
    e digits
digits
    digit | digit digits
e
    e | e+ | e- | E | E+ | E-
```

FIGURE D.4 – Grammaire de JSON

```
{
    city: {
        postalcode: 64130,
        name: 'Mauléon-Licharre',
        countrycode: 'FR',
        lat: 43.2253,
        lng: -0.885833
    }
}
```

FIGURE D.5 – Un exemple JSON de l'information sur la ville de Mauléon-Licharre.

D.4 DOM

DOM (*Document Object Model*) est une représentation orientée objet des documents XML et HTML qui fournit une API pour modifier le contenu, la structure et le style de ces documents.

À l'origine, les navigateurs spécifiques, tels que Netscape Navigator et Microsoft Internet Explorer, fournissaient différentes façons propres pour manipuler des documents HTML en utilisant JavaScript. DOM a été créé par le W3C pour fournir une façon indépendante de la plate-forme et du navigateur pour réaliser les mêmes tâches. DOM représente la structure d'un document XML ou HTML comme une hiérarchie d'objets, ce qui est idéal pour l'analyse par des outils XML standards.

JavaScript dispose de nombreuses API pour exploiter ces structures DOM pour l'analyse et la manipulation du document. C'est l'une des principales manières pour atteindre des changements asynchrones d'une page Web que nous voyons dans une application AJAX. L'autre fonction importante du DOM est qu'il fournit une manière standard pour JavaScript afin d'attacher des évènements à des éléments figurant sur une page Web. Cela permet d'enrichir des interfaces utilisateur en donnant la possibilité aux utilisateurs d'interagir avec la page au-delà des liens simples et des éléments de formulaire. Un bon exemple est la fonctionnalité de *glisser-déposer* qui permet aux utilisateurs de faire glisser des parties de la page autour de l'écran, et de les déposer pour déclencher des éléments spécifiques de la fonctionnalité. Ce type de fonctionnalité utilisée n'existait auparavant que dans les applications de bureau mais, grâce au DOM, elle est dorénavant aussi présente dans les navigateurs Web.

D.5 CSS

CSS (*Cascading Style Sheets*) fournit une méthode unifiée pour contrôler l'apparence des éléments d'interface utilisateur dans une application Web. Nous pouvons utiliser les CSS pour changer presque n'importe quel aspect de l'affichage de la page Web : taille de polices, couleur, espacement, positionnement des éléments, etc.

Il y a deux façons pour inclure une feuille de style CSS dans le code HTML d'une page Web :

- définition directe de la CSS dans le contenu HTML via la balise `<style>`. Par exemple : `<style type="text/css"> déclaration des styles </style>` ;
- déclaration d'un lien vers le fichier CSS via la balise `<link>`. Par exemple : `<link type="text/css" rel="stylesheet" href="style.css"/>`.

La première spécification CSS a été publiée en 1996, puis la spécification CSS2 est sortie en 1998. La spécification CSS3 a été développée depuis 1999 et est largement utilisée depuis 2011. Actuellement, le W3C est en train de développer la spécification CSS4 qui n'est encore supportée par aucun navigateur Web. La figure D.6 illustre un exemple de CSS2 de l'affichage d'une zone (nommée `info`) dans une page Web.

Dans une application AJAX, une très bonne utilisation de CSS permet de fournir la rétroaction ("*feedback*") de l'interface utilisateur (avec des animations et transitions dirigée par CSS), ou d'indiquer les parties de la page avec lesquelles l'utilisateur peut

```
#info {
  display: block; /* la zone est affichée comme un bloc */
  font-size: 12pt; /* la taille de la police est de 12pixels */
  font-family: Arial; /* la fonte est Arial */
  background-color: #33CCFF; /* la couleur de l'arrière-plan */
  border: 2px #0000FF solid; /* la bordure de la zone */
}
```

FIGURE D.6 – Exemple de CSS

interagir (avec les changements de couleur ou l'apparence des parties déclenchées, par exemple, par survol). Par exemple, nous pouvons utiliser les transitions CSS pour indiquer que certaines parties de l'application sont en attente d'une requête HTTP de traitement sur le serveur.

D.6 XMLHttpRequest

XMLHttpRequest est un objet ActiveX [67] ou JavaScript [68] permettant d'envoyer et de recevoir des requêtes et réponses HTTP vers et à partir des serveurs Web. Il reçoit depuis un serveur des données au format XML, JSON, ou un texte simple à l'aide de requêtes HTTP. L'avantage principal de XMLHttpRequest est le fait qu'il soit *asynchrone*, c'est à dire que la page entière ne doit plus être rechargée en totalité lorsqu'une partie doit changer. Cela peut faire gagner du temps et une meilleure interaction avec l'utilisateur.

La réponse du serveur, qu'elle soit un document XML (`responseXML`) ou bien une chaîne de texte (`responseText`), peut être transmise côté client et permet au développeur d'utiliser JavaScript (Figure D.7) pour mettre à jour certaines parties de l'interface utilisateur de l'application Web.

Le code côté serveur peut être implémenté avec un langage de traitement tel que PHP, Java... Le figure D.9 illustre un simple exemple du code PHP qui fait l'addition de deux numéros et est appelé par le code AJAX présenté dans la figure D.8.

67. Microsoft a implémenté en premier XMLHttpRequest dans Internet Explorer 5 pour Windows en tant qu'objet ActiveX.

68. Le projet Mozilla a fourni une version JavaScript-native avec une API compatible dans le navigateur Mozilla.

```
1  /* Fonction pour créer un objet XMLHttpRequest utilisé dans les
       applications AJAX */
2  function createXHR() {
3      var request = null;
4      if (window.XMLHttpRequest) {
5          request = new XMLHttpRequest(); /* pour Mozilla, Firefox, ... */
6      }
7      else if (window.ActiveXObject) { /* pour Internet Explorer */
8          try {
9              request = new ActiveXObject("Msxml2.XMLHTTP");
10         }
11         catch (e) {
12             try {
13               request = new ActiveXObject("Microsoft.XMLHTTP");
14             }
15           catch (e) {}
16         }
17     }
18     return request;
19 }
```

FIGURE D.7 – Fonction JavaScript pour créer un objet XMLHttpRequest

D.7 JavaScript

JavaScript est le liant permettant d'assembler une application AJAX. Il effectue plusieurs rôles dans le développement AJAX :

- contrôler les requêtes HTTP en utilisant XMLHttpRequest ;
- analyser le résultat provenant du serveur, en utilisant soit les méthodes de manipulation du DOM, soit les méthodes personnalisées en fonction du format d'échange de données utilisé ;
- présenter les données du résultat sur l'interface utilisateur, soit en utilisant les méthodes de manipulation du DOM pour insérer du contenu dans la page Web (en mettant à jour la propriété `innerHTML` d'un élément), soit en modifiant les propriétés CSS des éléments.

En raison de sa longue histoire d'utilisation dans la programmation légère du Web, JavaScript n'a pas été considéré comme un "langage de programmation sérieux" par certains développeurs d'applications traditionnelles, même s'il s'agit d'un langage dynamique capable de supporter des méthodologies de programmation orientée objet. Cette mauvaise perception de JavaScript est en train de changer rapidement à mesure que les techniques de développement AJAX étendent la puissance et les fonctionnalités des applications Web. À la suite de l'avènement d'AJAX, JavaScript semble être maintenant l'objet d'une sorte de renaissance, et la croissance explosive du nombre des bibliothèques JavaScript disponibles pour le développement AJAX est une conséquence.

