Nancy-Université
Université Henri Poincaré

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

➢ Contact SCD Nancy 1 : theses.sciences@scd.uhp-nancy.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Nancy-Université
Université Henri Poincaré

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR Sciences & Technologies

Coordination of Distributed Activities in Dynamic Situations. The Case of Inter-organizational Crisis Management

THÈSE

présentée et soutenue publiquement le 14/10/2011

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Jörn Franke

Composition du jury

Rapporteurs :	Mohand-Said Hacid, Professeur, LIRIS-UFR d'Informatique, Université Claude Bernard Lyon 1 Hervé Pingaud, Professeur, Directeur CUFR, Université Jean-François Champollion
Examinateurs :	Isabelle Chrisment, Professeur, Université Henri Poincaré, Nancy I, France Julie Dugdale, Co-Directrice de Magma, Université Pierre Mendes, Grenoble II, France Chihab Hanachi, Professeur, IRIT, Université de Toulouse 1, France Cédric Ulmer, Expert de la Recherche, SAP AG, France
Directeur de thèse :	François Charoy, Maître de conférence, HDR, à l'Université Henri Poincaré, LORIA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Mis en page avec la classe thloria.

Nancy-Université
Université Henri Poincaré

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR Sciences & Technologies

Coordination des activités réparties dans des situations dynamiques : le cas de la gestion de crise inter-organisationnel

THÈSE

présentée et soutenue publiquement le 14/10/2011

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Jörn Franke

Composition du jury

Rapporteurs :	Mohand-Said Hacid, Professeur, LIRIS-UFR d'Informatique, Université Claude Bernard Lyon 1 Hervé Pingaud, Professeur, Directeur CUFR, Université Jean-François Champollion
Examinateurs :	Isabelle Chrisment, Professeur, Université Henri Poincaré, Nancy I, France Julie Dugdale, Co-Directrice de Magma, Université Pierre Mendes, Grenoble II, France Chihab Hanachi, Professeur, IRIT, Université de Toulouse 1, France Cédric Ulmer, Expert de la Recherche, SAP AG, France
Directeur de thèse :	François Charoy, Maître de conférence, HDR, à l'Université Henri Poincaré, LORIA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Mis en page avec la classe thloria.

Remerciements

Research is never done in isolation and this holds also true for this thesis. I had the opportunity to work in two inspiring environments to conduct my research : the SCORE team of the Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA-INRIA-CNRS) in Nancy, France and SAP Research of SAP AG. I would like to thank my colleagues in these teams for the support, advice and friendship.

Special thanks go to my PhD supervisor, François Charoy, for his guidance, support, encouragement, patience and availability, although we had to work most of the time over large distances (between 900 km - 16000 km) together. Nevertheless, we managed well and I had the opportunity to learn much about academic research. His expertise and experience in the research area helped me to master this challenging topic. This was also invaluable for the success of my research within the industry. Furthermore, he encouraged me to search for academic cooperations to develop new innovations. This network is also of high value for my future.

I would also like to thank equally my industrial advisor and team manager, Cédric Ulmer, at SAP Research. Without him I would have never learned the secrets of successful industry research, such as successfully patenting, advertising research within the company, supervising students, transferring research to the company and interacting with customers. His guidance and management skills provided me freedom to pursue my research interests and to have success as a researcher in an industry environment. Furthermore, he provided me contacts to disaster managers and people within the company, which will be beneficial to push the ideas into products or standards.

Clearly, I would never have an understanding of the problems of disaster response management, if domain experts in Germany, France and the US, would have not spend time with me to explain their work or to discuss my developed concepts. I deeply appreciate this. I hope the results of the thesis will be beneficial for their future work.

Another thank goes to the SAP research team and specially the SAP research PhDs. I enjoyed much our interactions and discussions, although our research topics were very different. Nevertheless, I think our research benefits implicitly from these interactions. Paul El Khoury, Azzedine Benameur and Gabriel Serme provided special advice on various industry research related topics. Special thanks go to Géraldine Bous, Corentin Follenfant and Coralie Haese for reviewing parts of this thesis. Thanks also to Muhammad Rizwan Saeed, who contributed to parts of the implementation of the Google Wave extension.

At LORIA, I found at least equal support for my research. I benefited much from the discussions with my colleagues and the PhD students. Both made me always feel welcome when I visited Nancy, although I visited them rather infrequently. Special thank goes to Karim Dahman who challenged my ideas in discussions and helped significantly to organize the student experiments that we conducted in this thesis. Claudia-Lavinia Ignat, Gérald Oster and Pascal Urso provided me new perspectives on my research from the point of distributed collaborative text editing. This allowed me to reflect on my research.

Furthermore, I would like to thank the jury for their willingness to provide their expertise on the thesis subject. I am grateful to the rapporteurs, Mohand-Said Hacid, Professeur, Université Claude Bernard Lyon 1 and Herve Pingaud, Professeur, École des Mines d'Albi, for the time they spent on this manuscript and their valuable comments

to it. Equally, I appreciate the guidance of my internal referent of the thesis, Isabelle Chrisment, Professeur, Université Henri Poincaré, who also accepted to be part of the jury. Furthermore, I am grateful to Julie Dugdale, Directrice de Magma, Université Pierre Mendes, who joined the jury to provide her expertise as an important member of the research community on information systems for crisis response and management.

The writing of the thesis benefited also from the help of Stephen McIlvenna, who checked parts of the English writing of the thesis.

Finally - last but not least - I would thank my family and friends in Germany for supporting me in my career in a foreign country.

Table des matières

Chapitre 1
Introduction

1.1 Contributions 18
1.2 Thesis Outline 19

Chapitre 2
Context

2.1 Introduction 21
2.2 Coordination of Activities by People of Different Organizations in Dynamic Situations 22
2.2.1 Motivational Example : Coordination in the Disaster Response . . . 22
2.2.2 Background : Disaster Response Management 25
2.2.3 Summary 32
2.3 Research Problems and Contributions 34
2.3.1 Problem Statement 34
2.3.2 Research Questions and Method 36
2.3.3 Publications 37
2.4 Conclusion 39

Chapitre 3
State of the Art

3.1 Introduction 41
3.2 Coordination of Activities in Dynamic Situations 42
3.2.1 Process-Based Coordination 42
3.2.2 Artifact-Based Coordination 49
3.2.3 Rule-Based Coordination 52
3.2.4 Integrated Approaches 54

3.2.5 Summary 56
3.3 Coordination by People of Different Organizations 56
3.4 Conclusion 60

Chapitre 4
Framework for Coordination of Activities

4.1 Introduction 61
4.2 Modeling Coordination of Activities 62
4.2.1 Activity Type 63
4.2.2 Activity 65
4.2.3 Temporal Dependency 65
4.2.4 Summary 67
4.3 Verification 68
4.3.1 Translation of a Model into a Temporal Constraint Network 69
4.3.2 Checking Satisfiability of a Temporal Constraint Network Using State of the Art Algorithms 71
4.3.3 Performance Considerations 74
4.3.4 Summary 75
4.4 Detecting Deviations from the Model and How Activities Are Executed . . 76
4.4.1 Tracking Execution of Activities 76
4.4.2 Dependency Violation 77
4.4.3 Unsynchronized Dependencies 81
4.4.4 Summary 85
4.5 Example 85
4.6 Related Work 87
4.7 Conclusion 88

Chapitre 5
Inter-Organizational Coordination

5.1 Introduction 92
5.2 Sharing Activities between Organizations 92
5.2.1 Activity Workspace 93
5.2.2 Example for Sharing of Activities 94
5.2.3 Replicating Shared Activities in Different Activity Workspaces . . . 95
5.2.4 Summary 95

5.3 Verifying an Activity Workspace Containing Shared Activities 96
5.3.1 Problem Statement 96
5.3.2 Requirements for Verification 97
5.3.3 Discussion of Requirements 99
5.3.4 Protocol for Verification of an AW Containing Shared Activities . . 100
5.3.5 Summary 101
5.4 Detecting Deviations from the Model and How Shared Activities Are Executed 102
5.4.1 Tracking the Execution of Shared Activities 102
5.4.2 Conflicting State Changes of the Same Shared Activity 103
5.4.3 Conflicting Views on the Causal Order of State Changes of Dependent Shared Activities 110
5.4.4 Detecting Violation of Temporal Dependencies Involving Shared Activities 116
5.4.5 Detecting Unsynchronized Dependencies Involving Shared Activities 118
5.4.6 Incomplete Propagation of State Changes of a Shared Activity . . . 120
5.4.7 Summary 122
5.5 Related Work 123
5.6 Conclusion 124

Chapitre 6
Implementation

6.1 Introduction 127
6.2 Main components 128
6.3 Integration into Google Wave™ 128
6.3.1 Preliminaries 129
6.3.2 Extension - Architecture 129
6.3.3 Extension : Persistence of Activity Workspaces and Replicating Shared Activities 131
6.3.4 Gadget - Graphical Modeling Tool 132
6.3.5 Robot - Distributed Coordination 134
6.4 Conclusion 135

Chapitre 7
Evaluation

7.1 Introduction 137
7.2 Interviews 138
7.2.1 Selected Comments 139
7.2.2 Discussion 142
7.3 Design of an Experiment 142
7.3.1 Design Details 143
7.3.2 Data Sources 149
7.3.3 Validation of the Experiment Design 151
7.3.4 Discussion 155
7.4 Conclusion 156

Chapitre 8
Conclusion and Perspectives

8.1 Contributions 157
8.2 Perspectives for Future Research 159

Annexe A
Case Study Prototype

A.1 Modeling of Activities 163
A.2 Modeling of Dependencies 163
A.3 Sharing of Activities 165
A.4 State Change of Shared Activities and Dependency Violation 168
A.5 Concurrent State Changes of the Same Shared Activity 169
A.6 Verification 169

Annexe B
Allen's Path Consistency Method

Annexe C
Dependency State Machines

Annexe D
Acronyms

Annexe E
Résumé de la Thèse

Bibliographie **197**

Table des figures

2.1 Example Setting 1 . . . 24
2.2 Example Setting 2 . . . 25
2.3 Disaster Management Lifecycle . . . 29
2.4 Example for a Whiteboard used in a command center for disaster response [DKU+08] . . . 32
2.5 Special Structured Messages : 4-fach Vordruck [DKU+08] . . . 33

3.1 Example for a process modeled using BPMN . . . 44
3.2 Example model of a response process for responding to a train accident with hazardous material (from the perspective of the police) . . . 47
3.3 Example of a model of an artifact and the corresponding activities . . . 50
3.4 Example of a model of rules . . . 52

4.1 Example for an activity type : Simple Field Operation . . . 63
4.2 Example for an activity type with governance roles for approval : Complex Field Operation . . . 64
4.3 Types of temporal dependencies between states of activities (based on Allen's temporal interval relationships [All83]) . . . 66
4.4 Example for a model containing two activities with a temporal dependency 67
4.5 Example for a model with errors . . . 68
4.6 Example for a constraint network . . . 70
4.7 Example for translation of a model into a temporal constraint network . . . 72
4.8 Example for an unsatisfiable constraint network . . . 72
4.9 Example dependency state machines representing the dependencies contains, starts and overlaps . . . 79
4.10 Example for a model that may lead to an unsynchronized dependency . . . 82
4.11 Example for the evolution of a model during a disaster response in four steps 85
4.12 Activity type used in this example . . . 86

5.1 Examples for activity workspaces . . . 93
5.2 Examples for sharing activities . . . 94
5.3 Example for integrating a shared activity in an AW . . . 95
5.4 Example for verification of AWs containing shared activities . . . 98
5.5 Example for propagating a state change of a shared activity to another AW 103
5.6 Example activity type . . . 104

5.7 Example for detecting conflicts caused by state changes of the shared activity "Transport Sandbags" . . . 104
5.8 Activity type of example . . . 108
5.9 Example for valid and diverging state change sequences . . . 108
5.10 Example for valid and invalid state change sequences with two conflicting state changes . . . 109
5.11 Example for a different order of state changes in different AWs . . . 112
5.12 Example for a situation where a causal order of state changes cannot be established using the standard vector clocks approach . . . 115

6.1 Example for three wave servers with replicated waves . . . 130
6.2 Architecture of the extensions to Google Wave™ . . . 131
6.3 Example for the data model of our extension . . . 132
6.4 Screenshot of the "Gadget" for modeling activities and dependencies : Graphical Notation . . . 133
6.5 Screenshot of the "Gadget" for modeling activities and dependencies : Table Notation . . . 134

7.1 Example for a specification of a LEGO® object : building to protect the power generator . . . 145
7.2 Example for a specification of a LEGO® object : power generator . . . 145
7.3 Example for a specification of LEGO® components . . . 146
7.4 Locations of the five different teams . . . 148
7.5 Example for a coordination flow between the different teams in an experiment 149
7.6 Activity type used for all activities in the experiment . . . 150
7.7 Result example 1 : chat overview of activities . . . 153
7.8 Result example 2 : chat coordination problems . . . 154
7.9 Result example 3 : wave coordination problems . . . 154
7.10 Two constructed objects by different groups . . . 155

A.1 Screenshot : modeling activities . . . 164
A.2 Screenshot : modeling dependencies . . . 164
A.3 Screenshot : sharing activities . . . 165
A.4 Screenshot : shared activity can be integrated into model . . . 166
A.5 Screenshot : shared activity is integrated into model . . . 167
A.6 Screenshot : state change of shared activity . . . 168
A.7 Screenshot : state change of shared activity leads to dependency violation in another activity workspace . . . 169
A.8 Screenshot : concurrent state change of the same shared activity . . . 170
A.9 Screenshot : verification of a model in an activity workspace . . . 171

B.1 Example for an inconsistent constraint network . . . 177

C.1 Dependency State Machine Notation . . . 179
C.2 Dependency State Machine : Starts . . . 180
C.3 Dependency State Machine : Contains . . . 180

C.4 Dependency State Machine : Overlaps 180
C.5 Dependency State Machine : Meets 181
C.6 Dependency State Machine : Precedes 181
C.7 Dependency State Machine : Equals 181
C.8 Dependency State Machine : Finishes 182

E.1 Exemple simplifié 186
E.2 Exemple de modèle d'un processus de réponse pour répondre à un accident de train avec des matières dangereuses (du point de vue de la police) . . . 188
E.3 Exemple pour un type d'activité avec des rôles de gouvernance pour l'approbation : Opération complex sur le terrain 190
E.4 Exemple d'un modèle contenant deux activités avec une dépendance temporelle 191
E.5 Exemple : Transmettre une activité à une autre organisation 192
E.6 Exemple : dépendances exprimées entre des activités 193

Liste des tableaux

2.1 Lifecycle phases of disaster management 28
2.2 Elements of a disaster response plan (non-exhaustive list) based on plans provided by disaster managers of the SoKNOS project 31

4.1 Constraint notation and abbreviation 69
4.2 Matrix notation of constraint network 70
4.3 Matrix notation of unsatisfiable constraint network 73
4.4 Example for a trace 78

B.1 Constraint notation and abbreviation 175
B.2 Corrected transitivity table (based on [All83]) 176
B.3 Matrix notation of inconsistent constraint network 178

Thesis Manuscript : Coordination of Distributed Activities in Dynamic Situations

Chapitre 1

Introduction

Recently we have seen several large scale disasters affecting humans all over the world. Examples are not only natural disasters, such as Hurricane Katrina in 2005 [Qua05, Unk06] or the Haiti earthquake in 2010 [Nat10], but also man-made disasters, such as the September 11/2001 terrorist attacks on the world trade center [IfCIS03]. At the beginning of this year, a natural and human made disaster occurred in Japan, where earthquakes, tsunamis and a nuclear incident led to a huge multi-organizational response [Shi11].

During these disasters, several hundred organizations, such as police, fire brigade or humanitarian aid organizations, respond with the goal to save people and support them to live a normal life again [CK06, Ola10]. The responses of the different stakeholders need to be coordinated to deal with scarce resources, different expertise and capabilities to help the people as much as possible. Different tasks, such as search and rescue, sheltering and medical treatment, need to be carried out. These tasks can be performed by many organizations with different goals and operating procedures. For example, during the 9/11 terrorist attacks, three organizations were responsible for mass feeding at one disaster site [IfCIS03] : Red Cross, a fire brigade and Salvation Army. This led to duplication of efforts or issues with the quality of food delivered, because the fire fighters were not expert in transporting food provided by volunteers. Hurricane Katrina showed similar challenges. Search and rescue operations were conducted by different organizations [Unk06]. Each of them had their own priorities and goals. This led to cases where responders were forced to leave people on highways, where they had no sheltering or food provision and nobody else took responsibility to provide it to them. Multiple rescue teams were sent to the same locations, but some locations were not searched and victims not rescued.

The situation can be dynamic : it evolves in sometimes unexpected ways, goals shift and priorities of the organizations change. For instance, during Hurricane Katrina there was the priority to evacuate the people to a certain location and to provide food as well as shelter there [Unk06]. However, the location became overcrowded, so that people were moved to another location, but no food was available at the new shelter.

The aforementioned problems are related to coordination of the response involving public, private and non-profit organizations [Wac00, Dra03]. Inadequate support for coordination may lead to inaction, double efforts or contradictory actions by different response organizations. The problems increase with the dynamics of crisis situations and shifting goals of the involved organizations. Activities may have relations to activities of other

organizations, e.g. search and rescue is performed by one organization, but sheltering by another. However, each organization coordinates the response from its own perspective and relies on the information provided by other organizations.

Current tools used for coordination in a disaster response, such as telephone, fax or e-mail, have limitations with respect to coordination in dynamic situations. All information is stored in a large pile of unrelated messages or it exists in the heads of people. This makes it difficult to get an overview on the relations between what has been done, what is currently going on and what are the next steps. This is especially challenging on the inter-organizational level, where it is only possible to rely on the information provided by other organizations. To the best of our knowledge, no adequate information system support exists for coordination in dynamic situations taking into account the relations between activities on the inter-organizational level.

1.1 Contributions

The goal of this thesis is to support the coordination of activities in dynamic situations. The disaster response management domain motivates our research. We propose a process-based information system support to address these problems. Such an approach makes explicit the activities and their relations, i.e. what has been done, what is going on and what are the next steps. It needs to take into account the inter-organizational level as well as the fact that only limited information may exist on what is going on depending on what other organizations are willing to provide.

Process-based information system support has been applied to the domain of emergency response management, but it cannot take into account sufficiently the dynamics of the situation. It does not consider shifting goals and priorities adequately. Other approaches developed in the business context show similar limitations. Current process-based systems require also sharing information with every stakeholder on the inter-organizational level, which is not always desired. Other collaborative systems are very restricted with respect to coordination support.

We advocate in this thesis to address the coordination of activities in dynamic situations by provision of a framework. Activities and their relations need to be consistently modeled using this framework. Shifting goals leading to a reassessment of activities and their relations need to be highlighted to the users. They can then deal with them by communicating with the stakeholders of the activities.

Given autonomous organizations, such as police, military, fire brigade or Red Cross, we need to consider privacy, regulations or strategic intentions that prevent that all information on what is done by an organization is shared with everybody. For example, the police could not exchange information about crime investigations at the 9/11 terrorist attacks disaster site with everybody [IfCIS03]. Hence, we suggest sharing only selected activities and their current state between people of selected organizations. They can establish relations between internal and shared activities. We extend the framework for coordination of activities in dynamic situations to take into account shared activities. Consequently, shifting goals of different organizations can be highlighted to all people with whom the activity has been shared.

1.2 Thesis Outline

The thesis is structured as follows to address the objectives stated.

We introduce in chapter 2 disaster response management as *the context* for coordination of activities by people of different organizations in dynamic situations. We illustrate this with a motivational example and provide insights on how the disaster managers in the SoKNOS project [DPZM09] coordinate a disaster response with currently available tools. We explain, given this context, the research problems that we want to address, and our proposed solutions.

The *state of the art* in chapter 3 presents different approaches used for coordinating activities within and between organizations. We identify advantages and limitations with respect to their contribution to a solution of the research problems defined in chapter 2.

We address in chapter 4 the problem of coordination in dynamic situations by proposing a *framework for implementing coordination of activities in dynamic situations*. This framework deals with the gaps in the state of the art in the area of coordination of activities in dynamic situations. It provides functionality for defining activities ad-hoc and making their relations explicit in a model. We allow a richer description of relations than proposed in the state of the art. The model can be verified using established formalisms to ensure consistency. Governance roles describe accountability and responsibility for activities, so that every stakeholder is aware of it. Deviations from the model and how activities have been performed can be detected. This enables highlighting to the user the impact of shifting goals. Contrary to the state of the art on process-based information systems, the process is not enforced by a system. Thus, the state of the art cannot take into account shifting goals, because it assumes that they are under the control of the users or a system, which is not always the case in a disaster as we explain in chapter 2.

We *extend this framework to the inter-organizational level* in chapter 5. The framework defined in chapter 4 does not take into account the coordination by people of different organizations. The state of the art provides only limited solutions, because it requires sharing at least a basic model of activities and their relations with everybody or it only allows sharing selected unstructured information, which is difficult to use for coordination. We propose an alternative, where each organization maintains its own activity workspace, where it manages its internal activities using the framework proposed in chapter 4. Selected activities can be replicated in the workspaces of different organizations. This means the organizations can take into account privacy, regulatory, strategic or other reasons by sharing only selected activities with selected organizations. Relations can be established between internal and shared activities in a workspace. We detect and handle conflicts that can occur when managing shared activities replicated in different workspaces using our proposed framework. Conflicts can lead to a diverging view on the state of activities as well as their relations. Handling of these conflicts means that a partial shared common view of the activities and their relations needs to be ensured.

The contributions presented in chapters 4 and 5 are *implemented* in chapter 6 to demonstrate technical feasibility. We describe a library implementing the algorithms and data structures of chapters 4 and 5. This library can be used in various applications, because our concepts can be incorporated in different contexts (e.g. collaboration suites or social networking services). We present how parts of the library are used by an extension

of the collaboration service Google Wave. This shows how our concepts can be seen in the context of other software tools used in disaster response, such as text message exchange or maps. Furthermore, Google Wave provides a decentralized collaboration infrastructure suiting to our concepts. The extension is used in chapter 7 for evaluation in experiments.

We provide *initial evaluation* of the concepts and the implementation in chapter 7 to validate them with respect to the research questions stated in chapter 2. They are evaluated from two angles : comments by disaster managers and experiments with students. Four disaster managers comment on the concepts of chapters 4 and 5 presented to them. This gives us preliminary insights on how the concepts could suit to their working context. We also designed an experiment to be able to evaluate the concepts implemented in the prototype extension. Existing experiments in this area do not address exactly the evaluation of our research questions. First experiments conducted with students provide us insights on the value of the experiment design. Further iterations can provide us validated empirical insights on the value of the concepts implemented in the prototype with respect to the research questions in chapter 2.

We *reflect on the contributions and their evaluation* in chapter 8. We describe perspectives that will be addressed by our future work.

Chapitre 2

Context

Contents

2.1 Introduction 21
2.2 Coordination of Activities by People of Different Organizations in Dynamic Situations 22
2.2.1 Motivational Example : Coordination in the Disaster Response 22
2.2.2 Background : Disaster Response Management 25
2.2.3 Summary 32
2.3 Research Problems and Contributions 34
2.3.1 Problem Statement 34
2.3.2 Research Questions and Method 36
2.3.3 Publications 37
2.4 Conclusion 39

2.1 Introduction

Responding to a disaster in a coordinated manner is crucial for an effective and efficient response [Dra03]. Negative examples for coordination can be found in recent disasters, such as Hurricane Katrina in 2005 [Qua05, Unk06] or the Haiti Earthquake in 2010 [Nat10].

Several issues can be found in the disaster response to Hurricane Katrina [Unk06]. People were rescued, but were left at places where there was no shelter or food provision. Furthermore, even when people were rescued and care was provided, this has sometimes changed during the response. Some people had to move from an area where they were evacuated to, but they did not receive food or medical attention in their new shelter.

A disaster response is different from the day-to-day routine : an organization may perform tasks that it has never done before (e.g. the fire fighters need to ensure food supply) or it has to work together with other organizations, with which it has never worked before. The dynamic situation of a disaster also requires that the activities are coordinated according to shifting goals. This means the response evolves over time and it has to be considered what has been done, what is currently going on and what are the next steps.

During a disaster, different organizations work at different stages of the response together. For example, during the response to the 9/11 terrorist attacks, the Federal Bureau of Investigation (FBI) conducted crime investigations at the disaster site and was relying on resource provision by other organizations [GHSHM03]. They required food and containers from these other organizations. However, due to the classified nature of the crime investigations, it was important that they could coordinate selectively with other organizations. Without the ability to disclose information to selected organizations on what they are doing where, they would have needed to duplicate efforts. For instance, they would have had to organize their own food supply. This means that autonomous organizations need to share information in order to coordinate, but this is restricted due to privacy, regulatory, strategic or other reasons. However, autonomous organizations need also to take into account the previously mentioned problem of shifting goals of other organizations. For example, the FBI increased their personnel at the disaster site during the 9/11 terrorist attacks for more in-depth crime investigations [GHSHM03], but another organization had to provide subsequently more food for the additional investigators.

The problems of coordination stated before have also been identified as crucial in the disaster response management literature [Qua83]. Existing tools for coordination used in the disaster response, such as phone, e-mail or fax are limited with respect to these issues. This leads to problems related to coordination so that inaction, double efforts or conflicting actions occur. The reason is that it is not always clear what has been already done by whom, what is currently going on and what are the next steps. The main contribution of this thesis is to address this gap by providing an information system supporting people of different organizations to coordinate their activities in dynamic situations.

The goal of this chapter is to analyze these problems in more detail in the context of disaster response management in section 2.2. We motivate this context with a realistic scenario in disaster response management developed with disaster managers, such as fire fighters and police men, from the SoKNOS project [DPZM09]. We then provide background information about disaster management to define the scope of our research. Current tools for coordination, such as phone, fax or e-mail, used by disaster managers in the SoKNOS project are described afterwards. This facilitates a better understanding of the problems that can occur when coordinating in an inter-organizational disaster response. We derive research problems from this context in section 2.3, and conclude in section 2.4.

2.2 Coordination of Activities by People of Different Organizations in Dynamic Situations

2.2.1 Motivational Example : Coordination in the Disaster Response

First Part : Coordination of Activities in Dynamic Situations

The first part of the motivational example deals with coordination of activities in dynamic situations. The aim of coordination is an efficient and effective use of resources,

skills and capabilities to meet a goal. It should be avoided that inaction occurs, double efforts or conflicting actions are performed. Our definition of coordination is based on coordination theory proposed by Malone and Crowston [MC94], which has been also applied to the disaster response context [CSRU08]. Coordination of activities in our context means to relate what has been done, what is currently done and what are the next steps to reach a goal. A dynamic situation in the sense here is characterized by shifting goals. These shifting goals lead to reassessment of the goals and the activities that need to be done. These goals and the corresponding activities are defined by humans based on their judgment of the complex situation. A reassessment of activities has to take into account their relations, i.e. what has been done and what is currently going on.

We illustrate an example for these concepts in Figure 2.1 [1]. The fire fighter commander in the command center (illustrated on the right of the figure) tells the fire fighters in the field (illustrated on the left of the figure) to protect the residential area from a flood. This can be compared to a goal of the fire fighters in the field. He details this with further instructions to build a dam, transport sandbags and fill sandbags. These are the activities that need to be coordinated towards the goal.

During the fulfillment of these activities, some of the fire fighters get injured, because the rain-sodden ground makes it difficult to move. This is reported to the fire fighter commander. The fire fighter commander orders an additional unit to transport the injured fire fighters to a hospital. The goal shifts towards treating the fire fighters first before continuing with building the dam. Later, he decides that the protection of the residential area can continue as planned. The remaining fire fighters in the field are also able to continue with this task. Then, the fire fighter commander receives a request that some firefighting units are needed urgently at another disaster site. The goal shifts again to fulfill this request. This means sandbags cannot be transported and filled anymore by the fire fighters. A replacement needs to be found for them, but this has less priority than the urgent request.

We only described here one example for coordination of activities in dynamic situations. During a real disaster, there will be of course more goals, more activities and more frequent shifts of goals. Particularly, on the inter-organizational level, where there are many organizations, each with their own goals and activities that are related to each other.

Second Part : Coordination by People of Different Organizations

The second part of the example is based on the previous one, but coordination is now done by people of different organizations to deal with scarce resources, different expertise and capabilities. This means that an organization performs activities that are dependent on the activities of other organizations or it performs activities on which other organizations are dependent. Coordinating their activities implies that there is an exchange about what different organizations are doing. However, this exchange between autonomous organizations is restricted due to privacy, regulatory, strategic or other reasons. People in different organizations can decide what information they provide about their activities

1. Map is based on [ope]

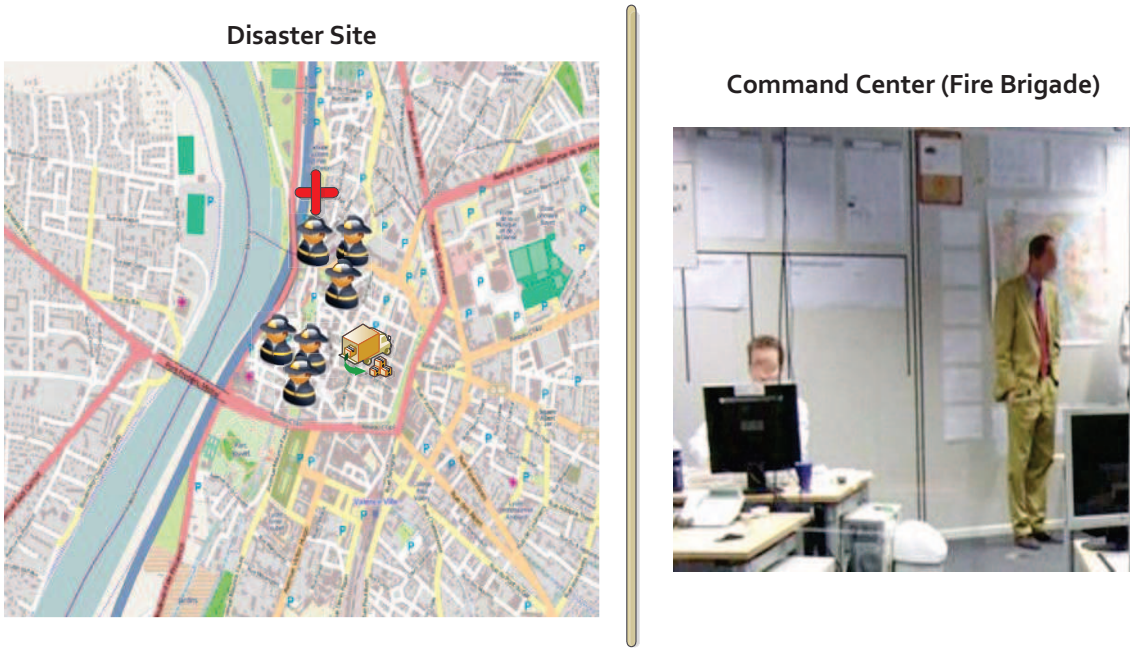


FIGURE 2.1 – Example Setting 1

to people from other organizations. No organization can have a full overview of what is coordinated by people of different organizations.

We describe an example of such a situation in Figure 2.2[2]. The situation in the field is illustrated on a map. The military is responsible for filling sandbags, transporting sandbags and building a dam. The purpose is to protect a chemistry plant from the flood. An expert in the coordination center of the department has suggested these measures to the mayor. The mayor has approved these measures and proposed them to the military commander in the command center. The military commander orders these measures to his units in the field. Additionally, the military also provides sandbags for the fire fighters to build a dam to protect the residential area from the flood.

The police forces have to evacuate the residential area in case it is threatened by the flood. This may involve activities such as warning people, transporting people, organizing shelter and determining the number of people to be evacuated. The police create a special unit for evacuation, because it is not a typical task of the police and should not interfere with crime investigations related to plundering in the area. These crime investigations and related information should not be disclosed to other organizations. A preventative measure to evacuate the area has been declined by the mayor for now, because the current risk that the area gets flooded is low given the protection measures of the fire fighters. The costs of evacuation are not justified given the risk. However, this depends also on the development of the flood in the future.

We illustrate in the figure the command center of the fire fighters is illustrated as well as a coordination center of the department. The coordination center has been established, because the incident developed to a larger incident covering several departments. Thus,

2. Map is based on [ope]

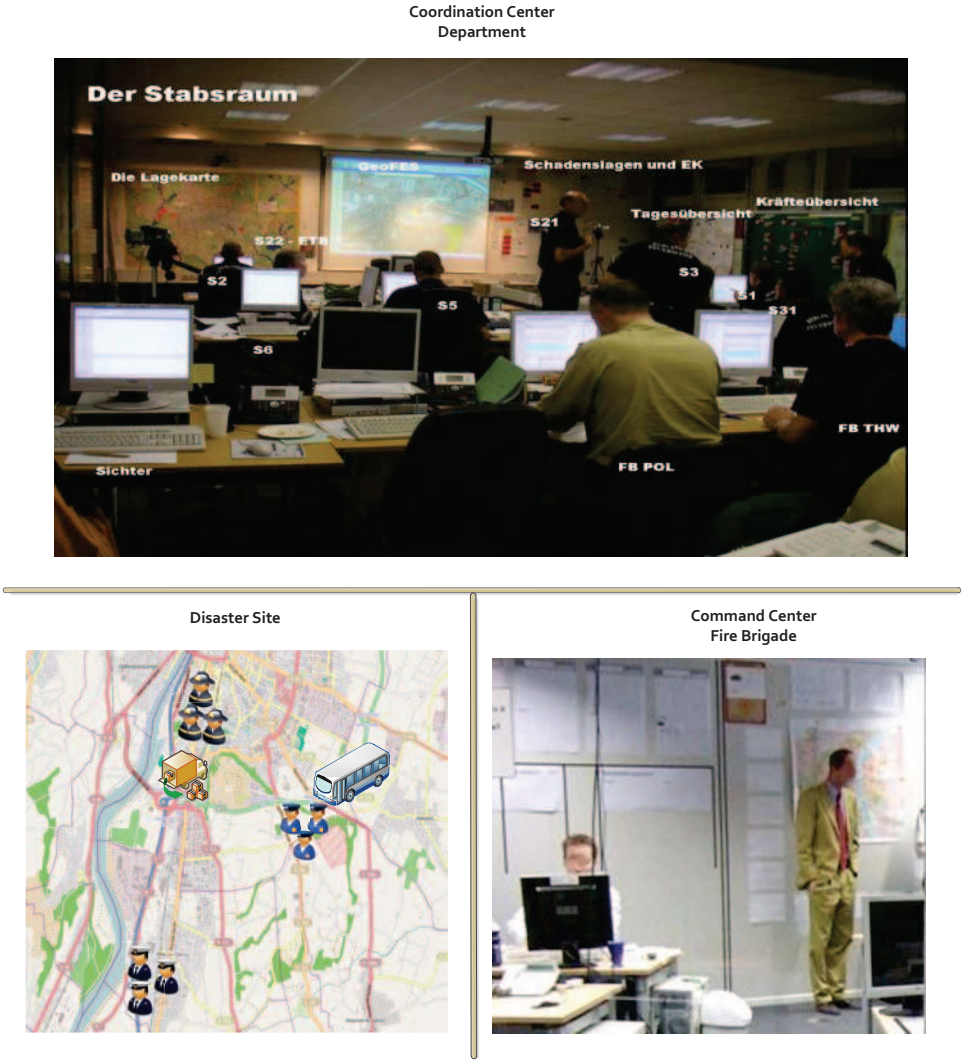


FIGURE 2.2 – Example Setting 2

it supports several organizations to coordinate their response. Some of the stakeholders described in this example are part of this coordination center, but also other stakeholders not explicitly described in the figure. This is another example how the situation can develop over several disaster sites with new organizations involved.

We described here the need of different organizations to coordinate. They do this by exchanging information about what they are doing. This information exchange is restricted and no organization has a full overview on what needs to be coordinated. Nevertheless, they need to take into account the dynamics of the situation as stated in the first part of the example.

2.2.2 Background : Disaster Response Management

We now give background information about disaster response management. This facilitates a better understanding of the functionality of technologies in the state of the art

(cf. chapter 3) and our concepts (cf. chapters 4 and 5) in the context of this domain. We explain how a disaster is different from day-to-day routines, but also what the limits are with respect to technology support. Afterwards, we address where our concepts fit in the lifecycle of managing a disaster. Both will define in which situations we aim to improve support for coordination.

What is a disaster ?

In order to better understand the context, we start by defining a disaster and differentiating it from emergencies or catastrophes. The term disaster is often used synonymously with other terms, such as emergency or catastrophe [DBBSPP09]. A distinction needs to be made between them to better understand the context of dynamic situations. Hence, a definition based on the one of Quarantelli is used [Qua05] (cf. also [DA79, Dra03, Wei00, Kle99, Vid10]).

An **emergency** can be seen as part of the day-to-day work of public safety organizations. A public safety organization in Germany can be defined as such by law. This is similar to other countries. Examples of public safety organizations in the SoKNOS project were police or fire brigade [DPZM09]. There are usually not a lot of organizations involved in an emergency response. Each organization involved has clear goals and tasks with only minor deviation from the routine. Examples for emergencies can be small traffic accidents or fighting a manageable fire in a house.

A **disaster** is significantly different from an emergency. During a disaster, the day-to-day social processes of the community are affected. For example, going to work, going to school or simply going shopping is impossible. Many organizations are involved in a disaster response. These organizations face new and unforeseen challenges. Tasks of day-to-day routines may become less important than activities to respond to a disaster. Goals of the organizations may shift depending on the development of the response. Given the examples described before, there can be a goal to protect a chemistry plant from a flood, but this may fail and now a residential area needs to be protected or evacuated. Plans are important, but may require arbitrary adaption or cannot be followed. Some activities are dependent on the fulfillment of other activities, e.g. a dam cannot be built without filling and transporting sandbags. Another example is that an area should only be evacuated if it cannot be protected anymore. This involves also challenges to get an accurate situation overview on the response activities, because priorities may change due to shifting goals. More precisely, it is difficult to get an overview on the relations between activities given the dynamics of the situation. In addition to public safety organizations, other organizations are on the scene, such as commercial, non-commercial and humanitarian aid organizations. Furthermore, new organizations may emerge. For instance, victims who help each other or more formal coordination bodies by different organizations [SQ85, QD77]. During the disaster response, different organizations need to coordinate their efforts, but it cannot be anticipated with whom it needs to be coordinated. Furthermore, the different organizations are limited on what they can exchange with other organizations. For example, the military cannot disclose everything to the fire brigade or the Red Cross cannot disclose everything to the police. We do not explicitly distinguish a disaster from a **crisis** and use them synonymously. A disaster is not limited to the aftermath of natural hazards,

such as an earthquake or a flood, but may also be applicable to human-made, such as a terrorist attack. Disasters do not need to have an agent, because they are seen as social constructions [Qua91]. This means for anything to be categorized a disaster it needs to affect the day to day basic social processes of people in a community significantly. The rationale behind this is that, for example, an earthquake in an inhabited area would not be considered as a disaster, since it does not affect anybody. Typical examples for disasters can be found in Hurricane Katrina (in a later stage) (cf. description in [Qua05, Unk06]), Haiti earthquake in 2010 (in a later stage) (cf. [Nat10]), the floods in Germany in 2002 [Sac03] or the September 9/11 attacks (cf. description in [IfCIS03]).

A **catastrophe** is characterized by its heavy impact on the community and particularly the infrastructure. For instance, response forces may be killed or seriously affected, communication infrastructure and community infrastructure is completely broken down. An example for a catastrophe is the fall of nuclear bombs on Hiroshima and Nagasaki in 1945 [CO46]. In this case the infrastructure broke down, responders from public safety organizations where either killed or seriously injured and basic medical treatment was impossible. Another example is the Haiti earthquake in the first days after the event, where people of the government as well as of humanitarian aid organizations (e.g. United Nations Stabilizations Mission in Haiti) were seriously affected or killed (cf. [Nat10]). It is very difficult to support the response in a catastrophe with any information or communication technology. However, to some extent there can still be support from a large distance, i.e. from countries where the communication infrastructure is not affected and where response capabilities still exist.

The borders between the terms emergency, disaster and catastrophe can be blurry. A catastrophe can be reduced in severity to a disaster when a response can be coordinated via a communication network. An emergency can turn into a disaster, e.g. when a routine fire becomes unmanageable (cf. the Mann Gulch Disaster [Wei93]).

Nevertheless, all of them involve to a different extent the coordination of activities by people of different organizations. The dynamics are defined by the type of the incident (emergency, disaster or catastrophe). We are mainly interested in dynamic situations similar to a disaster, because based on our interaction with disaster managers in the SoKNOS project these situations are the most problematic ones in comparison to emergencies. One important problem in a catastrophe is to establish a communication network, which we see as a prerequisite for concepts defined in this thesis and coordination in a disaster in general. However, activities for managing disasters are not limited to the phase after a disaster, but need also to be performed before it occurs. Thus, it is important to understand when which activities should be coordinated with respect to managing a disaster.

Disaster Management

We explain here when we aim to support coordination of activities with respect to managing a disaster. Disaster Management can be described as a lifecycle [WdB85, PM02, Tuf06, Cho08, Sch08] as illustrated in Figure 2.3 with the following phases : mitigation, preparedness, response and recovery. Table 2.1 describes the phases of disaster management and typical activities within the phases (cf. also [WdB85, PM02, Cho08, Dra03, Qua91]).

TABLE 2.1 – Lifecycle phases of disaster management

Phase	**Description**	**Example : Activities**	**Time Frame**
Preparedness	Establishment of the foundation for disaster response and recovery	Discovering & analyzing vulnerabilities and risks Develop plans for response and recovery Prepare resources and material Training and education of responders and citizens Maintain relationships with people of other organizations	On-Going
Mitigation	Reduce the vulnerabilities and risks of a community	Take into account risk and vulnerabilities when designing infrastructure (e.g. build higher dams to protect from a flood) Transfer risks (e.g. insurance) Distribute population to lessen the impact on it	On-Going
Response	Respond to an event that could not be adequately mitigated to reduce the exposure of the community to the impact	Warning Evacuation Shelter Feeding Search and Rescue Coordination within and between organizations	few hours to days
Recovery	Return to routine community processes (i.e. “normal life”)	Damage assessment Restoration of buildings Restoration of community services Donation and aid management Coordination within and between organizations Debriefing of response and recovery activities as an input for the preparedness phase	few weeks to years

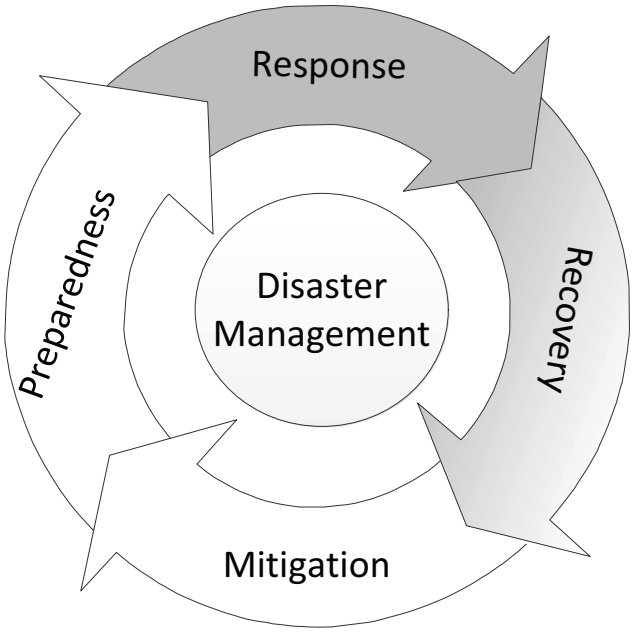


FIGURE 2.3 – Disaster Management Lifecycle

The focus in this thesis is the response phase and to some extent the recovery phase, because they characterize dynamic situations, where coordination by people of different organizations is a challenge. So far, we discussed only the dynamics of the situation faced in a disaster response, but not what problems occur in practice with respect to current tools used for coordination.