```
function sum(num1, num2) {
        /* créer un objet XMLHttpRequest */
        var xhr = createXHR();

        /* utiliser la méthode POST pour communiquer avec le serveur */
        xhr.open("POST", "getsum.php", true);
        xhr.setRequestHeader("Content-type",
            "application/x-www-form-urlencoded");

        /* envoyer les paramètres au serveur */
        xhr.send("a=" + num1 + "&b=" + num2);

        /* détecter les changements d'état de l'objet XMLHttpRequest */
        xhr.onreadystatechange = function() {
              if (xhr.readyState == 4) { /* le serveur a fini son travail */
                    if (xhr.status == 200) { /* les données réponses sont
                        successivement retournées */
                           sum = xhr.responseText;  /* récupérer les
                               données sous forme de texte brut */
                    }
                    else {
                           /* sinon, faire un avertissement */
                           alert("Error: " + xhr.status + " " +
                               xhr.statusText);
                    }
              }
         }
}
```

FIGURE D.8 – Une fonction en AJAX pour demander au serveur de calculer la somme de deux nombres

```
<?php
/* récupérer les valeurs des paramètres */
$variable1 = $_POST["a"];
$variable1 = $_POST["b"];

/* calculer la somme de deux variables */
$somme = $variable1 + $variable2;

/* afficher la somme */
echo $somme;
?>
```

FIGURE D.9 – Un exemple en PHP pour calculer la somme de deux nombres

Annexe E

Web sémantique

Sommaire

Cette annexe a pour but de montrer / expliquer de façon simple et courte des concepts du Web sémantique utilisés dans les travaux de thèse.

E.1 Web sémantique et RDF

Le Web sémantique [SH01, Ogb02, Pal01] est défini comme "*une extension du Web actuel dans lequel on donne un sens bien défini à l'information pour permettre aux machines et aux gens de travailler en coopération*" [BLHL01]. Les informations sur le Net sont principalement créées pour la lecture humaine (*e.g.* document textuel, photographique, etc.). Elle sont donc difficilement expoitables par les machines. Le Web sémantique vise à permettre aux machines de comprendre et d'exploiter automatiquement des documents et des données sémantiques. Le défi du Web sémantique est de fournir un langage qui exprime à la fois des données et des règles pour raisonner sur les données de la toile.

Il permet aussi aux moteurs de recherche de fournir des résultats mieux ciblés en réponse à des requêtes des utilisateurs. En effet, les moteurs de recherche courants fournissent souvent une grande quantité de résultats dont certains n'ont que peu ou pas de rapport avec la requête initiale de l'utilisateur, le principe de base de la sélection d'une page étant l'occurrence des mots clés de la requête dans son contenu et dans les balises `<meta>` associées quand celles-ci existent. Les résultats des recherches peuvent être affinés en tenant compte des informations supplémentaires concernant la requête. Par exemple, un moteur de recherche pourra prendre en compte le fait qu'une requête

concerne un article dont l'auteur se nomme "Fontaine" pour ne sélectionner que les articles dont l'auteur se nomme ainsi et non pas se contenter de retourner tous les articles dans lesquels le mot "fontaine" apparaît.

Le Web sémantique peut être considéré comme une série de couches (Figure E.1), à partir de la base avec une couche de représentation comme RDF, puis au-delà des nouveaux langages présentant les ontologies, les règles, les preuves et les logiques.

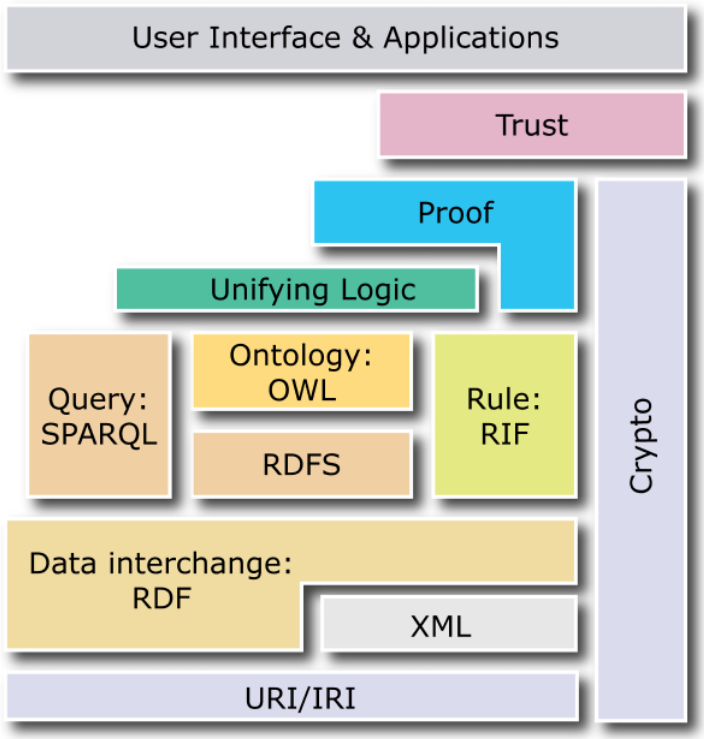


FIGURE E.1 – Architecture du Web sémantique conçue par Tim Barners-Lee

Un des composants de base du Web sémantique est RDF (*Resource Description Framework* - www.w3.org/RDF), développé par le W3C pour exprimer des méta-données structurées décrivant des ressources du Web. Comme nous allons le voir plus en détail par la suite, un modèle RDF est un ensemble de triplets représentant un graphe orienté. Ce graphe peut être sérialisé sous différentes formes comme la syntaxe RDF/XML [BM04] ou encore Notation3 [BL06]. Ces sérialisations, notamment RDF/XML, sont très utiles pour l'échange des métadonnées entre machines et leur utilisation par des agents. Elles sont par contre peu appropriées pour la lecture, la compréhension et l'édition des modèles par un utilisateur humain.

Dans le reste de cette section, nous détaillons les concepts importants du Web sémantique, pour aider à expliquer les fondements de cette nouvelle technologie Web.

RDF constitue un modèle de déclarations (*statement*) faites au sujet des ressources. Une ressource est n'importe quoi identifié par une URI (*Uniform Resource Identifier*) associée. Une ressource est une structure uniforme de trois parties : le sujet, le prédicat et l'objet. Par exemple : l'auteur [prédicat] de "Le voyage aux Pyrénées de James David Forbes" [sujet] est Jean-Pierre Daraux [objet]. RDF permet d'exprimer de telles déclarations d'une manière formelle que les agents logiciels peuvent lire et agir. Il permet d'exprimer une collection de déclarations sous forme graphique, comme une série de triplets (sujet, prédicat, objet).

Sur le Web, nous pouvons associer *une chose* à une URI au lieu de donner son nom. Tout ce qui se trouve sur le Web peut avoir une URI. L'URI est le fondement de l'Internet. Alors que la plupart des autres parties du Web peuvent être remplacées, l'URI contient le reste du Web dans son ensemble. Vous êtes probablement déjà familiers avec une forme d'URI : l'URL ou *Uniform Resource Locator*. Une URL est l'adresse qui vous permet de visiter une page Web, comme : `http://www.luongthenhan.com`. L'URL permet à un navigateur Web de localiser une ressource spécifique. En plus des URL, il existe d'autres formes d'URI comme par exemple les `"mailto:"` qui sont utilisées pour encoder des adresses e-mail, ou le moins bien connu `"mid:"` qui référence les messages ou des parties de messages SMTP/MIME.

Prenons un exemple. Quelqu'un a dit que le livre français "Le Voyage Aux Pyrénées De James David Forbes en 1835" a été écrit par Jean Pierre Daraux. Cependant, aucune machine ne peut traiter cette information sauf si elle est exprimée sous une forme formelle. Pour cela, RDF donne un moyen de faire des déclarations qui sont compréhensibles par la machine. Toutefois, la machine ne peut pas vraiment "comprendre", mais elle peut y faire face d'une manière qui semble plausible (Figure E.2).

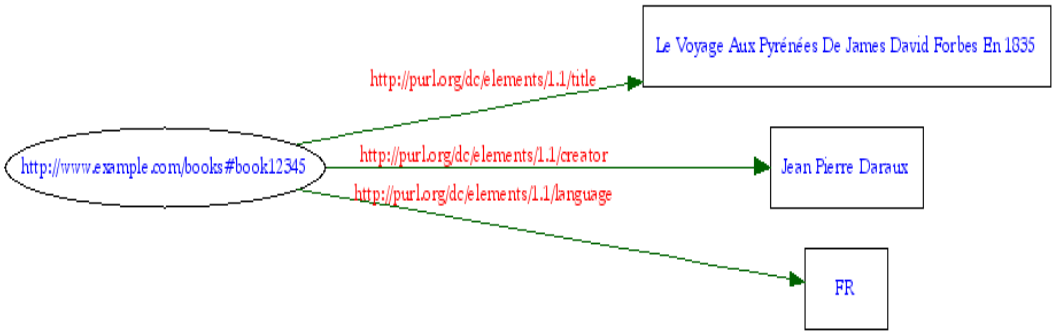


FIGURE E.2 – Graphe RDF décrivant un livre

Dans RDF, les URI peuvent être utilisées pour identifier :
- des individus (*e.g.* Jean Pierre Daraux) ;
- des choses (*e.g.* Personne) ;
- des propriétés de ces choses (*e.g.* titre, nom) ;
- des valeurs de ces propriétés (*e.g.* `mailto: em@w3.org` est la valeur de la propriété boîte aux lettres).

RDF fournit également une syntaxe XML (appelée RDF/XML) pour enregistrer et échanger ces graphes. La figure E.3 est un petit exemple de RDF exprimé en RDF/XML correspondant au graphe de la figure E.2 :

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xml:base="http://www.example.com/books">
  <rdf:Description rdf:ID="book12345">
    <dc:title>Le Voyage Aux Pyrénées De James David Forbes En
        1835</dc:title>
    <dc:creator>Jean-Pierre Daraux</dc:creator>
    <dc:language>FR</dc:language>
  </rdf:Description>
</rdf:RDF>
```

FIGURE E.3 – Exemple de RDF

La figure E.4 est un petit exemple de RDF en Notation3 (N3) correspondant au graphe de la figure E.2 :

```
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
<http://www.example.com/books#book12345> dc:creator "Jean-Pierre Daraux";
  dc:language "FR";
  dc:title "Le Voyage Aux Pyrénées De James David Forbes En 1835".
```

FIGURE E.4 – Exemple de Notation3

E.2 Schéma RDF (RDFS)

L'utilisation des URI est inutile si nous ne devons jamais décrire ce qu'ils signifient. C'est là que des schémas et des ontologies entrent en jeu. Un schéma et une ontologie sont des façons de décrire le sens et la relation des termes. Cette description (en RDF

également) aide les systèmes informatiques à utiliser des termes plus facilement, et de décider comment les convertir entre eux. Prenons un exemple simple : "*un père est un parent* et décrivons cet exemple sous la forme d'un schéma RDFS (Figure E.5).

```
@prefix ex: <http://example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
# A father is a type of parent:
ex:father rdfs:subClassOf ex:parent .
```

FIGURE E.5 – Exemple RDFS

Nous pouvons également définir quelques propriétés dans le schéma qui peuvent être utilisées comme des prédicats de déclarations RDF. RDFS permet d'appliquer certaines contraintes simples sur les propriétés. Par exemple, `rdfs:domain` permet de restreindre l'utilisation d'une propriété sur une certaine catégorie de ressources ; `rdfs:range` déclare que la valeur d'une propriété doit être d'une certaine catégorie de ressources...

E.3 Langage d'ontologie OWL

En informatique, une ontologie est "*un document définissant de façon formelle les relations entre les termes*" [Wiki]. Elle possède une taxonomie et un ensemble de règles d'inférence.

La taxonomie définit des classes d'objets et les relations entre elles. Par exemple, une adresse peut être définie comme un type de lieu, et les codes postaux de ville peuvent être définis pour s'appliquer seulement à des lieux...