Tool Support for Coordination of a Disaster Response in Practice

We research in this paragraph the tools used in practice for inter-organizational coordination of activities in the disaster response by the disaster managers, such as police men and fire fighters, within the SoKNOS project [DPZM09]. This will help us to understand better their functionalities and limitations. The main aim of the SoKNOS project was to investigate technical integration of systems of different organizations in a disaster. Integration should facilitate exchange of information between different organizations to provide a better overview of the situation. The use case was a large-scale flood based on previous flood disasters in Germany. Many organizations have been involved in these disasters. The main integration technology was based on Service-Oriented Architectures (SOA) [Erl05] and Service Component Architectures (SCA) [MR09]. Within the SoKNOS project, several interviews have been conducted with partners in the project (e.g. Technische Hilfswerk (THW), University of the Police, Dortmund Police, Fire Brigade of Berlin and Cologne [DKU+08]). A half-day workshop has been carried out in January 2009 with some of them (University of the Police, Fire Brigade in Berlin and Cologne) on the topic

of inter-organizational collaboration in a disaster response. Inter-organizational collaboration does not only include public safety organizations, but also private organizations responsible for public safety (e.g. Deutsche Bahn or churches). This workshop helped us to better understand the limitations of inter-organizational collaboration involving human actors.

An important artifact for coordination of the disaster response is a plan [GH05, Qua86], which can also be understood as a "resource for action" [Suc87], i.e. reasoning and reflecting on it based on the situation faced [Bar97]. It is used by public safety organizations, humanitarian aid organizations or private companies. One major challenge for any response plan is that a disaster is by definition something new and difficult to predict. This is contrary to the response procedures for an emergency, which can be planned in more detail. Going back to the motivational example, it is difficult to plan that the fire fighters are injured or that the military has to provide sandbags for the fire brigade. Several elements of a disaster response plan based on the plans made available by disaster managers in the SoKNOS project are described in Table 2.2. This list is not exhaustive, because the available plans are not representative of all possible plans, but the list provides a sufficient overview to understand the artifact. Disaster response plans are not detailed (cf. also [Suc87, Qua86]) and contain generic activities. There are few relations between activities documented in plans, because they are subject to the concrete situation faced when using the plan [Dek03]. It is very costly to develop detailed plans for disasters that may never happen. Furthermore, there is still the risk that plans become too specific, which may lead to the case that plans become rendered useless if the situation in a disaster is slightly different from the plan. This means they cannot specify all situations that may apply [Suc87]. It is impossible to anticipate every situation [Qua86] and adaptation as well as development of new plans can be necessary [Wei00, Suc87]. It would also be very costly to maintain detailed plans and adjust them to the object of risk of a disaster. For example, demographics of a community change and this affects how the community can be evacuated. Another example is that the response procedures of other organizations can change over time. Some disaster response plans in the SoKNOS project were codified as law applying to several response organizations (e.g. several different firefighting organizations in Germany, cf. FwDV 100 [fwd]) or that have been adapted by other organizations (cf. DRK-DV 100 of the German Red Cross [Rot00]). In this case, they deal with very generic aspects of organizational structure and response in disasters.

When activities are conducted then there needs to be a way to track their progress. For instance, the fire fighter commander in the motivational example needs to keep track of the activities for protecting the residential area from the flood, for transporting injured fire fighters and the activities of the military. Tracking is also used to monitor deviations between the plan and what has been done. Furthermore, other activities - not anticipated in the plan or performed by others - need to be described in order to be coordinated. Whiteboards [WPT06, GC10, BCSR07, MPBW07, Dra03] can be used for this purpose, when people need to share a common overview of the activities. New technologies allow sharing of Whiteboards over wide geographical distances using the Internet. Since it is a shared artifact, access to it needs to be managed to prevent it becoming useless. For instance, in the case of several people wanting to adapt the content of the Whiteboard to the situation [MPBW07]. Whiteboards only have an information sharing function, but do

TABLE 2.2 – Elements of a disaster response plan (non-exhaustive list) based on plans provided by disaster managers of the SoKNOS project

Element	**Description**
Activity	Generic activities (e.g. sheltering, evacuation)
Organizational Structure in Case of Disaster	A special organizational structure or sub-organization (e.g. staff or command center). This is used to process information relevant for coordination in an agile manner
Interfaces with other Organizations	Anticipated collaboration between organizations or experts (e.g. police may have to collaborate with fire fighters during an evacuation)
Tools for Coordination/-Communication	Tools for coordination/communication, such as described in this section
Material and Resource Types	Type of Materials (e.g. breathing mask) or Resources (e.g. humans)
Disaster-Specific Information	Disaster-specific information (e.g. scientific knowledge) about specific situations (e.g. what to do in case of contamination with chemical material)

not provide any means to reason on the current situation. For example, it is not possible to display deviations from what was planned. The traditional non-electronic Whiteboard faces the problem of space limitations and its content cannot be easily communicated to others in different locations.

A Mission Diary [BLJ10, Bor92, MPBW07] or electronic log is used to record what has happened and when in terms of major events, activities or information exchanges with others. For example, the military can record which activities it has performed for the fire brigade and this can be used for informing new staff after a shift changeover. The main aim of the mission diary is to understand the reasoning behind decisions, such as about specific activities. It can also be used for funding purposes of the response by an organization, i.e. as a document to claim funding from the government for response actions. Other purposes are debriefings and providing relevant information to new stakeholders joining the response.

People need to exchange information about the situation with others to coordinate their activities. These other people can be found in their own social network. For instance, friends, colleagues or people of other organizations they have worked with in previous disasters or exercises. Going back to the motivational example, the fire fighter commander knows the police commander from previous disasters and they can exchange information about the status of building a dam or evacuating the residential area. If stakeholders are situated in different locations, then different tools for communication are used [GC10, MPBW07, Dra03]. For example, phone, fax, e-mail or radio can be used. They differ in two important aspects : (1) Instant communication/feedback (e.g. phone or radio) vs. asynchronous communication (e.g. e-mail or fax) and (2) dominating communication paradigm, such as one-to-one, one-to-many or many-to-many. It depends on the organization and situation faced, which of them is more adequate. Usually a lot of messages are

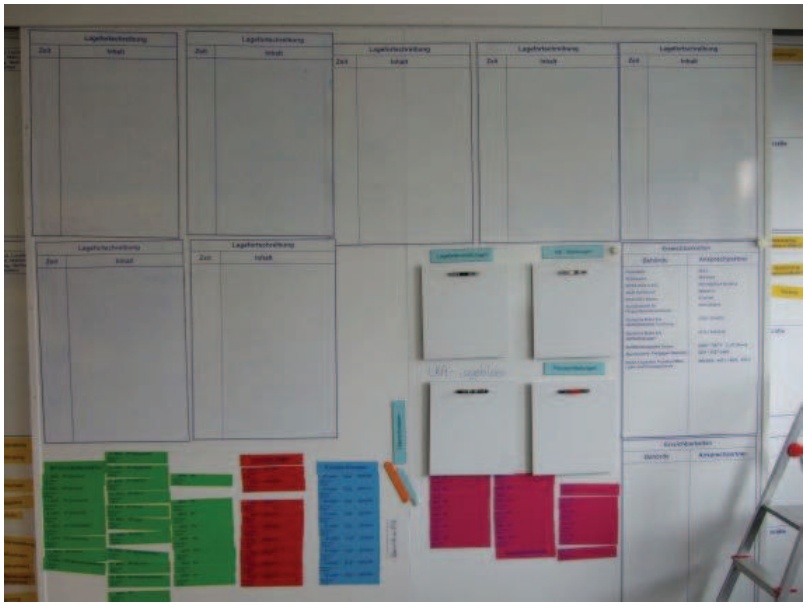


FIGURE 2.4 – Example for a Whiteboard used in a command center for disaster response [DKU+08]

exchanged and their relations are not clear. Recently, other tools have been used for communication in a disaster response, such as Internet social network services (e.g. Twitter or Facebook) [VHSP10]. However, they are currently predominantly used by organizations not specialized in public safety or by individuals. They are similar to the tools mentioned, but provide another communication channel with a potentially larger audience.

Communication can be based on pre-defined templates, called structured situation messages [BLJ10, Bor92]. An example from the SoKNOS project for a structured situation message is the "4-fach Vordruck" (four layer template, cf. Figure 2.5). It is also used for communicating commands and feedback via its content field. It defines time and date of a message. An important aspect is that the receivers of the message can be defined. Prioritization of messages can be articulated. Finally, additional comments can be described (e.g. which communication tool to use). At the moment, this template is filled out mostly by hand. Although it is possible to define the receiver of a message or to forward it to other interested stakeholders, this is still a manual error-prone process. It is not possible to relate the messages to each other.

2.2.3 Summary

We illustrated via a motivational example the facets of coordination of activities in dynamic situations and coordination by people of different organizations. The dynamics of the situation are characterized by shifting goals leading to reassessment of activities and goals. A reassessment has to take into account the relations between activities. Coordination on the inter-organizational level needs to take this into account as well, but it is

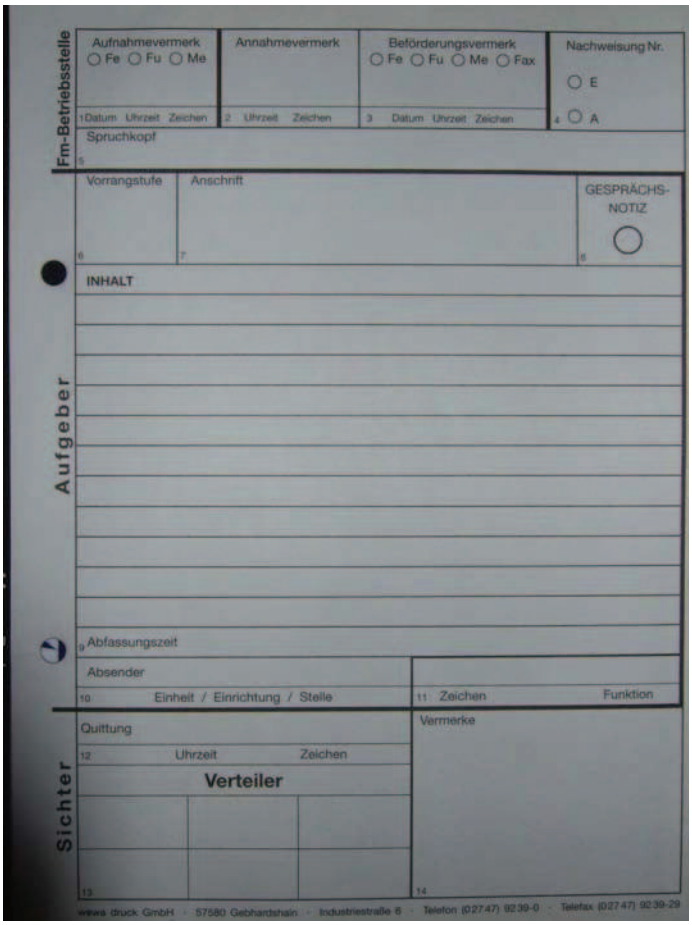


FIGURE 2.5 – Special Structured Messages : 4-fach Vordruck [DKU+08]

limited by the fact that an organization cannot share all information with every other organization. We explained these problems using an example about an inter-organizational disaster response.

We gave background information from the disaster management literature to distinguish situations with very low dynamics (emergencies) from high dynamics (disaster) to overwhelming dynamics (catastrophes) with respect to coordination. We then positioned our research in the disaster management lifecycle describing the activities that are performed before, during and after a disaster. The main emphasis is on the disaster response phase and to some extent the recovery phase.

Afterwards, we highlighted the problems occurring in practice in disaster response with respect to tool support for coordination of activities. This was based on a study with disaster managers in the SoKNOS project. For example, what has been done can be partially found in the mission diary. What is currently being done can be partially found on Whiteboards. What will be done can be partially found in plans. Much of this information can also be found in a large collection of unrelated messages exchanged ad-hoc between different people of different organizations. Unrelated information makes it difficult to coordinate, because every time something happens this information needs to be related again to deal with the challenges faced. In the next section, we will investigate research problems related to information system support in this context .

2.3 Research Problems and Contributions

2.3.1 Problem Statement

First problem : Coordination of Activities in Dynamic Situations

We have seen in the previous section that the dynamics of the situation affects coordination of activities. For example, shifting goals lead to reassessment of goals and activities. However, some activities have already been performed or are in progress. This has an effect on what can be done and may cause deviations from what was expected to be done. Shifting goals may also require managing the situation pro-actively, i.e. to define what are the next steps. All this is not encouraged by current tools, which support ad-hoc coordination to some extent, but all the relations between activities are hidden in a large collection of messages. Information is stored in unrelated artifacts, which makes it challenging to establish and understand the relations between activities. This leads to a less complete overview on the situation and thus potentially to less appropriate coordination of activities.

If we go back to the first part of the example, we can identify several relations between activities. Given the shifting goal of the fire fighters in the field that need to rescue their colleagues, there will be deviations from what is expected from them (building a dam) and what they are doing (helping their colleagues). Another example for a deviation is that the fire fighters start preparing the building of dam (moving units to the area and securing it), because the flood is getting worse. However, they do not wait for the sandbags being filled and transported, because they need to secure the area as part of building a dam. Furthermore, the fire fighter commander needs to ensure that injured fire fighters

are transported to the hospital and treated there. He also needs to keep in mind that there is still a need to build a dam and thus that filling as well as transporting of sandbags may need to be performed later. This can be done by monitoring what the fire fighter units that have been sent on another mission are currently doing. After they have finished they may come back. However, they may also be replaced by units from other organizations as illustrated in the second part of the motivational example. It may also be decided to cancel building a dam and sandbags should not be filled and transported anymore to the disaster site. Otherwise there will be unnecessary efforts.

Even in this rather simple example, we find a lot of activities and relations between them. These activities and relations are not given from the beginning of the response and emerge as the situation develops. For example, the police may need to perform additional evacuation measures when the residential area is flooded, such as ordering rescue boats. Thus, it is important not only to support monitoring of the activities, but also their relations.

Another important aspect is to define the governance for activities, such as accountability and responsibilities. The role of every stakeholder should be clear. For example, the fire fighters in the field cannot decide themselves that they leave the area to fulfill another mission. A further example on the inter-organizational level is that the military commander may decide that he gives the command to start filling and transporting sandbags for the fire brigade, but the fire fighter commander should be able to cancel these activities when they are not needed anymore. If governance is not defined properly then inaction or conflicting action may occur.

It should also be noted that the fire fighter commander has to gather and monitor other data, e.g. rising of flood levels or increase of the affected population in the area. This means he manages the activities in conjunction with other tasks. Thus, information system support that notifies the user about deviations caused by shifting goals is highly desired.

Second problem : Coordination by People of Different Organizations

On the inter-organizational level we need to consider coordination by people of autonomous organizations. This means that not all information can be shared with everybody. This leads to different partial views on the situation. Current tools used for coordination also have limitations in this case : all information, such as who are the receivers of information related to an activity, is hidden in different unrelated messages, which may even contain conflicting information. This can also potentially limit coordination effectiveness, because it is difficult for the users to have a partial shared view on the activities and their relations.

When we go back to the second part of the motivational example, we can identify several issues arising in this context. For example, what happens if the fire fighter commander decides not to protect the residential area from the flood anymore ? He has to inform the military that filling and transporting of sandbags is not needed anymore, otherwise they arrive at the disaster site without any use for them and they could be more useful at other sites. The fire fighter commander may also need to inform the police about this, so that the area is evacuated by them. He may also want to inform other organizations,

such as Red Cross, because they may treat people in the residential area. Later, the fire fighter commander may evaluate the situation differently and he decides to protect the residential area from the flood. This has an effect on the evacuation and other processes. Although this is a simplified example, this means that a lot of messages are sent around and some of them even may conflict. For instance, protection of residential area is given up, but then later it is picked up again.

This can lead to confusion of the different organizations involved. For example, one organization may only read the message that protection of residential area is given up and take appropriate actions without considering the follow-up message that the residential area is again to be protected from the flood. Some organizations may also not be informed, e.g. the fire fighter commander forgets to notify the Red Cross that the protection of the residential area has been given up. The problem here is to synchronize the information about activities. Nevertheless, not everything what is currently going on is shared with everybody. For example, the police won't disclose information about crime investigation related to plundering in the area.

We argue that information system support can help to relate activities of different organizations taking into account that not everything about the activities is shared with everybody. It can thus support creating a partial shared common view on the activities and their relations by the different organizations.

2.3.2 Research Questions and Method

We want to explore in this thesis the following research questions :

1. How is it possible to support coordination of activities in dynamic situations by an information system ?
2. How is it possible to support coordination by people of different organizations by an information system ?
3. How can the concepts addressing the first two research questions be integrated ?

We discover in the state of the art in chapter 3 that current information systems provide only limited answers to the research questions. Particularly, an integrated information system support addressing the third research question offers only very basic functionality.

Given the analysis of the limitations identified in the state of the art, we propose our first contribution for coordinating activities in dynamic situations : a framework for temporal coordination of activities. Activities and temporal dependencies between them can be modeled in a flexible manner. The consistency of the model can be verified using well-established formal methods based on Allen's interval logic [All83]. It is not required to fully specify all activities and dependencies, because this is very difficult to achieve in a dynamic situation subject to continuous change. Deviations from the model and how activities have been executed can be detected. They can be visualized to the user to illustrate the impact of shifting goals not taking into account all relations between activities. Governance roles are introduced to allow flexible definition of accountability and responsibility for activities.

We extend this framework to the inter-organizational level in our second contribution. This means we integrate the concept for coordination of activities in dynamic situations

with coordination by people of different organizations : selected activities can be shared between people of selected other organizations. Sharing of activities is voluntary : it is not required to share all activities with every organization, because of privacy, regulatory, strategic or other reasons. Shared activities are replicated in the workspaces of the selected organizations. Temporal dependencies can be created between shared and the other activities. The workspace supports the functionality of the framework previously developed. A state change of a shared activity is propagated to all workspaces. The state can be changed in any workspace according to governance roles. Conflicts causing a diverging view on activities and dependencies in different workspaces are detected and handled to ensure a converging view. This enables a partial shared view on the activities and dependencies by different organizations.

We validate the technical feasibility of the concepts by a proof of concept implementation of the algorithms and data structures as a Java library and a prototype implemented as an extension to the distributed open collaboration service Google Wave. This also shows how our approach can work in context with other tools used in crisis management, such as maps, text or videos. Furthermore, it addresses partially the research problems related to the inter-organizational dimension of coordination.

Technical feasibility of the concepts tells us that it is possible to design information system support addressing the research questions. However, empirical validation can give us insights into the advantages and limitations. Thus, we evaluate the concepts from a disaster response perspective. We conducted interviews with four experienced disaster managers. These interviews give first insights into the concepts in their work context.

Finally, we aim at validation of the concepts from a domain-independent perspective. We design an experiment for evaluating tool support addressing the research questions explained beforehand. An experiment can be repeated in a controlled manner at comparably lower cost with a specific focus on the goals of the prototype. We performed experiments with the prototype and other tools, where different distributed student teams needed to coordinate the construction of an object. These experiments provide insights into advantages and limitations of the experiment method itself. Thus, further experiments can validate the concepts implemented in the prototype. Although experiments cannot give insights with respect to disaster response management, they are complementary to evaluation of tool support for coordination in a disaster exercise. These exercises are more cost-intensive, do often not have the focus on tools for coordination between different organizations and are difficult to repeat. Experiments can help to deal with this issue by comparing the results of disaster exercises and experiments. They may also inform the validation of tools used for coordination in a dynamic enterprise context.

2.3.3 Publications

Related results have been published at peer-reviewed conferences and workshops in the technical field, but also in disaster management [FCU10c, FFU10, FCU10b, FC10, FCEK10, FCU10a, FWC+11, FCU11] :

– Franke, Jörn ; Charoy, François ; Ulmer, Cédric : *Handling Conflicts in Autonomous Coordination of Distributed Collaborative Activities*, 20th IEEE International Conference on Collaboration Technologies and Infrastructures (WETICE'2011),

Paris, France, 27-29 June, 2011. This paper deals with technical aspects related to the inter-organizational dimension of the research problems presented in chapter 5. More particularly, we focused on the conflicts caused by diverging views on activities and dependencies and how they can be handled to ensure a converging view. We contributed with novel conflict detection and handling mechanisms in an inter-organizational setting.

- Franke, Jörn ; Widera, Adam ; Charoy, François ; Hellingrath, Bernd ; Ulmer, Cédric : *Reference Process Models and Systems for Inter-Organizational Ad-Hoc Coordination - Supply Chain Management in Humanitarian Operations*, short paper, 8th International Conference on Information Systems for Crisis Response and Management (ISCRAM'2011), Lisbon, Portugal, 8-11 May, 2011. Coordination by people of different organizations does not only require adequate tool support, but also a common understanding of each other's processes. We discuss how reference process models, integrated into our tool, can create a common understanding between different humanitarian aid organizations, so that they can coordinate more effectively and efficiently.
- Franke, Jörn ; Charoy, François ; Ulmer, Cédric : *Coordination and Situational Awareness for Inter-Organizational Disaster Response*, 10th IEEE International Conference on Technologies for Homeland Security (HST'2010), Boston, Massachusetts, US, 8-10 November, 2010. We explain in this technical paper related to chapter 5 how activities can be shared in different workspaces and how verification of a model containing shared activities can be done.
- Franke, Jörn ; Charoy, François ; El Khoury, Paul : *Collaborative Coordination of Activities with Temporal Dependencies*, 18th International Conference on Cooperative Information Systems (COOPIS'2010) / OnTheMove (OTM) Conferences, Crete, Greece, 26-29 October, 2010. This technical paper contributes with a framework for coordination of activities in dynamic situations presented in chapter 4.
- Franke, Jörn ; Charoy, François : *Design of a Collaborative Disaster Response Process Management System*, 9th International Conference on the Design of Cooperative Systems (COOP'2010), Aix-En-Provence, France, 19-21 May, 2010. We discuss in this paper the domain concepts explained in chapter 2. We explain why current process-based approaches provide only a limited solution. Furthermore, we discuss possible evaluations of the concepts presented in chapters 4 and 5.
- Franke, Jörn ; Charoy, François ; Ulmer, Cédric : *A Model for Temporal Coordination of Disaster Response Activities*, 7th International Conference on Information Systems for Crisis Response and Management (ISCRAM'2010), Seattle, Washington, US, 2-5 May, 2010. We describe in this paper our proposed model for temporal coordination of activities in chapter 4 and results from interviews about this model.
- Franke, Jörn ; Charoy, François ; Ulmer, Cédric : *Pervasive Emergency Response Process Management System*, 8th IEEE International Conference on Pervasive Computing and Communications (PerCom'2010), First Annual Workshop on Pervasive Networks for Emergency Management (PerNEM'2010) Mannheim, Germany, 2 April, 2010. We briefly outline the framework presented in chapter 4 and discuss the problems with an extension to the inter-organizational level. We argue why contemporary approaches in distributed systems do not address them adequately.

Furthermore, we discuss possible technical implementations.
– Franke, Jörn ; Charoy, François ; Ulmer, Cédric : *Un modèle centré activité distribué pour la coordination des acteurs de la crise*, Workshop Interdisciplinaire sur la Sécurité Globale (WISG'2010), Troyes, France, February, 2010. This paper contributes with an integrated research plan to address the research problems stated here.

2.4 Conclusion

We have described in this chapter the context and problems related to coordination of activities by people of different organizations in dynamic situations. We utilized disaster response management as a critical example for this. Existing tools in this domain have limitations with respect to coordination of activities. The information related to coordination is stored in unrelated artifacts, such as mission diaries or plans, and unrelated ad-hoc messages exchanged between people of different organizations. Particularly, it is difficult to get an overview what has been done, what is currently going on and what are the next steps, i.e. the relations between activities. However, it is important to deal with shifting goals, because consequent reassessment of activities needs to take into account their relations. Coordination on the inter-organizational level introduces the problem that not all information about activities can be provided to everybody due to privacy, regulatory, strategic or other reasons. This means that there is and can be only a partial view on the activities and their relations. Furthermore, it is difficult to provide the relevant stakeholders with always up-to-date information, because the relevant stakeholders can be found in different unrelated messages. We assume that information system support can overcome these limitations of current tools (cf. also [BBSBMD$^+$06]).

Chapitre 3

State of the Art

Contents

3.1 Introduction 41
3.2 Coordination of Activities in Dynamic Situations 42
3.2.1 Process-Based Coordination 42
3.2.2 Artifact-Based Coordination 49
3.2.3 Rule-Based Coordination 52
3.2.4 Integrated Approaches 54
3.2.5 Summary 56
3.3 Coordination by People of Different Organizations 56
3.4 Conclusion 60

3.1 Introduction

Coordination of activities within and between organizations is of key importance in many business domains, such as distributed development projects or supply chain management. Dynamic situations as they can occur in many contexts require adequate support for coordination of activities to achieve goals - even when these goals shift. The domain disaster response management is an extreme case for this, but research in context of this domain can inform the design of information systems addressing the research problems presented in chapter 2. The goal of this chapter is to investigate how contemporary information systems address them. We mainly present here tools that have been developed and used in the business context to coordinate activities according to business goals. The reason is the majority of tools have been designed for this context. However, we will also address tools targeting an emergency response use case. We will discuss their advantages and limitations in dynamic situations given the context in chapter 2.

We identify in section 3.2 four categories of approaches that can be used to coordinate activities in dynamic situations. We report for each category several information systems and discuss how they can be applied given the disaster response context. The state of the art of information systems supporting coordination by people of different organizations

is presented in section 3.3. We investigate information systems that support sharing information between people of different organizations. As we have described in chapter 2, sharing of information about activities is necessary to coordinate them. Consequently, we analyze how these information systems support coordination using this shared information. We conclude the gaps in the state of the art with respect to the research problems in section 3.4.

3.2 Coordination of Activities in Dynamic Situations

We present in this section four different categories of approaches related to coordination of activities in dynamic situations : process-based coordination, artifact-based coordination, rule-based coordination and combinations of them. Process-based coordination is used to make relations between activities explicit by specifying their control-flow. Artifact-based coordination can be compared to coordination of activities mediated via a business artifact (cf. also [Bar98, Bar00]), i.e. the relations between activities are modeled explicitly between activities and an artifact (e.g. an invoice). Rule-based coordination is used for defining relations between activities implicitly as a set of rules that should be followed during execution of activities. Finally, all these approaches can be combined together in several ways.

We categorize different technical solutions according to their main focus. All the approaches are illustrated via an example from the business domain, because they have been developed primarily for supporting business operations. We discuss advantages and limitations of these approaches from the perspective of coordination of activities in dynamic situations given the disaster response context presented in the previous chapter.

3.2.1 Process-Based Coordination

Process-based coordination can be seen under the umbrella of business process management [vdAtHW03]. The discipline of business process management is broad [Ham10, vBR10, LD98] and the focus here is its technical dimension. The activities in business processes can be managed and coordinated using a business process management system (BPMS) or workflow system [DHL01, vdAtHW03, OAWtH10]. The scope of these systems are operational routine business processes that have few predictable exceptions during process execution and the execution frequency of one process is high [DvdAtH05]. Examples for these types of processes are administrative processes, such as travel booking or approval processes.

Processes are fully specified in these systems from a control-flow, resource and data perspective. The control-flow makes the dependencies (relations) between activities explicit by describing activity sequences, e.g. activity “B” follows activity “A”. Each activity of the business process is part of a sequence. These sequences or parts of them may also be executed in parallel. Furthermore, alternative sequences can be described. A business process has a clear start and a clear end. The control-flow describes a full specification of all possible activity sequences of a business process. The resource perspective defines who is supposed to execute an activity. The data perspective describes the data that is

processed by activities. A workflow system or process management system can support the coordination of a business process by executing automated activities in the process, assigning human resources to activities (asking them for input) and keeping track of the current execution state (what activities are currently executed and have been executed) according to the predefined business process model. More precisely, the process is enforced by the system without possible deviations. The process models need to be correct, otherwise they cannot be coordinated properly by a workflow system.

Recently, the Object Management Group (OMG) published the Business Process Modeling Notation (BPMN) standard version 2.0. This standard describes how business processes could be modeled and executed within a workflow system [bpm]. Figure 3.1 illustrates an example for a process modeled using this standard : the loan management process. A customer requests a loan from the bank. The bank clerk enters some data about the loan. A decision is made, if there is a need for an extra check of the loan based on the data of the loan. An extra check means that a background check and history check of previous loans of the customer is initiated. This check is done by the credit bureau. In any case, the manager decides to approve or not approve the loan. The bank clerk has to finalize the loan application in the end.

However, the standard workflow approach is not seen as very flexible and drawbacks have been identified in different work contexts (cf. [BBS95, Gri00, BR10]). For example, the process model describing the relations between activities cannot be changed once the process has started. Furthermore, the process is enforced by the system. This makes coordination supported by workflow systems unsuitable for dynamic situations. This led to the development of a variety of approaches to support flexibility in the coordination of a process, but still based on the same idea of a fully specified process model describing sequential relations between activities (cf. Figure 3.1). These approaches may provide flexibility from a control-flow, data and resource perspective.

Flexible Process Management Systems

The ADEPT approach supports a wide range of extensions to address the limitations of the standard workflow approach [Rin04, DR09]. For example, it allows adding, removing or skipping activities during process execution. The main goal of this approach is to ensure data consistency when performing these operations. Another goal is that the process model remains syntactically correct (e.g. does not contain deadlocks) after performing these operations.

Grigori et al. optimize the scheduling of activities by anticipating which activities can be already executed given their preconditions [GCG01, Gri01]. Thus, it weakens the strict sequential order of activities by supporting overlapping execution of activities. The processing of data streams using a workflow system is similar [BP07, Bio08]. Although the main focus of these approaches is the data perspective, they allow more flexible definition of the relations between activities and can deal to a limited extent with shifting goals. For instance, it is already possible to review a document due to a deadline, although the document has not been completed.

The GroupProcess system allows modeling of a process ad-hoc using a simpler notation as illustrated before in Figure 3.1 [HEN01, Hut04]. This means the follow-up activities

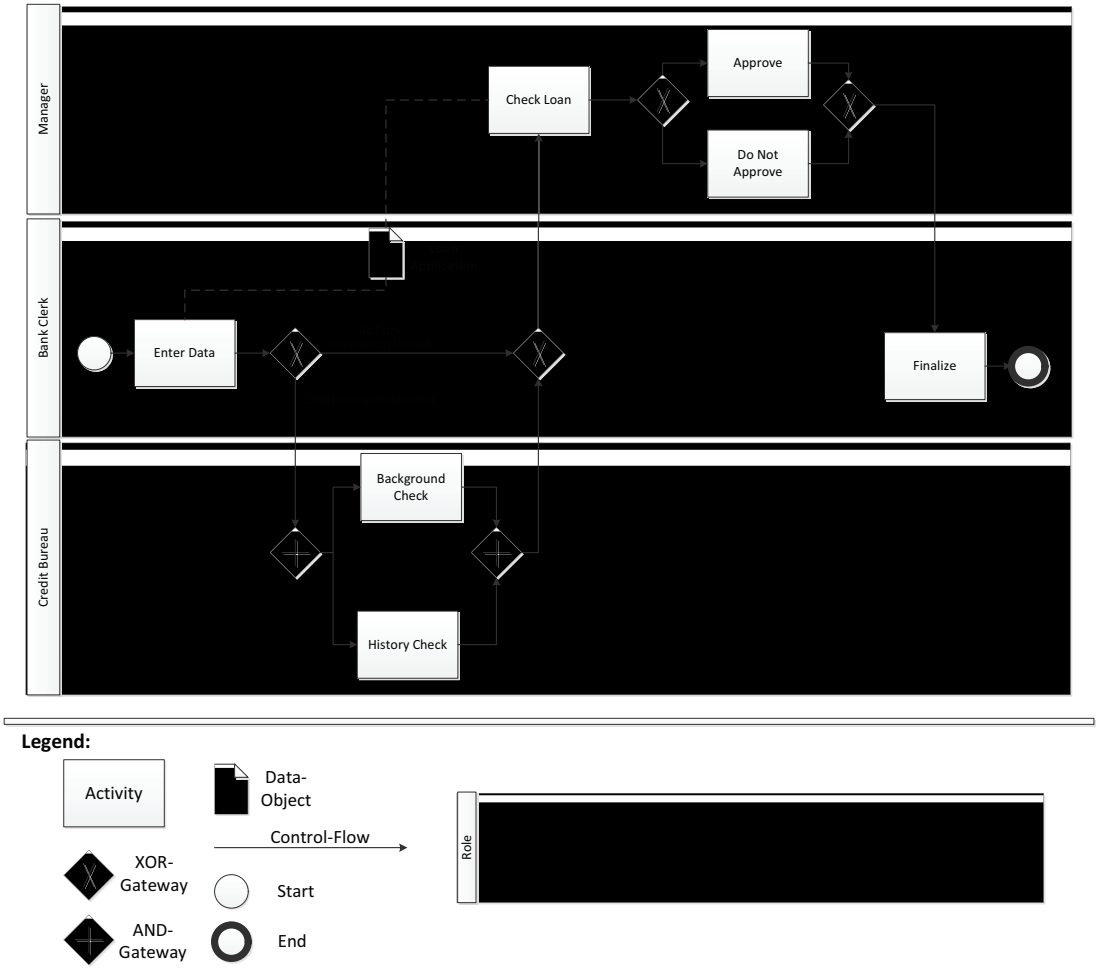


FIGURE 3.1 – Example for a process modeled using BPMN

of a process can be modeled later when they are known. The correctness of the process can only be verified after it has been completely specified. This means that errors can be detected only very late. The main use case is to support collaboration on documents. It has been integrated into the collaboration system Lotus Notes.

The ActionWorkflow approach postulates to describe an activity as negotiation loops between "customer" and "performer" [MMWFF92]. The negotiation loop consists of the following phases involving the two roles : Request (Customer), Confirm (Performer), Implement (Performer) and Evaluate (Customer). Each phase in the loop can be detailed by further negotiation activities. This way of describing the coordination of activities can be seen as a very simplified version of the process modeling notation presented before, because it basically only allows defining sequential dependencies between activities without specification of alternatives.

Other approaches allow flexible delegation of activities to other people [Gaa10, GZCG10]. This does not change the process model itself, but allows different people to work on activities to deal with organizational exceptions. For example, an activity is delegated to a proxy, because the employee, who was supposed to fulfill the activity, became ill. While still an important topic, this type of research is out of scope of the thesis.

The presented solutions are generic without any specific domain context. We investigate in the next paragraph process management systems that have been designed for an emergency response use case.

Emergency Response Process Management Systems

Process management systems have also been proposed as a use case for coordinating routine emergencies. They have been also enriched with some flexibility to meet the needs of this use case.

The ActionWorkflow approach described before has been proposed for coordinating a crisis response scenario [MMBA99]. Main motivation was the integration of different systems and the possibility to simulate the process.

The ADEPT approach has also been adapted to a emergency response scenario [RW07]. Main motivation was that the process model of a currently executed process can be modified according to the needs of the situation.

The collaboration management infrastructure (CMI) provides not only collaboration support (e.g. writing text), but also coordination of activities using a process-based approach similar to the one described in the beginning of this subsection. The use case is also a crisis response and it has been presented to emergency managers [GSBC00]. This approach allows defining templates that can be used to escalate currently running processes (e.g. an exception occurs). These templates can be defined dynamically.

Fahland et al. propose to describe different emergency response scenarios and associate them with processes [FW09]. These scenarios can be composed to build dynamically a process for the current situation. It is is based on well-established formalisms, such as Petri nets. The specification of scenarios seems to be detailed and complex. The approach has been originally developed for dynamic system integration.

The ROME4U process management system has been developed in context of the European WORKPAD project [dLMDG07, dL09]. The main focus of this project was ensuring

inter-operability between different emergency response organizations using specific technologies, such as peer to peer or semantic technologies. The ROME4U system monitors processes and their context (e.g. temperature). If the context changes (e.g. temperature change) then the processes can be "repaired" to fit to the new context. Repairing means, for example, adding new activities to a currently executed process.

Reijers et al. explore how resources can be scheduled to activities in a process using algorithms for swarm intelligence [RJVzMA07]. Their use case is an emergency response scenario involving one organization. It has been developed together with a fire fighter department. Similarly, the WIFA approach has been extended to take into account resource constraints in an emergency response scenario [WR09].

Discussion : Using a Business Process Management System for Coordination of Activities in a Dynamic Situation

The process-based approach provides some useful functionality for addressing our research questions. It allows defining the next steps and it is possible to monitor what has been done as well as what is currently in progress. The process is made explicit, so that the user can understand the relations between activities [vdAtHW03]. It can be visualized to the user similar to the process illustrated in Figure 3.1 or according to a timeline. There are also a lot of approaches for verifying correctness of the business process model (cf. [FFJ+09]). This is needed to ensure that the model does not contain syntactic errors. If a process model contains syntactic errors then a process management system cannot support properly the coordination of this process.

However, using a business process management system for managing response processes in emergency situations or even in a disaster context has been questioned by various scholars [GSBC00, Den06, dLMDG07, FW09, FCEK10, FCU10b]. This is why flexible extensions of the concept have been proposed. Nevertheless, they are still rooted in the original approach, which aims at supporting fully specified routine processes executed similarly many times. Thus, they still have similar limitations with respect to the research questions stated here [Dek03]. They are useful in situations where exceptions are not the rule, but not in dynamic situations where everything can be an exception. Georgakopoulos et al. describe critics by domain experts related to the usage of fully specified business process models [GSBC00]. Other scholars explain that their approach for supporting emergency response processes only considers standardized routines and it does not take into account exceptions from the control-flow [RJVzMA07]. Cugola et al. report difficulties to adapt business processes in dynamic situations, because a full specification of a business process can be complex [Cug98].

During the research of the SoKNOS project, several disaster response processes have been documented using business process modeling languages. These processes are based on interviews with domain experts with several years of experience in the management of disaster responses. They describe how an organization coordinates the response from its own perspective. Figure E.2 illustrates a result of this effort. We show only a few activities in detail due to space restrictions. It is about a response process to a train accident with hazardous material from the perspective of the police. It has been modeled using event-driven process chains [Sch00], but it has been shown that other business process modeling

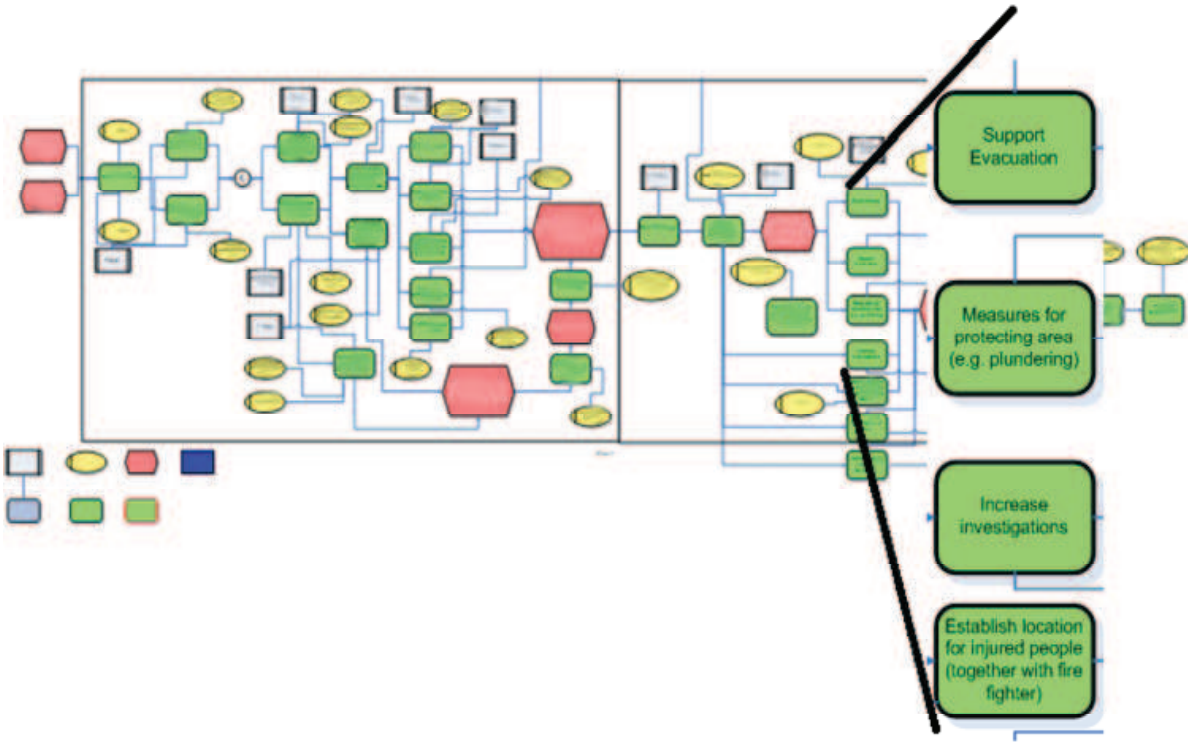


FIGURE 3.2 – Example model of a response process for responding to a train accident with hazardous material (from the perspective of the police)

languages are utilized similarly by process modelers. Thus, the results would be similar with other business process modeling languages [RD07, Rec08].

The first part of the process model describes activities that inform other organizations about a train accident with hazardous material. Furthermore, it describes the activities for creating a command center coordinating the response from the perspective of the police. This command center may be created by the police of the states (Landespolizei) or the federal police (Bundespolizei) depending on the disaster and the regulations who should be responsible. In the second half, activities are described for responding to a disaster. Examples for those activities are : evacuation, measures for protecting the area from plundering, crime investigation (it could have been a terrorist attack) or establishing a location for injured people together with the fire fighters. These process modeling examples and our interactions with domain experts in the SoKNOS project supported our understanding of the problems related to process-based approaches.