Les règles d'inférence dans les ontologies sont un outil très puissant à utiliser pour déduire les relations entre les ressources sur le Web. Par exemple, on dit que "Bayonne est une sous-préfecture" et qu'une sous-préfecture est une ville, alors Bayonne est aussi une ville.

Les ontologies peuvent valoriser le fonctionnement du Web de plusieurs façons. On peut les utiliser de façon simple pour améliorer la pertinence des recherches. Par exemple, un programme de recherche ne peut rechercher que les pages faisant référence à un concept précis au lieu de celles qui utilisent des mots-clés ambigus. De plus, ce marquage facilite le développement de programmes qui peuvent s'atteler à des questions compliquées dont les réponses ne se trouvent pas sur une seule page Web. Supposons que l'on veut chercher des informations concernant Monsieur Luong qui a présenté son article lors d'une conférence DocEng2011. On ne connaît pas son prénom, mais on sait qu'il a travaillé pour le LIUPPA. Un programme de recherche "intelligent" peut filtrer les pages

des personnes dont le nom est Luong et puis trouver ceux qui travaillent pour le LIUPPA et qui ont assisté à DocEng2011.

OWL (*Web Ontology Language*) est un langage informatique basé sur une syntaxe RDF permettant de définir des ontologies Web ; autrement dit, il permet de définir des terminologies pour décrire des domaines spécifiques.

OWL peut être considéré comme une extension de RDF et RDFS, mais il est plus expressif que RDF et RDFS. Comme RDFS, OWL présente les concepts de classe, de ressource, de littéral et de propriétés des sous-classes, de sous-propriétés. De plus, OWL ajoute les concepts de classes équivalentes, de propriétés équivalentes, d'égalité de deux ressources, de leurs différences, du contraire, de symétrie et de cardinalité (Figure E.6).

```
@prefix :      <http://example.com/pizzas.owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
:Pizza rdfs:subClassOf
               [ a owl:Restriction ;
                 owl:onProperty :hasBase ;
                 owl:someValuesFrom :PizzaBase ] ;
              owl:disjointWith :PizzaBase .

:NonVegetarianPizza  owl:equivalentClass
               [  owl:intersectionOf
                   (  [owl:complementOf :VegetarianPizza]
                      :Pizza  )  ] .

:isIngredientOf
            a owl:TransitiveProperty , owl:ObjectProperty ;
            owl:inverseOf :hasIngredient .
```

FIGURE E.6 – Exemple OWL (Source : www.obitko.com)

E.4 Langage de requête SPARQL

SPARQL (*SPARQL Protocol and RDF Query Language* - www.w3.org/TR/rdf-sparql-query) est un langage de requête permettant d'interroger une base de triplets RDF. Il est devenu une recommandation W3C le 15 janvier 2008 issue de la recherche sur le Web sémantique. SPARQL est considéré comme l'équivalent de SQL. En effet, en SQL on accède aux données d'une base de données via ce langage de requête, et avec SPARQL, on accède aux données du Web des données. Toutefois, SPARQL est différent de SQL. Alors que SQL est un langage de requête adapté aux bases de données relationnelles, SPARQL est adapté à la structure spécifique des graphes RDF et s'appuie sur les triplets qui les constituent.

Il y a deux types des requêtes SPARQL : les requêtes interrogatives et les requêtes constructives. Une requête interrogative est représentée par le mot clé "`SELECT`" et permet d'extraire du graphe RDF un sous-graphe correspondant à un ensemble de ressources vérifiant les conditions définies dans une clause "`WHERE`. Une requête constructive est représentée par le mot clé "`CONSTRUCT`" et permet de construire un nouveau graphe conforme à la nouvelle ontologie, dans le cas où on veut transformer les données de départ dans un nouveau système.

Supposons par exemple que nous avons un graphe RDF contenant des informations géographiques (Figure E.7).

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:gn="http://www.geonames.org/ontology#"
   xmlns:wgs84_pos="http://www.w3.org/2003/01/geo/wgs84_pos#">
   <gn:City>
      <gn:name>Bayonne</gn:name>
      <gn:population>44506</gn:population>
      <wgs84_pos:lat>43.4936</wgs84_pos:lat>
      <wgs84_pos:long>-1.475</wgs84_pos:long>
   </gn:City>
   <gn:City>
      <gn:name>Pau</gn:name>
      <gn:population>84036</gn:population>
      <wgs84_pos:lat>43.3017</wgs84_pos:lat>
      <wgs84_pos:long>-0.3686</wgs84_pos:long>
   </gn:City>
</rdf:RDF>
```

FIGURE E.7 – Exemple de RDF sur des informations géographiques

Nous pouvons utiliser une requête interrogative SPARQL pour retourner une liste des noms de ville avec le nombre d'habitants (Figure E.8). Le résultat obtenu est deux lignes de couple (`nom, population`) correspondant aux conditions de la clause `WHERE` : ("Bayonne", 44506) et ("Pau", 84036).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX gn: <http://www.geonames.org/ontology#>
SELECT DISTINCT ?nom ?population
WHERE { ?ville rdf:type gn:City.
        ?ville gn:name ?nom.
        ?ville gn:population ?population
}
```

FIGURE E.8 – Exemple de SPARQL interrogative

La requête constructive SPARQL suivante (Figure E.9) est utilisée pour générer des nouvelles données RDF à partir des triplets RDF dans la figure E.7.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX gn: <http://www.geonames.org/ontology#>
PREFIX wm: <http://erozate.iutbayonne.univ-pau.fr/windmash#>
CONSTRUCT { ?x wm:nomVille ?nom }
WHERE { ?ville rdf:type gn:City.
        ?ville gn:name ?nom.
}
```

FIGURE E.9 – Exemple de SPARQL constructive

Le résultat obtenu est donc (Figure E.10) :

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:wm="http://erozate.iutbayonne.univ-pau.fr/windmash#">
  <rdf:Description>
    <wm:nomVille>Bayonne</wm:nomVille>
  </rdf:Description>
  <rdf:Description>
    <wm:nomVille>Pau</wm:nomVille>
  </rdf:Description>
</rdf:RDF>
```

FIGURE E.10 – Résultat d'une requête SPARQL constructive

Annexe F

Compléments de l'API WIND et de WINDMash

Sommaire

F.1 Classes et méthodes de l'API WIND

Les classes du modèle générique unifié (Figure F.1) relatives aux trois phases (*Contenu*, *Interface* et *Interaction*) ont été implémentées dans l'API WIND. Nous listons ci-dessous les classes principales de l'API WIND selon un ordre alphabétique.

- `WIND` : Espace de noms regroupant les objets de l'API WIND

- `WIND.Annotation` : Permet de créer un objet de la classe `Annotation`.
  `var annotation = new WIND.Annotation("ville", "Mauléon-Licharre", new Array(mp));`
  Cet exemple va créer une annotation sur l'entité nommé *Mauléon-Licharre* dont la sémantique est une *ville* et la représentation est une partie `mp` sur la carte.

- `WIND.Calculation` : Permet de créer un objet de la classe `Calculation`. Une opération de calcul permet de créer une nouvelle annotation à partir d'une annotation de départ sur laquelle on applique un calcul. Du fait que les annotations sont de nature géographique, les opérations autorisées sont elles-mêmes de nature géographique : calcul de distance, d'orientation...
  `var react = new WIND.Calculation(annot, "prefecture_of_town", m);`

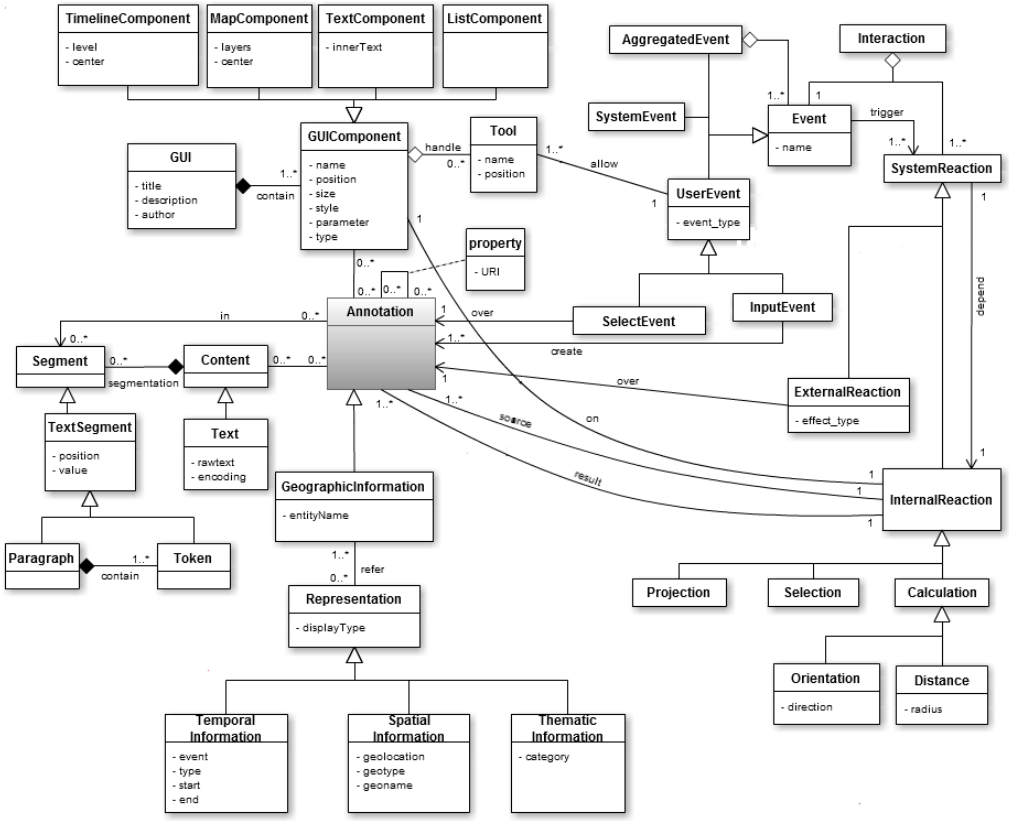


FIGURE F.1 – Diagramme des classes implémentées dans l'API WIND

Cet exemple va calculer la préfecture de la ville annotée (représentée par le paramètre `annot`) et renvoyer le résultat qui est une annotation sur cette préfecture sur la carte `m`.

– `WIND.Displayer` : Permet de créer un objet de la classe `Displayer`. C'est la classe mère des classes `Text`, `Map`, `List` et `Timeline`.

– `WIND.ExternalReaction` : Permet de créer un objet de la classe `ExternalReaction`.
```
var reaction = new WIND.ExternalReaction(annot, "zoom");
```
Cet exemple décrit une réaction externe du système où la carte va faire un zoom avant sur la représentation de l'annotation `annot`.

– `WIND.GUI` : Permet de créer un objet de la classe `GUI`.
```
var doc = new WIND.GUI("mydiv", "author" : "The Nhan LUONG", "title" :
"exemple", "description" : "exemple pour la documentation");
```

Cet exemple va créer une interface utilisateur de l'application qui contiendra par la suite tous les composants d'interface souhaités (texte, carte, frise...) avec un titre et une description de l'application.

- `WIND.InputEvent` : Permet de créer un objet de la classe `InputEvent`.
  `var evt1 = new WIND.InputEvent("Input Lieux", bouton1);`
  Cet événement correspond à l'outil *bouton1* (de type `Tool`) qui va permettre à l'utilisateur d'annoter sur un composant d'interface.

- `WIND.Interaction` : Permet de créer un objet de la classe `Interaction`.
  `var interaction = new WIND.Interaction(evt, new Array(react1,react2,-react3));`
  Cet exemple va définir une interaction avec l'événement `evt` et un ensemble (tableau) de réactions (ici trois : `react1`, `react2` et `react3`).

- `WIND.InternalReaction` : Permet de créer un objet de la classe `InternalReaction`. C'est la classe mère de trois classes `Selection`, `Projection` et `Calculation`.

- `WIND.List` : Construit un objet de type `List` qui affiche les items sous forme d'une liste à puces.

- `WIND.Map` : Permettre de créer un objet `Map`.
  `var carte = new WIND.Map("mymap", 'top': 10, 'left': 20, 'width': 600, 'height': 400, 'name': "carte", 'type': 'Google Street', 'longitude': -0.9331, 'latitude': 45.9236, 'zoom': 5, 'draggable': false, 'resizable' : false, 'color': '#FF9900', 'border': '#FF9900 2px solid', 'header': false, 'removable': false, 'configurable': false);`
  Cet exemple va créer un composant cartographique dans le DIV *mymap* de la page Web. Il est positionné à la position à 10px de haut, à 20px de gauche ; il est en 600px de largeur et 400px en hauteur. Ce composant cartographique va afficher la couche *Google Street* et se centre à l'endroit dont les coordonnées sont -0,9331 de longitude et 45,9236 de latitude, sur un niveau de zoom 5. De plus la carte n'est pas supprimable.