Based on these modeling efforts and our domain research presented in chapter 2, we identified the following issues of business process management systems for supporting coordination of activities in a dynamic situation :

– The current process-based approaches encourage definition of all possible execution alternatives in one process model, which is difficult to do in a dynamic situation.

They are usually not known from the beginning [Kle99, LN07], are subject to continuous change or there are so many that the model gets too complex. This means that activities and their relations are specified ad-hoc. Thus, it cannot be required that all possible relations are specified (cf. also chapter 2). Given the motivational example in chapter 2, it is not known from the beginning that the military will also transport and fill sandbags for the fire fighters.

- A business process is enforced by a workflow system. It is not possible to detect deviations from what is specified in the process model and how activities are executed. This limits the ability to deal with shifting goals. Given the example in chapter 2, the fire fighters may already start the activity of building a dam (e.g. going to the disaster site and preparing the area), because the flood quickly gets worse, without checking if the filling of sandbags has already been started. Enforcement would constrain the user unnecessarily. It is also problematic to enforce dependencies out of the control of the system as they occur often in human-driven processes as, for example, in crisis management.
- It is only possible to define who is responsible for executing an activity within a process. It is not possible to define, for instance, who can cancel an activity. This means the governance of activities is very limited. Going back to the motivational example, it would not be possible to define that the fire fighter commander can cancel the activity of transporting sandbags in case it is not needed anymore.
- Activities cannot be only described in a sequence or as parallel sequences (cf. also [Gon08]), whereby activities in a dynamic situation can also, for example, overlap, should take place at the same time or should start at the same time. For instance, the activities fill sandbags, transport sandbags and build dam for protecting an area from a flood usually overlap, because it is possible that building of a dam can start when already some sandbags have arrived. They are rarely sequential or strictly parallel. Sequential modeling makes sense for business processes that can be highly standardized and are frequently executed. It makes less sense for dynamic situations with many exceptions. This problem has already been identified in the BPM community by Dumas et al. : "[..] even today's workflow management systems enforce unnecessary constraints on the process logic [..] processes are made more sequential than they need to be" [DvdAtH05]. For example, it was very difficult to decide if activities in the process model given in Figure E.2 should be executed in parallel or in sequence.
- Modeling a process correctly takes time. For example, Figure E.2 shows the state of the process model directly after the interview. It has to be corrected afterwards to describe a syntactically correct model that can be coordinated by a workflow system. Even if the process could be defined to some extent before the response, changing it takes also time, because the user has to take into account the full specification with all possible alternatives.

Further processes have been documented with fire fighters (disaster with many injured people) and police men (rampage in a school). This has shown similar results. An attempt with the technical relief agency (Technisches Hilfswerk, THW) has not lead to any documented process model, because it provides only services for other organizations and thus their activities are partially coordinated by other organizations. Although this is just a

small sample, it shows already what problems can occur.

We conclude that the process-based approach provides some useful functionality with respect to our research questions, such as an explicit model of the relations between activities. However, there are major limitations, such as requiring full specification of the process and enforcement of the process by the system. Deviations from the process caused by shifting goals cannot be detected. Although it is possible to change the process description, this needs to be done in advance and cannot be highlighted to the user if there have been already deviations from it. However, this is not sufficient. For example, the fire fighters do not know that they will get injured and need to treat their colleagues. Another problem is to do it syntactically correct, because it is difficult for the user to find (1) the correct positions where to make the change and (2) to discover the parts of the process models that are erroneously defined. The user would also need to have access rights to modify a process and it is an open and complex issue how these rights should be assigned [Rei00]. Particularly, given the fact that the users can work for potentially different organizations. Furthermore, the specification of relations between activities is not very rich and only allows specifying sequential dependencies.

Finally, the approaches assume that the business goal is to some extent well-defined and does not change, because they expect that the processes are executed rather frequently with only few exceptions. However, in a disaster there are many exceptions due to shifting goals and current process-based approaches require a lot of effort by the user to deal with them.

3.2.2 Artifact-Based Coordination

The idea of artifact-based coordination is to mediate coordination via business artifacts (e.g. a loan application). This approach may also be known as case-handling. Artifact-based coordination is of practical relevance and has been successfully applied as an alternative to process-based coordination in a business context [BCK+07].

Artifacts usually have a lifecycle with states and state transitions between them (cf. [BCK+07]). Depending on the **state** of an artifact certain **activities** can be carried out. As a result of execution activities, the state of the artifact is modified.

In Figure 3.3, an example for a model for artifact-based coordination is illustrated. The artifact mediating coordination is the loan application. It is illustrated on the left of the figure. It can be in various states, such as “Initial”, “Checked”, “Approved”, “Not Approved” or “Final”. Depending on the state of the artifact, only selected activities can be performed. For instance, “Enter Data” can only be performed if the loan application is in state “Initial”, but not when it is in state “Final”. These actions are illustrated on the right of the figure.

Information System Support

The approach by [BGH+07, GS07] proposes to model the aforementioned artifacts. Services (comparable to an activity) can be executed to change the state of one or more artifacts. Each service may have preconditions (state of one or more artifact) that need

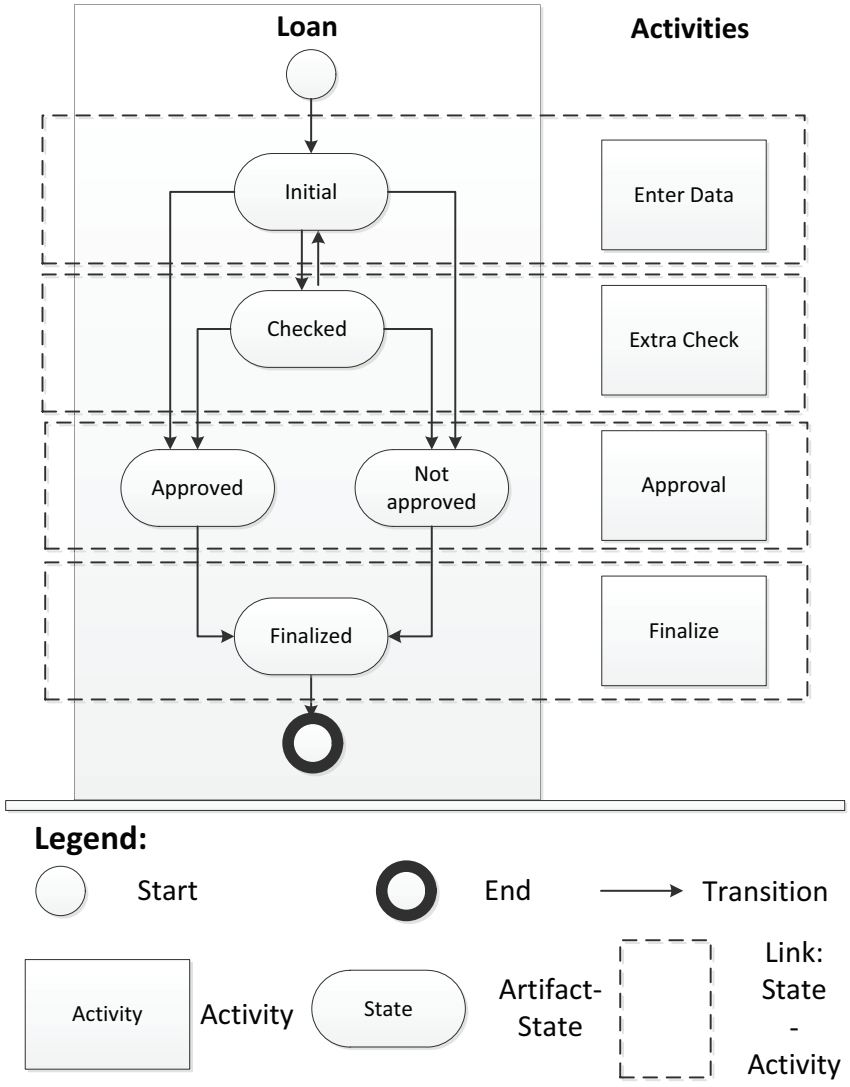


FIGURE 3.3 – Example of a model of an artifact and the corresponding activities

to be fulfilled before execution of a service. Their approach allows also verifying the correctness of the model.

The case handling approach has been implemented in the commercial FLOWer system [vdAWG05]. Cases can be compared to artifacts and have a state. Activity execution leads to change of the state of a case. Activities can have a certain state of a case as precondition.

The data-driven process management approach can be seen as variant of the artifact-based approach [MRH07]. During the execution of a process, the state of one or more data objects can change. Each process may have preconditions, which are the states of one or more data objects. The detection of errors in the model has been considered.

Another variant of the artifact-based approach is the product-based approach [Van09, VRvdA11]. A product data model describes the dependencies between a product and its sub-products. Activities can create products, if the required sub-products have been produced. Different execution strategies can be employed to determine the activities that can be executed next. This can be based, for instance, on cost or time. These models can be also verified for correctness.

The SAP Status and Action Management (SAM) approach is an example that has been implemented in an industry solution [HWK09, HWK10]. Business objects in this approach have a state. A state is defined by the values of the attributes of a business object. Transactions (comparable to activities) change the state of a business object. Furthermore, they can have a certain state of business objects as preconditions. It is unclear if detection of errors in the model of business objects and transactions has been incorporated in the original SAM.

Discussion : Using Artifacts for Coordination of Activities in a Dynamic Situation

The artifact-based approach has similar functionality as the process-based approach. The user can derive from the state of an artifact the next steps. Furthermore, it can be monitored what is currently done. It is also possible to track what has been done. The main difference to the process-based approach is that coordination of activities is mediated via the state of an artifact.

We identified the following limitations with respect to the research problems. Some of these limitations are similar to the ones of the process-based approach mentioned before. It requires full specification of the artifacts and associated activities. Detecting deviations caused by shifting goals is not possible, because activities and their relations are prescribed by the artifacts in advance. Similarly to the process-based approach, the specification of relations between activities is not very rich. Basically, it allows only the same type of relations as the process-based approach : sequential dependencies and parallel sequences. This is defined by the transitions of states in an artifact, which are sequential. Associating activities with a state of a business artifact allows parallel execution of activities. To the best of our knowledge, the artifact-based approach does not take into account governance aspects.

Finally, we could not model artifacts and activities together with disaster response managers. It was unclear what should be a relevant artifact in a crisis, e.g. an artifact could be a house in a residential area, fire trucks or the dam. The approach seems to be

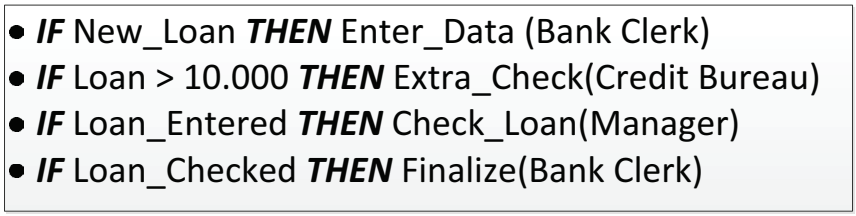


FIGURE 3.4 – Example of a model of rules

more suitable for digitalized business artifacts, such as an invoice or a contract, but less for objects in a disaster response. We came to the conclusion that artifact-based approaches are too limited given our research context.

3.2.3 Rule-Based Coordination

Rule-based or constraint-based coordination [KRSR98] is about defining a set of **rules** that are applied to a given set of **facts** to derive new facts. Facts can be about activities and rules can specify the relations between activities. The rules and facts can be entered by the user, but facts can also be derived from other sources (e.g. sensor data). The facts and rules can be visualized to the user differently. A **rule** can be described as follows :

"**IF** condition **THEN** fact".

A fact can be any kind of data, e.g. the age of the loan applicant or that an activity has been executed. A condition describes a constraint on an existing set of facts. For example, background check has been executed. This constraint can either be true or false. This is a very generic model and more detailed formalisms exist to describe constraints. This generic concept can be adapted to build a more specific concept for coordination of activities.

Figure 3.4 describes a simple example for a rule-based coordination model based on the loan application example used before. This example contains only four rules, but it can be expected that real-world scenarios consist of a lot of more rules. Basically the rules express that when a new loan application arrives the data about it has to be entered by the bank clerk. An extra check has to be performed if the loan is above 80.000 Euro and the manager has to approve the loan. The order of steps is irrelevant as long as they adhere to the rules. For instance, the manager can approve the loan without waiting for the result of the extra check.

Rules can be seen as the building blocks of the process-based or artifact-based approach. Indeed, these concepts can be expressed using rules [KRSR98]. However, these rules are usually enforced to ensure adherence to a process or artifact model. Other concepts can be expressed using rules. The problem is to define a concept which is predictable and useful for humans using it. Different rule formalisms can be used to describe

rules (e.g. linear temporal logic [Pes08]). This enables different expressiveness as well as reasoning on rules. Rules are very flexible and do not require a full specification as the process-based or artifact-based approaches. It is important that rules are not conflicting. For example, the following two rules can be conflicting : (1) IF 'A" then "B" and (2) IF "A" then "NOT B". It is not possible that "B" and "NOT B" are result of the same condition "A". Conflicting rules can be detected using various techniques depending on what is defined as a conflict. Several approaches have been proposed for supporting coordination of activities by using rules.

Information System Support

Skaf et al. describe cooperative software development activities that are coordinated by rules to ensure data consistency [SCG96, SC99]. The rules are described in a simple temporal logic based on a standard logic language. It has similarities with linear temporal logic. The activities and their rules can be defined ad-hoc, but it is not clear how the set of rules can be verified for correctness, i.e. how can a user make sure that the rules are not conflicting.

The Declare approach is a more recent approach for coordinating activities using rules (described there as constraints) [Pes08]. This work presents an approach based on linear temporal logic (LTL). These LTL formulas define the relations between activities. An algorithm for detecting conflicting rules exists, but depending on the specified rules it can have an exponential complexity in the worst case. One limitations identified by the author is that the approach cannot express the lifecycle of an activity. For example, it is not possible to define that an activity starts and afterwards it can be canceled or fail. Their approach would, for example, allow that an activity can fail without having it started.

The DISC framework is another rule-based (or declarative) approach for coordination of activities [ZPG10]. Rules are described using event calculus. Relations are not established between activities directly, but as rules between events (facts) generated by activities (e.g. activity "Transport Sandbags" starts). A modeled set of rules is used to generate different alternative sequences for executing activities. Depending on the defined rules, many different execution sequences are possible. One of the generated sequences is selected by the user for execution. During execution new possible sequences can be generated if the set of events or rules changes. If it is not possible to generate a sequence then the modeled set of rules is conflicting. Generation of possible activity sequences can be computationally complex.

The Event Condition Action (ECA) system [DHL90, Kri95] proposes to model conditions (C) between events (E) to trigger an action (A). Its main purpose is to coordinate database transactions, which are not exactly comparable with activities in our sense. It is unclear how conflicting rules are detected.

Another rule-based coordination system is the Freeflow approach [DHMZ96]. Constraints can be defined between states of different activities. The states of an activity are fixed in their modeling concept. It is unclear how conflicting rules are detected. Intelligent To-Do Lists are similar [WPT06]. However, constraints can be defined between tasks (activities).

Discussion : Using Rules for Coordination of Activities in a Dynamic Situation

A rule-based approach for coordination of activities addresses some of the research problems stated before. They do not require full specification of a process or an artifact in advance. They are able to establish many different types of relations between activities. This goes beyond sequential dependencies. For example, it is also possible to describe that activities have to start at the same time or overlap. Verification for correctness can be based on well-established formalisms (e.g. linear temporal logic). Furthermore, deviations caused by shifting goals can be detected by finding rules where the conditions for executing an activity cannot be fulfilled.

However, this flexibility comes with a price : It can be computational complex to verify the rules, so that no conflicting rules are defined (cf. for linear temporal logic [Pes08]). This is problematic in a dynamic situation where the rules can change very often. Using a rule-based approach has the disadvantages that the relations between activities are hidden in a potentially large set of rules (cf. also [WW06, Rei00, Sch11]). It is difficult to visualize rules, because usually only textual or formal descriptions are available. A visualization similar to the process-based or artifact-based approaches would be helpful for the user to better understand the relations between activities. Although some of the approaches propose some visualization for logic rules (cf. [Pes08]), it seems to be still difficult to understand by people who are not experts in logic languages, which has also been acknowledged by the authors of these approaches (cf. also [Rei00]). This means it takes a lot of time for the user to describe the relations properly, which is again problematic in a dynamic situation.

Furthermore, it is difficult to understand by the user what could be done in case of a deviation caused by shifting goals or what the impact of this is. It is already a problem for the user to understand what could be done next [Rei00].

Finally, another problem is to define what should be described with rules. Rules are very expressive, but coordination in a dynamic situation cannot be about expressing everything. We have stated that it should support the user to understand what has been done, what is currently going on and what are the next steps.

3.2.4 Integrated Approaches

The advantages and limitations of the previously described approaches have led to proposals to integrate two or more of them. We present them in the subsequent paragraphs.

Integrating Rules and Processes

Different approaches have been proposed to integrate rules and processes. For example, rules can be used to adapt business processes when an exception occurs. This has been suggested by Müller et al. [MGR04]. They utilize ECA rules and business process models as they have been described before. Furthermore, they provide a framework to detect conflicts in the ECA rule set. Rules are used to automatically adapt a currently executed process, i.e. they address processes which are primarily driven by a system and not humans. Automatic adaption is useful in case of an exception (e.g. data not available).

This is similar to the philosophy of business process management systems. Further related approaches can be found in [SSO01, Ada07, Pes08]. The Worklet approach allows underspecifying some activities in a process [Ada07]. Rules are used during execution to select process fragments to specify these activities.

Raposo et al. propose to specify temporal rules for synchronizing processes [RMR00]. These temporal rules can also be visualized according to a timeline (cf. [All83]). Nevertheless, these rules are enforced and the model of temporal rules as well as workflows is cannot be modified once it is executed by the system.

Integrating Rules and Artifacts

Several approaches have been suggested to integrate rules and artifacts. The Cordys approach is an example for this [dM09a, dM09b]. Similar to the artifact-based approach, activities can be executed given the state of a case. Execution of activities may change the state of a case. Rules can define that activities can only be executed when some conditions over the case data are fulfilled (e.g. income $<$ 50.000 Euro). Thus, relations between artifacts and activities can be more fine-grained.

Document-based workflows are very similar to this, but the concepts are described differently and rule formalisms, such as linear temporal logic, are employed (cf. [RRS09, Rah10, PBB10]). An example would be to represent the artifact as a document and depending on the state of the document different services (comparable to activities) can be executed. These services manipulate the state of the document and are constrained by the rules imposed on them. Detection of conflicts between rules can be done using standard mechanism provided by LTL (cf. [Pes08]).

The Prosyt system [Cug98] is similar with this respect, but the focus is explicitly not on enforcing the rules, but allowing to deviate from them and eventually reconcile later. Detection of conflicting rules has not been considered.

Integrating Artifacts and Processes

Few approaches integrate artifacts and processes. For example, Wang et al. model the artifacts generated by activities or consumed by activities explicitly in the process model [WA05]. This is also supported by the business process modeling notation described in the previous subsection about process-based approaches. This notation allows to model data objects manipulated or consumed by activities.

Discussion : Using Integrated Approaches for Coordination of Activities in a Dynamic Situation

Integrated approaches combine the advantages of the previously presented approaches for coordination of activities in dynamic situations. For example, integrating rules and processes allows a richer specification of the relations between activities in a process. The same holds for integrating artifacts and rules.

However, integrated approaches do not overcome the limitations of the previously presented approaches with respect to our research questions in context of the disaster response scenario. For example, integrating rules and sequential business processes requires

still rules, where the relations are not made explicit. Similarly, integrating artifacts and rules or artifacts and processes requires still fully specified artifacts or processes. Thus, they have the same limitations.

Finally, the user has to specify more when using an integrated approach (e.g. a process and rules). This means verification for correctness can be also computationally more complex. He has also to decide what should be specified as an artifact, rule or process. Given our process modeling experiments with domain experts and the context we gave in chapter 2, we do not assume this is feasible for the user.

3.2.5 Summary

We presented in this section different approaches for coordination of activities in dynamic situations and categorized them. Basically all approaches have been developed for a business context. The problem is that some of the approaches (e.g. process-based or artifact-based ones) assume to some extent well-defined business goals. For example, they require full specification of a business process or an artifact. Furthermore, these approaches assume that business goals do not shift radically, as it can happen in a disaster response. They require that the activities to reach a business goal will always be executed similarly with few exceptions.

On the other hand, there are rule-based approaches that can deal to some extent with this issue. However, contrary to process-based or artifact-based approaches, these rules are usually specified using textual notations. It is not obvious for the user what can be done next, because it cannot be visualized in a simple way as for example a business process or an artifact. Using well-established formal analysis tools for rules, such as linear temporal logic, can make it computationally complex to verify correctness of a given set of rules. This makes it difficult to modify this rule set, because verification needs to be performed after each modification and this takes time. This is problematic in a dynamic situation.

Integrated approaches combine the advantages of individual ones, but also inherit their limitations. Furthermore, they require more specification efforts by the user than individual approaches. We did not consider in this section how these approaches can support coordination by people of different organizations.

3.3 Coordination by People of Different Organizations

This section introduces technical approaches for coordination by people of different organizations. In the previous chapter, we have motivated that people of different organizations need to coordinate their activities. They need to share information about what they are doing with others. This is limited by the fact that information related to activities cannot be provided to everybody due to privacy, regulatory, strategic or other reasons. Moreover, it should be possible to provide only selected information to selected organizations. This shared information needs to be related to other activities in order to coordinate them.

We distinguish in this section two different types of approaches for doing this : one is an extension of the process-based approaches mentioned before to the inter-organizational level and the other one is based on distributed collaboration systems. Clearly, inter-operability of the different systems in this context is important and we see it as pre-requisite. Inter-operability of information systems for disaster management is another major research stream that has to take into account the particularities of this domain (cf. the IsyCRI project [TBP09, TBP10] or the SoKNOS project [DPZM09]).

Inter-organizational Process-Based Information Systems

Inter-organizational process management systems research dates back more than a decade. The basic underlying paradigm is that a public version of the business process model is shared with all organizations taking part in this business process. Each organization is responsible for executing a certain part of this public business process. They can relate a private process to their part of the public business process, which is not disclosed to the others. This means the details on how an organization executes its part are hidden. Besides this, they are similar to the process-based coordination approaches presented previously.

Several of these approaches have been described in the literature (e.g. [GH98, MM00, vdAW01, CDT06, MM05, Mon07, SO04, FYG09, SYY06]). Another variant are choreographies [FYG09]. The public process may also be known as a contract between different organizations [GAHL00].

Extensions allow only limited modification of the public process during execution compared to the approaches presented in the previous section (e.g. [RB07, RWR06, GAHL00]). The difficulty here is to synchronize these modifications from users of different organizations. Proclets can be used to define a fixed public communication model between processes and private processes can be instantiated in a flexible manner depending on the messages exchanged between them [vdABEW01, MRvdA$^+$10].

All these approaches require a full specification of the private and public processes. However, there are also other type of systems that enable exchange of information between different organizations without requiring a full specification of a model for coordination or sharing a public process with every organization.

Distributed Collaboration Systems

Collaboration systems enable users to work together on information, such as text, videos or pictures. Many examples for these types of systems can be found in the research domain Computer Supported Collaborative Work (CSCW) (cf. [RB91, Beg98, BS00]). These systems allow sharing of selected information with users of other organizations. However, the idea behind most of these systems is that they are under the control of one organization, which could theoretically have access to all the data of all the users - even if they are from another organization. This would not address adequately the research questions stated in chapter 2. In the following, we show some examples, where this is not the case, i.e. the exchange of information is done between different systems under control of different organizations. We do not consider here systems supporting simple exchange of messages, such as e-mail or instant messengers, because they are similar to the tools

presented already in chapter 2.

The collaboration service Google Wave allows sharing rich-text documents (called Waves) with different people and replicate them to the servers of the organizations the people belong to [Fer09, TP10]. This means selected documents can be exchanged between people of selected organizations. Different Waves can be linked using hyperlinks that allow navigating from one document to another. This can only be done if the user has access rights and it is replicated on the server of the user. Changes to a Wave are propagated to all servers where the Wave is replicated. Thus, all users with whom a Wave has been shared receive them. Google Wave is in process of being open-sourced as Apache Wave [Apa].

Distributed revision control systems (cf. [O'S09, Loe09]) enable sharing selected artifacts, such as source code files or documentation, between users. Each user can have its own server and shared artifacts are replicated to their server. A user can decide to propagate updates of the shared artifact to all its replica. The users can then decide to integrate updates, propagated by others, into their replicated version of the artifact. Links can be established by the users between different artifacts.

Peer to peer Wikis facilitate sharing of text structured as a "Wiki" between different peers (e.g. [OMMD10]). A "Wiki" consists basically of different pages that can be linked with each other via hyperlinks. These different peers can be under the control of different organizations and thus they can also control what information is exchanged. Once a text is shared between different peers, these peers receive also updates to this text propagated by any other peer.

Although all these solutions address to some extent the research problem, they are very limited with respect to coordination of activities. Basically, they allow sharing of unstructured information, such as text, videos or images. This makes it very difficult to get an overview on the relations between activities, because there is no agreement on how they should be modeled. This also means that it is difficult to detect errors in the model. Furthermore, they have limited capabilities to highlight the impact of shifting goals and consequent reassessment of activities taking into account the relations between them.

We find only few approaches trying to deal with this issue. Shared to-do lists contain tasks (comparable to activities) that can be accessed by users [KHW93]. These tasks can be distributed to servers under the control of different organizations. Everybody can add new tasks or modify them (e.g. their state). Dependencies can be modeled between tasks, but only sequential ones. It is not possible to detect deviations between the model and how tasks have been executed. It is unclear how correctness of the model of tasks and dependencies can be ensured.

Recently, several approaches have been described that allow collaborative modeling of a business process between people of different organizations in a distributed fashion (cf. [Kre10, pro, gra, SAP10]). They have been implemented as extensions to distributed collaboration platforms, such as Google Wave or SAP StreamWork. A similar approach is the more advanced Caramba platform [Dus04]. These approaches only allow modeling of a business process and the whole business process model is shared with selected users. Changes of the model are propagated to every user with whom the model has been shared. It is unclear how errors can be detected in these models. Furthermore, they do not provide any coordination support, such as highlighting deviations from what has been done and

what is defined in the model.

Discussion : Using Approaches for Coordination by People of Different Organizations

Inter-organizational process management systems are mostly based on the process management approach described in the previous section. Artifact-based or rule-based coordination has not drawn much attention with respect to the inter-organizational dimension of coordination.

However, the inter-organizational process-based coordination approach has several limitations with respect to the problem that not all information about activities can be provided to all organizations. Nearly all of them require defining a public process that is shared with all organizations that take part in the execution of the process. Although the public process is generic, this means all organizations involved know it and this is not desired in our scenario. It is very difficult to define such a public process. It is already difficult to define in advance which organization should be involved. Going back to the motivational example of chapter 2, it is not clear if it should involve the military, the fire brigade and the police or only military and fire brigade. There are arguments for both cases, because they are directly or indirectly dependent on each other. For example, the fire brigade depends on the military and the police depend on the fire brigade, but the military does not depend on the police and vice versa. However, the fire brigade does not depend on the military in the beginning. Current approaches do not seem to consider this. Another question is who should define this public process. There needs to be an agreement of all organizations on the exact process. Such an agreement is time consuming and not always feasible given the dynamics of the situation where an organization has to act quickly.

Distributed collaboration systems enable sharing of selected information with people from selected other organizations. Organizations can keep control over what is shared with people from other organizations. Nevertheless, it is hardly possible to model activities and their relations in a structured way, because these systems mostly deal with unstructured information, such as text, video or images. This makes it difficult to synchronize information about activities and their relations between people of different organizations. Furthermore, this unstructured information is of limited use for supporting coordination by an information system.

Some first approaches try to address this issue, but they are still very limited with respect to coordination capabilities compared to the approaches presented in the previous section (cf. also [TAH+08]). It is difficult to get an overview what is currently going on or what can be the next steps. Another problem is that it is unclear how they ensure correctness of the model of activities and their relations.

Finally, it is unclear how the shared information, once it changes, should be synchronized between different organizations and particularly how to deal with conflicts during synchronization (cf. also [GM94, RB91]). For example, one user describes an activity as failed and another user defines that the same activity has been canceled. They have a diverging view on the activity and may perform contradictory actions. A failed activity may, for instance, be instantiated another time, although it is not needed anymore since

it has been canceled.

3.4 Conclusion

We conclude that the approaches for coordination of activities in dynamic situations are limited with respect to the relations that can be defined between activities, the enforcement of a process that does not take into account shifting goals, the requirement to specify a full process with all alternative execution paths or they do not make the relations between activities explicit. Some of these approaches cannot be easily visualized to the user, so it is difficult to understand the relations between activities. Furthermore, verification for a correct model can be computationally complex. This limits the use of these approaches in dynamic situations with many changes. The expressiveness of governance for activities is very limited only allowing definition of responsibility for its execution, but it is not clear who can, for example, cancel an activity.

Current approaches for coordination by people of different organizations do not consider sufficiently that not all information about activities can be provided to everybody. This is due to privacy, regulatory, strategic or other reasons. Furthermore, they do not take into account sufficiently conflicts when synchronizing shared information about activities. Finally, these approaches are very restricted with respect to coordination of activities in dynamic situations.

Thus, we introduce in the subsequent chapters an alternative approach addressing the problem of coordination of activities by people of different organizations in dynamic situations.

Chapitre 4

Framework for Coordination of Activities

Contents

4.1 Introduction 61
4.2 Modeling Coordination of Activities 62
4.2.1 Activity Type 63
4.2.2 Activity 65
4.2.3 Temporal Dependency 65
4.2.4 Summary 67
4.3 Verification 68
4.3.1 Translation of a Model into a Temporal Constraint Network . . 69
4.3.2 Checking Satisfiability of a Temporal Constraint Network Using State of the Art Algorithms 71
4.3.3 Performance Considerations 74
4.3.4 Summary 75
4.4 Detecting Deviations from the Model and How Activities Are Executed 76
4.4.1 Tracking Execution of Activities 76
4.4.2 Dependency Violation 77
4.4.3 Unsynchronized Dependencies 81
4.4.4 Summary 85
4.5 Example 85
4.6 Related Work 87
4.7 Conclusion 88

4.1 Introduction

We address in this chapter the first research question related to coordination of activities in dynamic situations. We describe how (1) activities, their relations and roles

governing them are modeled, (2) the model of activities and their relations is verified to detect errors and (3) deviations from what is defined in the model and how activities are executed can be detected. We leverage a process-based approach as described in the state of the art in chapter 3. This means that we make activities and their relations explicit in a model. We go beyond of the state of the art by (1) not enforcing the process model, but highlighting deviations from what is defined in the model and how activities are executed, (2) providing a richer description of relations in the form of temporal dependencies beyond sequential ones, (3) not requiring full specification of a model and (4) allowing a richer description of governance for an activity. More precisely, we propose a **framework for coordination of activities in dynamic situations**.

We explain how coordination can be described in a model by a user in section 4.2. This allows defining what has been done, what is currently going on and what are the next steps. We provide a richer description of relations in the form of temporal dependencies to enable a more accurate view on what needs to be coordinated. Governance roles enable the user to describe accountability and responsibility for an activity. Afterwards, we show in section 4.3 how this model can be verified in predictable time. This is important, because the model can change very often due to the dynamics of the situation and verification can delay adding or removing of elements. Afterwards, we describe how deviations between what is defined in the model and how activities are executed can be detected. These deviations result from shifting goals in a dynamic situation and subsequent reassessment of activities not taking into account their dependencies. The deviations are highlighted to the user so that appropriate actions, such as communication with the stakeholders of the activity, can be initiated. This is presented in the section 4.4. An example of our framework is illustrated in section 4.5. We discuss our framework in the context of related work in section 4.6 and conclude in section 4.7.

4.2 Modeling Coordination of Activities

We have explained using the motivational example in chapter 2 that it is important to get an overview on the relations between activities. For example, the fire fighter commander wants to define that the following activities for protection of a residential area from the flood have been planned : fill sandbags, transport sandbags and build a dam. However, these activities should only be performed when the overall activity of protection of a residential area from a flood is performed. This means that the planning for this activity has been completed and that the risk evaluation has shown that the area really needs to be protected. Furthermore, the current state of activities should be represented in the model, so that the commander has an overview of what is currently going on. We explain in this section how the fire fighter commander can define activities and their temporal dependencies in a model.

We argue that not every activity in the model should be treated the same, but there are different types of activities. For example, there are very simple activities, such as warning of people, and more complex ones requiring approvals, such as evacuation of a residential area. Activity types can be defined beforehand and reused to create activities based on them in an ad-hoc manner. They allow the definition of the lifecycle of an activity

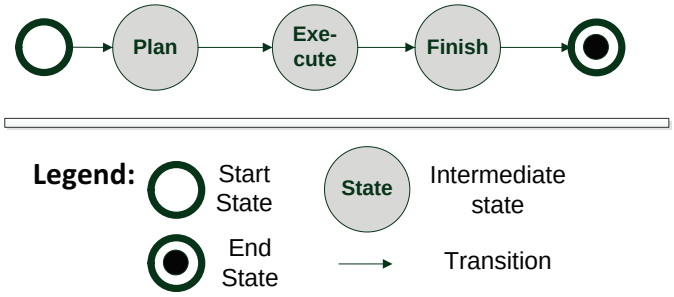


FIGURE 4.1 – Example for an activity type : Simple Field Operation

in the form of states and transitions. Governance roles define who can decide about the transition from one state to another. For example, the fire fighter commander can decide to change from a planning state to the execute state. Every activity has a current state and further information can be attached to it.

A temporal dependency describes the relation between activities. More precisely, it articulates what has been done, what is currently done and what are the next steps. This dependency is based on Allen's thirteen time interval relationships providing a richer description of the dependencies between activities than the state of the art [All83]. For example, it can be defined that the activity for building a dam should only be executed when the protection of a residential area from a flood commenced execution. It is not required to specify all possible dependencies in a model or specify them at all.

We introduce in the following subsections three modeling constructs : the activity type, the activity based on an activity type and temporal dependencies between activities. We summarize the model in the last subsection. We decided to use few modeling elements, because our process modeling efforts with disaster managers showed that it is very difficult to keep all information in a model up-to-date during a crisis (cf. chapter 3).

4.2.1 Activity Type

An activity type characterizes the management lifecycle of an activity, more precisely, the nature of an activity. For example, different activities exist ranging from very simple ones, such as warning of people to more complex ones for releasing a response team to build a dam. The main difference between simple and complex activities is the number of states in the management lifecycle. The activity type can be compared to a template that is used to create activities in an ad-hoc manner. We assume that most of the activity types are pre-defined when modeling activities.

In a basic form, an activity type can describe that an activity needs to be planned, then executed and then finished. For instance, an activity for warning the people can be defined on such a simple lifecycle. This simple lifecycle is illustrated in Figure 4.1.

Furthermore, the user can define different governance roles in this management lifecycle. Going back to the motivational example in chapter two, a military commander coordinating the transportation of sandbags can order the execution of this activity, but forces in the field may decide when this is finished or when it has failed. It can also be defined that the fire fighter commander can cancel it. More complex lifecycles can be

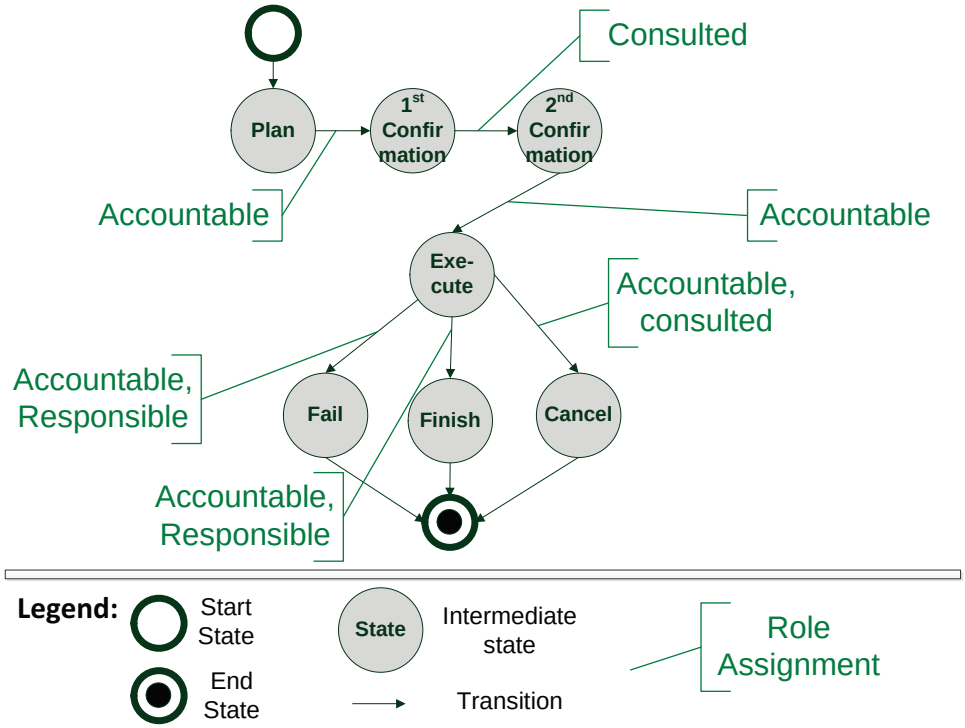


FIGURE 4.2 – Example for an activity type with governance roles for approval : Complex Field Operation

defined using this mechanism, e.g. activities that require approval by several people, such as evacuation of a residential area. An example for these lifecycles is illustrated in Figure 4.2.

We assume that only few different activity types will be used and that one activity type can be used for many different activities. We now introduce the activity type formally :

Definition 1 *An activity type* $at_d = (S, st, se, f, G)$ *represents the management lifecycle of an activity where*

- S *is a finite set of activity states*
- $st \in S$ *describes the start state of an activity type*
- $se \in S$ *describes the end state of an activity type (i.e. a state where no further transition is possible)*
- $st \neq se$ *a start state is not an end state*
- $f : S \rightarrow S$ *is a transition function defining the possible transitions from one state to another for one activity type*
- G *describes the governance roles* $g_1, ..., g_n$*, which describe who can transit from which state to another state :* $g_x \subseteq f$*.*

The lifecycle must not contain strongly connected components (i.e. cycles, cf. [Tar72]), because they can lead to confusion. For example, an activity is changed from state "Execute" to "Fail" and then again from "Fail" to "Execute". This is difficult to display and understood by the user. In particular, if the activity status is shared with other users.

4.2.2 Activity

An activity describes a human action that is performed in the real world, such as transporting sandbags or evacuating people. It is based on an activity type. It describes the actual activity to be performed and its current state. For example, it says that an activity for transporting a sandbag is based on the activity type for simple field operations. The activity "Transport Sandbags" is currently in the state "Plan" of the management lifecycle of the activity type "Simple Field Operation". We now introduce the activity formally :

Definition 2 *An activity is defined as* $a_i = (uid, name, cs, cat, A)$ *where*
- *uid is a unique identifier of the activity. This will be relevant in chapter 5.*
- *name describes the activity*
- $cs \in cat.S$ *is the current state of the activity. On creation it must be the start state st of an activity type.*
- $cat \in AT = (at_1, .., at_n)$ *one activity type in the set of existing activity types*
- A *describes the assignment of users* U *to roles in the activity type* $cat.G : A = U \times cat.G$ *by the creator of an activity.*

Any further data can be attached to the activity. Activities are independent from each other. This means they can change their state independently of any other activities. We observed during our interactions with disaster managers that many activities are running in parallel through different stages in the disaster response.

4.2.3 Temporal Dependency

A temporal dependency can be established between states of activities. It provides a richer description than the sequential ones found in the state of the art.

We use Allen's proposed thirteen time interval relationships for describing different types of temporal dependencies [All83]. A state of an activity can be compared to a temporal interval, because an activity is in a state for some time. In Figure 4.3, we illustrate seven of them, because the other six are the inverse of the first six. For example, the interval relation "overlaps" has the inverse "overlapped by". This provides a greater level of flexibility regarding the type of dependency that can be modeled. Allen's time interval relationships also have the following properties [All83] : qualitative, exhaustive and distinct. They are qualitative and do not require to define exact time points (e.g. in 5 hours and 5 minutes). We do not assume that it is always possible to define such exact points in times and if it would be possible then they needed to be updated constantly to reflect the current situation. However, our framework does not prevent defining exact points in time. Exhaustiveness and distinctness are other desired properties, because they reduce ambiguity about what is meant by a temporal interval relationship.

We chose these relationships over other rule formalisms, such as linear temporal logic or first order logics [LWZ08], because we assume that they can be understood more easily by humans than other logics. More precisely, they can articulate explicitly the relations between activities. We assume that they are easier to adopt by crisis managers. Finally, there is a natural visualization of the relationships according to a timeline.

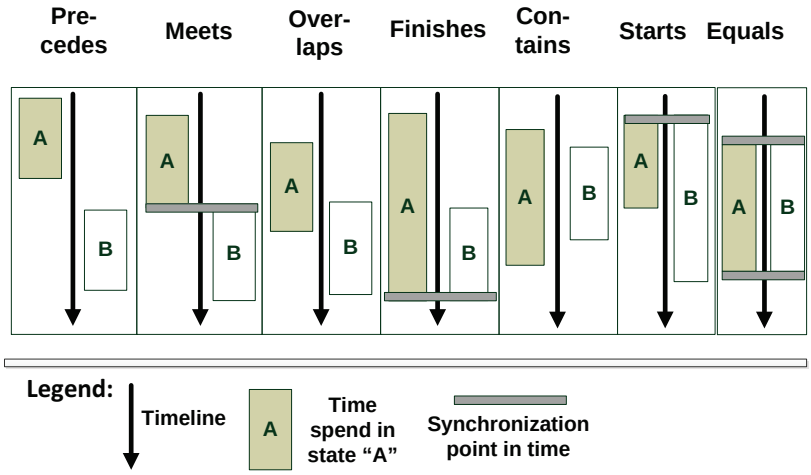


FIGURE 4.3 – Types of temporal dependencies between states of activities (based on Allen's temporal interval relationships [All83])

A temporal dependency can be established between two states of two different activities (see Definition 3). We chose to define temporal dependencies on the state level to be able to deal with exceptional situations. For example, an activity might fail and this requires initiating counter-measures.