- `WIND.Map.Itinerary` : Permet de créer un objet `Map.Intinerary`.
  `var iti = new WIND.Map.Itinerary(new Array( new OpenLayers.Geometry.Point(2,45), new OpenLayers.Geometry.Point(3,46)), "route");`
  Cet exemple va tracer un itinéraire entre deux points de coordonnées (2,45) de latitude et (3,46) de longitude par les routes sur la carte grâce au service de Google.

- `WIND.Map.Marker` : Permet de créer un marqueur sur la carte.
  `var marqueur = new WIND.Map.Marker("POINT(0.5678 45.5682)", "marker.png", 15);`

Cet exemple va mettre un marqueur (l'icône *marker.png*) avec une taille de 15 pixels sur le point de coordonnées(0.5678, 45.5682) sur la carte.

– `WIND.Map.Part` : Permet de créer un objet `Map.Part`.
`var part = new WIND.Map.Part("POINT(0.43, 46.56)", "EPSG:4326", "strokeColor:#FF9900,strokeWidth:3,strokeOpacity:0.8,fillColor:#FFFF00, fillOpacity:0.4");`
Cet exemple va créer et dessiner un point sur la carte à l'endroit dont les coordonnées sont 0,43 de longitude et 46,56 de latitude. Ce point va être une zone sensible pour déclencher des interactions du système que le développeur pourra définir.

– `WIND.Projection` : Permet de créer un objet de la classe `Projection`. L'opération de projection consiste à transférer une annotation présente sur un composant d'interface vers un autre composant d'interface. Dans cette opération, le système doit calculer, en cours d'interaction, la représentation de l'annotation transférée vers le composant de destination.
`var react = new WIND.Projection(annot, m);`
Cet exemple va copier l'annotation `annot` dans le composant d'interface `m`.

– `WIND.Representation` : Permet de créer un objet de la classe `Representation`.
`var rep = new WIND.Representation("map", annot, m);`
Cet exemple va créer une représentation cartographique ("*map*") de l'annotation `annot` dans le composant d'interface `m`.

– `WIND.SelectEvent` : Permet de créer un objet de la classe `SelectEvent`. `var evt1 = new WIND.SelectEvent("click", annot1);`
Cet exemple définit un événement correspondant au clic gauche sur l'annotation `annot1`.

– `WIND.Selection` : Permet de créer un objet de la classe `Selection`. L'opération de sélection permet au système de déterminer quelle annotation a été sélectionnée par l'utilisateur parmi toutes les annotations affichées sur un composant d'interface.
`var react = new WIND.Selection(new Array(annot1,annot2,annot3));`
Cet exemple va définir une sélection d'une annotation parmi les trois annotations `annot1`, `annot2` et `annot3`.

– `WIND.SensiblePart` : Permet de créer un objet de la classe `SensiblePart`. C'est la classe mère des classes `Text.Part`, `Map.Part`, `List.Part` et `Timeline.Part`.

– `WIND.SystemReaction` : Permet de créer un objet de la classe `SystemReaction`. C'est la classe mère de deux classes `InternalReaction` et `ExternalReaction`.

– `WIND.Text` : Permet de créer un objet `Text`.
```
var texte = new WIND.Text("mytext", 'top': 100, 'left': 10, 'width': 400,
'height': 180, 'draggable': false, 'resizable' : false, 'color': '#0033CC',
'border': '#0033CC 2px solid', 'header': false, 'removable': false, 'con-
figurable': false);
```
Cet exemple va créer un composant textuel dans le DIV *mytext* de la page Web. Il est positionné à la position à 100px de haut, à 10px de gauche ; il a 400px de largeur et 180px en hauteur.

– `WIND.Text.AnnotationButton` : Permet de créer un outil de type `Text.AnnotationButton`. Cette classe hérite de la classe `Tool`.
```
var buttonlieux = new WIND.Text.AnnotationButton("Lieux", "Choisir des
lieux dans le texte", "bottom", "place", "red");
```
Cet exemple va créer un bouton d'annotation sur le composant texte. Ce bouton est situé en bas du composant textuel. Il est de couleur rouge avec comme titre "Lieux" et comme description "Choisir des lieux dans le texte". En annotant des mots dans le texte avec ce bouton, l'utilisateur va créer une annotation de type "place".

– `WIND.Text.Paragraph` : Permet de créer un objet `Paragraph`.
```
var para = new WIND.Text.Paragraph("mytext_para1", "Je suis allé à Biar-
ritz.");
```
Cet exemple va créer un paragraphe dans un composant textuel avec le contenu "*Je suis allé à Biarritz.*".

– `WIND.Text.Part` : Permet de créer un objet `Text.Part`.
```
var tp = new WIND.Text.Part(1, 4, 5);
```
Cet exemple va créer une partie du composant textuel sur deux mots (token) : le 4e et le 5e du premier paragraphe.

– `WIND.Timeline` : Construit un objet de type frise chronologique qui affiche une frise chronologique sur la page.

– `WIND.Tool` : Permet de créer un objet de la classe `Tool`. Cette classe sera être implémentée dans chaque composant d'interface.

– `WIND.UserEvent` : Permet de créer un objet de la classe `UserEvent`. C'est la classe mère de deux classes `SelectEvent` et `InputEvent`.

Les détails des classes et leurs méthodes sont décrits en ligne sur http://erozate.iutbayonne.univ-pau.fr/Nhan/windapi/doc/.

F.2 Liste des modules de WINDMash

L'utilisation de WINDMash commence par une connexion à l'environnement avec un compte d'utilisateur existant ou bien par la création d'un nouveau compte. Lorsqu'un utilisateur se connecte, WINDMash crée sur le serveur son répertoire dont le nom est son identifiant. Chaque utilisateur possède alors un répertoire dans `windmash/data/`. Ses applications sont enregistrées dans ce répertoire et s'identifient par une session en 10 caractères générés aléatoirement (par exemple, `M1g6XyqScK`) au moment de la création. L'utilisateur peut choisir le langage de l'interface parmi l'anglais (par défaut), le français et le vietnamien.

Le processus de conception d'une application se compose de trois phases : *Contenu*, *Interface* et *Interaction*. Sur chaque phase, l'utilisateur dispose d'un environnement visuel/graphique avec des modules pour créer son application interactive sans programmation.

Sur la phase Contenu, l'utilisateur crée une chaîne de traitement sur les contenus géographiques. Lorsque l'exécution de la chaîne de traitement se termine, les contenus créés sont mis en évidence dans le menu à gauche, sous la zone "Contenus générés". Côté serveur, quelques fichiers sont générés dont par exemple :

- `sessionName_dataFacet.js` : la description graphique de la phase Contenu ;
- `sessionName_dc0.txt` : le texte manipulé en entrée (si l'utilisateur travaille avec un texte) ;
- `sessionName_dc*.rdf` : les descriptions RDF générées décrivant les données manipulées par la suite de la conception. Ces contenus seront servis à consommer par les afficheurs dans la phase Interface.