Definition 3 *A temporal dependency is defined as* $d_i = (a_s, s_s, a_d, s_d, type)$ *where*

- a_s *is the source activity*
- s_s *is the state of the source activity, whereby* $s_s \neq a_s.cat.st \neq a_s.cat.se$ *(no dependencies between start and end states are allowed)*
- a_d *is the destination activity*
- s_d *is the state of the destination activity, whereby* $s_d \neq a_d.cat.st \neq a_d.cat.se$ *(no dependencies between start and end states are allowed)*
- *type is the type of temporal dependency*

The only limitation is that temporal dependencies cannot be defined between start or end states of the activity type of an activity. The reason is that an activity is in the start state when it is created, but a temporal dependency can only be defined after an activity is created. For example, let us assume there are two activities "A" and "B". It is not possible to define that after activity "B" is in a state "Execute", that activity "A" should enter a start state. This would require creation of a dependency to activity "A" before activity "A" is created. Another reason is that an activity is in the end state in an infinite point of time. For instance, let us assume there are two activities "A" and "B". It is not possible to define that after activity "A" has been in the end state, that activity "B" can enter state "Execute". This would imply that another state exist after an end state, which is excluded by definition (cf. Definition 1).

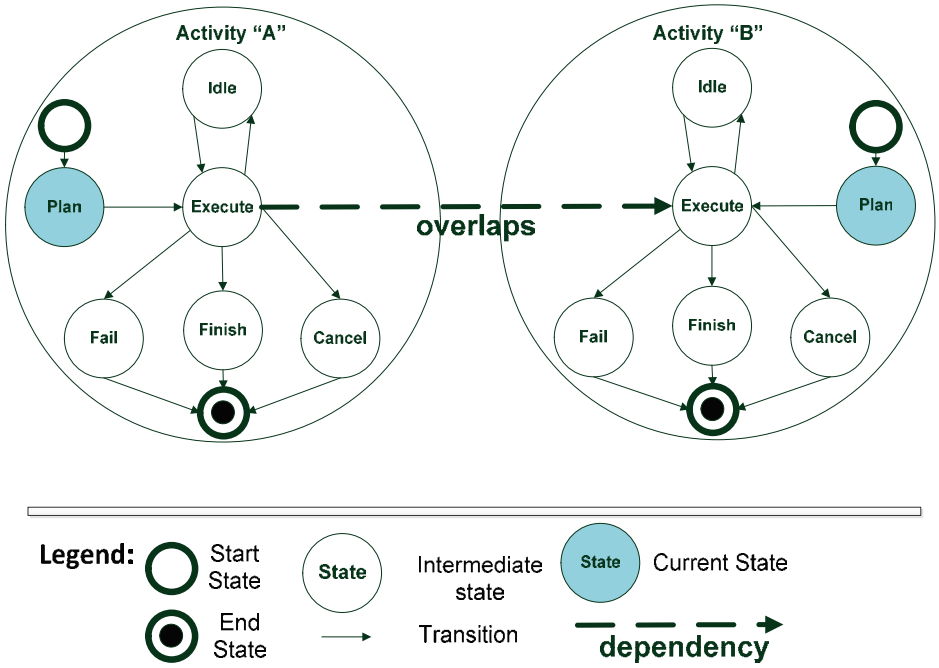


FIGURE 4.4 – Example for a model containing two activities with a temporal dependency

We illustrate in Figure 4.4 a simplified example of two activities "A" and "B" with a dependency "overlaps" between the states "Execute" of the two activities. This dependency means that activity "A" should enter state "Execute" before activity "B" enters state "Execute". Activity "A" should leave state "Execute" (e.g. by entering state "Finish", "Fail" or "Cancel") before activity "B" leaves state "Execute".

4.2.4 Summary

We presented in this section how the coordination of activities can be modeled. We propose to describe activities based on an activity type. The activity type can be compared to a template. It defines the management lifecycle of an activity and the associated governance roles. For instance, it is possible to define who can execute or cancel an activity. The relations between activities can be modeled explicitly as temporal dependencies between activities. They provide a richer description than sequential dependencies found in the state of the art in chapter three, because they are based on Allen's thirteen time interval relationships. Contrary to the process-based approaches explained in the state of the art, we do not require the user to fully specify all possible dependencies between activities and we allow a richer set of dependencies beyond sequential ones. These dependencies can be visualized according to a timeline, which we assume is easier to understand than text-based rules definition. We provide few modeling elements to reduce the complexity of modeling during a crisis. Of course, any other data can be attached to activities depending on the needs of the organizations, for example resources or geographical information. However, the main focus here is their coordination with temporal dependencies.

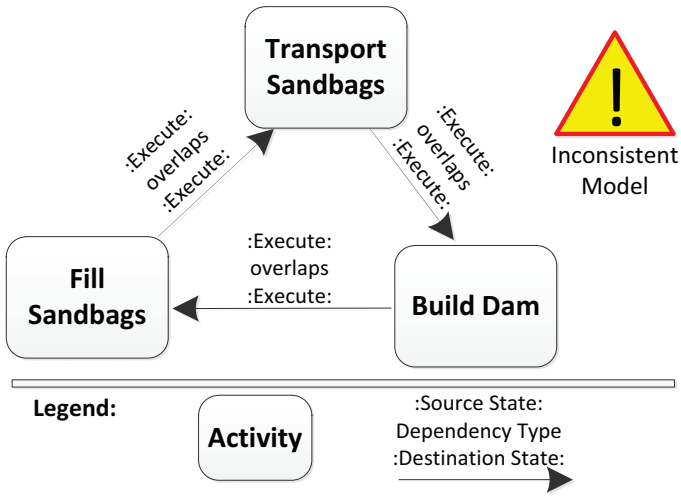


FIGURE 4.5 – Example for a model with errors

4.3 Verification

We cannot allow arbitrarily modeling of activities and temporal dependencies, because this can lead to errors in the model. An example for an erroneous model is illustrated in Figure 4.5. Three activities “Fill Sandbags”, “Transport Sandbags” and “Build Dam” are modeled. The activities are based on the activity type illustrated in Figure 4.1. A dependency “overlaps” is established between the activity “Fill Sandbags” in state “Execute” and the activity “Transport Sandbags” in state “Execute”. Another dependency “overlaps” is established between “Transport Sandbags” in state “Execute” and “Build Dam” in state “Execute”. Finally a dependency “overlaps” is established between “Build Dam” in state “Execute” and “Transport Sandbags” in state “Execute”. This basically means that the activity “Fill Sandbags” (or any other activity in this example) in state “Execute” overlaps itself, which is not possible.

Other errors in the model may not be that simple and can be very difficult to detect manually by the user. Particularly in dynamic situations, such as a disaster, where there is hardly time for “manual” checks. These errors are related to the temporal dependencies between activities. If the temporal dependencies can never be satisfied then the model is erroneous. We propose to rely on existing well-studied formalisms to perform verification of a model with activities and temporal dependencies. This means that the model of activities and dependencies needs to be translated into such a formal model so that we can perform the verification using well-established methods. Thus, we propose the following two step verification procedure :

1. Translation of the model into a temporal constraint network.
2. Detecting errors in the temporal constraint network.

We explain the two steps and their motivation in the following subsections. The verification procedure needs to be executed every time a dependency is added or removed from the

TABLE 4.1 – Constraint notation and abbreviation

Constraint Name	**Inverse Name**	**Abbreviation**	**Abbreviation Inverse**
Precedes	Preceded by	p	p^{-1}
Meets	Met by	m	m^{-1}
Starts	Started by	s	s^{-1}
Overlaps	Overlapped by	o	o^{-1}
Finishes	Finished by	f	f^{-1}
During	Contains	d	d^{-1}
Equals	Equals	e	e

model to be able to highlight to the user which dependency caused an error in the model.

4.3.1 Translation of a Model into a Temporal Constraint Network

As the first step of the verification procedure, we propose to translate a model into Allen's formalism for temporal reasoning (cf. [All83]), i.e. a temporal constraint network. This has the following advantages. We can rely on already existing and well-understood algorithms for detecting errors and results are available on the computational complexity [All83, Neb97, NB95]. Erroneous temporal constraint networks are also described as unsatisfiable, because in these cases there are no solutions, so that the constraints can be satisfied.

Definition of a Temporal Constraint Network

We define in Definition 4 a temporal constraint network. This is important to understand the translation of a model into such a temporal constraint network.

Definition 4 *A temporal constraint network is defined as* $CN = N \times N \times C$ *where*

- N *is the set of nodes in the constraint network. A node corresponds to a time interval of a undefined finite length.*
- $C \subseteq DT$ *is the set of constraints between two nodes. Multiple constraints (e.g. precedes, meets) between two nodes describe that one of them is possible (e.g. precedes or meets).*
- $DT = \{precedes, precededby, meets, metby, overlaps, overlappedby, finishes, finishedby, contains, during, starts, startedby, equals\}$ *is the set of possible constraints based on Allen's interval relationships [All83]. Table 4.1 illustrates the notations and abbreviation used for these constraints.*

We distinguish between the graphical representation of a constraint network and the data representation as a matrix. They are explained in the next paragraph using an example.

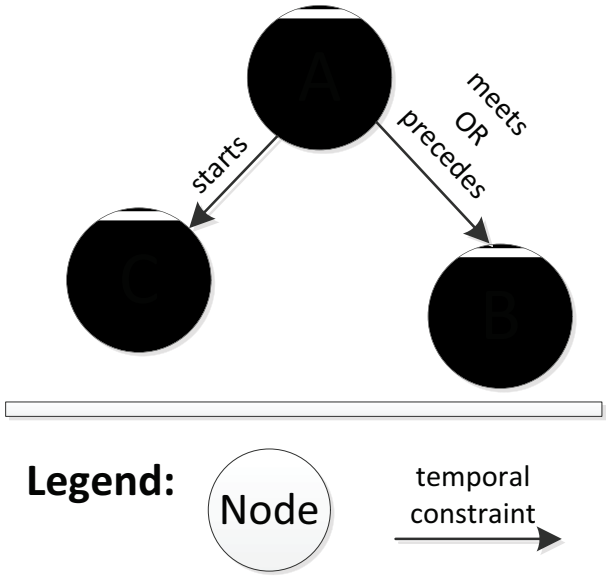


FIGURE 4.6 – Example for a constraint network

TABLE 4.2 – Matrix notation of constraint network

	A	**B**	**C**
A	e	$m\ p$	s
B	$m^{-1}\ p^{-1}$	e	all
C	s^{-1}	all	e

Example for a Temporal Constraint Network

We illustrate in Figure 4.6 an example for a constraint network. A matrix version of the same constraint network is described in Table 4.2. The matrix version provides more information and is used by the algorithm presented later to check for errors, i.e. unsatisfiable constraints. A node is referring to itself by using the equals (e) constraint, e.g. Node "A" has a constraint e (equals) to itself. The constraint from node "A" to node "B" in the example is "$m\ p$", which means that Node "A" either meets (m) or precedes (p) Node "B". This also means that the constraint in the other direction, i.e. from node "B" to node "A", needs to be the inverse of the constraint from node "A" to "B". In the example, the inverse of the constraint "$m\ p$" is "$m^{-1}\ p^{-1}$" (met by or preceded by). For all other relations, for which no constraint is defined, it is initially defined that all constraints are possible : *all*. For example, no constraint is defined between node "B" and node "C" in Figure 4.6. This means initially all constraints are possible between them.

Translation of Model into Temporal Constraint Network

We explain in this paragraph how a model of activities and temporal dependencies can be translated into a temporal constraint network. The translation adheres to the following rules :

1. States of an activity (as defined in the activity type) are translated into nodes of the constraint network. A state describes a time interval of an undefined finite length. As mentioned, time intervals correspond to nodes in a constraint network.
2. State transitions of an activity (as defined in the activity type) are translated into the constraint "meets" between the corresponding nodes in the constraint network. The inverse direction is translated into the constraint "met by".
3. A dependency is translated into the corresponding constraint between nodes in the constraint network (representing states of different activities) and the inverse direction is defined by the inverse constraint.
4. For all other nodes that do not have constraints between them defined, we define that all constraints are possible (precedes or preceded by or meets or met by or overlaps or overlapped by or finishes or finished by or contains or during or starts or started by or equals).

These rules cover the translation of all model elements relevant for temporal coordination to elements in the temporal constraint network.

Figure 4.7 illustrates an example of a translation of a model into a temporal constraint network. On the left, a model consisting of three activities with three dependencies between them is described based on the example in Figure 4.5. The activities are based on the same activity type previously illustrated in Figure 4.1. After applying the previously described translation rules, we obtain the constraint network illustrated on the right of the figure. The nodes of the constraint network correspond to states of the activities. The transitions between states within one activity and the dependencies between states of activities have been translated to the corresponding constraints between nodes.

If the model is changed then the existing constraint network can be updated or extended using the aforementioned translation rules.

4.3.2 Checking Satisfiability of a Temporal Constraint Network Using State of the Art Algorithms

Given a constraint network CN from a translated model, we want to check if it is satisfiable to make sure that it does not contain errors. Figure 4.8 illustrates an example for an unsatisfiable temporal constraint network. Table 4.3 describes the corresponding matrix notation. Temporal intervals are represented as nodes. There are three nodes in the network : "A", "B" and "C". A constraint "overlaps" is established between node "A" and node "B". The same constraint is established between nodes "B" and "C". Finally, the constraint "overlaps" is established between nodes "C" and "A". This means that the temporal interval represented by node "A" would overlap itself. This is not consistent from a temporal perspective, because it would imply that the temporal interval has several start or end points.

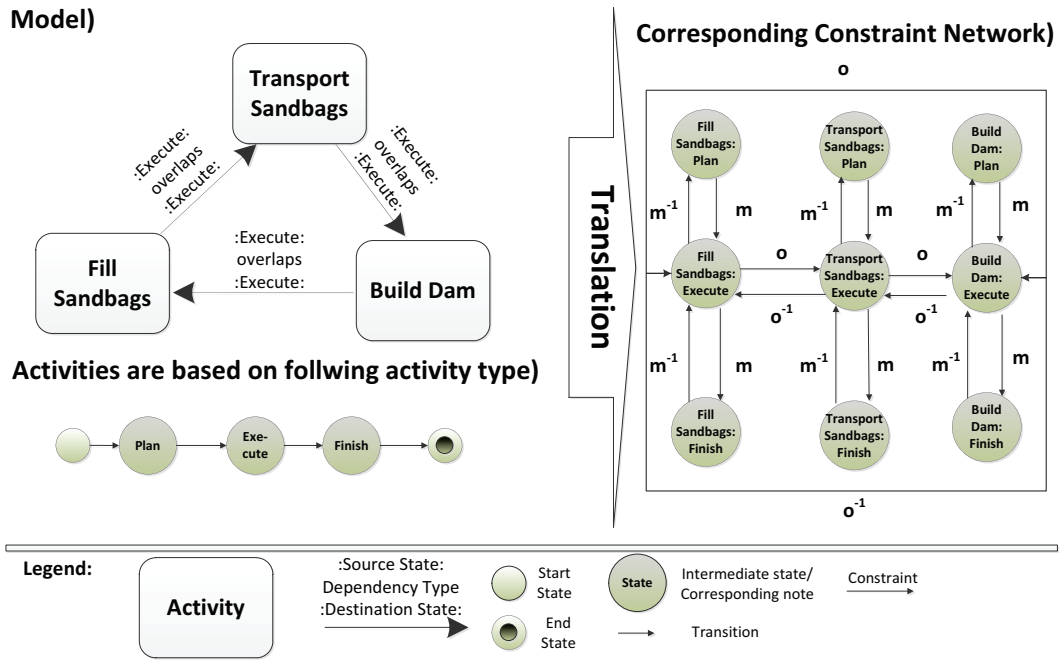


FIGURE 4.7 – Example for translation of a model into a temporal constraint network

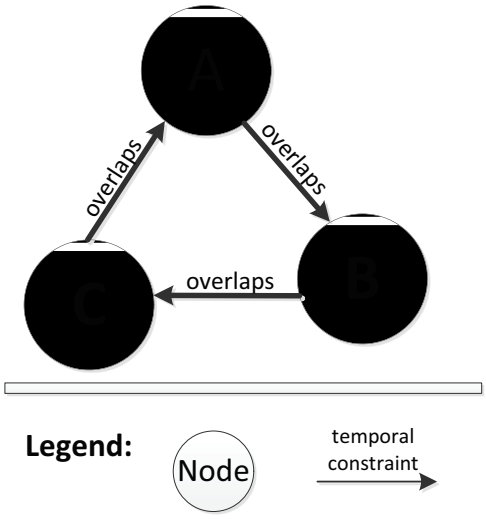


FIGURE 4.8 – Example for an unsatisfiable constraint network

TABLE 4.3 – Matrix notation of unsatisfiable constraint network

	A	**B**	**C**
A	e	o	o^{-1}
B	o^{-1}	e	o
C	o	o^{-1}	e

There are two different solutions to check satisfiability of translated temporal constraint networks :

– Enforcing local satisfiability, e.g. path consistency algorithm [All83]
– Using search algorithms, e.g. back-tracking [Kum92, LR92]

Enforcing local consistency is usually computationally less complex than using search algorithms. Search algorithms can have exponential complexity [Kum92, LR92] depending on the constraints in the temporal constraint network. However, algorithms for enforcing local consistency may not be able to detect all unsatisfiable temporal constraint networks, if the set of possible constraints in the temporal constraint network is not restricted. We decided to restrict the set of possible constraints to have a predictable and acceptable performance for verification of our model, since it needs to be performed every time a dependency is added or removed. We will show that this restricted set of constraints is sufficient for our model. This means we use an algorithm enforcing local satisfiability for checking satisfiability of a constraint network. More particularly, we use Allen's proposed path consistency algorithm [All83] (cf. Appendix B for details). We discuss in the next paragraph the limitations of this choice with respect to the possible constraints that can be expressed.

Completeness of Adapted Path Consistency Algorithm

As Allen notes, the path consistency algorithm is not complete. He provides as an example an unsatisfiable constraint network, which is wrongly reported as satisfiable by the path consistency algorithm (see [All83]). This is a typical problem of algorithms enforcing local consistency as it has been described before. Although it does not detect wrongly a satisfiable constraint network as unsatisfiable, it may detect wrongly an unsatisfiable constraint network as satisfiable. This is a problem in our case, because we want that the user is able to rely on the results of verification and it may have fatal consequences if the model contains errors, but is reported as correct.

Several researchers have investigated how the set of possible constraints need to be restricted, so that the path consistency method always leads to a correct result when checking a temporal constraint network. Particularly, they have found 18 maximal tractable sub-algebras of Allen's interval algebra (cf. [KJJ03]). Maximal tractable means that each of the sub-algebras cannot be extended by further combinations of temporal interval relationships without becoming intractable, i.e. checking satisfiability can have exponential complexity in the worst case. However, there is only one maximal tractable sub-algebra containing all basic relationships as it is required by our approach : The ORD-Horn-Class [NB95]. The ORD-Horn-Class is also the largest maximal tractable sub-algebra of Allen's interval algebra. It contains around 10% of all combinations of the full algebra. More

precisely, it contains 868 different combinations of temporal interval relationships [NB95], while the full algebra contains $2^{13} = 8192$ combinations. The ORD-Horn class provides support for all basic interval relationships and in addition also some combinations of them, i.e. our approach can be extended beyond the basic interval relations. For example, we can use combinations of basic interval relationships to simplify dependencies. A simple dependency would be that an activity "A" in state "Execute" has to be executed together with activity "B" in state "Execute". The dependency "together" can be represented using the constraint "starts OR during OR finishes". This constraint is also part of the ORD-Horn class. The path consistency method proposed by Allen can be used to correctly check satisfiability of a temporal constraint network where the constraints are elements of the ORD-Horn-Class [NB95].

Summary

We described in this subsection two different possibilities to check satisfiability of temporal constraint networks : search-algorithms or enforcing local consistency. We propose to use a special algorithm enforcing local consistency : the path consistency algorithm suggested by Allen for reasoning on temporal constraint networks. A satisfiable temporal constraint network means successful verification, whereby an unsatisfiable temporal constraint networks means failed verification. The algorithm is able to correctly verify a temporal constraint network derived from a translated model.

4.3.3 Performance Considerations

We have presented in this section an approach for verifying a model described using our proposed modeling elements. Verification is important for the user to detect if there is one possible solution so that the dependencies between activities can be satisfied. This feature is crucial, particularly in dynamic situations, otherwise there will be coordination based on wrong assumptions. An important aspect of verification is the performance of the approach. We expect that in a dynamic situation dependencies are added or removed at any point in time and every time the model needs to be verified again to be able to highlight to the user which dependencies caused an error. Thus, we investigate in the two subsequent paragraphs the complexity of each step of the verification procedure and conclude the overall performance [3].

First Step : Translation of the Model into a Temporal Constraint Network

Translating an activity into a constraint network can be done in linear time, i.e. $O(N)$, whereby N is the number of states in the activity type of the activity. However, when considering translation of all activities in a model (L), we need to take into account that all states of one activity have to define constraints to the states of all other activities (cf. section 4.3.1). This leads to $O(L * N * L * N) = O(L^2 * N^2)$ complexity. Translating a dependency into a constraint network requires just the modification of one entry in the matrix between two nodes, i.e. $O(1)$. In total, the complexity of translation of a given

3. We will use the well-known O-Notation [Knu76] to analyze the worst case theoretical complexity.

model is $O(L^2 * N^2)$. This can also be derived from the fact that a model translated into a temporal constraint network can be represented as a matrix with $L * N * L * N$ entries. This means there are $L * N$ nodes in the temporal constraint network. Please note that we assume for simplicity reasons that each activity type has the same size (N), but similar complexity observations can be made when incorporating the exact number of states of an activity type of each activity. However, we assume that the complete model is not given from the beginning and thus the performance is better, when activities and dependencies are added one after another $O(2 * N * (L - 1))$. Basically, if an activity is added then we need to add a row of the length $N * (L - 1)$ and a column of the length $N * (L - 1)$ to the matrix representing a temporal constraint network.

Second Step : Checking Satisfiability of Temporal Constraint Network

The path consistency algorithm used for checking satisfiability of a temporal constraint network has a worst-case complexity of $O(K^3)$, whereby K is the number of nodes in the temporal constraint network. This means the complexity is polynomial [All83]. Adding a node and a constraint to a constraint network already processed by the path consistency algorithm and processing it afterwards again has linear complexity, i.e. $O(K)$ [All83].

Overall Performance

The verification procedure has an overall complexity of $O(L^2 * N^2 + K^3)$ for a given model. Basically, the path consistency algorithm (second step) determines the performance of the approach, because the operations executed in this algorithm in each iteration are more complex than the ones in the translation procedure (cf. also Appendix B). Adding elements to an already verified model has only linear complexity, i.e. $O(L * N + K)$. The procedure can be further technically optimized, for example, when adding an activity or dependency to the model then only activities which are (in-)directly connected via dependencies need to be verified. The others are independent by definition and thus not relevant for verification. Although this does not change the theoretical complexity, it can improve significantly the practical performance in the implementation, because we expect that a user creates much more activities than dependencies. This means we expect many activities not (in-)directly connected with each other via a dependency. Our own implementation and also experimental results from other researchers (cf. [vBM96]) illustrate that the performance is acceptable.

4.3.4 Summary

We discussed in this section how errors in a model of activities and dependencies can be detected. Errors mean that it is impossible to satisfy the dependencies defined in the model. We proposed to translate the model into a temporal constraint network to detect these errors. This enables us to use the well-understood path consistency algorithm for checking their satisfiability. Verification needs to be performed every time the user adds or removes a dependency from a model to display to the user the dependency that lead to an error in the model. Thus, the user can ensure correctness of the model. We have shown

that this can be done in predictable and acceptable time. This suits dynamic situations, where we expect many ad-hoc changes to the model of activities and dependencies.

4.4 Detecting Deviations from the Model and How Activities Are Executed

We explained in chapter 2 that a dynamic situation can lead to shifting goals of an organization. These shifting goals can lead to a reassessment of activities. This has to consider the relations between activities. Given the dynamics of the situation, we cannot expect that the model is updated by users in an instant to reflect the situation, because it takes time to understand the impact of a new situation. This may lead to cases where the users may need to deviate from the execution order of activities defined by the dependencies in the model. This can be highlighted to the user, so that it can be taken into account later. Most of the process-based approaches presented in the state of the art in chapter 3 enforce the dependencies between activities. This is not possible in our case for the reasons mentioned. Going back to the motivational example, the fire fighters may start execution of building a dam, because the situation gets worse. They need to prepare and secure the area. However, they do not take into account that filling and transporting of sandbags has not yet been given the order to start. This can be highlighted to the fire fighter commander and he can give the order to start them by changing their state.

We describe in section 4.4.1 how the execution order of state changes of activities can be tracked in a trace. This is used to detect deviations from what has been defined in the model and how activities are executed. Deviations lead to violation of dependencies. We present in section 4.4.2 how this can be detected. There are cases where it is not possible to detect if a deviation leads to violation of a dependency or not. We explain how this can be addressed in the section 4.4.3. Both types of deviations can be highlighted to the user.

4.4.1 Tracking Execution of Activities

The execution of activities can be tracked to detect deviations from what has been defined in the model and how the activities have been executed. Execution of activities means that they change their current state. The state change is added to a trace that tracks the execution of activities. The trace can be roughly compared to the mission diary as illustrated in chapter 2. Thus, the trace contains what has been done.

State Change

A state change is initiated by the user. The state change is defined as a set of one or more different activities entering simultaneously from their current state a new state. This is needed, for example, to start two or more activities at the same time. We define this formally as :

Definition 5 *A **state change** is defined as* $SC = \{s_1, s_i, .., s_n\}, i = 1, .., n$ *where*

- $s_i = (a_i, a_i.cs, ns)$ *where*
  - a_i *is a activity as it has been described before.*
  - $a_i.cs$ *is the current state of the activity*
  - $ns \in a_i.cat.S$ *is the new state of the activity, whereby it holds that there is a transition from the current state to the new state* $a_i.cs \rightarrow ns \in a_i.cat.f$
- $a_i \neq a_j \forall i, j = 1, .., ni \neq j$ *Each element* s_i *of the state change has to contain a different activities*

A state change must not contain the same activity several times, because this would mean that it can change into different states at the same time.

Trace

A trace is needed to detect deviations between the model and how activities are executed. It tracks the execution order of state changes of activities. We define it formally as :

Definition 6 *A* ***trace*** *is defined as finite number of state changes* $T = (SC_1, SC_i, .., SC_n), i = 1, .., n$*, where*
- SC_i *is the i-th state change entry of the trace*
- n *is the number of entries in the trace*

We illustrate in Table 4.4 an example trace of a model with
- Set of existing activity types $AT = \{at_1\}$, with $at_1 = (S, st, se, f, G)$
  - $S = \{$"$Start$", "$Plan$", "$Execute$", "$Finish$", "$Obsolete$"$\}$
  - $st =$ "$Start$"
  - $se =$ "$Obsolete$"
  - f contains the following transitions
    - "$Start$" $\rightarrow$ "$Plan$"
    - "$Plan$" $\rightarrow$ "$Execute$"
    - "$Execute$" $\rightarrow$ "$Finish$"
    - "$Finish$" $\rightarrow$ "$Obsolete$"
  - $G = g_1 = f$: every user can change the state
- Two activities "A" and "B"
  - $a_1 = (1,$ "A", "$Start$"$, at_1, g_1 = user)$
  - $a_2 = (2,$ "B", "$Start$"$, at_1, g_1 = user)$
- A dependency $d_1 = (a_1,$ "$Execute$"$, a_2,$ "$Execute$", "$starts$"$)$

The example trace contains three state changes. Activity "A" and "B" are changed to the state "Plan" in step 1 and 2 respectively. They are simultaneously changed to state "Execute" in step 3.

4.4.2 Dependency Violation

State changes of activities can lead to violation of temporal dependencies. This means that the activities are executed in a different order from what is defined in the model.

Table 4.4 – Example for a trace

Number of State Change Entry	**State Change**
1	$SC_1 = \{s_0 = (\text{“}A\text{”}, \text{“}Start\text{”}, \text{“}Plan\text{”})\}$
2	$SC_2 = \{s_0 = (\text{“}B\text{”}, \text{“}Start\text{”}, \text{“}Plan\text{”})\}$
3	$SC_3 = \{s_0 = (\text{“}A\text{”}, \text{“}Plan\text{”}, \text{“}Execute\text{”}), s_1 = (\text{“}B\text{”}, \text{“}Plan\text{”}, \text{“}Execute\text{”})\}$
..	..

These violated dependencies should be highlighted to the user to display relations between activities that are affected by shifting goals.

We defined already in Definition 5 that a trace can only contain state changes of activities according to their lifecycle. This means we need to define mechanisms that makes it possible to decide given a trace if a dependency is violated or not.

We propose to represent dependencies as deterministic finite state machines. A trace is inputted into the dependency and depending on the trace they change to a state “Violated” or “Neutral”. Afterwards, we present an algorithm that inputs state changes into the finite state machines of the dependencies. We illustrate the algorithm with an example. Finally, we discuss the performance of our approach.

Representing Dependencies as Finite State Machines

We use an adapted version of finite state machines to decide if a trace is accepted by a dependency or not. If it is accepted then the dependency is “Neutral” and if it is not accepted then the dependency is “Violated”. This version of a finite state machine is called dependency state machine.

Definition 7 *We define a dependency state machine as* $DSM = (\Upsilon, \Omega, s, e, tg)$ *where*
- Υ *is the finite set of input symbols*
- Ω *is the finite set of states*
- $s \in \Omega$ *is the the current state. On creation it describes the start state.*
- $e \in \Omega$ *is the end states or acceptance state*
- $tg : \Omega \times \Upsilon \rightarrow \Omega$ *is the transition function that defines given an input to which other state*

The transition function tg of a DSM of a temporal dependency d can be described using the following constructs :
- $d.a_s : Sa$ source activity of the dependency $d.a_s$ changes into the state $Sa \in d.a_s.cat$
- $\neg(d.a_s : Sa)$ source activity of the dependency $d.a_s$ changes into any other successor state of $Sa \in d.a_s.cat$
- $d.a_d : Sb$ destination activity of the dependency $d.a_d$ changes into the state $Sb \in d.a_d.cat$
- $\neg(d.a_d : Sb)$ destination activity of the dependency $d.a_d$ changes into any other successor state of $Sb \in d.a_d.cat$

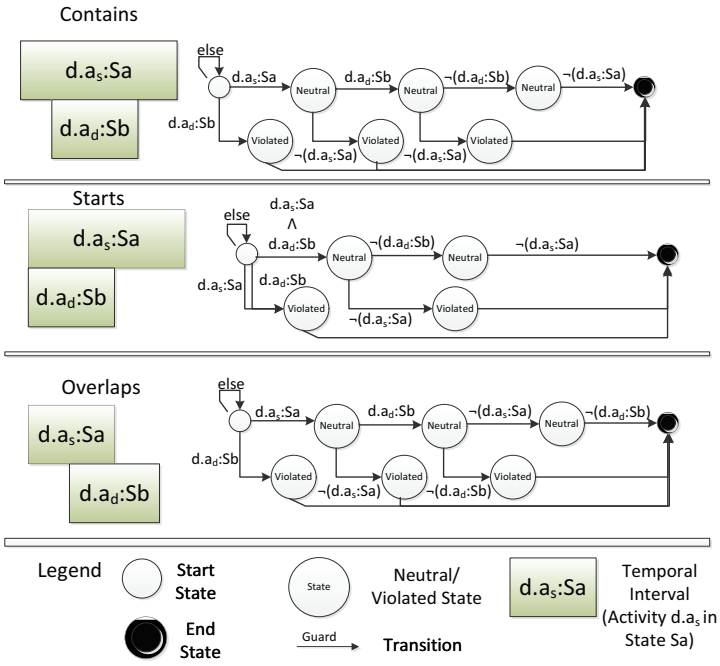


FIGURE 4.9 – Example dependency state machines representing the dependencies contains, starts and overlaps

- $d.a_s : Sa \wedge d.a_d : Sb$ source activity of the dependency $d.a_s$ changes into the state $Sa \in d.a_s.cat$ and destination activity of dependency $a.a_d$ changes into the state $Sb \in d.a_d.cat$
- $\neg(d.a_s : Sa \wedge d.a_d : Sb)$ source activity of the dependency $d.a_s$ changes into any successor state of $Sa \in d.a_s.cat$ and destination activity of dependency $a.a_d$ changes into any successor state of $Sb \in d.a_d.cat$
- $\neg(d.a_s : Sa) \wedge d.a_d : Sb$ source activity of the dependency $d.a_s$ changes into any successor state of $Sa \in d.a_s.cat$ and destination activity of dependency $a.a_d$ changes into state $Sb \in d.a_d.cat$
- $d.a_s : Sa \wedge \neg(d.a_d : Sb)$ source activity of the dependency $d.a_s$ changes into state $Sa \in d.a_s.cat$ and destination activity of dependency $a.a_d$ changes into any successor state of $Sb \in d.a_d.cat$
- *else* any other state change of source activity of the dependency $d.a_s$ or destination activity of dependency $a.a_d$

The main differences to a finite state machine are that there is only one end state and a transition function is defined supporting logic expressions with a special semantic related to our proposed model. These semantics take automatically into account the activity and states associated with a dependency (cf. Definition 3). We illustrate in Figure 4.9 examples for the dependency state machines for the dependencies contains, starts and overlaps (see Appendix C for the complete list of dependency state machines).

We mentioned before that Allen's temporal interval logic also permits combinations of temporal interval relationships (e.g. starts OR meets). These can be also represented using our proposed dependency finite state machine representation. The finite state machines for combinations of dependencies need not to be created manually. The finite state machine representation of the basic interval relations can be merged automatically to a finite state machine representing the combinations of temporal interval relationships.

Algorithm for Detecting Violation of Temporal Dependencies by State Changes

We describe in Algorithm 3 how violated dependencies can be detected, so that they can be highlighted to the user. The user may then perform appropriate actions to deal with this issue. The algorithm takes as input a state change and has as output a list of violated dependencies by the state change.

Algorithm 1: Detect violation of dependencies when executing activities

input : State Change SC
output: A set V of violated dependencies

```
dependencylist ← GetAllDependencies(SC)
for i ← 0 to dependencylist.size - 1 do
    CheckDependency(dependencylist[i],SC);
    if GetState(dependencylist[i] == violated) then
        V ← dependencylist[i]
return V
```

The algorithm gets all dependencies related to the activities occurring in the state change ($GetAllDependencies(SC)$). It then inputs the state change into the finite state machine representation of the dependencies ($CheckDependency(dependencylist[i], SC)$). This results either in "Neutral" or "Violated" state of a dependency. All dependencies in state "Violated" are returned.

Example

We now present an example for detecting violation of dependencies based on the concepts described before. We assume two activities "A" and "B" based on the simple activity type in Figure 4.1 and a dependency "overlaps" between the state "Execute" of activity "A" ($d.a_s : Execute$) and state "Execute" of activity "B" ($d.a_d : Execute$). The corresponding dependency state machine can be found in Figure 4.9. Initially, both activities are in state "Plan". If now activity "B" changes into state "Execute" then the algorithm determines that there is one dependency and inputs the state change into the dependency. The dependency state machine transits given this condition into the state "Violate".

Performance Considerations

Similar to the verification procedure, the execution procedure needs to have an adequate performance. We expect that state changes of activities happen more often than changes to the structure of the model (e.g. adding activities or dependencies). We assume that the finite state machine representations of the dependencies are given. This means Algorithm 3 determines the performance of our approach. The function *GetAll Dependencies* iterates the list of dependencies associated with a model to find all dependencies associated with the state change SC. The complexity of this function depends on the size of the dependency list N and the number of state changes (L) in SC. Thus the complexity for this function is $O(N * L)$. The main part of the algorithm depends on the number of dependencies M associated with the activities. There is one for-loop determining the performance. This for-loop is exactly performed M times and thus the complexity of the algorithm is linear ($O(M)$). The function *CheckDependency* has a complexity of $O(1)$), because it only determines from the current state of the dependency the next state. In total the algorithm has a complexity of $O(N * L + M)$. This is approximately linear to the size of the list of all dependencies N. Thus, we conclude that the performance is sufficient for our scenario, where we expect that state changes of activities happen very often.

4.4.3 Unsynchronized Dependencies

We introduce here a solution to a special case, where it is not possible to decide if a dependency is violated or not. This means there is a deviation from the model, but this needs to be detected differently. An example for this problem is illustrated in Figure 4.10. There are two activities "A" and "B" with the same lifecycle. A dependency is established between the state "Beta" of activity "A" and the state "Beta" of activity "'B". If activity "A" changes into state "Beta" and activity "B" changes into state "Alpha" then it is not possible to decide if the dependency is violated or not, because the state "Beta" of activity "B" cannot be reached anymore. Thus the dependency state machine can never reach its end state. The user should be informed about this problem, so that appropriate actions similar to the ones of violated dependencies can be taken. This means for every state change it needs to be checked if it leads to unsynchronized dependencies.

Definition 8 *An unsynchronized dependency d with the finite state machine representation DST is defined as* $CT + FT \rightarrow DST.s \neq DST.e$*, where*
- *CT is the current trace*
- *FT is any future trace*

This means given the current trace there is no possible future trace where a dependency state machine can reach its end state.

Unsynchronized dependencies lead to a case where it is not possible to decide if they are violated or neutral, i.e. there can be a problem. The reason is that we allow alternative states in the activity type of an activity, e.g. an activity can be alternatively in state "Fail", "Finish" or "Cancel". We argue that this is necessary, because it cannot be predicted in which state an activity will be and it should be possible to select from alternatives.

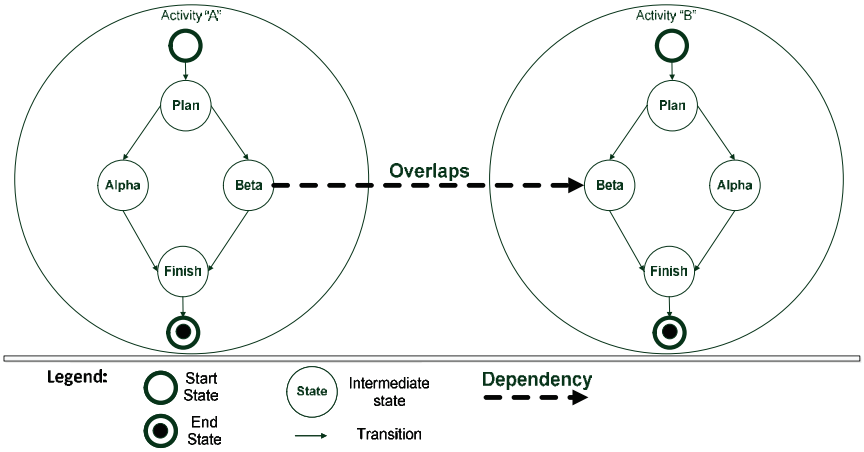


FIGURE 4.10 – Example for a model that may lead to an unsynchronized dependency

In the next paragraph, we describe an algorithm for detecting unsynchronized dependencies when a state change occurs. Afterwards, we give an example of the algorithm. Finally, we evaluate the performance of the algorithm, because it needs to be executed every time a state change occurs.

Algorithm for Detecting Unsynchronized Dependencies

The goal of the algorithm is as follows : Given a state change and the dependencies related to the activities changing their state, find a future trace FT, so that the dependency state machines can reach their end state. If such a trace cannot be found then this should be highlighted to the user as an unsynchronized dependency. The lifecycle of an activity and the dependencies can be represented as a graph (see Figure 4.10). The future trace can be determined by using graph analyses, more precisely by performing an adapted depth search algorithm. In Algorithm 2, we describe how unsynchronized dependencies can be detected when performing a state change. Each state change can lead to unsynchronized dependencies connected to the activities changing their state. The algorithm performs for each dependency associated with an activity changing the state the following check :

- Check for the source activity $d.a_s$ of a dependency d if from the current state $d.a_s.cs$ there are only paths through the $d.s_s$ source state of the activity of the dependency. If there are only paths involving $d.s_s$ and there is no paths of the destination activity $d.a_d$ of the dependency d through the destination state $d.s_d$ of the dependency d then the dependency d is unsynchronized.
- Check for the destination activity $d.a_d$ of a dependency d if from the current state $d.a_d.cs$ there are only paths through the $d.s_d$ destination state of the dependency. If there are only paths involving $d.s_d$ and there is no paths of the destination activity $d.a_s$ of the dependency d through the destination state $d.s_s$ of the dependency d then the dependency d is unsynchronized.

Algorithm 3 describes how the number of alternative paths from one activity state to another activity state can be determined. It uses a variant of depth-search to determine the number of paths from one activity state to another activity state based on the activity

Algorithm 2: Detect Unsynchronized Dependencies when Executing Activities

```
input  : State Change SC
output: A set U of unsynchronized dependencies
dependencylist ← GetAllDependencies(SC)
for i ← 0 to dependencylist.size - 1 do
    d ← dependencylist[i]
    numPathsToDependencySourceActivityState = GetNumPaths(d.a_s.cs,d.s_s)
    numPathsToSourceActivityEndState = GetNumPaths(d.a_s.cs,d.a_s.cat.se)
    numPathsToDependencyDestinationActivityState = GetNumPaths(d.a_d.cs,d.s_d)
    numPathsToDestinationActivityEndState = GetNumPaths(d.a_d.cs,d.a_d.cat.se)
    // Check if dependency is reachable from the current state of the source activity
    if numPathsToDependencySourceActivityState >0 then
        if numPathsToDependencySourceActivityState
        ==numPathsToSourceActivityEndState then
            // Reachable from current state of source activity of dependency
            if numPathsToDependencyDestinationActivityState ==0 then
                // Not reachable from current state of destination activity of
                dependency
                U ← d
    if numPathsToDependencyDestinationActivityState >0 then
        if numPathsToDependencyDestinationActivityState
        ==numPathsToDestinationActivityEndState then
            // Reachable from current state of destination activity of dependency
            if numPathsToDependencySourceActivityState ==0 then
                // Not reachable from current state of source activity of dependency
                U ← d
return U
```

type of an activity. It is required that the activity type does not contain cycles, which we excluded by definition (cf. Definition 1). Similarly to depth-search, it processes all nodes in a graph that it has not visited. It starts from the node st_1 and finds all different combinations of nodes (different paths) to reach the node st_2 in a graph. Since it marks already visited paths, it is not possible that the same path is counted twice.

Example

We describe an example based on the original problem illustrated in Figure 4.10. We will discuss two cases :

1. No unsynchronized dependency
2. Unsynchronized dependency

The first case can occur, when activity "A" changes into state "Alpha" and activity "'B" changes into state "Alpha". It can also occur, when activity "A" changes into state "Beta" and afterwards activity "B" changes into state "Beta".

Algorithm 3: GetNumPaths : Determine the number of paths from one state of the activity to another state of the activity

```
input  : Activity a, determine paths from state st1 to st2
output: Number of paths from st1 to st2

numOfPaths = 0
if st1==st2 then
    return 1
Stack.put(st1)
while Stack.size>0 do
    CurrentState = Stack.fetch()
    if CurrentState == st2 then
        if PathContainsUnvisitedStates(CurrentPath)==true then
            numOfPaths = numOfPaths + 1
        else
            return numOfPaths
        MarkAllStatesInPathAsVisited(CurrentPath)
    CurrentPath.add(CurrentState)
    // Fill the Stack
    Stack.addItems(GetStateSuccessor(CurrentState,a.cat))
return numOfPaths
```

The second case occurs, when activity "A" changes into state "Beta" and activity "B" changes into state "Alpha". The algorithm checks for every dependency connected with activity "B" if there are unsynchronized dependencies. There is one dependency in the example. It checks first if the source activity state of the dependency ("Beta") is reachable from the current state ("Beta"). It is reachable, because the current state of activity "A" is the source state of the dependency. This also means it is the only path to the end state. Thus, the dependency cannot be "avoided". It checks then if the dependency is reachable from the current state of activity "B". The current state of activity "B" is "Alpha" and there is no path from the current state to the destination state ("Beta") of the destination activity "B" of the dependency. This means the dependency is unsynchronized.

Performance Considerations

We identified two factors influencing the performance of the algorithm. The first one is the number of dependencies N associated with the activities changing their state (L). We have previously explained that this leads to a complexity of $O(N * L)$. The second factor is the function $GetNumPaths$, which is a variant of depth-search. Depth-Search has a complexity of $O(V + E)$, whereby V is number of activity states and E is the number of transitions between activity states. This function is called four times and thus the complexity for the second factor is $O(4 * (V + E)) = O(V + E)$. The total complexity is $O(N * L * (V + E))$. This is approximately linear complexity and thus acceptable for our scenario.

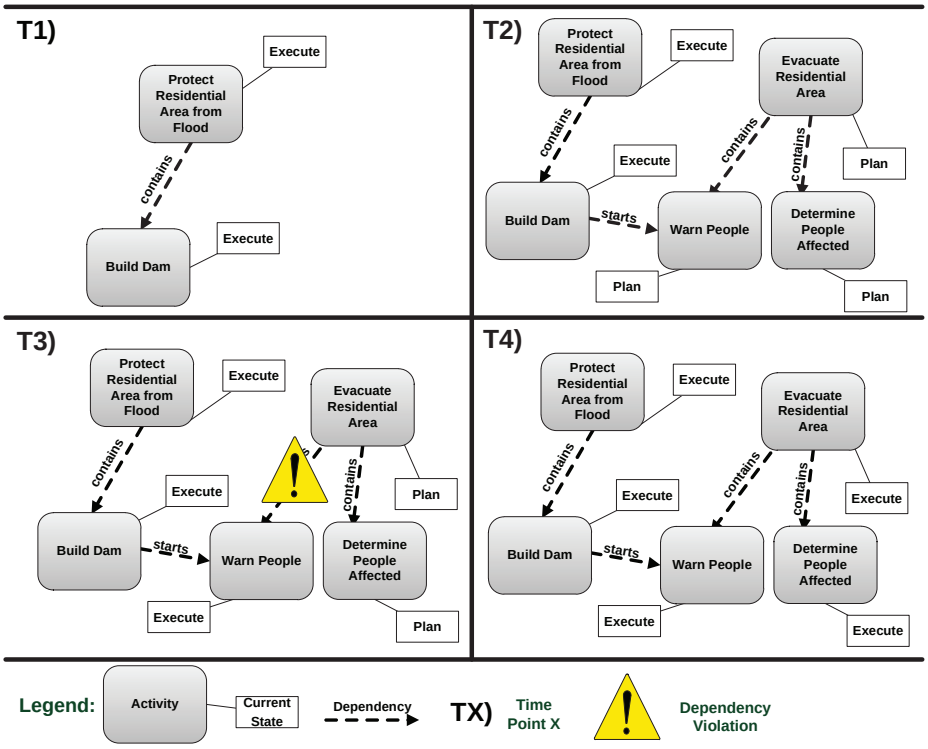


FIGURE 4.11 – Example for the evolution of a model during a disaster response in four steps

4.4.4 Summary

We presented in this section how deviations from the model and how activities are executed can be detected. We introduced the notion of a state change of one or more activities and the trace, which contains all the state changes that took place of the activities in the model. This trace is utilized to detect violated and unsynchronized dependencies. These dependencies indicate that there has been a deviation from the model. Deviations can occur, because the model cannot be always updated by users in an instant to reflect shifting goals. Particularly, it can be difficult in dynamic situations to anticipate these shifts. These deviations can be highlighted to the user and appropriate actions can be taken.

4.5 Example

The goal of this section is to understand when and how the concepts previously described are applied by a user to coordinate activities in dynamic situations. We utilize the motivational disaster response example introduced in chapter two. Figure 4.11 illustrates the evolution of a model of a disaster response in four steps. All activities in this example are based on the activity type illustrated in Figure 5.6.

In the first step, the user models the activities “Protect Residential Area from Flood” and “Build Dam”. The user changes them into state “Plan”. There are no violated or un-

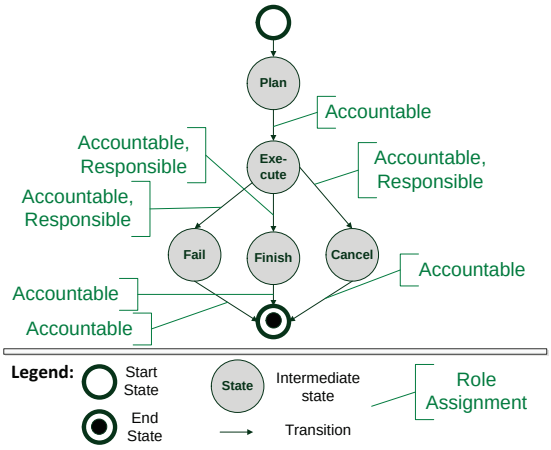


FIGURE 4.12 – Activity type used in this example

synchronized dependencies, because there are no dependencies modeled. The user creates a dependency "contains" between the state "Execute" of the activities. This means that as long as "Protect Residential Area from Flood" is executed, the activity "Build Dam" can be executed. After adding the dependency, the model is verified. The model is error-free. The user then changes the activity "Protect Residential Area from Flood" into state "Execute". There are no violated or unsynchronized dependencies.

In the second step, the user models further activities : "Warn People", "Evacuation of Residential Area" and "Determine People Affected". The user changes them into state "Plan". There are no violated or unsynchronized dependencies. The user establishes a dependency "contains" between the state "Execute" of the activity "Evacuation of Residential Area" and the state "Execute" of activity "Warn People". The same type of dependency is established between the state "Execute" of the activity "Evacuation of Residential Area" and the state "Execute" of activity "Determine People Affected". Another dependency "starts" is established between the activity "Build Dam" in state "Fail" and activity "Warn People" in state "Execute". This means when building a dam fails then the people are warned that the area is going to be executed. The addition of these dependencies does not introduce errors in the model.

In the third step, the user changes the activity "Build Dam" into state "Fail" and the activity "Warn People" into "Execute". The activity "Protect Residential Area" is also changed into state "Fail". All state changes do not lead to unsynchronized dependencies. However, the dependency "contains" between state "Execute" of the activity "Evacuation of Residential Area" and the state "Execute" of "Warn People" is violated, because the activity "Evacuation of Residential Area" is still in state "Plan". This is highlighted to the user.

In the fourth step, the user decides to resolve this issue by changing the activity "Evacuation of Residential Area" into state "Execute" and afterwards changing activity "Determine People Affected" into state "Execute". The violated dependency is marked by the user as resolved and is not highlighted anymore.

4.6 Related Work

This section researches selected related work to concepts of the framework. Kafeza et al. use Allen's interval logic to verify a schedule of activities [KK00]. The schedule is based on quantitative definition of time and it is fixed. This is not suitable for our scenario, because using quantitative definition of time requires more modeling and planning effort. Furthermore, these times are subject to continuous change and thus are very difficult to update in a dynamic situation. Lu et al. use Allen's interval logic to describe rules to create variants of business process models [LSPG06].

Raposo et al. use Allen's temporal interval relationships to describe synchronization points between fixed workflows that are enforced by a system [RMR00]. These synchronization points are also enforced and it is not possible to deviate from the model. The problem of unsynchronized dependencies is addressed by using timeouts. This is a break with Allen's temporal interval relationships, because they are qualitative and a timeout quantitative. For example, if an activity related via a dependency to another activity is not executed after 15 seconds then an exception is raised. Furthermore, their mechanism would also affect dependencies, which are not unsynchronized, but where activity execution has taken longer than the time-out permits. This is not acceptable in our case. We showed in this chapter how the detection of unsynchronized dependencies can be solved qualitatively without affecting dependencies which are not unsynchronized. The authors use Petri nets to describe the temporal interval relationships. This means their model is fixed and difficult to modify once it is executed [EKR95]. It can thus not deal with a dynamic situation. Furthermore, the verification procedure evaluates Petri net properties (e.g. deadlock or livelock) and not temporal properties. The computational complexity of this verification procedure can be exponential in the general case [Mur89].

Leymann et al. describe how constraints between activities can be defined using Allen's temporal interval relationships [LUW10]. They do not support an activity lifecycle, governance roles or flexible definition of activities and dependencies. Their work is based on, to some extent, the constraint-based workflow model of the Declare approach (cf. [Pes08]), but uses a different formalism. It is unclear how they detect deviations from the model and how activities are executed.

Timed Petri nets are used to analyze static properties of a concurrent system with quantitative time as constraints on the state of the system [BM83]. Temporal process algebras have been described to investigate the static analysis of actions or services with quantitative time dependencies (e.g. [HNSY92, GG10]). Timed Petri nets and temporal process algebras can be useful for verification, but they do not consider qualitative time aspects sufficiently. Petri nets have known limitations for verification of modified processes already in execution [EKR95, Rei00]. Furthermore, both do not consider at all deviations from what is defined in the model and how activities are executed.

The visualization of rules described in other formalisms seems to be difficult (cf. [Pes08]). Allen's temporal interval relationships have common characteristics, for example, with relationships between activities in project management applications (e.g. the Gantt chart [Jon88]). We assume thus that they can be more easily and more quickly understood by the user. Furthermore, contrary to other formalisms, they make the relations between activities explicit, because they can be visualized along a timeline.

Related work is not able to address the research questions stated in this thesis with respect to coordination of activities in dynamic situations sufficiently. It is either limited with respect to flexible modeling of activities and their dependencies without a priori full specification of the process, it does not provide adequate verification functionality, it does not provide mechanisms to detect deviations from what is defined in the model and how activities are executed or it does not provide the definition of more advanced governance roles as we have it introduced in this chapter.

4.7 Conclusion

We have presented in this chapter a framework for temporal coordination of activities. It addresses the research problem stated in chapter 2 related to coordination of activities in dynamic situations and has the following contributions beyond the state of the art :

- Flexible ad-hoc modeling of activities and temporal dependencies : we proposed a modeling language that allows defining activities and temporal dependencies between them. We do not require a full specification of the model, because such a specification is not available in dynamic situations. Furthermore, we provide a richer description of dependencies than contemporary process-based approaches presented in the state of the art in chapter 3. We expect that the dependencies are easier to understand and visualize than rules, which are expressed in text form. Modeling is the foundation for describing the relations between activities, which is important for their coordination.
- Definition of governance roles associated with activities : governance roles enable the user to describe accountability and responsibility of an activity. For example, it is possible to define who can cancel an activity. They represent a generic concept to govern activities, because they describe who can change from one state to another state of an activity. Thus, the concept is not limited to the disaster response domain.
- Ensuring correctness of the model ad-hoc : we defined a procedure to detect errors in a model. The first step of this procedure is to translate the model into a temporal constraint network. In the second step, we adapt the path consistency algorithm proposed by Allen [All83] for checking satisfiability in a temporal constraint network. This means we can rely on well-understood formalisms where the limitations are known. We argue that our verification procedure can be performed in reasonable predictable time. This is necessary, because the model may change often depending on the situation. The user can use this information to ensure a correct model.
- Detecting deviations from what is defined in the model and how activities are executed : deviations are caused by shifting goals and reassessment of activities. This has to take into account their dependencies. We identified two different types of deviations : violation of temporal dependencies and unsynchronized dependencies. Deviations indicate that the dependencies have not been taken into account, when activities and their relations have been reassessed. The reason is that it is not always possible to update the dependencies in the model in time to deal with shifting goals, because the dynamics of the situation do not permit it. Dependencies are not enforced, because this would constrain the user too much. It is also problematic to

enforce dependencies out of the scope of a system (cf. also chapter 3). The detection of deviations can be performed in reasonable and predictable time. They can be highlighted to the users so that they can take them into account. For example, they can communicate with the stakeholders of activities or initiate counter-actions.

This framework addresses the first research question on how to design information system support for coordination of activities in dynamic situations. It needs to be integrated with an approach addressing the inter-organizational dimension described in the second research question as we have explained in chapter 2.

Chapitre 5

Inter-Organizational Coordination

Contents

5.1 Introduction 92

5.2 Sharing Activities between Organizations 92

5.2.1 Activity Workspace 93

5.2.2 Example for Sharing of Activities 94

5.2.3 Replicating Shared Activities in Different Activity Workspaces 95

5.2.4 Summary 95

5.3 Verifying an Activity Workspace Containing Shared Activities 96

5.3.1 Problem Statement 96

5.3.2 Requirements for Verification 97

5.3.3 Discussion of Requirements 99

5.3.4 Protocol for Verification of an AW Containing Shared Activities 100

5.3.5 Summary 101

5.4 Detecting Deviations from the Model and How Shared Activities Are Executed 102

5.4.1 Tracking the Execution of Shared Activities 102

5.4.2 Conflicting State Changes of the Same Shared Activity 103

5.4.3 Conflicting Views on the Causal Order of State Changes of Dependent Shared Activities 110

5.4.4 Detecting Violation of Temporal Dependencies Involving Shared Activities 116

5.4.5 Detecting Unsynchronized Dependencies Involving Shared Activities 118

5.4.6 Incomplete Propagation of State Changes of a Shared Activity 120

5.4.7 Summary 122

5.5 Related Work 123

5.6 Conclusion 124

5.1 Introduction

Coordination by people of different organizations is another important aspect of the research problems stated in chapter 2 (cf. [Qua83, Dra03]). More particularly, we identified the problem of how people can provide information about what they are doing to people of other organizations. This needs to take into account privacy, regulatory, strategic or other reasons. This means not all information about what an organization is doing can be provided to everybody. We have explained in chapter 3 that contemporary approaches require disclosure of too much information about what needs to be coordinated or provide only little support for coordination. This is too restrictive with respect to our motivational example illustrated in chapter 2. Similarly, existing tools for coordination used by the disaster managers provide only limited capabilities to relate activities (cf. chapter 2). This information is hidden in a large pile of unrelated messages exchanged between different organizations.

We propose to leverage the framework of chapter 4 by extending it to the inter-organizational level. The main idea is that every organization can have a partially shared common view on activities and their dependencies given the restriction that not everything can be shared. We go beyond the state of the art by (1) allowing sharing selected activities between selected organizations and (2) extending the framework for coordination proposed in chapter 4 to incorporate shared activities.

We introduce the concept of sharing activities between organizations in section 5.2. We demonstrate in section 5.3 how a model, containing shared activities with dependencies established between activities in different organizations, can be verified. Deviations from what is defined in the model and how shared activities are executed can be detected and highlighted to the participants of different organizations (cf. section 5.4). This means shifting goals involving activities of other organizations can be taken into account. Related work to the mechanisms described in this chapter is presented in section 5.5. Finally, we summarize the results in section 5.6.

5.2 Sharing Activities between Organizations

We have seen in chapter 2 that the sharing of information between organizations is restricted. Given the motivational example in chapter two, the police cannot share information about crime investigations with the fire brigade. However, the fire brigade can share information about the progress of building a dam with the police. The police want to start evacuation of the residential area only when building of a dam to protect it fails. This means once information about what is going on has been shared, this information needs to be related to other activities by people in different organizations.

Every organization needs to keep control over what they want to share and with whom. A centralized solution where everything needs to be shared with a central party is thus not possible in our scenario. However, in order to coordinate on the inter-organizational level selected information needs to be shared between selected organizations.

We explain in the following subsections how organizations can manage their activities using their own workspace and how selected activities can be shared with selected

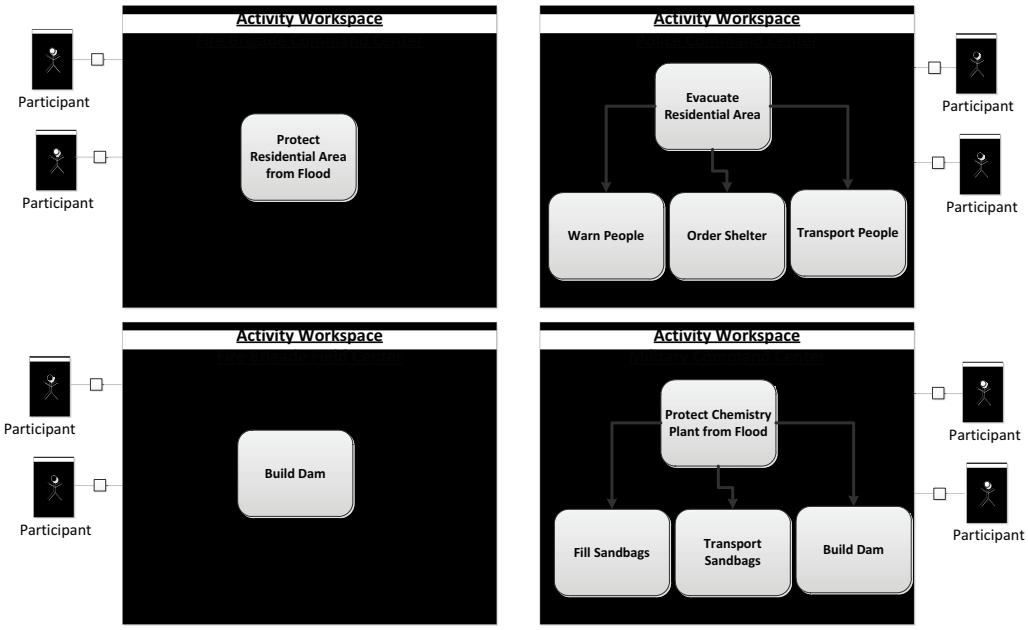


FIGURE 5.1 – Examples for activity workspaces

organizations. Shared activities are replicated in the workspaces of the corresponding organizations.

5.2.1 Activity Workspace

An "Activity Workspace" (AW) is used by participants of an organization to coordinate activities according to the framework presented in the previous section. Each organization can have its own AW. It can keep control over the activities and dependencies in its AW. They are not disclosed to other organizations. An AW provides functionality for defining a model, verifying it, detecting deviations between the model and execution of activities as described in the framework in the previous chapter. It is defined as follows :

Definition 9 *An Activity Workspace (AW) is used by participants for managing the following elements using the framework of chapter 4 :* (AC, D, ATF, U)*, where* AC *is the set of activities (cf. Definition 2), the set of Dependencies (cf. Definition 3), the set of activity types* ATF *(cf. Definition 1) and the set of participants* U*.*

We illustrate in Figure 5.1 four different AWs of the fire brigade command center, the fire fighters in the field, the police and the military. Each of them has different activities and dependencies modeled. However, there needs to be a mechanism to relate activities in the AWs of different organizations. We provide in the next subsection an example how this can be done.

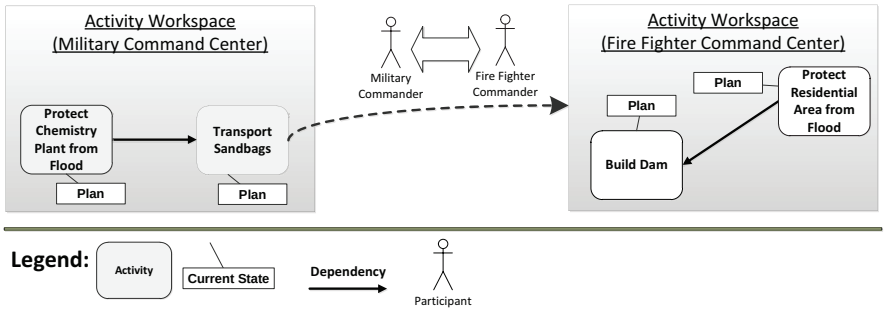


FIGURE 5.2 – Examples for sharing activities

5.2.2 Example for Sharing of Activities

Each organization keeps control over the activities and dependencies modeled in their AW. Activities need to be related to activities of other organizations. People within one organization need to decide which information they want to disclose to other organizations. We explained in chapter 2 that people of different organizations exchange information to coordinate with other people in their social network. For example, they may know people of other organizations from previous disasters, exercises or day-to-day work. This is a well-accepted practice and needs to be supported by the concepts described in this thesis.

Thus, we propose that participants of an AW can share selected activities with participants of AWs of other organizations. This means that any information about the selected activity is exchanged with them, for instance, the current state of an activity. A participant can instruct his AW to send the information about the activity to another AW of the other participant, so that the other AW can highlight it to its participants. This requires that a participant knows the address of another participant, i.e. the name and the AW where he is registered.

We illustrate in Figure 5.2 an example for sharing of activities between people of different organizations. This is based on the motivational example presented in chapter 2. The military provides sandbags to the fire brigade, so that they can build a dam. The military commander shares the activity "Transport Sandbags" with the fire fighter commander in the command center. He knows the fire fighter commander from a previous disaster. The idea is that the fire fighter commander can use this shared activity for coordinating it with other related activities, such as protecting the residential area from a flood.

After an activity has been shared, it can be integrated into the AW. This means that the shared activity can be used as any other activity in the workspace. For instance, dependencies can be established between shared activities and other activities in the AW. We show how the previously shared activity has been integrated by the fire fighter commander in Figure 5.3. A dependency has been established between the shared activity "Transport Sandbags" and the internal activity "Protect Residential Area from Flood".

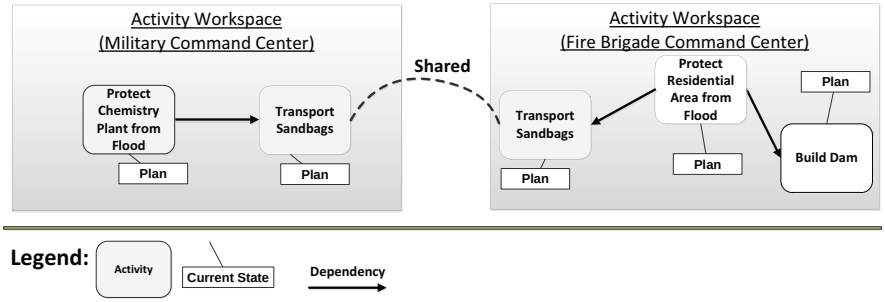


FIGURE 5.3 – Example for integrating a shared activity in an AW

5.2.3 Replicating Shared Activities in Different Activity Workspaces

Integrating a shared activity in a workspace means that they have a representation in the workspace, so that they can be handled as any other activity. Thus, they are replicated in the other workspace. We decided to replicate shared activities in the different workspaces optimistically (cf. [Coh00]). This means any updates to a shared activity, such as state changes, are propagated in an instant to all AWs where the shared activity is replicated. We chose optimistic over pessimistic replication for several reasons. A pessimistic approach would mean to lock the activity or even parts of a model for a period of time in which no state changes can be entered by the user. This is necessary, because a pessimistic approach requires that all replicas confirm an update, before it is applied. This can be time-consuming in dynamic scenarios with less reliable network connections. This would unnecessarily limit the possible interaction with the system (cf. also [GM94]). A pessimistic approach can lead to inaction, because people have to wait until they can provide input or receive input to perform an action. They could not capture and communicate the state of their activities in time. This is contrary to what happens in disaster response management as explained in chapter 2. Thus, a pessimistic approach is not possible in our context.

5.2.4 Summary

We proposed in this section an approach where selected activities can be shared between people of different organizations. These shared activities are integrated in the AWs of different organizations by replicating them optimistically. Pessimistic replication does not adequately address our scenario explained in chapter 2, because it cannot deal with the dynamic situations. Sharing of activities is voluntary, so that the organizations do not need to disclose everything about what they are doing due to privacy, regulatory, strategic or other reasons. More precisely, they can share selected activities with selected organizations, if they perceive it as beneficial. We investigate in the following sections how shared activities replicated in different workspaces can be incorporated in the framework

proposed in the previous chapter.

5.3 Verifying an Activity Workspace Containing Shared Activities

Given the example presented in chapter 2, the military provides sandbags to the fire fighters, so that they can build a dam (cf. also Figure 5.3). The military commander shares the activity "Transport Sandbags" with the fire fighter commander, so that the fire fighter commander can be aware of the state of the activity. The military commander establishes a dependency from the activity "Fill Sandbags" to the activity "Transport Sandbags". The fire fighter commander establishes a dependency from "Transport Sandbags" to "Protect Residential Area from Flood". He now wants to verify the AW containing the shared activity "Transport Sandbags", the activity "Protect Residential Area from Flood" and a dependency between them. The fire fighter commander may want to take into account as well the non-shared activity "Fill Sandbags" of the military and the dependency of the shared activity "Transport Sandbags". The reason is that he wants to make sure that their dependencies established to the shared activity "Transport Sandbags" do not lead to unsatisfiable dependencies in other AWs of other organizations he is working with.

Shared activities can be replicated in different AWs of different organizations. We explained that dependencies can be established between different activities in different workspaces. The model of activities and dependencies in one AW can be correct, but replicated activities are part of several workspaces. They establish inter-dependencies between activities in different AWs. Verification can also consider these inter-dependencies of a replicated activity in a different AW. This is useful to make sure that establishing dependencies to shared activities does not lead to unsatisfiable dependencies in the AWs of other organizations. This implies that we need to specify a protocol for exchanging information about activities and dependencies between different AWs. This protocol is initiated when participants of one AW want to perform verification of their AW containing shared activities.

We stated in chapter two that organizations cannot disclose all information to everybody. However, considering the inter-dependencies to shared activities in different AWs, as required by verification, is contradictory to this, because this requires disclosing information about them including their dependencies to non-shared activities.

We explain this problem in section 5.3.1 with a more detailed example. We describe the requirements for a verification protocol executed between different AWs in section 5.3.2. In section 5.3.3, we discuss the requirements for a protocol for verification of a model containing shared activities. We design this protocol in section 5.3.4 and summarize in section 5.3.5.

5.3.1 Problem Statement

We illustrate the problem of unsatisfiable dependencies established between shared activities by an example in Figure 5.4. Four AWs are described in this figure : "Alpha", "Beta", "Gamma" and "Delta".

AW "Alpha" contains the activities "A" and "B". A dependency "overlaps" is established between them. Activity "B" has been shared and integrated in AW "Beta". AW "Beta" contains also activity "C". A dependency "overlaps" is described between shared activity "B" and activity "C". Activity "C" also has been shared with AW "Delta". AW "Delta" contains also activity "D". A dependency "overlaps" is established between shared activity "C" and activity "D". Activity "D" and activity "A" have been shared with AW "Gamma". A dependency "overlaps" is established between shared activity "A" and shared activity "D".

Each AW (cf. Definition 9) individually contains a correct model (cf. verification procedure in section 4.3) in this example. However, dependencies can be established between shared activities in different AWs. Thus, we need to consider the dependencies established in different AWs (see Definition 10).

Definition 10 *The global model* $GAW = \cup AW_i, i = 1, .., n$ *is composed of all activities and dependencies in all AWs where* n *is the number of AWs*

Using the verification algorithm proposed by our framework in the previous chapter, the global model is erroneous in this example. Thus, we analyze the requirements for a protocol for verification of a global model (see Definition 10).

5.3.2 Requirements for Verification

Verification of AWs containing shared activities needs to take into account the restriction that not all information can be shared with everybody. We identified in this context the following requirements to consider when verifying an AW_i containing shared activities :

- Making only necessary information available : verification may require that information about activities and dependencies in other AWs needs to be made available to AW_i which has not been shared before.
- Limiting the scope of verification : only relevant activities and dependencies in AWs of other collaborating organizations should be considered for verification of AW_i.

We discuss them in more detail in the subsequent paragraphs.

Making Only Necessary Information Available

Verification can also take into account the dependencies of the shared activities in AW_i to other non-shared activities in other AWs, where the shared activities of AW_i have been replicated. For example, in Figure 5.4, Activity Workspace "Alpha" contains the shared activity "A", which has also a dependency in Activity Workspace "Beta" to different non-shared activities (e.g. activity "C"') with Activity Workspace "Alpha". This means the previously non-shared information, such as non-shared activities or dependencies, needs to be made available to AW_i.

Limiting the Scope of Verification

Dependencies can be established between shared activities of different AWs. For example, in Figure 5.4, there are only dependencies between shared activities. Activity Workspace

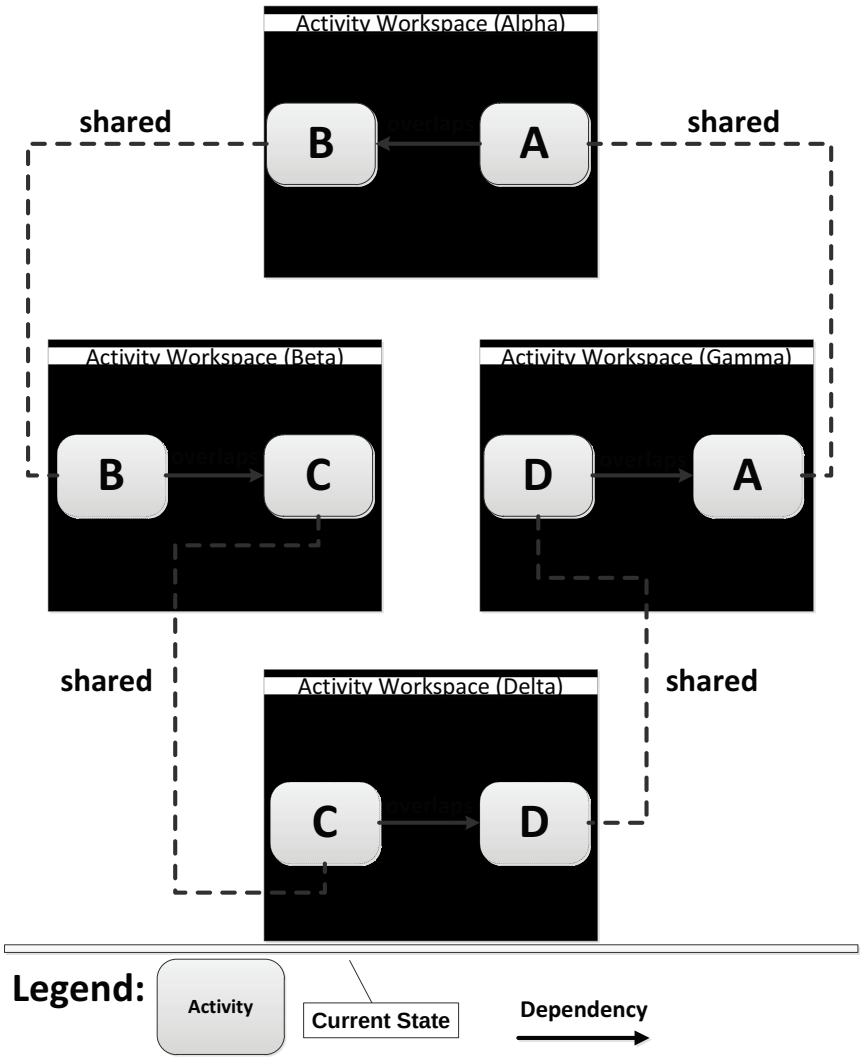


FIGURE 5.4 – Example for verification of AWs containing shared activities

"Alpha" has dependencies to shared activities "A" and "B" replicated also in Activity Workspace "Beta" and Activity Workspace "Gamma".

We say that Activity Workspace "Alpha" is directly connected to Activity Workspace "Beta" and Activity Workspace "Gamma", because it contains replicated shared activities from these workspaces. However, in Activity Workspace "Beta" there is also a dependency between the shared activity "B" and the shared activity "C", which is replicated in Activity Workspace "Delta", but not in Activity Workspace "Alpha". We say that Activity Workspace "Delta" is indirectly connected with Activity Workspace "Alpha", because they have no common shared activities, but there are dependencies between their shared activities in another workspace (Activity Workspace "Gamma").

Verification may consider only directly connected AWs or also indirectly connected AWs. For example, from the perspective of Activity Workspace "Alpha", the Activity Workspaces "Beta" and "Gamma" are directly connected. Activity Workspace "Delta" is indirectly connected.

5.3.3 Discussion of Requirements

Participants of one organization can collaborate with participants of other organizations. We assume that these organizations want to work to together. They may establish independently dependencies to shared activities and this can cause an erroneous definition of dependencies between activities of different organizations. Thus, we argue that they should include the dependencies to shared activities in other AWs for verification. Going back to the motivational example in chapter two, the military wants to work together with the fire fighters so that they can respond collectively to a flood. They can be made aware that the dependencies related to their collaboration are erroneously defined. However, this means that information about the models in different AWs sharing activities need to be exchanged. We need to address in this context the two requirements mentioned before, because they influence what needs to be shared between organizations for verification.

We assume that participants want to verify the activities and dependencies in their AW_i containing shared activities. We discuss our decisions made with respect to the requirements for verification of shared activities.

Firstly, we argue that the scope of the verification should only take into account directly connected AWs (cf. section 5.3.2). The reason is that the organization maintaining AW_i only works directly together with the organizations it has activities shared with. It is less beneficial if also the activities of indirectly connected AWs of other organizations are taken into account. In the motivational example of chapter two, the military works together with the fire brigade, because they provide sandbags to them. However, the military does not work together with the police. The activities of the police are only indirectly related to the activities of the military via the activities of the fire brigade. If transporting sandbags of the military fails then building of a dam by the fire fighters may fail. Thus, the evacuation of the residential area by the police is started. The military wants to verify now its workspace containing shared activities with the fire fighters. Let us assume they take also into account the activities of the police and the verification procedure detects an erroneous definition of dependencies. It would be difficult to resolve them, because the

military would have to communicate with the police via the fire brigade. This can be very time consuming in a more complex scenario involving many organizations. We also argue that erroneous definition of dependencies should not affect the work of organizations that are not directly collaborating with each other.

Secondly, the information needed for verification should not require to share the current state of a non-shared activity, the name or any other further information attached to a non-shared activity. The disclosure of additional information is limited. For example, a participant of the activity workspace "Alpha" in the example illustrated in Figure 5.4 wants to verify the model defined in this workspace, but also taking into account the directly connected Activity Workspace "Beta" and Activity Workspace "Gamma". Verification requires the unique identifier of activity "C" and activity "D", their activity types without governance roles, as well as the dependencies from these activities to the shared activities "A" and "B". We describe the information required for verification by an AW from another AW as the relevant snapshot (cf. Definition 11). The unique identifiers are needed to merge the relevant snapshots of different AWs in order to verify them (cf. for all elements of an AW Definition 9). Shared activities should only appear once in this merged snapshot and they can be identified via their unique identifier (cf. Definition 2).

Definition 11 *The relevant snapshot of an AW is defined as :* $SN = \{\{UAI \times AT\}, D\}$. *The information needed for verification is the set of unique activity identifiers* UAI*, their assigned activity type in the set of activity types without governance roles* AT *and the dependencies* D*.*

It should be noted that the relevant snapshot does not need to consider all activities and dependencies of an AW. It is sufficient to consider the activities (in-)directly dependent on shared activities. Thus, the relevant snapshot does not need to contain all elements of an activity workspace (cf. Definition 9 and Definition 10).

5.3.4 Protocol for Verification of an AW Containing Shared Activities

The protocol presented in this subsection takes into account the design requirements explained before : only the relevant snapshots (cf. Definition 11) of AWs sharing activities with AW_i are considered. The underlying idea is that participants of AW_i can be made aware of any errors in the model of their AW taking into account the dependencies of shared activities in another AW. However, the protocol can only consider the activities and dependencies in other AWs at a certain point in time. The activities and dependencies in other AWs may change after the protocol has been executed. In this case, the protocol has to be executed by AW_i again. The protocol is inspired from the snapshot algorithm in distributed systems [CL85]. However, there are important differences. A snapshot algorithm considers the messages exchanged between processes (comparable with an AW) influencing the state of a process. The protocol consists of the following steps :

1. AW_i verifies that its relevant snapshot SN_i is consistent.
2. AW_i adds its relevant snapshot SN_i to a global snapshot $GC = SN_i$

3. AW_i requests a relevant snapshot SN from all AWs it has activities shared with ($AW_{allshared}$). We assume that this information is available.
4. $\forall AW_x \in AW_{allshared}$ send their relevant snapshot SN_x to AW_i
5. As soon as AW_i receives a snapshot SN_x of AW_x it merges it into the global snapshot $GSN = GSN \cup SN_x$. It can be compared to a merger as described in Definition 10, but it only contains necessary information for verification. Shared activities are only represented once in this global snapshot. They can be identified via their unique identifiers. All dependencies established to them can be established to the unique representation of the shared activity.
6. If all relevant snapshots of all AWs it has activities shared with ($AW_{allshared}$) have been received then it verifies them according to the verification procedure defined in chapter 4.

The main benefit of the verification protocol is that the participants of AW_i can be made aware of the errors in the model taking into account the activities and dependencies of other organizations they are working with. The participants of AW_i can contact the other participants with whom activities have been shared to resolve errors in the models. For example, the fire fighter commander can discuss the problem with the military commander. It is also possible to remove dependencies established between shared activities (cf. section 4.3). Then, they can perform verification of AW_i again without contacting the other participants to check if the errors have been resolved.

5.3.5 Summary

We argued in this section that verification of a model in one AW containing shared activities should also incorporate the dependencies between the shared activities and other activities in other AWs. The reason is that organizations having shared activities are working together and thus should be informed of any errors in their AW containing shared activities. However, we do not consider organizations indirectly working together, because resolving errors with them is difficult with little further benefits.

We designed a protocol for verification of AWs containing shared activities. An AW can initiate this protocol if the dependencies of shared activities have changed. Verification needs also to take into account the dependencies between non-shared activities and shared activities in other AWs. Only the unique identifiers of activities, their activity type without governance roles, and the dependencies, need to be disclosed between organizations working together. This means only the structure of the model is disclosed, but not the meaning of the elements in the model, such as the activity name, the governance roles, or the users. Thus, the disclosure of non-shared information is limited. When an erroneous model is detected in one AW, participants of this AW can contact the participants of other AWs to resolve this problem. The participant can also remove dependencies to shared activities and verify the model again to find if the dependency is the reason for this problem.

The protocol makes participants of one organization aware of any erroneous definition of dependencies taking into account the activities and dependencies of other organizations

with whom they collaborate. They can thus detect misunderstandings about the dependencies between activities in a rather short time, which is not possible with the tools described in chapter 2.

5.4 Detecting Deviations from the Model and How Shared Activities Are Executed

We extend in this section the functionality of the framework for highlighting shifting goals involving shared activities to the user. Revisiting the motivational example of chapter 2, it is now possible that the fire fighter commander can cancel transporting sandbags and the military is notified about this, because it affects the dependency to filling sandbags. The military can now cancel filling the sandbags and use the resources somewhere else. However, conflicts can occur when working with replicated activities. For example, the fire fighter commander declares transportation of sandbags from the military as canceled, but concurrently the military declares the same activity as failed. The military now looks for alternatives for transporting sandbags and may order new units to deliver the sandbags to the fire fighters, although they do not need them anymore. Both have a diverging view on the state of the activity. Another related problem is that we need to deal with the impact of unreliable connections. For example, the fact that building of a dam has failed is communicated to the fire brigade command center, but not to the police, because this information got lost during transmission.

We describe in this section the tracking of the execution of shared activities based on the framework introduced in the previous chapter (cf. section 5.4.1). This means that state changes of shared activities need to be propagated optimistically to all AWs where the shared activity is replicated (cf. section 5.2.3). This can cause conflicts leading to a diverging view on the state of activities or dependencies in different AWs.

We identify two different types of conflicts that need to be handled to ensure eventually a converging view on the activities and dependencies in different AWs. The first conflict, concurrent state changes of the same activity by different participants, is addressed in section 5.4.2. The second conflict, different views on the causal order of state changes of shared activities in different AWs, is addressed in section 5.4.3. We adapt the algorithms of the framework for detecting dependency violation and unsynchronized dependencies in section 5.4.4 and 5.4.5. They have to consider the handling of conflicts mentioned before.

State changes may not be completely propagated to all AWs where a shared activity is replicated. We investigate the impact of this with respect to ensuring a converging view on the state of activities and dependencies in section 5.4.6. We present a protocol for synchronizing missing state changes between AWs to ensure eventually a converging view and discuss the implications of this synchronization mechanism for the user. We summarize the contributions in section 5.4.7.

5.4.1 Tracking the Execution of Shared Activities

State changes of shared activities need to be propagated optimistically to all AWs where the activity is replicated. Every organization is aware of the current state of a

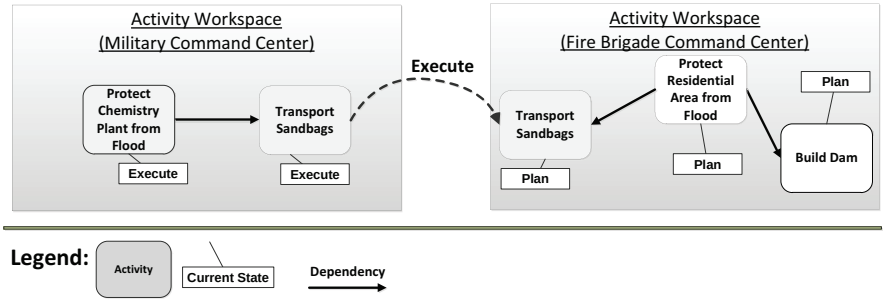


FIGURE 5.5 – Example for propagating a state change of a shared activity to another AW

shared activity eventually. This allows detecting deviations from what is defined in the model of its AW and how activities are executed. According to our framework presented in chapter 4, state changes of activities are tracked in a trace. Each AW has its own model of activities and dependencies, because the organizations are autonomous and only share selected activities. Consequently, each AW has its own trace according to Definition 6.

This also means that there is no common global trace shared among AWs. However, shared activities can change their state. Every AW where this shared activity is replicated needs to know state changes of a shared activity to add them to its own trace. It should be noted that state changes can be initiated from any AW where the shared activity is replicated.

We already described in subsection 5.2.3 that optimistic propagation of state changes seems to be the only possibility to address research questions presented in chapter 2. We assume that it is known with whom an activity has been shared.

We present in Figure 5.5 an example for propagating a state change of a shared activity. There, two AWs of the military and the fire brigade are illustrated. The military has changed the non-shared activity "Protect Chemistry Plant from Flood" to the state "Execute". It also changed the shared activity "Transport Sandbags" to the state "Execute". The state change of the shared activity is propagated to the AW of the fire brigade, where the shared activity "Transport Sandbags" is replicated.

However, optimistic propagation can lead to conflicts and we explain how they can be detected and handled in the subsequent subsections.