Sur la phase Interface, dans le menu à gauche, il y a deux parties : "Afficheurs" et "Contenus générés" (venant de la phase Contenu). Pour afficher les afficheurs, il faut sélectionner les afficheurs souhaités et les déposer dans la zone de travail. Puis, il faut choisir le contenu à mettre dans l'afficheur qui va manipuler le contenu choisi (par glisser-déposer). Lors du dépôt d'un afficheur sur la couche (*layer*) Interface, une miniature est créée à gauche sous la rubrique "Afficheurs générés" (une instance de l'afficheur). En cliquant sur le bouton "Sauvegarder" ou en passant à la phase Interaction (l'enregistrement se fait automatiquement), les fichiers sont générés côté serveur :

- `sessionName_interfaceFacet.js` : la description graphique de la phase Interface.
- `sessionName_interface.rdf` : la description RDF des afficheurs créés avec l'interface de l'application et des URL des contenus éventuellement associés à chaque afficheur..

Sur la phase Interaction, l'utilisateur crée un diagramme d'interaction avec les briques du langage visuel basé sur le formalisme graphique du diagramme de séquence UML pour décrire une interaction. Les fichiers sont générés côté serveur :

– `sessionName_interactionFacet.js` : la description graphique de la phase Interaction.
– `sessionName_interaction.rdf` : la description RDF des interactions de l'application.

Enfin, les fichiers RDF des 3 facettes sont fusionnés pour créer l'application dirigée par le contenu, intégrant les interactions et bien affichée (interface) grâce à l'API WIND.

F.3 Architecture de l'environnement WINDMash

L'architecture WINDMash [69] est divisée en deux parties : le côté client et le côté serveur (Figure F.2).

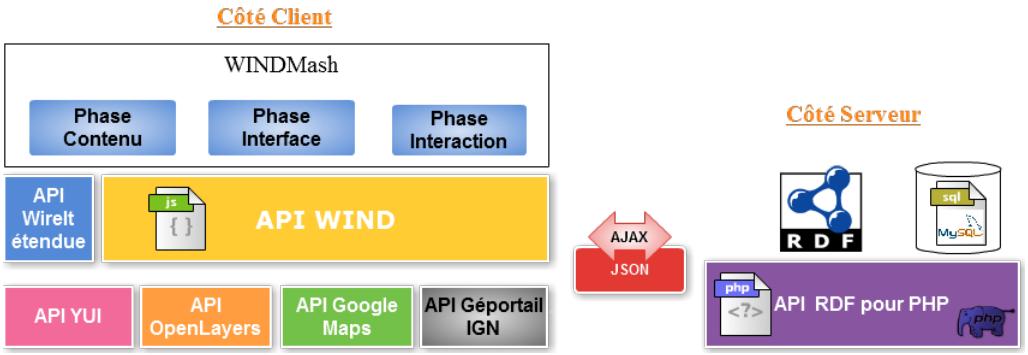


FIGURE F.2 – Technologies Web utilisées pour implémenter WINDMash

Côté client, nous avons pleinement tiré parti des avantages de JavaScript en utilisant notre API WIND v2.0 qui est basée sur les quatre API présentées dans la section suivante et l'API WireIt v0.5 qui est une bibliothèque JavaScript open-source pour créer des modules graphiques (URL : `http://neyric.github.com/wireit/`, open-source, usage sous licence MIT).

Côté serveur, le langage de programmation PHP est utile pour traiter les services Web et les transformations (par exemple, les fichiers `execution.php`, `generateData.php`, `generateInterface.php`, `generateInteraction.php`, `parser.php`...). Nous l'avons utilisé pour stocker des données RDF sur le serveur, car il permet de décrire, d'agréger, de partager et de réutiliser des descriptions de chaque information contenue dans chaque phase. Notre code côté serveur est basé sur l'API RAP v0.9.6 (RDF API for PHP - URL : `http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/`, open-source, usage sous licence LGPL [70]). Nous utilisons également deux bases de données MySQL v5.5 (open-source,

69. WINDMash v5 - URL : `http://erozate.iutbayonne.univ-pau.fr/Nhan/windmash/`
70. GNU Lesser General Public License

usage sous licence GPL [71]) pour enregistrer quelques caractéristiques de notre environnement et PostgreSQL v9.1 (open-source, usage sous licence PostgreSQL similaire aux licences MIT et BSD) pour stocker les données géographiques.

Nous avons choisi la technologie AJAX pour la communication asynchrone avec le serveur et le format JSON pour le codage des données (en raison de sa légèreté, simplicité et efficacité).

F.4 API utilisées dans WIND

L'API WIND [72] a été implémentée en basant sur quelques API suivantes :

- YUI 2.9 (Yahoo User Interface) est un ensemble d'utilitaires et de commandes pour construire des applications Web interactives riches en utilisant des techniques telles que DOM et AJAX. URL: `http://developer.yahoo.com/yui/`, open-source, usage sous licence BSD.
- OpenLayers 2.10 est une bibliothèque JavaScript pour afficher les données cartographiques sur les navigateurs Web. URL : `http://openlayers.org/`, open-source, usage sous licence FreeBSD.
- Google Maps contient un service de cartographie Web par lequel nous pouvons obtenir les directions et les lieux sur la carte. URL: `http://code.google.com/intl/en/apis/maps/index.html`, libre à utiliser mais avec une clé inscrite sur le site de Google Maps API.
- Géoportail IGN est un service de cartographie offrant des données géographiques en France. URL : `http://api.ign.fr`, libre à utiliser mais avec une clé inscrite sur le site IGN API et renouvelée par an.

71. GNU General Public License
72. WIND API v2.0 - URL : `http://erozate.iutbayonne.univ-pau.fr/Nhan/windapi/`

Résumé

Le point de départ de cette thèse a pour origine les difficultés constatées dans la communauté EIAH pour concevoir des applications éducatives exploitant des informations géographiques. A terme, il s'agit de proposer un nouveau cadre susceptible de rendre possible et opérationnel la conception d'applications éducatives portées par les spécialistes d'un domaine (et donc en particulier par les pédagogues) de façon autonome.