5.4.2 Conflicting State Changes of the Same Shared Activity

We introduce in this subsection the problem of conflicting state changes of the same shared activity. A shared activity is replicated in different AWs. Participants in different AWs can set the replicated activity into two different states concurrently. Each AW has now a diverging view on the state of the activity. Based on their own view the participants may perform contradictory actions. The goal of handling this conflict is to reach eventually a converging view on the shared activity to avoid contradictory actions.

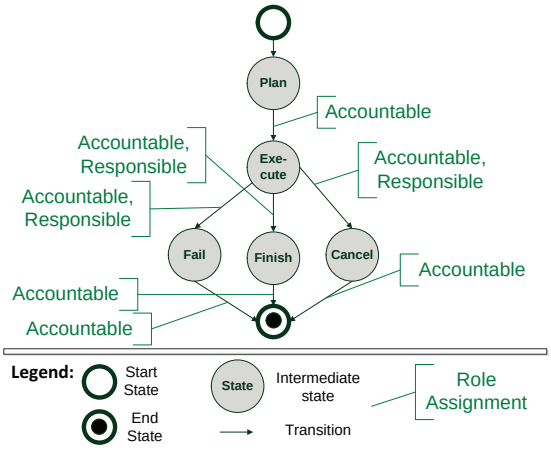


FIGURE 5.6 – Example activity type

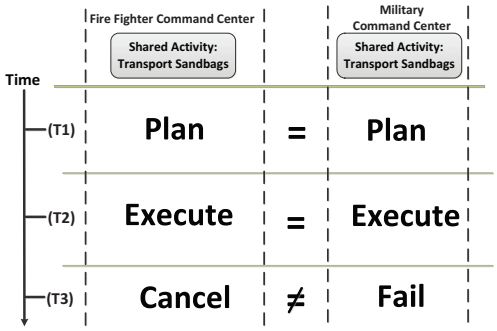


FIGURE 5.7 – Example for detecting conflicts caused by state changes of the shared activity "Transport Sandbags"

Problem Statement

We illustrate the problem via an example activity type illustrated in Figure 5.6. A conflict can occur if a participant of one AW sets an activity based on this activity type to state "Cancel" and a participant of another AW to "Fail" concurrently.

This type of conflict is illustrated in Fig. 5.7. In the first two steps, (T1) and (T2) the shared activity "Transport Sandbags" is changed to state "Plan" and to state "Execute" respectively without any conflict. The activity "Transport Sandbags" is changed by the military commander in the military command center to state "Fail" and by the fire fighter commander in the command center to state "Cancel" in the third step (T3). The conflict can be detected based on the activity type and the trace of state changes of the activity. In the activity type, it is impossible to change from state "Execute" to "Fail" and at the same time from "Execute" to "Cancel". Thus, there is a conflict.

We define this problem formally as follows.

Definition 12 *Conflicting state change trace of an activity in an AW : Let* T_{AW} *be the trace of an AW and* $T_{AW_a} \subseteq T_{AW}$ *the state changes of an activity "a" in the trace of*

the AW. A conflict occurs when : $\exists(s_j \in SC_i \wedge s_k \in SC_l) \rightarrow (s_j.cs = s_k.cs \wedge s_j.ns \neq s_k.ns), SC_i, SC_l \in T_{AW_a} \forall i, j, k, l$. *SC describes a state change according to Definition 5.*

This definition means that there is conflict in the trace of state changes of an activity if there are two or more state changes originating from the same current state of the same activity. It is not possible to transit twice through the same state without causing conflicts in the trace of an activity with the activity type. This is only possible when the activity type has cycles, which we excluded by definition (cf. Definition 1).

The problem here seems to be similar to state machine replication and distributed consensus [Lam98, Sch99]. Replicated state machines represent the states of a replicated service. The goal is fault-tolerance, i.e. one or more of the replica may fail, but still it is possible to recover a correct result based on the non-failed replicas. Different replicated state machines of the same service may have a different state due to faults. The client needs to determine now what the correct state is to resolve this. Consensus protocols determine the correct state by agreeing on the state with all replicated state machines. This requires 2*F+1 replicated state machines, with F describing the maximum number of faulty replica.

However, there are differences to our research problem : firstly, there is not the notion of a correct state. Participants of different AWs set the state of a shared activity, because they have a legitimate reason for it. For example, the military commander sets the activity "Transport Sandbags" to fail, because the truck has an accident. Concurrently, the fire fighter commander set it to cancel, because they do not need the sandbags anymore. This makes it difficult to agree on a "correct" state. Secondly, we do not expect that there will be many replications of a shared activity, so that agreeing on a state as outlined before is possible. Thus, we need a different approach to handle this conflict.

Motivation for Automatic Handling of Conflicting State Change Traces of a Shared Activity

It is possible to detect conflicts caused by concurrent state changes based on Definition 12. These detected conflicts can be highlighted to the users and they can resolve them by agreeing on the same state. However, this takes time and it is not guaranteed that users will eventually handle them. This means that they will continue with a diverging view and this makes coordination difficult. This may even introduce more conflicts. Thus, we propose to resolve conflicts in a state change trace of a shared activity automatically to reduce the burden of the user. This allows avoiding contradictory actions by reducing the time that the conflict is resolved.

Our approach is inspired from an idea of [Lam78]. There, a central authority decides on the correct value. This does not work in our scenario, because every organization is autonomous and controls the activities as well as dependencies in their workspace. We propose alternatively to utilize the governance roles specified by the creator of an activity (cf. Definition 2) to decide which state change should be chosen. For instance, we assume the previous example of the shared activity "Transport Sandbags" (cf. section 5.2.2). The military commander changes the state to "Fail" and the fire fighter commander to "Cancel". Since the fire fighter commander has the accountability for the activity, he is higher in

the role hierarchy than the military commander who is responsible. This means the final state in both AWs is "Cancel" for the shared activity "Transport Sandbags".

The main rationale behind this is that the strategic intent of the stakeholder of the activity can be preserved. The one higher in the role hierarchy should have a better strategic overview with respect to the activity and its relations with other activities. Furthermore, it allows flexibility on who can decide what and it is transparent for everyone involved. Each AW applies the same algorithm to handle the conflict when receiving a state change. This means no extra communication is needed to reach a converging view, because the algorithm ensures the same result given that state changes are eventually received by all AWs where the shared activity is replicated. We describe the algorithm in more detail in the following paragraph.

Algorithm

Algorithm 4 describes how the conflict can be handled based on the strategic intent and governance roles. The algorithm handles one conflict in the state change trace of a shared activity. It is performed every time a state change is added to the trace. It sets the current state of the shared activity and marks diverging state changes as ignored. This is useful for the algorithms presented later to determine if a dependency is violated or unsynchronized. One central point of this is the definition of a valid state change sequence (see Definition 13).

Definition 13 *A valid sequence of state changes of an activity "a" is defined as a* $VS = \{SC_1, SC_i, .., SC_n\} \subseteq T_{AW_a}$ *where*

- SC_1 *(cf. Definition 5) is the state change from the start state of the activity type of activity "a"*
- SC_n *(cf. Definition 5) is the state change to the current state of activity "a"*
- *Given the state changes* $SC_i, i = 0, .., n$ *(cf. Definition 5) and the activity type of activity a all states on a path between the start state and the current state have occurred (i.e. they have been part of state changes) and have not been marked as ignore by Algorithm 4.*

There are two situations when this is not the case : (1) the conflict has not yet been handled in all AWs or (2) not all state changes have been received yet. The first case leads to a situation where based on the conflicting state change further state changes have been performed (cf. example in the next paragraph), i.e. the view on the activity diverges further, because of the original conflicting state change. The other diverging state changes should not be considered.

Algorithm 4 implies that governance roles need to form a hierarchy (e.g. "accountable" > "responsible" > "consulted"). If this is not the case then it is possible to notify the different stakeholders and they have to resolve the conflict manually.

In the beginning of the algorithm, the markings of state changes in the trace are removed for initializing the algorithm ($RemoveMarkingFromTrace$). The markings determine if a state change should be ignored or not. An ignored state change is one that is part of the conflict or a diverging state change. This is relevant for the algorithms presented

Algorithm 4: Handling one conflict in the trace of state changes of the one shared activity

input : Trace T_{AW_a} of activity a in AW with conflicting state changes
output: Setting of the current state as defined by governance roles and mark all other state changes as ignore

```
RemoveMarkingFromTrace (T_AWa);
σ = GetValidStateChangeSequencesFromTrace(T_AWa);
conflictingstatechangeslist ← GetConflictingStateChanges(σ);
chosenStateChange ← conflictingstatechangeslist[0];
for i ← 0 to conflictingstatechangeslist.size - 1 do
    if chosenStateChange.role < conflictingstatechanges[i].role then
        chosenStateChange ← conflictingstatechangeslist[i];
MarkAllOtherStateChangesIgnore(chosenStateChange,conflictingstatechangeslist);

SetCurrentState(GetValidStateChangeSequencesFromTrace (σ)[lastentry]);
```

later for detecting deviations from what is defined in the model and how activities are executed. Algorithm 4 requires as input a valid sequence of state changes in the trace of a shared activity according to Definition 13. This is important to determine the current state of the activity.

The algorithm can determine conflicting state changes ($GetConflictingStateChanges(\sigma)$) in the trace of a shared activity according to Definition 12. It selects the state change set by the user with the highest governance role (*chosenStateChange*). All other conflicting state changes are marked as "ignore" ($MarkAllOtherStateChangesIgnore$ (*chosenStateChange*, $conflictingstatechangeslist$)). As mentioned, this information is used by the algorithms presented later to determine if a dependency is violated or unsynchronized. Finally, it sets the current state of the shared activity based on the last entry of the valid state change sequence ($SetCurrentState$).

Based on the example in Figure 5.7, the algorithm would return that the activity "Transport Sandbag" is in state "Cancel", when assuming that the role "accountable" is superior to the role "responsible".

Example

We illustrate an example for this in Figure 5.9. There, we have activity "a" replicated in two AWs. The activity type of the shared activity "a" is based on the activity type described in Figure 5.8.

As shown in Figure 5.9, there has been initially a conflict when changing from the start state to the state "Alpha" by activity workspace "X" and "Beta" by activity workspace "Y" in (T1). The state changes diverge further in (T2). In (T3) only activity workspace "X" changes the state of a shared activity. Activity workspace "Y" detects the conflict and marks the state change to state "Beta" as ignored. Furthermore, the state change to state "Beta 1" is ignored, because it is not part of a valid state change sequence anymore.

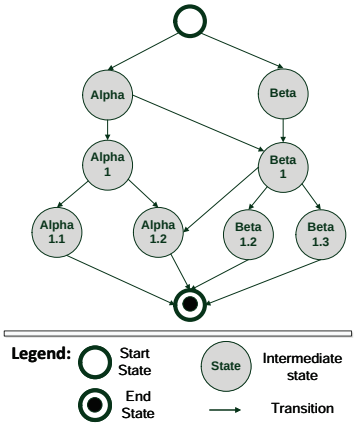


FIGURE 5.8 – Activity type of example

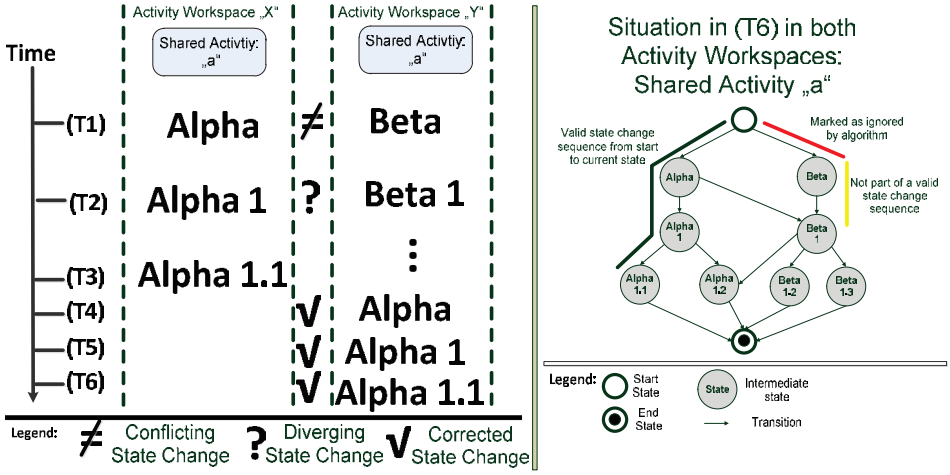


FIGURE 5.9 – Example for valid and diverging state change sequences

The current state is now “Alpha”. In subsequent time points, it is applied to all the other state changes. We illustrate the situation in (T6) also via the activity type on the right. There we see the valid state change sequence on the left and the sequences which should be ignored on the right.

Several Conflicts

We may identify several conflicts at the same time, because as we have seen before it is possible that a conflict leads to deviating state changes of an activity. We illustrate an example for this in Figure 5.10. Three AWs, “W”, “X” and “Y”, are described. They have shared activity “a”. At time point (T1), AW “W” and “X” change activity “a” into the same state “Alpha”. However, AW “Y” changes it into the state “Beta”. This is the first conflict that occurs. At time point (T2), AW “Y” changes its state into “Beta 1”, which means the view on the activity continues to diverge. At time point (T3), the second conflict occurs when AW “W” changes it into state “Alpha 1.2” and AW “X” changes it into state “Alpha

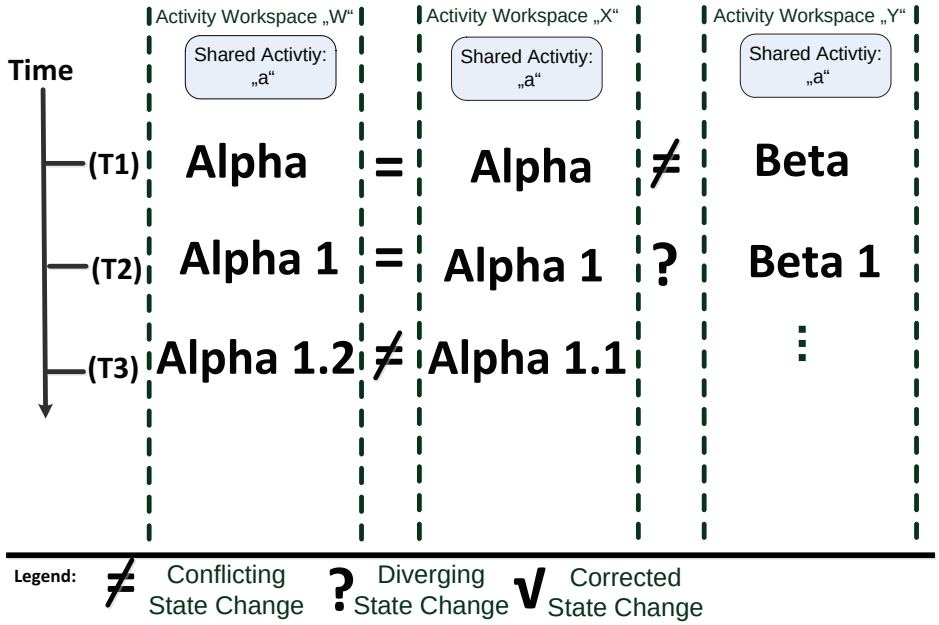


FIGURE 5.10 – Example for valid and invalid state change sequences with two conflicting state changes

1.1".

We propose to apply Algorithm 4 to the order of conflicts (see Definition 14).

Definition 14 *The order of conflicting state changes of the same shared activity "a" is determined by the transitions a.cat.f between states a.cat.S in its activity type a.cat.*

The order of conflicts is determined by the activity type, because it describes the transitions from the start (start state) to the end (end state). These transitions determine the order of state changes. Since an activity should only be in one state at the same time, there is only one possible order when resolving all conflicts. For example, the first conflict according to the activity type is the conflict between the states "Alpha" and "Beta". This conflict is resolved first by the algorithm. The result is that the state "Alpha" has been chosen and the state "Beta" is marked as "ignore". Furthermore, the state change from "Beta" to "Beta 1" is not considered anymore, because it is not part of a valid state change sequence. The reason is that the change to the state "Beta" is ignored (cf. Definition 13). According to Definition 12 there is still a conflicting state change trace after the execution of the algorithm. There is a conflict between the state "Alpha 1.1" and the state "Alpha 1.2". This means the algorithm is applied a second time. The result is that the state "Alpha 1.1" is chosen and the state "Alpha 1.2" is marked as ignored.

As mentioned, we assume that state changes arrive eventually. This means every AW can resolve eventually the conflict even if there have been deviations. We do not rely on the arrival order of state changes, because the activity type of the shared activity prescribes the order. We can thus determine the correct order of received state changes based on the activity type under the assumption that it does not contain cycles, which we excluded by definition (cf. Definition 1).

Summary

We presented in this subsection a conflict caused by concurrent state changes in the trace of a shared activity. It means that several participants try to change the state of the same activity from different AWs into different states concurrently. This leads to diverging views on shared activities. There needs to be a converged view on the state of shared activities, otherwise coordination will be difficult.

The problem seems to be similar to the one of replicated state machines, but there are differences in how we handle the conflict. The algorithm introduced and extended in the previous paragraph ensures a converging view. The underlying assumption is that eventually all state changes arrive at the AW and are part of its trace (cf. section 5.4.1). We do not require that they arrive in a certain order, because the order is defined by the activity type of a shared activity. We also do not need communication between the different AWs to resolve conflicts as, for example, distributed consensus [Lam98, Sch99]. Eventually, all AWs will have the same trace of state changes of a shared activity and thus they will apply the same algorithm under the same condition to resolve the conflict. In the meantime, they can resolve conflicts based on the already received state changes.

Consensus protocols fall short in our scenario, because there is not the notion of a faulty or "wrong" state of a shared activity. Each participant may have legitimate reasons for setting an activity to a state.

The goal of handling it here is to reach a converging view on the states of the activities based on the governance roles associated with them. The state change performed by the person with the highest governance role is selected and the other state changes are marked as ignored. The underlying assumption is that the person with the highest governance role for a state change has the best strategic overview related to the activity. Thus it is important that others align with this person. This can be defined in a flexible manner in the activity type, because this depends on the organization. For example, some organizations in the SoKNOS project emphasize more the leadership in the field while others emphasize leadership in the command center. Furthermore, it is transparent how the conflict is resolved. If two people with the same governance role perform conflicting state changes then the conflict can be highlighted to them and they have to resolve them manually. In any case, the resolved conflict needs to be highlighted to the stakeholders of an activity, so that they are aware of the converged view on the state and do not consider the old state of an activity any more. However, we expect that this type of conflict does not happen very often and thus conflict resolution is rarely required.

5.4.3 Conflicting Views on the Causal Order of State Changes of Dependent Shared Activities

We address in this subsection the problem of ensuring the same view on the causal order of state changes of dependent shared activities. Optimistic propagation of state changes of shared activities can lead to a situation where state changes arrive in a different order in different AWs. This can lead to cases where a deviation between what is defined in the model and how shared activities are executed is observed in one AW, but not in another. For example, a dependency is violated in one AW, but not in another one. The

participants of different AWs may thus initiate contradictory actions.

The main aim of handling this conflict is to ensure that all AWs observe the same causal order of state changes, although they receive them in a different order. This means they have the same view on deviations from what is defined in the model and how activities are executed.

We propose in this subsection an approach for ensuring the same causal order of state changes in all AWs using vector clocks [Mat89]. The vector clock approach has to be adapted to work in our proposed setting of optimistically replicated activities in different AWs. The adapted approach is reused in the next subsections to extend the functionality of the framework for detecting deviations between what is defined in the model and how activities have been executed.

Problem Statement

Optimistically propagated state changes of shared activities can arrive in a different order in different AWs. The order of state changes determines if there has been deviations from what is defined in the model of an AW and how activities have been executed (cf. section 4.4). If the order is not the same in all AWs then the AWs may have a different view on the state of dependencies. This can lead to contradicting actions of participants of different AWs, because they have a different view on the situation. Contrary to the previous conflict, we are not interested in the state changes of one shared activity, but in the state changes of several dependent shared activities.

We provide in Figure 5.11 an example for this conflict with the following involved organizations : military, fire brigade command center and fire brigade field. The military has shared the activity “Transport Sandbags” with the fire fighters in the command center. The fire fighters in the command center have shared the activity “Build Dam” with the fire fighters in the field. The fire fighters in the command center have established a dependency “overlaps” between the activities “Transport Sandbags” and “Build Dam” in the state “Execute”.

We show in the lower part of the figure the sequence of state changes. The fire fighters in the command center change both activities to state “Plan” and propagate the state change to the AWs of the military and the fire fighters in the field. The military changes the activity “Transport Sandbags” into the state “Execute”, but the propagation of the state change to the fire fighters in the command center is delayed. Before the state change arrives, the fire fighters in the command center change the activity “Build Dam” to the state “Execute”. Afterwards, the state change of the military of the activity is received by the AW of the fire fighters in the command center. It is not possible to determine if the state change was performed after a state change of activity “Build Dam” or not. This can also lead to problems if more than one activity has been shared between two or more AWs and these activities are dependent, because the order of state changes can be different in each AW.

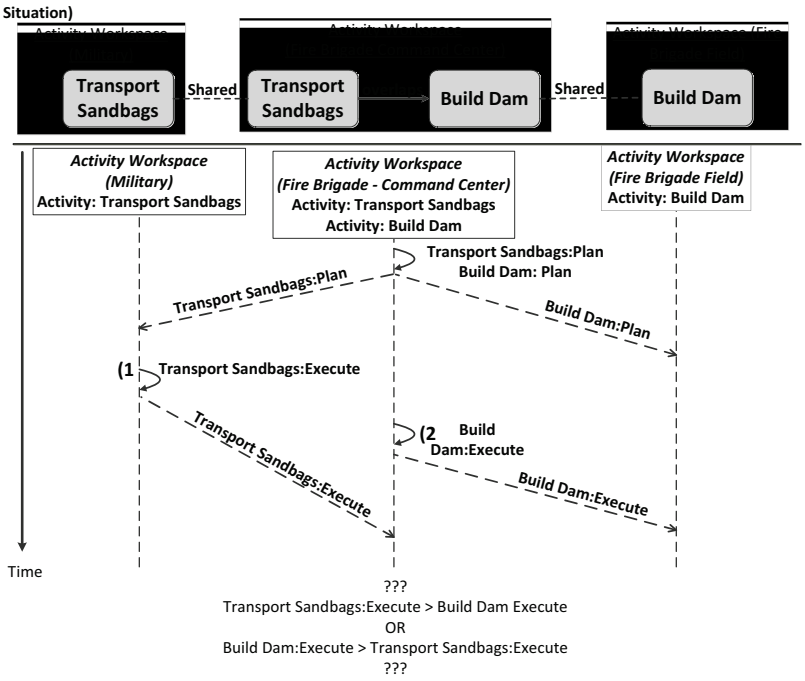


FIGURE 5.11 – Example for a different order of state changes in different AWs

Ensuring the Same View on the Causal Order of State Changes of Dependent Shared Activities

We need to ensure that all AWs can establish the same causal order of state changes of shared activities and can establish eventually the same view on the causal order of state changes, so that they have the same view on the deviations caused by state changes. This is described in Definition 15 based on the "happen before" relation proposed by Lamport [Lam78].

Definition 15 *Eventual global order :* $a_x : s_i < a_y : s_{i+1} \rightarrow C_j(a_x : s_i) < C_j(a_y : s_{i+1})$ $\forall AW j = 1,..n$ *sharing activity* x *and/or* y*. If a state change* $a_x : s_i$ *happens before state change* $a_y : s_{i+1}$ *then this needs to be equally observed in all AWs where the shared activities are replicated.*

Lamport also describes a logical clock function C to ensure the same global ordering of events in a distributed system. The events can be compared with state changes. Every time an event is sent from one process to the other processes in a distributed system, the time of a logical clock C is updated ($C = C + d$, whereby d is usually a value of 1) and attached to the event. Every time the event is received by a process, it updates its logical clock according to the highest clock value of all events received. However, this mechanism does not maintain causal relationships between events, while it is required by our framework. This means it creates the same order of events, but it is arbitrary. An extension of this mechanism can be realized with vector clocks [RS96]. We use vector clocks in our concept, because they can preserve the causal order of state changes.

Every time an AW sends a state change, it attaches a vector clock to it. The vector clock is a vector containing all the states of logical clocks $c_1, .., c_n$ known of the AWs : $V = (c_1, .., c_n)$. The sending AW_i increases its own entry in the vector clock before sending the state change : $V[i] = V[i] + 1$. When a state change with a vector clock has been received, it can be ordered among the existing state changes according to the following rule :

Definition 16 *Ordering State Changes Using Vector Clocks : state change s_i (with clock vector V_x) is partially ordered before state change s_j (with clock vector V_y), if : $V_x[k] \leq V_y[k] \forall k$. When this rule cannot be fulfilled, i.e. it holds $\neg(V_x < V_y) \wedge \neg(V_y < V_x)$, then the state changes occurred concurrently.*

A vector clock allows maintaining the causal relationship between events as it has been proven by Mattern [Mat89]. There it has been shown that received vector clocks build a lattice.

Extending Vector Clocks : Dynamic Organizational Network

The standard vector clock approach assumes a fixed set of AWs which is known in advance. This means each entry in the vector has an AW assigned to it, but it is not possible that new AWs are added. This cannot be assumed in our motivational example described in chapter 2. The police and the fire brigade are coordinating their activities using our concepts. The military may join at a later point in time to protect the chemistry plant from a flood and to transport sandbags for the fire brigade. The standard vector clock approach could not incorporate the military, because it was not known that they would join later.

We can utilize dynamic vector clocks proposed by Landes [Lan05]. The basic idea is that each entry in the vector clock has a unique identifier of the corresponding AW assigned to it and that only entries of two vector clocks are compared which have the same identifier. This is similar to Definition 16, but considering the unique identifiers of each AW.

However, it is important that an identifier is not reused in case an AW is not communicating anymore with other AWs. For instance, after a system crash or when an organization is not participating in a disaster response anymore. This is necessary to avoid known problems to establish a causal order of state changes in these cases [AKD00].

Extending Vector Clocks : Dealing with Finite Clocks

Another well-known limitation of vector clocks is that each logical clock in the vector is finite (cf. [AKD00]). Once the limit of a logical clock is reached, it is not possible anymore to establish a causal order of state changes. A solution is to globally reset all vector clocks at a certain point in time. This is error-prone and difficult to achieve. It may also require to use matrix clocks (cf. [LS87]), which are of larger size. It is desired to have a simpler approach.

We solve this problem by accepting this limitation of vector clocks. This means that we only include as many activities in an AW, so that the finite vector clock counter is

sufficient. We have seen before that an increase of a counter in a vector clock is related to the number of state changes, i.e. each state change leads to an increase of the vector clock counter. As we have explained before, each activity has an activity type describing the lifecycle (i.e. different sequences of state changes) of an activity. The maximum number of state changes can be derived from the lifecycle. They can be determined by calculating the longest path in a graph of the activity lifecycle. Calculating the longest path in a graph is generally a NP-complete problem and has exponential complexity. However, we defined that the activity type must not contain cycles (cf. Definition 1). Thus, the longest path problem has only polynomial complexity by using any shortest path algorithm [Sed03]. This means our approach has practical applicability, because we do not expect large lifecycles with very long paths.

For example, we assume that each activity type has an average longest path of 10 state changes and the counter c_i of AW i has a size of 32 bits. In this case, approximately 400000 activities can be modeled in an AW. We assume that this is enough, when considering around 500 organizations (cf. the Haiti earthquake [Ola10] or September 9/11 terrorist attacks [CK06]) and even if each organization would share with the other organization activities or has several AWs. In any case, if the counter of an AW should reach its limit then participants can switch to another AW and share some of the activities from the "old" AW with the "new" AW (e.g. all activities that have not reached an end state). The size of the vector clock can also be reduced further when using this mechanism. Please note that this limitation does not affect the number of AWs, but the number of activities in one AW.

Establishing the Same Causal Order of State Changes of Shared Activities in a Special Case

We described in the beginning of this section that a state change of a shared activity is propagated optimistically to all AWs where the activity is replicated. We use vector clocks to establish a causal order of state changes of activities. This is important for detecting violation of temporal dependencies, because this depends on the same sequential input of state changes.

However, there are special cases, where it is not possible to establish a causal order of state changes using the standard vector clock approach. Such a case is illustrated in Figure 5.12. There are three AWs : "Military Command Center", "Fire Brigade Command Center" and "Fire Brigade Field". The activity "Transport Sandbags" has been shared by the military commander with the fire fighter commander. It is replicated in the AW "Military Command Center" and AW "Fire Brigade Command Center". The activity "Build Dam" has been shared between the fire fighter commander and the fire fighters in the field. It is replicated in AW "Fire Brigade Command Center" and AW "Fire Brigade Field".

We assume that the activity "Transport Sandbags" is changed by the military into state "Execute" and the state change is propagated to the AW of the fire brigade command center (with vector clock $V_{Military} = ((1, \text{“}Military\text{”}),(0, \text{“}FireBrigadeCommandCenter\text{”}))$). Afterwards, the activity "Build Dam" is changed into state "Execute" and the state change is propagated to the AW of the fire fighter command center (with vector clock $V_{FireBrigadeField} = ((1, \text{“}FireBrigadeField\text{”}),(0, \text{“}FireBrigadeCommandCenter\text{”}))$. The

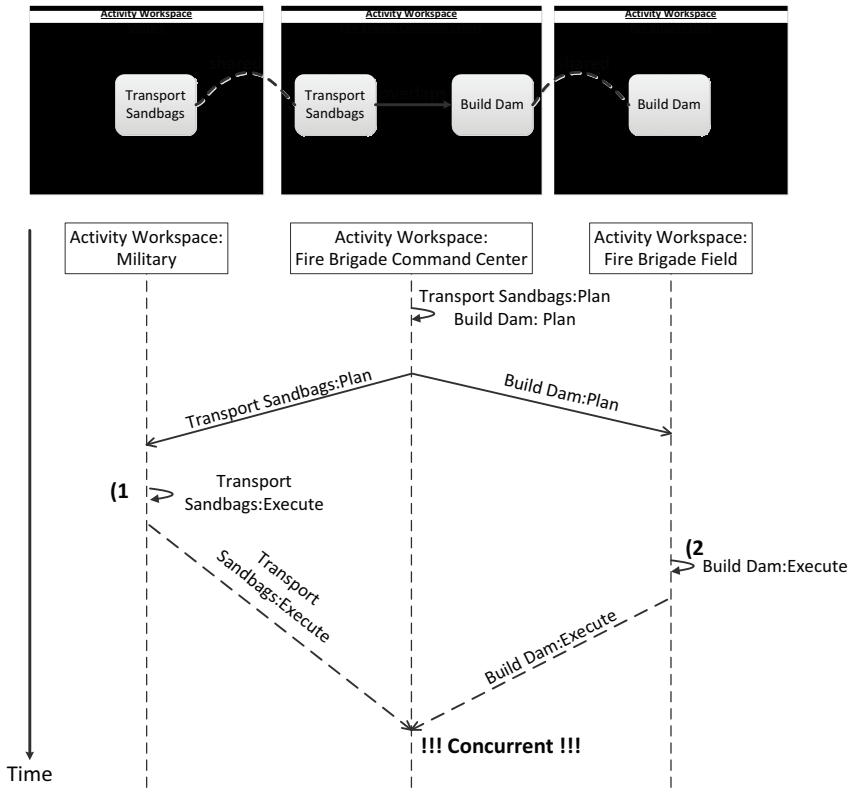


FIGURE 5.12 – Example for a situation where a causal order of state changes cannot be established using the standard vector clocks approach

AW of the fire brigade command center is never able to establish a causal order of state changes, because the AW of the military and the AW of the fire brigade in the field do not know their vector clocks. Thus, the AW of the fire brigade command center always observes a concurrent state change.

We address this issue by introducing the following rule into the protocol :

Definition 17 *Propagation Rule for Vector Clocks : When a state change with a vector clock is received by AW_i then it increases its own clock c_i and sends the updated clock vector to all AWs it shares activities with.*

This means for the previous example that when the state change of the military is received, the AW of the fire brigade command center updates its own vector clock. Afterwards, it sends the vector clock $V_{FireBrigadeCommandCenter}$ $= ((0, \text{“}FireBrigadeField\text{”}), (1, \text{“}FireBrigadeCommandCenter\text{”}))$ to the AW of the fire brigade in the field. When now the state of activity “Build Dam” is changed in the AW of the fire brigade in the field then it is sent together with the vector clock $V_{FireBrigadeField}= ((1, \text{“}FireBrigadeField\text{”}),(1, \text{“}FireBrigadeCommandCenter\text{”}))$ to the AW of the fire fighter commander in the command center. This AW can now observe that the state change of activity “Transport Sandbags” has happened before the state change of activity “Build Dam” when comparing the vector clocks.

The work by Nichols et al. provides a solution for a similar problem related to client-server communication [NCDL95]. However, we address here the communication of state changes to all stakeholders that have shared activities. Furthermore, we do not share everything with a central entity (server).

Summary

We have presented in this subsection mechanisms for ensuring eventually the same view on the causal order of of state changes in all AWs. This is important, otherwise it would not be clear what are the relations between what has happened, what is currently going on and what are the next steps. These mechanisms are exploited in the following subsections, where we extend the functionality of the framework described in chapter 4 for detecting deviations of what is defined in the model of an AW and how shared activities are executed.

5.4.4 Detecting Violation of Temporal Dependencies Involving Shared Activities

State changes of shared activities may arrive in a different order and are ordered by using vector clocks as mentioned before. There can be a delay between state changes arriving in a different order. This means there are times where it is not possible to decide if a dependency is violated or not. Another problem is that concurrent state changes may occur and that this has been handled (cf. Algorithm 4). Handling of this conflict may involve changing the state of a shared activity and it has to be re-evaluated if this leads to violation of dependencies.

Algorithm 5: Algorithm for Determining the State (neutral, violated, unknown) of a Dependency when Receiving a State Change

input : Received state change SC
output: Setting the state of each dependency affected by state change SC

```
a1 ← GetActivity (SC);
D ← GetDependencies (a1);
for i ← 0 to D.size - 1 do
    d ← D[i];
    a2 ← GetOtherActivity (a1,d);
    X1 ← GetValidStateChangeSequence (a1);
    X2 ← GetValidStateChangeSequence (a2);
    if (ContainsRelevantStateChangesOfDependency (X2,a2.at)==false) OR
    (ContainsRelevantStateChangesOfDependency (X1,a1.at)==false) then
        SetDependencyState (d,unknown);
    else
        X ← SortStateChanges (X1,X2);
        SetDependencyState (d,CheckDependency (d,X));
```

Thus, we have to extend the algorithm for detecting violation of temporal dependencies of the framework presented in chapter 4 to deal with this (cf. Algorithm 3). Given the previous example (cf. Figure 5.12), we assume that the AW of the fire fighter commander in the command center has received a state change of the activity "Build Dam" to "Execute". However, it has not received any message about a state change of activity "Transport Sandbags". However, it can still be the case that the delivery of the message of the state change of activity "Transport Sandbags" is delayed and it has to be waited until a state change of activity "Transport Sandbags" is received. During this waiting time the state of the dependency is neither neutral nor violated, but unknown.

Algorithm

Each AW can determine individually if a dependency is violated, neutral or unknown using Algorithm 5 given the received state changes of activities. It is executed every time a state change of a shared activity is received. When a state change SC is received by an AW, it retrieves all the dependencies associated with the activity changing its state ($GetDependencies$). For every dependency it gets the other activity associated with the dependency ($GetOtherActivity$). This can either be the source activity $d.a_s$ or the destination activity $d.a_d$ of the dependency d. It determines for both activities a valid state change sequence ($GetValidStateChangeSequence$). We have already described in subsection 5.4.2 how this can be done (cf. Definition 13). This means it takes into account the conflict resolving mechanism for concurrent state changes of the same shared activity as well as that not all state changes have been received yet (cf. section 5.4.2). The state of a dependency is unknown, if not all state changes affecting the dependency have been re-

ceived ($ContainsRelevantStateChangesOfDependency$). This can be determined based on the dependency state machine that has state changes as input (cf. Definition 7). This is different to the approach presented in the previous chapter. We cannot simply input a received state change into the dependency state machine as soon as it arrives, because we may not know the order between state changes. This order is only known when all state changes defined in the dependency state machine have arrived.

Otherwise it checks if the dependency is violated or neutral as described in Algorithm 3 in the previous chapter. It takes into account that the state changes arrive in a different order and sorts them according to their vector clocks ($SortStateChanges$). Using Algorithm 5, it can be detected if a dependency is violated, neutral or it cannot be decided given the set of already received state changes.

Discussion

As described before, we also assume here that state changes arrive eventually. This means the AWs will have eventually the same result when applying the algorithm, because the causal order of state changes is defined by the vector clock approach. The algorithm takes into account conflict handling mechanisms described before, because it considers only a valid state change sequence. However, conflict resolution can mean that a dependency changes its state from neutral or violated, because a new state has been chosen for a shared activity to ensure a converging view. This needs to be highlighted to the user, so that the new information can be taken into account.

Conclusion

We presented in this subsection how violation of dependencies involving shared activities can be detected. The main challenge here is that not all state changes have been received or that conflicting state changes have been handled and thus a dependency can be in an unknown state. This has not been covered by the framework described in the previous chapter. We described an algorithm that is able to detect if a dependency is in state violated, neutral or unknown given a state change. It delivers eventually the same result in all AWs and also takes into account the conflict handling mechanisms presented in the previous subsections. However, due to the conflict resolution mechanisms, a violated dependency can become neutral and vice versa. This new information needs to be displayed to the user. The algorithm is a little bit slower compared to the original algorithm in the framework, because state changes need to be sorted.

5.4.5 Detecting Unsynchronized Dependencies Involving Shared Activities

Similarly to the previous problem, the detection of violation of dependencies, the algorithm for detecting unsynchronized dependencies of the framework is also not able to deal with shared activities. The problem here is also that the state changes can arrive in a different order. For example, when considering the previous example (cf. Figure 5.12) that the AW of the fire fighter commander in the command center has received a state change

Algorithm 6: Algorithm for detecting unsynchronized dependencies when receiving a state change

input : Received state change SC
output: List of unsynchronized dependencies L

```
A ← GetActivities (SC);
for i=0; i<A.size;i++ do
    X ← GetValidStateChangeSequence (A [i]);
    MRSC ← GetMostRecentStateChange (X);
    OACT ← GetDependentActivities (X));
    for j=0; j<OACT.size;j++ do
        OAMSC = OAMSC ∪ GetMostRecentStateChange
        (GetValidStateChangeSequence (oact[j])));
    L ← DetectUnsynchronizedDependencies (MRSC,OAMSC);
return L ;
```

of the activity "Build Dam" to "Execute" and afterwards receives the state change of activity "Build Dam" to "Plan". The AW should consider for detection of unsynchronized dependencies only the state change of activity "Build Dam" to "Execute", because it relies on the current state of activities. This has to also take into account the conflict handling mechanisms described before, otherwise it may consider invalid state changes.

Algorithm

The algorithm for detecting unsynchronized dependencies presented in chapter 4 has as input the most recent state change of an activity and the most recent state change of dependent activities. Algorithm 6 describes how the most recent state change can be obtained from the received state changes of a shared activity.

The algorithm gets the activities of the state change in the first step (*GetActivities*). For all these activities, it determines a valid state change sequence (*GetValidState ChangeSequence*). Thus, it considers only the state changes that build a valid sequence and ignores the other ones (cf. Definition 13). It picks the most recent state change from the sequence of valid state changes (*GetMostRecentStateChange*). Then, it determines the most recent state changes of dependent activities (stored in $OAMSC$), because it is required to determine their current state. Afterwards, it performs the detection of unsynchronized dependencies according to the framework (*DetectUnsynchronizedDependencies*) based on the most recent one of the activity changing its state and the dependent activities. The output of the algorithm is a list of all dependencies which become unsynchronized by the state change.

Discussion

Similarly to the previous algorithm, eventually all AWs will obtain the same result when applying the algorithm, because they receive eventually the same state changes of

a shared activity in the same causal order. The algorithm considers the conflict handling mechanisms described before, because it utilizes only a valid state change sequence. It should be noted that the handling of the causal order of state changes is not required for our approach, because we only rely on the most recent state change (i.e. the current state of an activity). However, if a conflict handling mechanism leads to a new state of a shared activity to ensure a converged view, then a dependency may become unsynchronized and vice versa. This needs to be highlighted to the user, so that the new information can be taken into account.

Conclusion

We presented in this subsection an algorithm for detecting unsynchronized dependencies involving shared activities. It takes into account the conflict handling mechanism described before, because it considers only a valid state change sequence. However, due to the conflict resolution mechanisms, a dependency can become unsynchronized and vice versa. This new information should be highlighted to the user.

5.4.6 Incomplete Propagation of State Changes of a Shared Activity

Going back to the motivational example in chapter two, the transport of sandbags by the military for the fire fighters may fail. The fire commander never receives this information from the military and still assumes sandbags are transported. Thus, he does not look for alternatives. However, the military is not aware that the fire commander never received this information, because it may still be the case that the confirmation of reception is delayed. We discuss in this subsection the problem of incomplete propagation of state changes of shared activities. This means a state change of a shared activity is propagated to some, but not all, AWs where the activity is replicated. This can happen, for instance, due to unreliable communication. This leads to a diverging view on shared activities and dependencies in different AWs.

We describe how the AWs can synchronize each other to detect and handle incomplete propagation of state changes. The goal is to ensure eventually a converging view in all AWs. We state in the next paragraph the problem and propose an approach for synchronizing the state change trace of a shared activity. Afterwards we analyze how this affects detecting deviations from what is defined in the model and how activities are executed.

Synchronizing the State Change Trace of a Shared Activity

There are two approaches to address the problem stated before :

1. The problem is solved eventually : after another state change of any AW, the AWs synchronize each other.
2. The problem is solved by synchronizing on a timely basis.

The first approach means that after another state change, the different AWs can detect that there has been a state change they did not receive and synchronize. The second

approach means that the different AWs synchronize the state changes of a shared activity regularly. For example, they may synchronize every ten minutes. We suggest combining both, because the first approach may lead to a situation that a missing state change is only detected after some time. A synchronization protocol for both cases needs to ensure that eventually the different AWs have the same trace of state changes of a shared activity.

Our proposed synchronization protocol for a trace of state changes of a shared activity between AW i and AW j works as follows :

1. AW i and AW j exchange the traces of a shared activity as well as the vector clocks of the corresponding state changes in the trace
2. All duplicate state changes are merged into one entry and the vector clock from the earliest occurrence of the state change is taken
3. All missing entries are added from the trace of AW i to AW j and vice versa

This synchronization protocol is based on an AW-to-AW communication. There is no consensus mechanism involved in the sense that there is an agreement on a correct trace. Basically, the traces of different AWs are merged. The only conflict that can occur and that is not handled by the algorithms mentioned before is that the vector clocks for the same state change (e.g. from "Plan" to "Execute") may differ. This can happen in rare cases. For example, we assume two AWs containing a shared activity "a". We assume that the first AW changes the state of shared activity "a" from "Plan" to "Execute". However, the state change is not propagated. The second AW also changes the state of shared activity "a" from "Plan" to "Execute" later. After synchronization there may be two different vector clocks for the same state change.

We cannot utilize vector clocks to determine missing entries in a state change trace of shared activities. Vector clocks in our approach consider the state changes of all activities occurring in one AW and not of a single shared activity.

We argue that our protocol leads eventually to the same trace in all AWs, when all AWs are able to successfully synchronize themselves eventually. They can use the conflict handling mechanisms presented in the previous sections to find a converging view on a trace. However, there can be state changes which will never be part of a trace. For example, if a state is changed in one AW and it crashes before the state change can be propagated to any AW.

Effect of Synchronization on Detecting Violation of Dependencies

Eventual synchronization of propagated state changes of shared activities may lead to a situation where a dependency is changed from violated to neutral and vice versa. This is due to the second step in the synchronization procedure. However, this step ensures that eventually all AWs have the same view on the dependencies. More particularly, they observe the same causal order of state changes of a shared activity, which is important as we have argued before. We propose to highlight this change to the participants of an AW, so that they are aware of this. Synchronization does not affect the termination of Algorithm 5. The reason is that it does not require that all state changes have been received and thus termination is possible.

Effect of Synchronization on Detecting Unsynchronized Dependencies

Incomplete propagation and eventual synchronization of the state changes of shared activities does not affect unsynchronized dependencies, because it only relies on the fact that a state change occurred. However, conflict resolution as it has been described before (cf. section 5.4.2), may lead to cases where dependencies become unsynchronized and vice versa. This needs to be highlighted to the user.

Conclusion

We described in this subsection, how incomplete propagation of state changes of shared activities affects the concepts described before. We did not propose distributed state based transactional recovery algorithms (cf. [EAWJ02]). The reason is that a transactional recovery mechanism leads to an outdated state in the past in our scenario. However, activities taking place in the "real world" cannot be set back to a state in the past. The importance here is to provide a (partial) shared common view on the activities and dependencies in all AWs.

Of course, it is possible to make it unlikely that there is a problem of incomplete propagated state changes, e.g. by using a peer-to-peer network architecture (cf. [ATS04]). However, this only decreases the likelihood that it does not happen. Hence, we described here the consequences for the case that the "unlikely" happens. We described a synchronization mechanism that merges the state change trace of shared activities between different AWs. This may lead to a change of the state of a dependency in the AWs, e.g. from neutral to violated or vice versa. Nevertheless, the synchronization mechanism ensures that this is consistent in all AWs by providing eventually the same state changes of shared activities to all AWs.

5.4.7 Summary

We argued in the previous section that organizations can only share selected information with other organizations due to privacy, regulatory, strategic or other reasons. This means that they have a partial shared common view on what is going on, but not one organization knows everything. We proposed that state changes of shared activities are propagated optimistically, because pessimistic propagation seems to be not feasible in our research context (cf. chapter 2). However, we need to detect conflicts in the form of diverging views on the states of shared activities and dependencies. A diverging view would lead to typical coordination problems in the form of inaction, double efforts or conflicting actions, because there is no partial shared common view on the activities and dependencies.

We described how the following conflicts can be detected and handled to ensure a converging view on the states of shared activities and dependencies :

– Conflicting state changes of the same shared activity : state changes of a shared activity can be initiated from any AW. This may lead to a situation where a shared activity could be in two different states in different activity workspaces. If possible this should be handled eventually to ensure a converging view. This problem is

handled by choosing the state set by the user with the highest governance role. This preserves strategic intention of the state changes and follows the rationale that the one with the highest governance role has the best overview to deal with this conflict. Furthermore, it is transparent for everybody involved in a shared activity. However, it is important that the user is informed when the conflict is handled so that the new information is taken into account.

– Causal order of state changes : we extended standard vector clocks, so that each AW observes the same causal order of state changes of shared activities. The reason is that without such a mechanism each AW can observe the state changes of shared activities in a different order. Thus, there is a diverging view on the causal order of state changes of shared activities. This could lead, for example, to cases where dependencies between shared activities are in different states in different AWs.

Furthermore, we adapted the algorithms of the framework for detecting violation of dependencies and for detecting unsynchronized dependencies to incorporate the handling of conflicts. It turns out that these adaptations only introduce a small performance overhead compared to the algorithms in the framework.

Finally, we addressed the problem of incomplete state change propagation of shared activities. This means that state changes of shared activities are propagated to some, but not all, AWs. This leads to a different view on the states of shared activities. We describe how this can be detected by the AWs eventually after another state change or on a timely bases. We propose a synchronization mechanism so that all AWs become aware of all propagated state changes of shared activities, so that they can have eventually a converging view on the state of shared activities and dependencies.

It turns out that synchronization can lead to cases where a dependency switches its state from neutral to violated or vice versa. We argue that this is negligible, because it is more important to have a converging view in all AWs. This ensures a partial shared common view on the activities of the different organizations and their dependencies. Hence, it helps to avoid typical coordination problems. Furthermore, the new information can be highlighted to the user. Otherwise all stakeholders have a different view on the activities and dependencies, which would make coordination more difficult.

5.5 Related Work

We adapted in this chapter several mechanisms to extend our proposed framework in chapter 4 to the inter-organizational level. We investigate now their relationships to applications in other scenarios. We already described before the differences to the replicated state machine approach and its application area in section 5.4.2.

Approaches for distributed Constraint Satisfaction Problems (CSP) address the problem that variables and constraints are distributed to several agents. These agents have to find a solution so that all constraints are fulfilled [HY97]. This is similar to verification of a model containing shared activities, but in our case the model that needs to be verified is defined dynamically based on the shared activities. Other work only considers verification of inter-organizational workflows before executing them and it does not take into account a dynamic distributed setting such as ours [vdA00, KMR00, vGS03].

We also find many approaches addressing detecting and handling of conflicts in distributed collaborative work (cf. [GM94]). For example, collaborative image [SC02] or text editing [IROM06] with optimistic change propagation. These approaches deal with conflicts in unstructured documents (e.g. text or images) and not in structured models like our concept. Furthermore, there are more generic concepts that deal with replication of data (e.g. [BM08]). They do not consider the case that not everything is shared. The work by Taylor et al. is an exception to this [TI06]. They describe an approach for biological and biomedical communities where different sites share some data and import some, but not all, data from others. This data is not necessarily consistent and there is even a need that it diverges in case of disagreement on the facts by different scientists. It can create a reconciled view of the data based on ranking information of the data. This is different from our approach, where we propose to create a reconciled view based on the governance roles defined for the activity.

Distributed versioning systems can include a distributed issue or bug tracker (e.g. ticgit [tic]) . They allow defining an issue and assigning people to work on it. Different sites, comparable to our AWs, may decide to include the issues from other sites. Similar to our activities the issue can be in different states. However, they do not allow defining dependencies between different issues or to have different lifecycles for different issues. Furthermore, it is unclear how they resolve conflicts.

5.6 Conclusion

People need to decide what information they want to share about what they are doing with people of other organizations. This has privacy, regulatory, strategic or other reasons. We presented an approach where selected shared activities are replicated optimistically in activity workspaces of different organizations. An activity workspace provides the functionality of the framework presented in chapter 4. It has been extended to manage shared activities replicated optimistically in different activity workspaces. As far as we know, there exists no unique good theoretical solution to handle conflicts using this kind of replication mechanism with constraints (dependencies). Thus, we adapted different techniques in distributed systems to our setting. They should provide the user with timely information about problems with respect to coordination. Automatic handling of conflicts is optional. Empirical studies have to be conducted to determine advantages and perspectives for future research in this area.

Verification of an AW containing shared activities also needs to consider the dependencies to this shared activity in other AWs, otherwise it is not possible to detect all errors. We argued that the dependencies of shared activities in other organizations should be considered for verification, because they are working together and should be made aware of problems this may cause. We proposed a verification protocol for considering dependencies of shared activities in other AWs. This only requires disclosing the structure of dependencies and activities in other AWs with whom activities have been shared, but not their content or current state. Thus, only relevant information for verification is shared. Nevertheless, participants can be made aware of any misunderstanding with respect to the dependencies of activities of other organizations they are collaborating with.

State changes of shared activities are propagated optimistically to all AWs where the activity is replicated. It needs to ensure that eventually all AWs provide a converging view on the state changes of activities and on the state of dependencies. Two conflicts leading to diverging views have been identified : concurrent state changes of the same shared activity and diverging causal order of state changes of shared activities in different AWs. They are handled eventually to ensure a convergent view on the activities and dependencies. The algorithms for detecting deviations between the model and how activities are executed are extended to incorporate the conflict handling mechanisms. Ensuring a convergent view means that some of the replicated activities may need to change their state to a converged view. This may also affect dependencies (e.g. they may become violated or unsynchronized). This needs to be highlighted to the user, so they can take them into account. A converged view is necessary for coordination, because otherwise each organization may have diverging views on the activities and dependencies, which makes it very difficult to coordinate.

Finally, we dealt with the problem that state changes of a shared activity are not propagated to all AWs where the shared activity is replicated. This can happen, for example, if they are lost during transmission. Thus, we designed a protocol for synchronizing AWs with each other regularly to ensure a converging view on state changes of shared activities and the state of dependencies.

We argue that the concepts presented here support the coordination by people of different organizations more adequately than traditional tools used by end users, such as phone, fax or e-mail. When using these tools, the information is stored in unrelated messages making it difficult to relate the different activities, get a current overview on what is going on and detect diverging views on the activities as well as dependencies. Contrary to approaches in chapter 3, we take into account that not everything can be shared between everybody and we support coordination using partially shared information.

We explained in this and the previous chapter the concepts of an information system supporting coordination of activities by people of different organizations in dynamic situations. These concepts need to be validated technically, to determine if it is possible to realize such an information system, and empirically, to determine what are advantages and limitations with respect to the research questions.

Chapitre 6

Implementation

Contents

6.1 Introduction 127
6.2 Main components 128
6.3 Integration into Google Wave™ 128
6.3.1 Preliminaries 129
6.3.2 Extension - Architecture 129
6.3.3 Extension : Persistence of Activity Workspaces and Replicating Shared Activities 131
6.3.4 Gadget - Graphical Modeling Tool 132
6.3.5 Robot - Distributed Coordination 134
6.4 Conclusion 135

6.1 Introduction

We present in this chapter the implementation of the concepts described in chapters 4 and 5 with a special focus on implementation challenges. The main goal of the implementation was to demonstrate technical feasibility of the concepts. Other related goals have been :

1. Implement a proof of concept of the algorithms described before.
2. Leverage an existing collaboration platform to show how the concepts unfold in the context of different tools already used in disaster management, like maps, pictures or videos.
3. Use this extension for experiments with students (cf. next chapter).

We provide an overview of the main components of the implementation in the section 6.2. We address the extension of an existing distributed collaboration platform, Google Wave™, in section 6.3. This collaboration platform provides already an infrastructure for the concepts presented in chapters 4 and 5. Finally, we conclude in section 6.4.

6.2 Main components

The implementation consists of two main components :

1. Activity Management Engine : a Java library addressing the first goal of the implementation. It implements the algorithms and data structures described in chapters 4 and 5. The main idea of the library is to abstract from the network and persistence functionality, because we argue that the concepts presented in these chapters can be integrated into various applications. For example, the concepts of chapters 4 and 5 can be integrated into collaboration services, social networking services (e.g. Twitter [twi] or Facebook [fac]) or instant messaging services (e.g. [SAST09]). These services are not equally well-suited for addressing our research questions, but still part of our concepts can also be realized in them. Furthermore, we can implement more easily unit test cases for the concepts.
2. An extension to a collaboration service : Google Wave™ is extended to address the second and third goals of the implementation. The extension uses the Activity Management Engine and provides persistence as well as network functionality. Furthermore, it provides different graphical user interfaces for defining activities and dependencies. This can also be helpful for investigating the effect of the user interface on the usability of our concepts. It represents one possible realization of the concepts in a specific application.

6.3 Integration into Google Wave™

We extended the collaboration service Google Wave™ [Fer09, TP10] to realize parts of the concepts presented in chapters 4 and 5. The main motivation for implementing an extension to this collaboration service was :

- It is based on an open decentralized architecture that allows each organization to have an own server and keep control over the data and what is shared with others. This is also the underlying assumption of our approach. Furthermore, it is in process of being open sourced as Apache Wave [Apa]. The potential benefits are higher inter-operability between different collaboration platforms as well as a standard for collaboration facilitated by a federation of servers. Both can be seen as beneficial in the disaster response scenario, where different heterogeneous organizations need to work together.
- It provides already the Open Wave Federation Protocol standard for replicating optimistically shared information with selected participants of different organizations, but it does not incorporate the concepts presented in chapters 4 and 5.
- It provides rich “real-time” collaboration functionality for text, images, videos or maps. This means we can demonstrate how our concepts can be integrated with other functionality required in a disaster response.
- It can be extended by further collaboration capabilities.

We introduce in the section 6.3.1 the basic functionality of Google Wave™ relevant for the implementation. The architecture of the extension is presented in the section 6.3.2. The representations of activity workspaces as well as activities are explained in section 6.3.3.

The extension of the graphical user interface of Google Wave™ for modeling activities and dependencies is discussed in section 6.3.4 The functionality of the framework and propagation of state changes of shared activities replicated in different activity workspaces is presented in section 6.3.5.

6.3.1 Preliminaries

Google Wave™ is a collaboration service that allows "real-time" collaborative editing of distributed documents, called "Waves". It is in process of being open sourced under the framework of the Apache Foundation as "Apache Wave" [Apa]. The Open Wave Federation Protocol enables different Wave servers to communicate with each other. These servers host "Waves" that can be replicated over several servers. A "Wave" has a unique identifier provided by the server. Participants can register to different servers. They can be part of different "Waves". Participants of a "Wave" can modify it (e.g. editing text or inserting images). Changes are propagated optimistically to other servers and thus their participants. Participants can be invited to "Waves" by other participants of the "Wave". The participants of a "Wave" can be registered with different servers and the "Wave" is replicated to the servers of its participants.

We illustrate in Figure 6.1 a scenario consisting of three different federating Wave servers. "Wave 1" is replicated between the Wave server "Alpha" and "Beta", because participants from these two servers have been invited to this "Wave". It is not replicated with the Wave server "Gamma", because no participant from this server has been invited to it. "Wave 2" is replicated between the Wave server "Beta" and "Gamma", because it participants from these two servers has been invited to it.

The server where a "Wave" has been created manages and propagates optimistically the updates of text in a "Wave", because this is supported by the operational transformation approach to deal with concurrent text editing conflicts (cf. [NCDL95]). Other types of updates of the "Wave", e.g. by a Gadget (see below) inserted into it, are propagated optimistically without any conflict handling mechanism. The Wave Federation Architecture and the Open Wave Federation Protocol suit well to the concepts presented in chapter 5. Only selected "Waves" can be shared by inviting participants of different organizations using different Wave servers. The shared "Waves" are replicated to the servers of different organizations under their respective control.

6.3.2 Extension - Architecture

We present in this paragraph how Google Wave™ can be extended and provide an overview of our extension implementing parts of the concepts described before. Two different types of extensions are supported :

1. **Gadget** : a "Gadget" can be inserted into a "Wave". It is a graphical user interface that provides further collaborative functionality. For example, a map gadget can be inserted into a "Wave" and participants can collaboratively define landmarks on the map. Another example is a collaborative process modeler that can be used to model together with participants a business process (cf. chapter 3). A "Gadget" can persist data in its own state (e.g. landmarks or process models). The "Gadget"

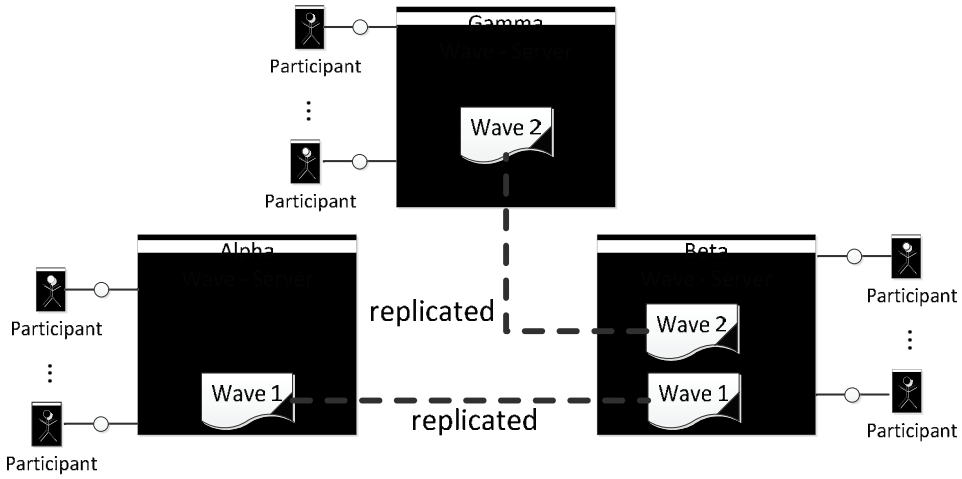


FIGURE 6.1 – Example for three wave servers with replicated waves

state is a key-value store that is part of the "Wave", where the "Gadget" has been inserted. This data is replicated optimistically to all Wave servers where the "Wave" is replicated. A "Gadget" is implemented using the Hyper Text Markup Language (HTML) and Javascript ([htm]). The "Gadget" logic can be stored on any server or on distributed fault-tolerant servers (e.g. Google App Engine [San09]).

2. **Robot** : a "Robot" is an automated participant that can be added to a "Wave". The main purpose is to connect a "Wave" with the "outer world" (e.g. stock market database, dictionary, social networks or other waves). It can react on events in a "Wave" (e.g. modification of text or adding of new participants), but also update information in a "Wave". Technically, it is realized as a stateless Servlet [ser] and requests are received and send to it by a Wave agent connected to a Wave server. The "Robot" is hosted on its own server, but can also be hosted in a distributed fault-tolerant way on several servers (e.g. Google App Engine). In this case, several different instances of the "Robot" on different servers can handle requests. The "Robot" can be realized in any language.

The architecture of our extension is illustrated in Figure 6.2.

We exploit both types of extensions to implement parts of the concepts presented in chapters 4 and 5. Particularly, we provide the following extensions :

1. A **graphical modeling tool** implemented as a "Gadget" that is inserted into a "Wave" : a participant can use it to describe models, i.e. creating activities and dependencies. Furthermore, a participant can perform state changes of activities. An erroneous model (cf. section 4.3) is highlighted to the participants as well as violated or unsynchronized dependencies (cf. section 4.4). Concurrent state changes can also be highlighted (cf. section 5.4.2). Shared activities are displayed to the participant and can be integrated into the model (cf. section 5.2). The "Gadget"

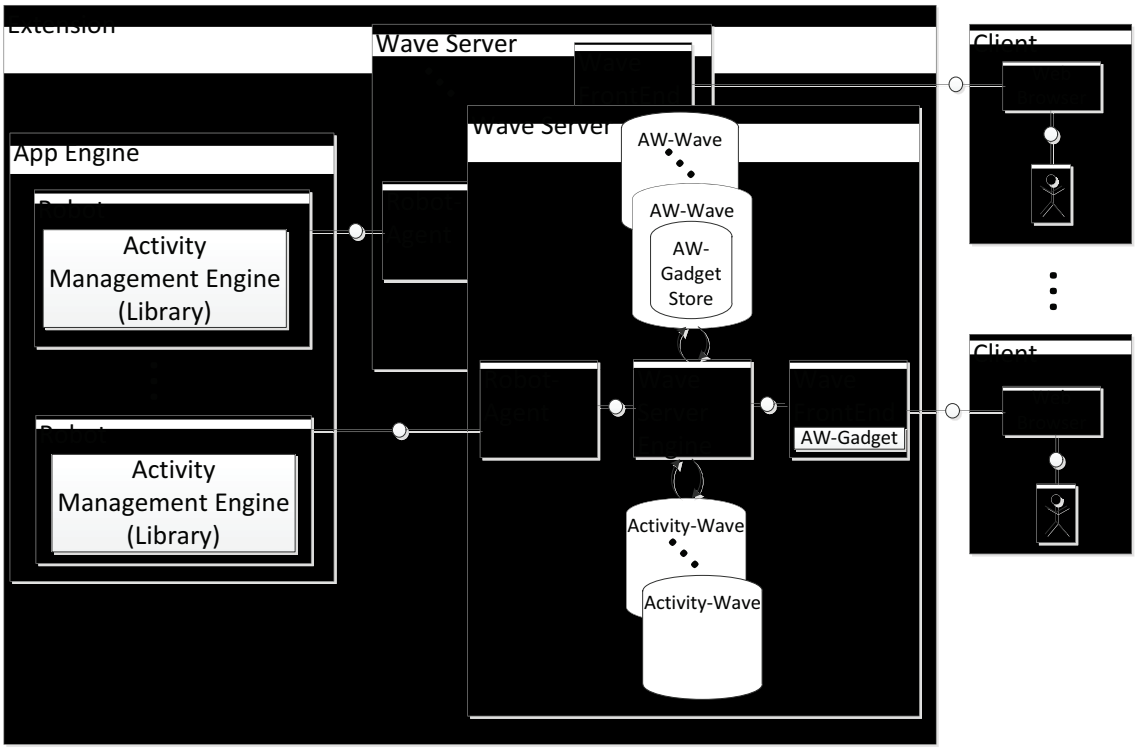


FIGURE 6.2 – Architecture of the extensions to Google Wave™

persists the model and state changes in the "Wave" where it has been inserted. We refer to a "Wave" containing the graphical modeling tool as "AW-Wave". The "AW-Wave" can be compared with an activity workspace.

2. A "Robot" to support **distributed coordination of activities** (cf. chapter 5) : it propagates state changes between shared activities modeled in models (cf. section 5.4) described in different "Waves" via the graphical modeling tool. Furthermore, it performs verification of models, detection of violated dependencies and detection of unsynchronized dependencies.

6.3.3 Extension : Persistence of Activity Workspaces and Replicating Shared Activities

We illustrate here an example how our extension stores models in different "AW-Waves". This is illustrated in Figure 6.3. It shows two different "AW-Waves" on the left and on the right. "Participant 1" is part of the "Wave - AW 1" and "Participant n" is part of the "Wave - AW n". Both are part of the "Activity Wave" of activity "Activity 1". This means the activity is shared between them. Both have also other activities in their "AW-Waves", which they have not been shared together, but they have established dependencies from their own activities to the shared activity "Activity 1".

Sharing of activities is done by representing an activity as a "Wave", called "Activity-Wave". Each activity in a model described using the graphical modeling tool has a link to an "Activity-Wave". An activity can shared by adding a participant to the "Activity-

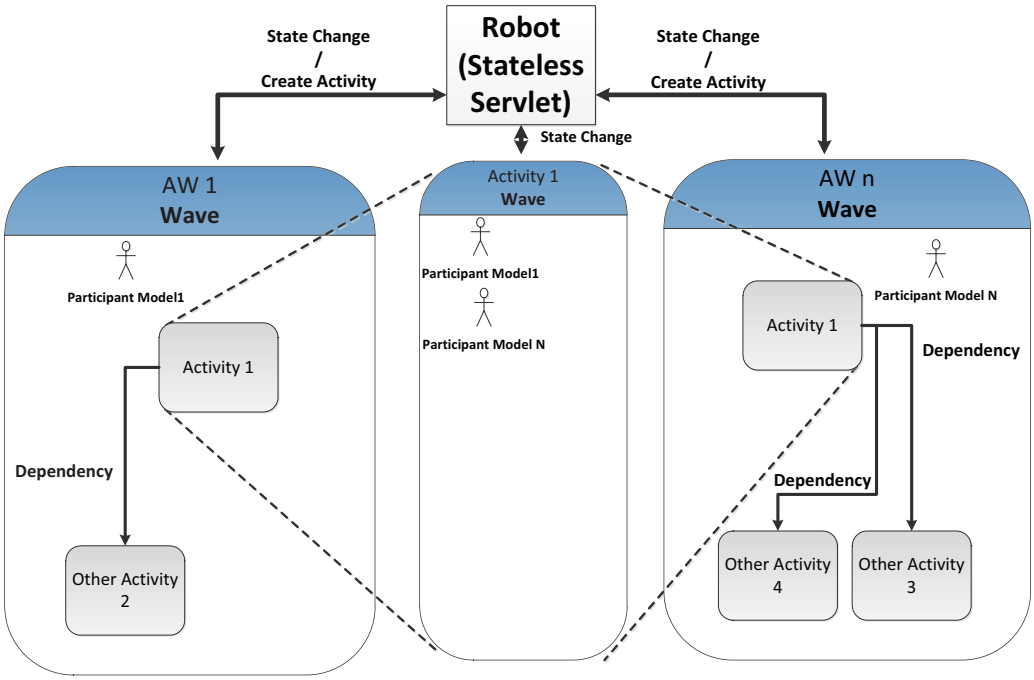


FIGURE 6.3 – Example for the data model of our extension

Wave". The "Activity-Wave" is then replicated by Google Wave™ to the server of the participant. The activity is then shown in the graphical modeling tool "AW-Wave" to its participant with the support of a "Robot" (see following subsections). It can be integrated into the model in the "AW-Wave" by a participant. This means sharing of activities is supported already by Google Wave™, because we exploit the replication mechanisms of "Waves" to different servers. Furthermore, since each "Wave" has a unique identifier, this can be reused for the unique identifier of an activity.

6.3.4 Gadget - Graphical Modeling Tool

We explain in this subsection the graphical modeling tool implemented as a "Gadget" in more detail. Basically, we have implemented two different ways of modeling : using a graphical notation and a table notation. The main motivation for this was that one notation may have advantages over the other in certain situations. For example, the table notation seems to have more advantages when entering quickly many activities and dependencies, whereas a graphical notation can provide a better visual overview to define what happened in the past, what happens now and what is to be executed next.

Figure 6.4 illustrates a screenshot of the "Gadget" using the graphical notation. Activities are represented as rounded rectangles showing the name of the activity and the current state. A state can be changed using the corresponding button on the toolbar on the left or by clicking on the activity and selecting the state change button.

Figure 6.5 illustrates a screenshot of the "Gadget" using the table notation. Activities

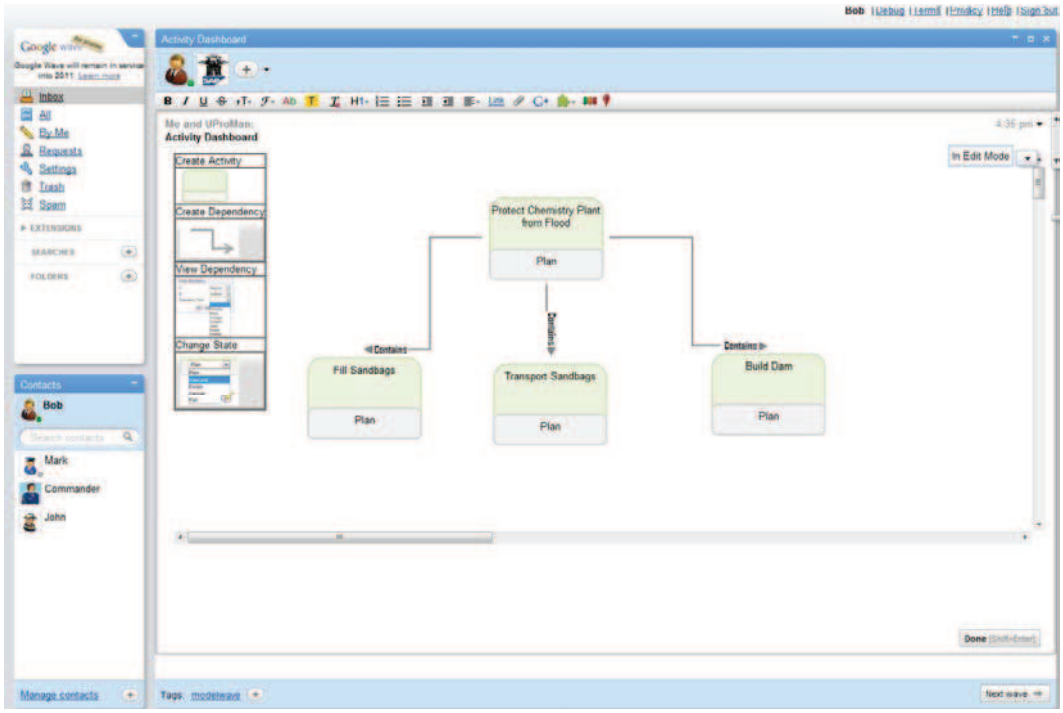


FIGURE 6.4 – Screenshot of the "Gadget" for modeling activities and dependencies : Graphical Notation

are represented as table rows and their dependencies are shown in an extra column. A state of an activity can be changed by selecting the row of the activity and there the new state of the activity. The state changes need to be published to persist them and propagate them to others. This can be done by clicking the button "Publish State Changes".

The "Gadget" implementing the graphical modeling tool loads and stores data in its state as key-value pairs. As mentioned before this data is part of the "Wave", where the "Gadget" has been inserted. It utilizes the following data (amongst other) :

- Model (cf. section 4.2) : activities and their dependencies. This includes references from a modeled activity to the corresponding "Activity-Wave".
- Trace (cf. section 5.4.1) : state changes of activities and associated vector clocks. This can be used by the Gadget to display the current state of activities as well as concurrent state changes of the same shared activity (cf. section 5.4.2).
- Verification results : the results of the verification of the model by the "Robot".
- Violated and unsynchronized dependencies : the results of the detection of violated and unsynchronized dependencies by the "Robot".

The graphical modeling tool does not yet support definition of customized activity types. It uses only one default activity type. This was due to the limited time available and is not related to technical restrictions.

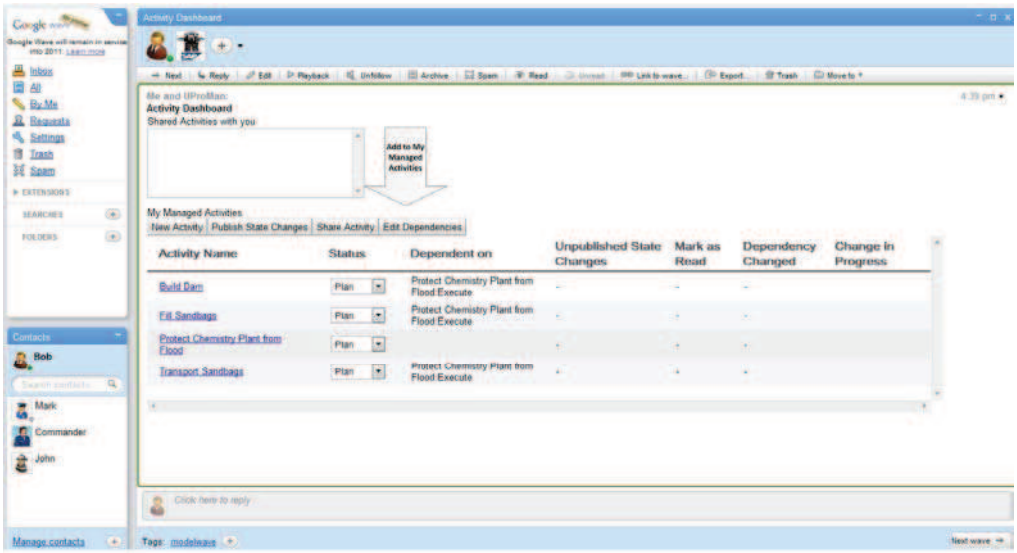


FIGURE 6.5 – Screenshot of the "Gadget" for modeling activities and dependencies : Table Notation

6.3.5 Robot - Distributed Coordination

We explain in this paragraph the functionality of the "Robot" enabling distributed coordination. A "Robot" connects a "Wave" with the "outer world" and can react on events happening in a "Wave". This is exploited by our extension so that it connects data of different "Waves" to enable the following functionality :

- Propagating state changes : Once a state change is stored by a "Gadget", the "Robot" sends it to all "AW-Waves" where the activity is replicated and the vector clock is also submitted to all other "AW-Waves" with which other activities have been shared (according to Definition 17). It stores a reference to the "AW-Waves", where an activity is replicated, in the corresponding "Activity-Wave".
- Verification : once a "Gadget" stores changes of the model, such as adding or removing a dependency, the "Robot" performs verification of the model and returns the results in the "Gadget" data store in the "AW-Wave". It also is able to perform verification of shared activities in different AWs as we have presented it in section 5.3. In this case, it collects the different models stored in different "AW-Waves", merges them and returns the verification results in the "AW-Wave". They are displayed by the "Gadget".
- Detecting dependency violation and unsynchronized dependencies : once a state change occurs, the "Robot" detects if it violates dependencies or leads to unsynchronized dependencies in all "AW-Waves" where the activities of the state change are replicated.

The “Robot” is realized on the Google App Engine [San09], which is a fault-tolerant distributed application server infrastructure. The “Robot” is thus not a centralized entity. This means all concepts presented in chapter 5 are still needed. Another possibility would have been an extension of the Open Wave Federation Protocol, which describes how different servers propagate optimistically information between them. This could lead to a better performance of the implementation and we would not rely on extra servers for the “Robot”. Furthermore, there is no need for a “Robot” as a participant of all “AW-Waves” and “Activity-Waves”. However, this was not possible at the time of implementation due to non-availability of the source code. This has changed with the publication of the open source version Apache Wave.

Verification and detection of violated as well as unsynchronized dependencies could also have been implemented within the “Gadget”. Alternatively, a “mixed” implementation within the “Robot” (or server) and the “Gadget” is possible. This depends also on the Web browser client. However, we suggest doing this at server side to prevent that every client has to do these tasks. Particularly, this can reduce the computational burden from mobile clients.

6.4 Conclusion

We presented in this chapter a Java library implementing the algorithms and data structures presented in chapters 4 and 5. The library abstracts from network or persistence functionality, because the concepts can be integrated into various applications fitting more or less to the concepts presented in the previous chapters. This library is used for implementing parts of the concept as an extension to the distributed collaboration service Google Wave™. The extension has the following characteristics :

- Sharing of activities is done by inviting participants to a special “Wave”, called “Activity Wave”. This type of “Wave” represents activities. Google Wave™ and Apache Wave servers already implement optimistic replication of these “Waves” using the Open Wave Federation Protocol. Furthermore, this protocol provides already a standard for unique identifiers of “Waves” that can serve as unique identifiers of activities.
- A graphical modeling tool implemented as a “Gadget” so that participants can model activities and dependencies. Modeled activities reference the corresponding “Activity Wave”. They can also perform state changes of activities in this tool. Shared activities can be integrated into this model. This modeling tool can be inserted into “Waves” representing the activity workspace explained in chapter five. These “Waves” are called “AW-Waves”. They display results of verification, concurrent state changes of the same shared activity, detecting dependency violation and unsynchronized dependencies. We support two different user interfaces : a table notation and a graphical notation. The table notation enables more rapid creation of activities and dependencies, while the graphical notation can provide a better overview on the dependencies between activities.
- A “Robot” as an automated participant of all “AW-Waves” and “Activity-Waves”. It ensures that state changes are propagated between “AW-Waves”. Furthermore, it verifies, detects dependency violations and unsynchronized dependencies in different

models.

We conclude that it is possible to realize the concepts presented in chapters 4 and 5. We illustrate in Appendix A a case study with the prototype extension. The various parts of the implementation will be optimized in the future. We want to develop a modeling tool for the activity type as well as user assignment, so that different activity types besides the default one can be supported. We plan to utilize the open source version of Google Wave™, called Apache Wave, to implement the concepts presented in chapters 4 and 5 on the server side without relying on a robot. This would also mean higher performance, because we do not rely on extra servers and a robot for propagating state changes of shared activities.

Chapitre 7

Evaluation

Contents

7.1 Introduction 137
7.2 Interviews 138
7.2.1 Selected Comments 139
7.2.2 Discussion 142
7.3 Design of an Experiment 142
7.3.1 Design Details 143
7.3.2 Data Sources 149
7.3.3 Validation of the Experiment Design 151
7.3.4 Discussion 155
7.4 Conclusion 156

7.1 Introduction

We evaluated in the previous chapter how the concepts described in chapters 4 and 5 can be realized technically to demonstrate technical feasibility. We want to validate in this chapter the concepts from a specific disaster response perspective and from a more generic coordination perspective as it has been stated in our research questions in chapter 2. It is expected that the two different perspectives will provide us complementary insights on advantages and perspectives for future research. Due to time and resource constraints, we were only able to perform initial evaluations and further work has to be done for a complete evaluation. Nevertheless, the contributions made here are useful for performing further evaluations and inform how these evaluations can be conducted. We contribute the following items for an initial exploratory evaluation of the concepts presented in chapters 4 and 5 as well as the prototype :

1. Comments by disaster response experts on the concepts presented in chapters 4 and 5 to explore how these domain-independent concepts could fit in their existing work context based on their experience.

2. Design of an experiment for validating tool support addressing the research problems stated in chapter 2. We evaluate the design of this experiment by conducting it three times with different tools. We expect from an experimental evaluation more generalized validation of tool support addressing the research questions, because experiments should be repeatable. Thus, we expect domain-independent results, but we cannot expect disaster response management specific results. Experiments can be seen as complementary to a domain specific evaluation in specific disaster exercises, where there is less control over the evolution of the exercise as well as influencing factors and their relationships. Further experiments need to be conducted to validate tool support and particularly the prototype developed in this thesis.

This chapter is structured as follows : in section 7.2, we present interviews with four domain experts about parts of the concepts described in this thesis. These interviews are different from the interviews that we conducted within the SoKNOS project with different disaster managers. Thus, they are complementary. In section 7.3, we design an experiment for evaluating tool support addressing the research questions. We validate the design by conducting first experiments with students.

7.2 Interviews

We present here selected comments from interviews with four different disaster managers. The main goal of these interviews was to explore how the concepts could fit into their existing disaster response work context based on their experience. We expect that their comments confirm the problems we stated in chapter 2 and that they provide initial feedback on the concepts we developed. This feedback can give perspectives on future research. We interviewed domain experts with several years of experience of managing a disaster response. These experts work for firefighting organizations in France and US as well as a large humanitarian aid organization in Germany. All interviews were conducted via phone and screen-sharing facilities, except the one with the French fire fighter commander, which was conducted in person. The sample of experts we chose was relatively small and the experts belong mostly to one type of disaster response organization. Thus, further interviews are needed to validate the concepts with experts from other type of organizations, such as police or military. However, since the disaster managers we interviewed belong to professional organizations, we still expect valuable comments on the concepts. Furthermore, these disaster managers do not work in isolation, but are in contact with other professional organizations.

An interview started with a general introduction of each other. In the first part of the interview, we presented our motivation (cf. chapter 2) and the concepts (cf. chapters 4 and 5) using an animated Powerpoint prototype. In the second part, the experts commented our motivation and the concepts. The phone interviews last around one hour and the interview in person around three hours. Transcripts have been created from the phone interviews and notes have been taken from the interview in person. The interviews were conducted in December 2009, February 2010, March 2010 and April 2010.

7.2.1 Selected Comments

We present in this subsection selected comments made by the disaster managers. We report the answers from the following perspectives in the subsequent paragraphs :

- Confirmation of the domain study conducted in chapter 2.
- Comments related to the concepts designed in chapters 4 and 5.

Confirmation of Domain Study

We present in this paragraph comments by the interviewed disaster managers confirming the results of the domain study conducted in chapter 2.

The fire fighter commander from Southern California reports similar problems as we have identified in chapter 2 : *"[..] the pile of messages in the inbox, [which contains] the reality as a situation [..] being able to put them in context and update them and coordinate them to create a common picture is the difficulty."*

The same commander provides as another example the incident action plan and its problems : *"We have an incident action plan, which each entity utilizes and keeps track of their activities. [This happens] more or less on a manual basis and [people enter] who is command and what actions are taken. [..] We update it every twelve hours and put [..] future actions and intended actions for the next twelve hours operational period. [..] A lot of this in larger incidents has gone to a web-based electronic data exchange. [This is a] software program, that does not alert you to inter-connection failures [and] that does not tie pieces of information together [..]. My experience with it [is] that it is not more than a collection of emails back and force. [They are] not in chronological order and [the software] does not necessarily go back and correct. [For example a message says] it is action A [..] and in 30 minutes you change your mind now its B, because there is another email [saying] this is B. [..] Later on you may say it [is] C, and A is still in the system. [..] If somebody looks at [the messages] as a sequence they going to think it is A, when it is really C. [..] This [should be] more of a system that can change [..] that has trigger points in it [..] and be more of an open actual working plan in real-time. [It should not be] a collection of information that is gone back and forth."*

The fire fighter commander from Washington states the benefits of awareness of what needs to be coordinated depending on the situation : *"[If different] missions were connected [then] you can, as a result, share resources, share assets, and [you can focus on] what was most important. There will be a priority process set, because of the limited amount of assets that you have, available to do all these different missions."*

However, there are also human-made problems, where any coordination support by a system reaches its limit as the commander of the humanitarian aid organization states : *"[For example, we had] 600 rescue dog teams - 600 ! [and] the result was the location of 6 dead bodies, i.e. there was a total discrepancy between real need and available responders. The responders [..] competed with each other to find some areas or villages, because they came with big media support from home. Now it is expected that they do something at the disaster site and nobody wants to say "hey i was not requested, i just flew over there, and there was no need for me". [..] This is one of the problems of coordination, which - with the increase of stake holders in the field of international disaster aid - gets more and more*

important."

This means if the intent or understanding for coordination by all stakeholders does not exist then any support by technology, such as we presented here, cannot make much difference.

The fire fighter commander from France states that depending on the hierarchy level different types of information representations are needed. For example, people in the field, fighting a disaster, need a very simple view, because they have little time to read and even less time for providing feedback. We provided in this thesis different types of views on the activities and dependencies (cf. chapter 6), but did not cover this problem explicitly.

The commander of the humanitarian aid organization states the need for flexible coordination of activities and gives examples for shifting goals : *"[For example], the problem with evacuation. [..] It is important to collect data who has been brought where. For example, when the people have been brought to different areas, or you have to divide [them], e.g. mother with kid is preferred. You have [also] the group of sick people, which you have to bring to another hospital and then you have care or retirement homes."*

The same commander emphasizes the need for governance : *"For example, [in an] evacuation, who decides that an area is going to be evacuated, where do you bring the people[..]."*

The fire fighter commander from Washington confirms the problem of shifting goals (end states) : *"You are reacting based on an end state, an ultimate objective, an ultimate goal or an ultimate concept of what this incident will look like when it is over. [..] The whole [response] will be tailored to mediating this end state, as opposed just to continuous analysis/reaction/analysis/reaction."*

Another thing which we did not explicitly cover before was mentioned by the commander of the humanitarian aid organization : *"[We do] not only need short information, [e.g.] short situation reports, but also situation pictures in the sense of maps, photos from the field. The command center in Germany can get [then] an overview of the situation [of the Tsunami in Phuket, Thailand in 2004]."*

It is important that the system is flexible, but also that activities of an organization need to be included into the plans of other organizations as the fire fighter commander from Southern California states : *"[The situation] changes rapidly and [..] communication has to go on within the own organization, [..] be input in the system and transported out. The effects of that change [need to] be included in everyone else plans."*

This confirms the need for inter-organizational coordination.