Le cycle de conception visé se veut court afin que les concepteurs soient en mesure d'évaluer très rapidement l'application conçue. Ils doivent disposer d'un environnement informatique les rendant complètement autonomes, c'est à dire non tributaires de tiers tels des spécialistes en programmation. Actuellement, nous parlons de spécialistes du domaine et pas seulement de pédagogues car le modèle ne cible pas uniquement des applications de type EIAH mais plus largement des applications fortement interactives. Pour ces applications, il s'agit de préciser et opérationnaliser sans programmation les formes d'interaction voulues entre l'utilisateur final (apprenant ou autre) et le système. Nous souhaitons définir des artefacts de modélisation adaptés aux personnes impliquées. Cette approche de modélisation se veut donc la plus "naturelle" et flexible possible tout en rendant opérationnelles ces représentations en limitant aussi les coûts de développement et en assurant la cohérence des systèmes construits. Le travail de thèse est donc un travail en Ingénierie Dirigée par les Modèles (IDM) des Interactions Homme-Machine (IHM).

Afin de pouvoir considérer les problèmes d'exécutabilité des modèles sur des situations d'interaction riches et ouvertes mais dont la complexité reste maîtrisable, nous avons choisi de construire cette contribution pour un domaine ciblé sur lequel vont porter les interactions à décrire. L'analyse des travaux précédents menés dans l'équipe T2i du laboratoire LIUPPA nous a donc conduit à nous centrer sur un domaine d'étude qui est celui des "*situations d'interaction finalisées valorisant des contenus à dimension géographique*", et d'envisager pour ce domaine particulier les problèmes d'exécutabilité des scénarios d'interaction que des non-informaticiens sont capables de spécifier de façon autonome.

La proposition scientifique est basée sur un modèle de conception piloté par les contenus à transmettre et par les interactions visées. Il est opérationnalisé dans une plateforme nommée WINDMash offrant aux concepteurs un environnement visuel de spécification et d'évaluation des interactions. Le modèle générique de description des applications Web géographiques comporte trois facettes permettant de représenter les contenus géographiques manipulés, de les afficher sur une interface graphique et de décrire leur comportement (en terme d'interactions) à l'aide d'un langage visuel adapté dont le for-

malisme graphique est inspiré du diagramme de séquence UML. Ce langage visuel se veut à la fois simple, afin de pouvoir être facilement appréhendé par des personnes non-informaticiennes, mais aussi puissant dans la richesse de description de l'interactivité. Grâce à une interface de programmation d'application (API WIND) dédiée que nous avons mise en œuvre, le concepteur d'application manipule notre environnement-auteur visuel (WINDMash) et, de ce fait, instancie le modèle générique de façon transparente. Cette transparence est toutefois palpable car le concepteur dispose d'une évaluation immédiate de sa spécification grâce à des mécanismes de transformation de modèles assurant l'exécutabilité.

Nos propositions scientifiques sont appliquées à la conception d'applications pédagogiques dans le domaine de la lecture active de documents à connotation géographique (cartes, textes, frises chronologiques). Pour ce domaine, un travail de fond a été mené pour identifier les objectifs pédagogiques ainsi que des scénarios d'interaction contribuant à l'acquisition de connaissances par des écoliers (cycle 3) et collégiens. En ce sens, ces travaux de thèse constituent donc un socle à partir duquel de nombreuses problématiques de recherche en EIAH peuvent trouver une concrétisation et des applications directes, faisant de WINDMash le point de départ et le réceptacle de travaux futurs de l'équipe T2i (analyse des traces, modules de diagnostic et de tutorat...).

Mots-clés : Interaction Homme-Machine, Information Géographique, Environnement visuel de type Mashup, Ingénierie Dirigée par les Modèles, Interface de Programmation d'Application (API) dédiée à la programmation d'interactions portant sur des contenus géographiques, Conception flexible pour prototypage rapide, Environnements Informatiques pour l'Apprentissage Humain

Abstract

The starting point of this thesis is to deal with the difficulties encountered in the TEL community for designing educational applications exploiting geographic information. Ultimately, we wish to propose a new framework allowing for the operational design of educational applications for experts in a specialized domain (and particularly for teachers).

The design cycle needs to be short so that designers will be able to quickly assess the intended application. They should have a computer environment allowing them to be completely autonomous, i.e. not dependent on third-party specialists in programming. Currently, we are addressing domain experts and not just teachers because the model does not only target TEL applications but also broadly interactive applications. For such applications, the task is to clarify and operationalize the different forms of interaction required between the end-user and the system without coding. We thus needed to define modeling artifacts suitable for the various people involved. This approach involves therefore modeling of a more 'natural' type, not just to be as flexible as possible, but while making such representations operational, also making sure to limit development costs and to ensure the consistency of any systems built. Our thesis thus covers work done on Model-Driven Engineering (MDE) of Human-Computer Interaction (HCI).

In order to consider the problems of executability for models with rich and open interaction situations but whose complexity still remains manageable, we chose to build our contribution to apply to one specific domain and targeting particular interactions needing to be described. The analysis of previous work carried out by the T2i team of LIUPPA led us to focus on their domain studying 'finalized interaction situations enhancing geographical dimension content', and to envisage for this specific domain, the executability problems of interaction scenarios which non computer scientists would be able to specify autonomously.

The scientific proposal offered here is based on a design process driven by content and interaction. It is operationalized on a platform called WINDMash offering designers a visual environment for specifying and evaluating interactions. The unified model for describing geographic Web applications has three parts for the purposes of representing geographic contents, displaying them on a graphical user interface (GUI) and describing their behavior using a visual language whose graphical formalism is based on the UML diagram sequence. This visual language is intended to be both simple enough for use by non computer scientists, as well as sufficiently powerful to be able to cover a wealth of interactivity description.

By means of an application programming interface (WIND API) which we coded, the application designer can manipulate our visual authoring environment (WINDMash) and, thus, instantiate the generic model seamlessly. Its transparency can also be expe-

rienced palpably because the designer can get an immediate assessment of whatever specification is required, by means of model transformation mechanisms ensuring executability.

In this particular case, our scientific proposal has been confined to the design of educational applications in the active reading domain of geographic documents (maps, texts, and timelines). For such a domain, substantial work has been conducted to identify the educational objectives and interaction scenarios assisting the acquisition of knowledge by pupils in primary school (third class) and junior secondary school. Nonetheless, this thesis will also serve not only as a basis from which many research issues in TEL should be able to find practical and direct applications using WINDMash as the starting point, but equally, as a receptacle for future work carried out by the T2i research team (including, for example, trace analysis, diagnostic modules or tutoring).

Keywords: Human-Computer Interaction, Geographic Information, Mashup Visual Environment, Model-Driven Engineering, Web Mapping API, Rapid Application Development, Technology Enhanced Learning