Comments Related to the Concepts Developed in this Thesis

The commander of the humanitarian aid organization described a limitation of a system that requires input of data by people in the field. His experience was based on a deployment of a system for transferring health data to the command center and hospitals. This limitation may also apply to other systems : *"The system never worked in practice, because the doctor or nurse at the disaster site, in particular if you have a lot of injured people, they do not enter anything into the computer. [..] They are busy with the situation and from this angle I think, I do not say there is a limit, but the probability, that you can break it down, to the disaster site is vague. You should consider [different] hierarchical*

levels, and the higher you go, [the more effective is the system].”

The fire fighter commander from Southern California highlights that it is important that people are willing to share some information, but that there is also motivation for sharing information : *“[Your approach] addresses the problem of coordination and sharing goals and objectives from one organization to others. It is that communication, updating of information, progress as far as plan, [..] that inherently seems to be the crux of all problems in the States. [Without sharing] a lot of information stays within each independent organization, and without [it] we duplicate services, we actually implement plans that interfere with the other goal and objective. [..] This is a critical [..], [but also] who does this communication and at what level authority is that communication link established.”*

The concepts of our approach have been mapped by the fire fighter commander from Southern California to existing concepts : *“We use time lines as markers for future action [and] we have what we call trigger points. [This means] when the incident advances to a certain point, it triggers other things. [..] That would fit into your model as well, using time lines, connecting inter-dependencies, and [defining] future actions.”*

He further highlights that it is important that the system does not enforce any behavior : *“[The] system presents the common picture to everyone, if the human user uses it. It is not a restricted system, it denotes the actions and is able to change. That is good. [The users] are not constrained to the system [and] it is a free system. That it needs to be.”*

The fire fighter commander from France states that feedback related to activities is important. The concept can be useful for propagating feedback (e.g. ”Build Dam” is executed), but also for getting feedback related to an activity (e.g. ”Build Dam” failed).

The fire fighter commander from Washington highlights the ability of the system to measure progress, but also coordinate the activities towards a goal and that this goal (end state) can shift :

“there is a couple things [about your approach], [..] it is a good way to measure progress, and the second thing is that you recognize that the end state will change, because of the dynamic situation of the incident you are involved in [..] the end state may need to be modified or you might have intermediate type of objectives.”

Another feature, which was not part of our research questions, was highlighted by the same commander : *“[The system] can put the information together to make sure the incident can be analyzed afterwards for cost recovery. In the States our system works on what resources where deployed and what action were taken for determining your reimbursement from the government. An accurate timeline and accurate record of all activities, those type of things are very important to analyze the event and to do better next time.”*

The fire fighter commander from France sums up where the approach can be useful : complex situation, large area, many organizations involved and it is a long enduring event. For example, storms or floods. This has been also confirmed in the other interviews. For example, the fire fighter commander from Southern California states :

“I think some real importance to the system is connecting different organizations in large incidents, where multiple resources, multiple interconnections need to be established and maintained. This is where the system is going to be useful, not necessarily in a smaller incidents, or single entity responses.”

This means that our concepts could address dynamic situations, such as disaster res-

ponses, and to a lesser extent more routine scenarios, which we described in chapter 2 as emergencies. However, still our system can also be used in those routine scenarios.

7.2.2 Discussion

We presented in this section findings from interviews with four experienced disaster managers. Although the sample of experts is small, the comments show already that our approach addresses real problems faced in a disaster response. However, we could confirm the results from the domain study described in chapter 2. We presented the concepts in an animated prototype and the disaster managers did not use the prototype themselves, but this issue will be addressed by the experiments conducted in the third section. Of course, our approach has to be seen in context of many other systems used in a disaster response (e.g. simulations or resource management systems). It is a tool amongst other tools that can be used to manage the disaster response.

Furthermore, we have to consider cultural differences in what technologies responders use. During our interviews within the SoKNOS project and the domain experts in the US, it seemed that the domain experts in the US were more open to use Internet technologies. For example, one expert from the US mentioned the use of Google Earth for real-time geographical coordination. On the other side, the domain experts we interviewed within the SoKNOS project seemed to prefer more their own data sources and servers.

However, the comments indicate that our approach is relevant for disaster managers and it addresses the flexibility and inter-organizational coordination functionality required in a disaster response. Future interviews are required for a full evaluation of the concepts. For instance, we did not cover explicitly conflict handling mechanisms as they have been described in chapter 5. The presented interviews are specific to the disaster response setting, but we are also interested in complementary domain-independent evaluations with respect to the research questions.

7.3 Design of an Experiment

We design in this section an experiment for evaluating tool support addressing the research questions stated in chapter 2. Due to time and resource constraints, we could not draw validated conclusions about specific tools. However, we could validate the experiment design by successfully conducting three experiments with students. We will demonstrate evidence that we have obtained from the experiment, which shows that the experiment can reproduce some typical coordination problems as we have described them in chapter 2.

Experiments are complementary to specific evaluations in a specific domain, such as disaster response management. However, disaster exercises usually do not have an explicit focus on coordination between different organizations, because such exercises would be very expensive and it is not guaranteed that the organizations who trained together will have to work together in a disaster. Furthermore, disaster specific aspects also play an important role, e.g. how to fight a wild fire. Thus a lot of different complex factors interact in crisis exercises involving several organizations, which are difficult to control, capture and

understand. This makes it difficult to interpret the role of the prototype representing novel concepts for coordination in a disaster response. Experiments can be useful to provide a better understanding how to interpret results obtained from disaster exercises involving our prototype. They can thus be seen as complementary. However, domain-independent experiments themselves do not allow drawing any conclusion with respect to the domain disaster response management. Experiments have also been used in other research on information systems for supporting a disaster response (e.g. [FBZ09, MDvdW10, SFA11]).

Unfortunately, only few experiments evaluating tool support addressing the research questions of this thesis are described in the literature (e.g. [WRZW09, MWR08]). However, they do only partially address dynamic situations as stated here and they also do not consider coordination by people of different organizations. Nevertheless, experiments are seen as important for addressing evaluation of concepts similar to the ones presented here [Pes08]. An important part of the experiment design is to develop a repeatable experiment, so that it can be conducted by others and that the conclusion drawn from one experiment can be validated in further experiments. This also means that the experiments should provide as a outcome data that allows drawing conclusions about tool support. For example, typical coordination problems as we have described in chapter 2 should occur during the experiment. We also need to be able to distinguish coordination problems caused by misunderstandings between the participants of the experiment from coordination problems caused by the tools used. Our research questions aim at the latter problem.

We explain the design details of the experiment in section 7.3.1. Afterwards, we describe how different data sources can be used for evaluating tool support in section 7.3.2. We report the results of three experiments with students and three different tools in section 7.3.3. We discuss them with respect to validation of the experiment design in section 7.3.4.

7.3.1 Design Details

The main task of the participants in the experiment is to coordinate the building of LEGO® [4] objects in different distributed teams. Their coordination is supported by specific tool. The idea for this type of experiment has been inspired from Lego Serious Play™, a method developed in management science. A variant of this method is also used to experiment collaboration on business strategies or web communication design [BVL04, SO08, CMF+09, SG11].

We decided to let the students coordinate the building of a LEGO® object in the "real world" and not a virtual object within the computer. This was more close to our use case, because a disaster response consists of actions in the real world. This type of experiment can easily be reproduced by others, because it requires only a widely available set of simple LEGO® blocks. Additionally, it can easily be extended to incorporate different dynamic situations, different numbers of teams or different types of activities.

In order to make the experiment repeatable, we describe a set of fixed factors, which we did not modify when conducting the experiments and a set of variable factors that we

4. LEGO® is a trademark of the LEGO Group of companies which does not sponsor, authorize or endorse this thesis

modified during the experiment. The fixed factors of each experiment are the setting (or the briefing of it), the number of teams and the type of operations. We varied in each experiment the tool that supported the coordination of activities by different distributed student teams. Although different students participated in each experiment, we expect that they have similar skills and experience with respect to the experiment. Other demographic groups can also be considered for these kind of experiments, such as experienced disaster managers, because the results obtained in experiments with them may differ (cf. [Dö3, Dÿ7]).

In order to reproduce typical coordination problems, no team had a full overview of what needs to be constructed as described in the specifications. Furthermore, we did not prescribe how the students should construct the LEGO® objects, but only what the result should be. Two different LEGO® objects had to be constructed in parallel by different teams. Thus, no team could anticipate the activities of other teams and had to rely on the tools to coordinate.

We provided the students a common language for coordinating the operations for constructing the LEGO® object. The idea is to reduce coordination problems caused by misunderstandings to put the emphasis on coordination problems caused by the tools used by the participants of the experiment.

We describe the basic setting for the experiment in the next paragraph. Afterwards, we present the different teams that needed to coordinate to construct the LEGO® object in the second paragraph. We exemplify the basic operations that have to be coordinated between the teams in the third paragraph. In the last paragraph, we introduce the three different tools used in our experiments.

Fixed Factor : Setting

As mentioned, two different LEGO® objects had to be built during the experiment

- A "power generator" (cf. Figure 7.2)
- A "building to protect the power generator" (cf. Figure 7.1)

The LEGO® objects consist of LEGO® components. LEGO® components themselves consist of standard LEGO® building bricks. The participants had a specification of the LEGO® objects (c.f. Figure 7.1) for the "building to protect the power generator", a specification for the "power generator" (cf. Figure 7.2) and a specification of the LEGO® components (illustrated in Figure 7.3 for the components of the "building to protect the power generator"). These specifications defined what the participants can build. We defined before the experiment different variants of the specifications, so that we could simulate shifting goals by replacing the specifications. These changes have to be introduced externally (e.g. by the researchers). For example, if the specification of the "Power generator" is modified then also the construction of the building to protect the power generator is affected. Furthermore, the relations between activities have to be considered in this case. For instance, some components or objects that are currently built may not be needed anymore. However, the specifications are all rather simple, so that they can be quickly understood by the participants of the experiments.

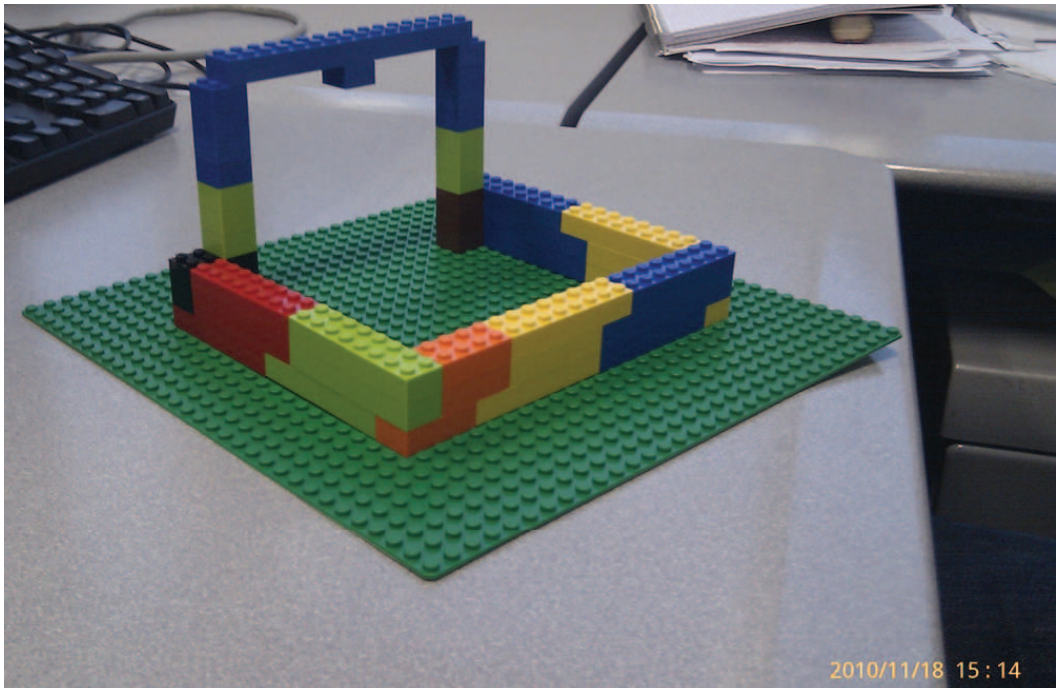


FIGURE 7.1 – Example for a specification of a LEGO® object : building to protect the power generator

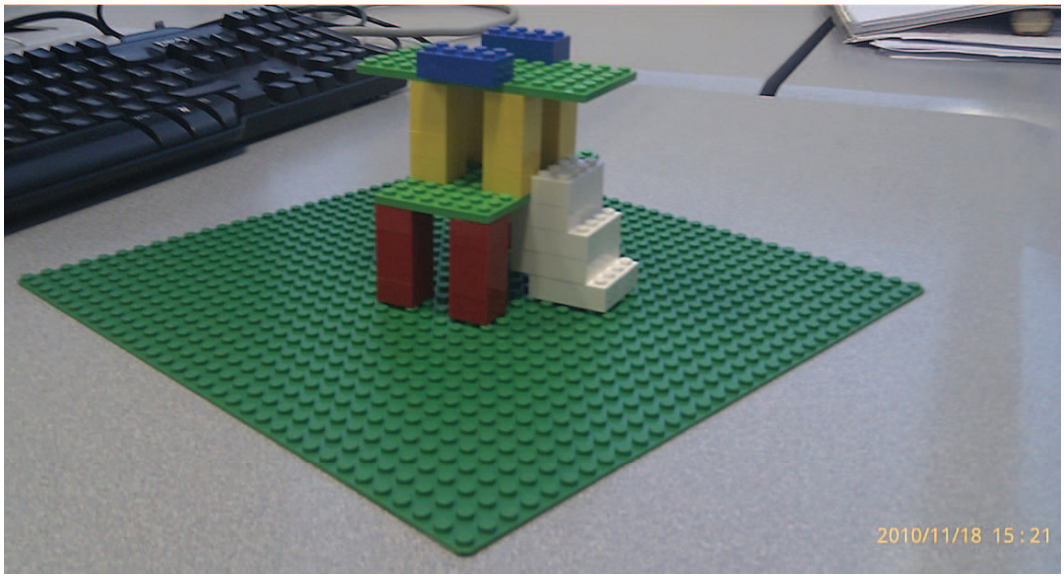


FIGURE 7.2 – Example for a specification of a LEGO® object : power generator

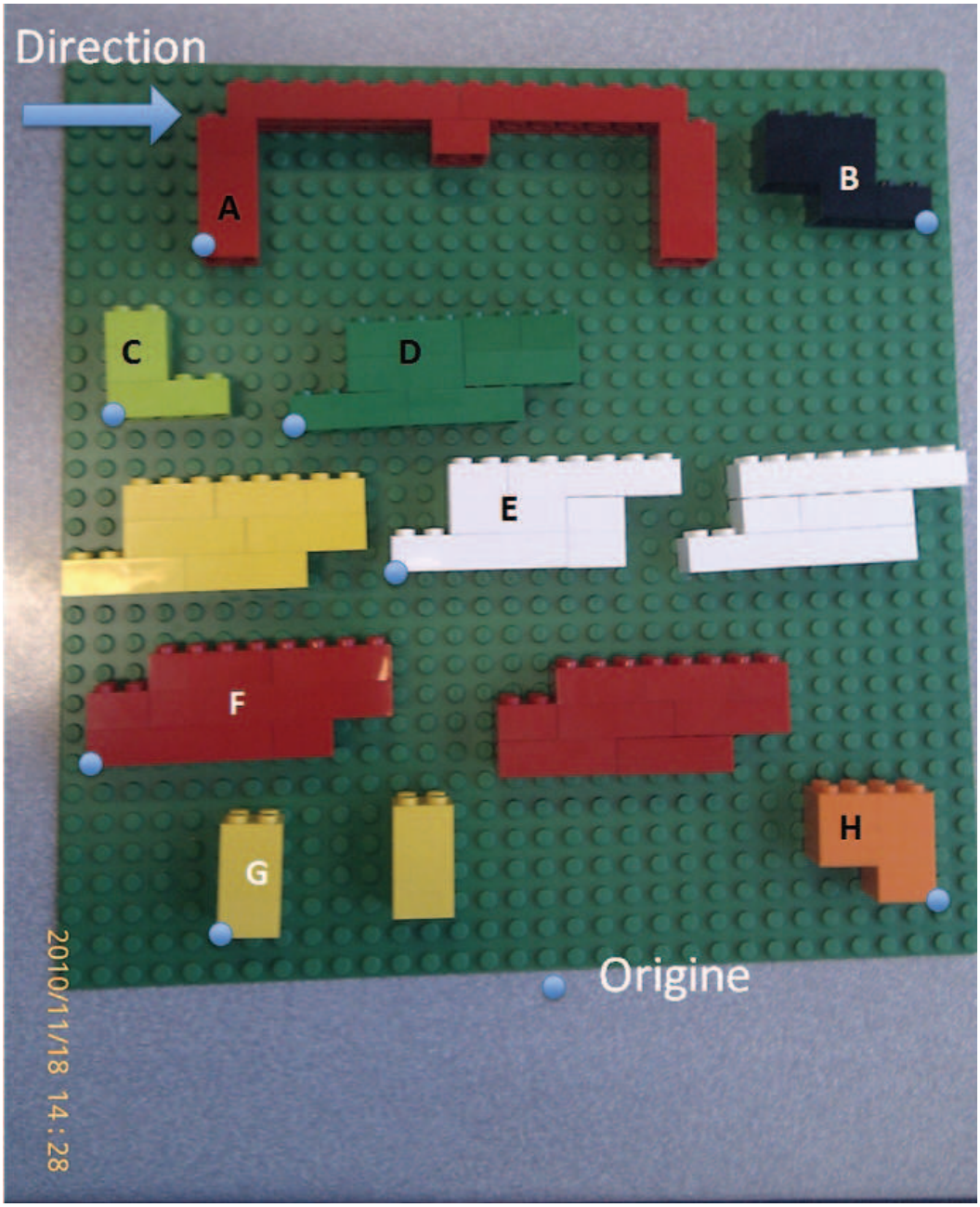


FIGURE 7.3 – Example for a specification of LEGO® components

Fixed Factor : Teams

The participants were divided in five different teams consisting of one or two students. Each team knows about the other team what they were roughly doing, but did not know necessarily the details. For instance, specifications were not known by everyone. This made it more difficult to anticipate for a team what the other team needed from them. Thus, they need to coordinate using the tool assigned to them. We considered for our experiments the following teams :

1. Architect : The architect gives orders to create components at the building site, to assemble components at the assembler site or to transport components between the building site and the assembler site. The architect has a specification of the LEGO® object for the "building to protect power generator" and the corresponding component specification.
2. Engineer : The engineer gives orders to create components at the building site or to transport components between the building site and the assembler site. The engineer assembles the components himself. The engineer has a specification of the LEGO® object for "power generator" and the corresponding component specification.
3. Assembler : The assembler assembles components according to the architect at the assembler site.
4. Builder : The builder builds components out of LEGO® bricks according to the architect or the engineer. The builder has the component specification of the components for "power generator" and "building to protect power generator", but not the specifications of the corresponding LEGO® objects.
5. Transporter : The transporter transports LEGO® bricks between the block site and the building site as well as between the building site and the assembler site according to the architect and the engineer.

Figure 7.4 describes the different locations and relationships between the sites of different teams. They were also physically in different locations in the room, so that they could not see what each other was doing. However, the transporter could be seen when transporting bricks or components. The architect team could not check if the instructions he gave to the assembler lead to a building according to the specification. He had to ask the assembler team to take from time to time a photo of the building. This means coordination was needed to achieve the construction of the LEGO® objects.

Fixed Factor : Operations

We proposed to the participants a set of instructions to introduce a common understanding about the basic operations that needed to be coordinated. We provided it to the students to avoid that they have to establish this themselves, which would have extended the time needed for the experiment and is not directly related to the research questions stated in chapter 2. The rationale behind this is to reduce the risk that coordination problems are introduced by a lack of understanding of each other.

Figure 7.5 illustrates an example for a coordination flow between the different teams using the notation proposed by us. The engineer team asks the architect team to create

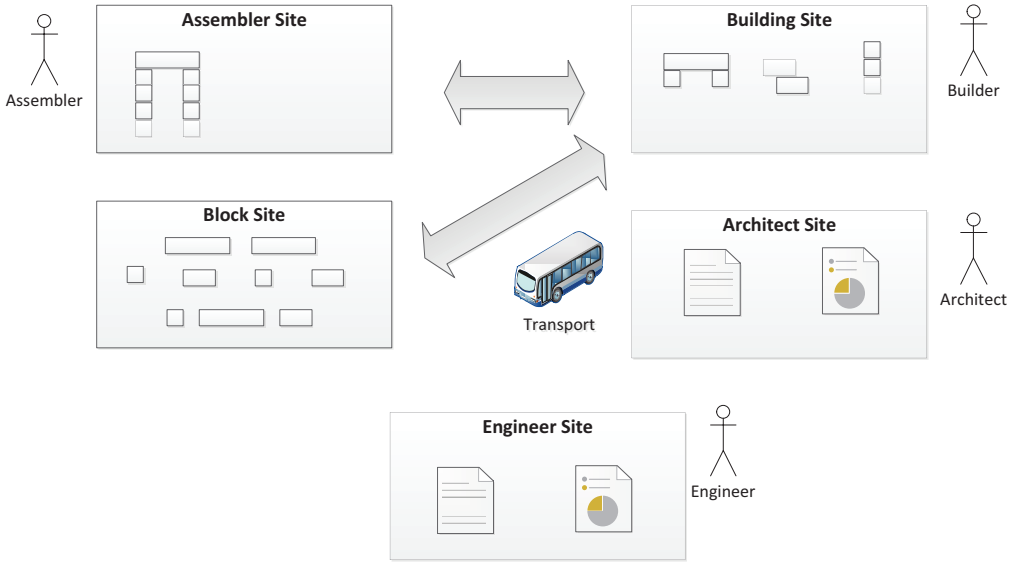


FIGURE 7.4 – Locations of the five different teams

a building for protecting the power plant. The architect team asks the builder team to build the components for the power plant according to the specification. The builder team asks the transporter to provide the LEGO® blocks required for each component. If the component has been constructed then the builder team notifies the architect team. The architect team asks the transporter team to transport it to the assembling site. The transporter team reports to the architect team that the transport has been performed. Finally, the architect team instructs the assembler team to assemble the component to a building. This was an idealized coordination flow that was also made known to the participants of the experiments we conducted. However, this idealized coordination flow does abstract from the fact that many orders can be in parallel or that orders can fail. This has also been observed in our experiments.

Variable Factor : Tools

We used three different tools within three different experiments :

- An instant messaging tool (Google Talk) : Every team had the other teams in their contact list and they could get in contact with any team. Text messages with any content could be exchanged between the participants. This tool was the closest one to current tools used in disaster management, such as phone, fax or e-mail (cf. chapter 2).
- Google Wave (without extension) : Every team had the other teams in their contact list and they could get in contact with any team. The tool could be used as the students want to use it. This tool was used, because our extension was based on Google Wave. This may allow us in future experiments to determine the difference of using Google Wave with and without extension.
- Google Wave (with our extension) : Every team had the other teams in their contact

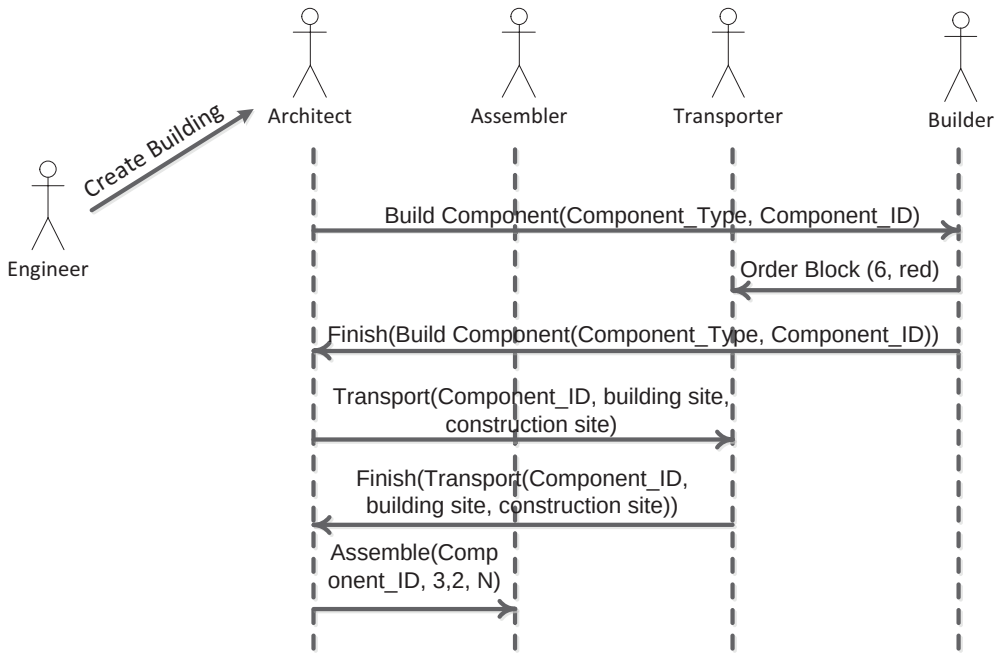


FIGURE 7.5 – Example for a coordination flow between the different teams in an experiment

list and they could get in contact with any team. The tool could be used as the students want to use it. The concepts have been limited as follows to keep more control over the variables of the experiment : There was only one activity type for every activity (cf. Figure 7.6) and no users or governance roles had to be defined. The definition of dependencies was optional. The reason was to keep the experiment simple. Further experiments can of course be extended to incorporate the other aspects. We only made the table-based notation for modeling activities and dependencies available. We expected that many activities would need to be created rapidly.

Of course, other tools can evaluated in the experiments.

7.3.2 Data Sources

Different data sources can be used to evaluate the tool support during the experiment. We present here different types of data sources that can be used in experiments : questionnaire, data from the tools and constructed objects. Other data sources are possible, such as observations, but have not been considered in the experiments that we conduct in the next subsection. We will show that the considered data sources can already provide sufficient insights on coordination problems caused by the tools. We provide more motivation for considered data sources in the subsequent paragraphs.

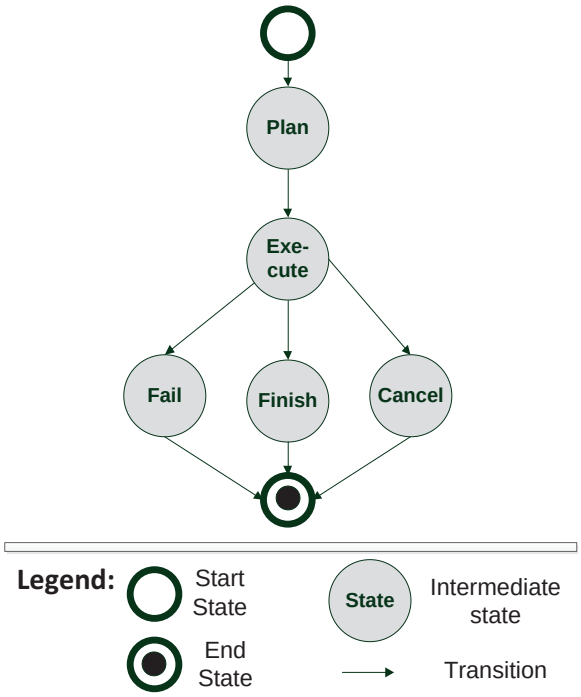


FIGURE 7.6 – Activity type used for all activities in the experiment

Questionnaire

We conducted a questionnaire before and after the experiment. The goal of the questionnaire before the experiment was to determine the knowledge of the students with respect to the tools and concepts used in the experiment. This can help to explain differences between experiments using the same tools. For example, we assume that participants who have a lot of experience in coordination will perform better than participants with little experience (cf. also [Dö3, Dÿ7]). The questionnaire before the experiment covered the following items :

- General Demographic data (e.g. age, sex, study background).
- Knowledge in business process management systems : This item is used to determine how exposed the students have been to the thinking in processes. An answer could be provided on a scale from 1 to 5 (no knowledge to very good knowledge). A free text field could be used to provide further insights.
- Experience in development projects : This item is used to determine to what extent the students have done teamwork. An answer could be provided on a scale from 1 to 5 (no experience to very good experience). A free text field could be used to provide further insights.
- Experience with LEGO®-related concepts : This item has been used to determine the experience and knowledge in constructing objects as it has to be done in the experiment. This was a free text field.
- Experience with chat tools in the daily life : This item has been used to determine the experience with communication tools, such as instant messenger, used in the experiment. An answer could be provided on a scale from 1 to 5 (never to daily). A

free text field could be used to provide further insights.
- Experience with collaboration tools in the daily life : This item has been used to determine the experience with collaboration tools, such as Google Wave, used in the experiment. An answer could be provided on a scale from 1 to 5 (less than once a week to several times per day). A free text field could be used to provide further insights.

The survey after the experiment was used to find out how the tool support affected the coordination between different teams. The goal was to separate team specific aspects (e.g. misunderstandings) from tool specific aspects. The survey after the experiment covered the following items :

- Main difficulties faced in the experiment : This item is used to determine what where the main issues as perceived by the students. It was a free text item.
- What tool has been used (Instant Messenger/Google Wave without extension/-Google Wave with extension).
- Advantages of the tool used : This item was a free text item.
- Disadvantages of the tool used : This item was a free text item.
- What were the problems faced with the architect/builder/assembler/engineer/transporter : These were free text items.

Tools

Another source for evaluating tool support is the data stored by the tools used for the experiments. It is important that the tools capture all conversations between the teams and that the participants use only the tool for coordination. Furthermore, the tool should allow reconstructing the view the participants had on the activities that need to be coordinated, so that problems can be better understood. This was true for all tools we used in the experiments (cf. next subsection). The instant messenger tool logged all conversations of the participants. Google Wave with and without our extension persisted all conversations in "Waves" (cf. chapter 6).

Constructed Objects

The objects constructed by the participants can be benchmarked against the specification. Divergence from the specification may hint at coordination problems caused by tools.

7.3.3 Validation of the Experiment Design

We briefly explain the results from our conducted experiments. These are preliminary results to evaluate the design of the experiment. Further experiments have to be conducted to evaluate tool support.

At the beginning of each experiment, the students have been briefed about the experiment setting, what they have to do and the tools they have to use. The briefing took around one hour. The experiments themselves took between two and three hours. In total

18 students participated in three experiments. All students are currently doing university-level studies. Twelve of the students have followed a course on component architectures at the École Supeérieure d'Informatique et Applications de Lorraine (ESIAL). Six further students have been doing an internship at SAP. All students did not know the goal of the experiment, but only the main objective to build the LEGO® objects, the setting and a tool assigned to them as we have explained before.

We present in the subsequent paragraphs results obtained from the following data sources : questionnaire, data of the tools and constructed LEGO® objects.

Pre-Experiment Questionnaire

Most of the students (12) have been Master students. There were only few undergraduate students (3) and few PhD students (3). Two students did other kind of studies. All students did their studies in the area of information technology.

Most of the students (11) had very few knowledge about business process management (own rating from 1 to 2). Four students estimated that they have an average knowledge (rating of 3). Three students said they have a good knowledge of business process management (rating of 4 to 5).

The experience in development projects was generally low. Fifteen of the students had only little experience (from 1 to 2). Two students had medium experience (average of 3) and one student had good experience (a rating of 4). Most of the students commented that they had experience from university projects together with other students. One had experience from a project in the industry.

Several students reported that they used LEGO® when they were young, but not anymore now. The others did not use LEGO® or related concepts.

Many students used chat tools every day to several times a week (16). Only few used it less (1 use it once a week and 1 never). Most students report that they use instant messengers from various companies.

This was different from the experience with collaboration tools. Many students (11) use them less than once a week to once a week. Four students use them several times a week. Only three students use them on a daily basis. The most frequent example given by them was Google Docs. One student mentioned Google Wave, but commented also that it is used rarely.

Post-Experiment Questionnaire

We deduce from the answer of the post-experiment questionnaire that the students could identify coordination problems related to misunderstanding and problems caused by the tools. In the beginning of the experiments there has been a short phase where the students had to get used to the tools and the experimental setting, which caused misunderstandings. For example, a specification was sometimes misunderstood in the beginning. The students also described specific problems related to the tools. For example, it was difficult for the students using the chat tool to get an overview of the already executed activities and the ones that are currently executed. However, in order to assess how they affected coordination, we need to analyze the data stored by the tools.

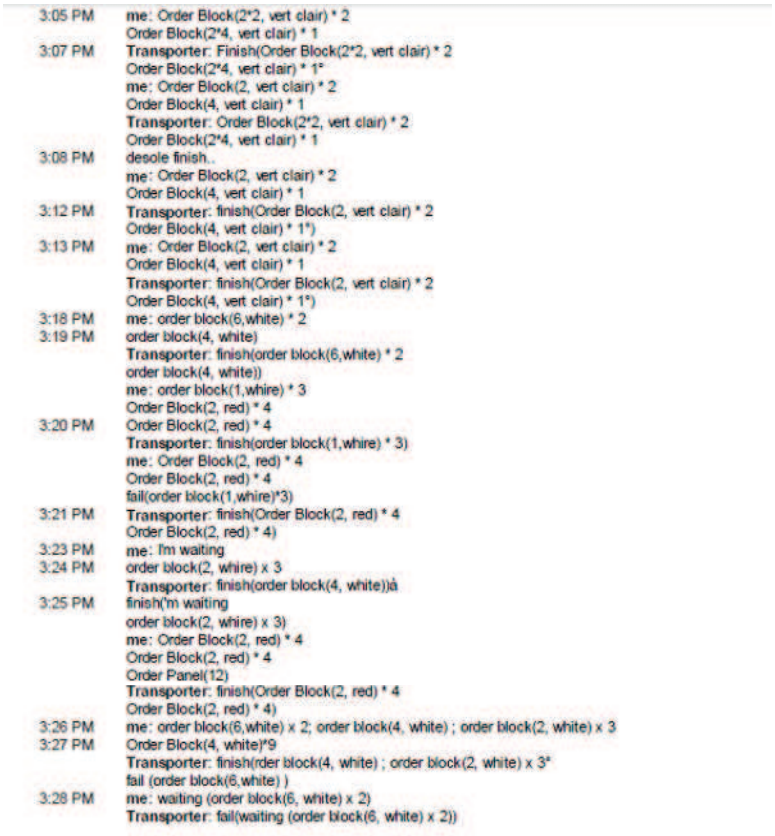


FIGURE 7.7 – Result example 1 : chat overview of activities

Data from the Tools

We have also some evidence from the tools supporting the comments made by the students. This includes for the instant messenger the conversations and for Google Wave the data persisted in the Wave. We describe now some examples for coordination problems that occurred during the experiments when using different tools [5].

We illustrate in Figure 7.7 an excerpt from the chat tool from the perspective of the builder team. It shows that it is difficult to get quickly an overview of the transporting activities and what has already been done or what is currently done. It is possible to derive performance indicator from this data. For example, we can calculate the number of activities that have never been finished or the number of activities that have been repeated unnecessarily several times.

Figure 7.8 shows an example for coordination problems that occurred during the experiment. The architect team got confused, because the builder confirms twice that the same blue piece has been finished, but it did not receive any information on the status of the white piece.

We present in Figure 7.9 another example for a coordination problem that occurred

5. Please note that the terminology in earlier experiments may deviate from the terminology described before. For example, we used the term "piece" instead of "component". However, this did not affect the experiment.

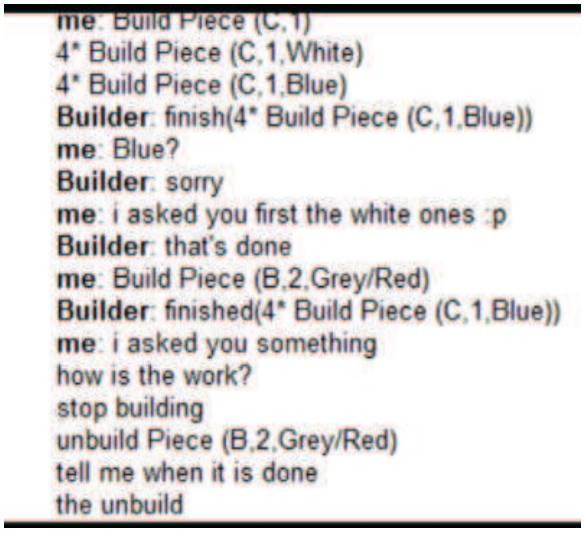


FIGURE 7.8 – Result example 2 : chat coordination problems

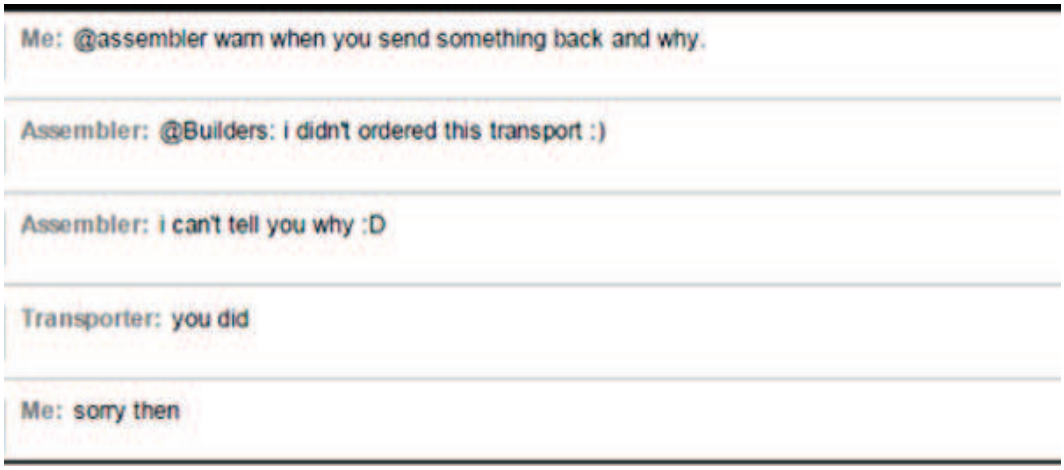


FIGURE 7.9 – Result example 3 : wave coordination problems

during the experiment when using Google Wave. The architect team did not remember that it gave an order for a transport and thought it has been given by the assembler team. They invited the transporter team to the Wave to resolve the problem. This also meant that they disclosed all exchanges with the assembler team to the transporter team.

Constructed Objects

We illustrate in Figure 7.10 two constructed objects by different teams. It can be observed that they do not comply to the specification (see Figure 7.1). For example, the object presented in the upper half has a hole in its wall. In the lower half, we see another constructed object by another student group. There, the wall has to be stabilized "manually" so that it does not collapse. However, other parts have been constructed according to specification.

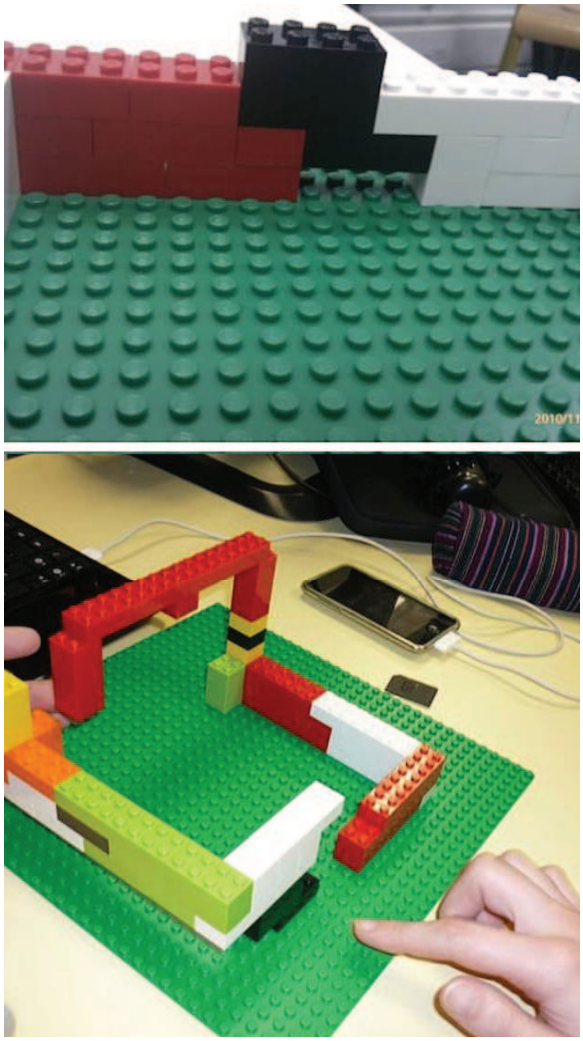


FIGURE 7.10 – Two constructed objects by different groups

Unfortunately, we cannot deduce that these errors are the result of coordination problems caused by the tools. It could also have been a misunderstanding. Further interviews with the Assembler team and Architect team could clarify the cause. However, the results show that the participants could coordinate the building of a LEGO® object and that they are able to provide as a result something similar to the specification.

7.3.4 Discussion

The results obtained from the conducted experiments provide us insights on the experiment design. Further experiments have to be conducted to validate the concepts implemented in our prototype. The results indicate that the experiment can be repeated, because we were able to repeat it several times. The resulting objects created by a coordinated effort of different teams indicate that the experiment is feasible. We were able to reproduce typical coordination problems caused by the tools. For example, we could show that there have been double efforts and inaction. Finally, we illustrated how coordination

problems caused by misunderstandings can be separated from coordination problems caused by the used tools. However, this is not always straight-forward and the results need to be carefully evaluated from different angles.

We acknowledge that the experiments can only provide initial insights. However, we have demonstrated that the experiment design can be used to evaluate tool support addressing the research questions. Only few experiment designs have been proposed in the literature and they do not address exactly the research problems stated in this thesis [WRZW09, MWR08]. It should be noted that we provide only the design of an experiment that investigates coordination problems caused by tools. The results cannot be directly transferred to the disaster response management domain, but they may help to interpret the results obtained, for instance, in disaster exercises. For instance, the results in experiments and exercises can be compared to get insights on how the tools affected coordination in a disaster exercise.

7.4 Conclusion

We presented in this chapter the evaluation of the concepts described in chapters 4 and 5 and their implementation. The main goal of the evaluation was to validate them with respect to the research questions. Particularly, we described two different approaches :

1. Interviews and presentation of the concepts to four experienced disaster managers. Further interviews have to be conducted to validate the results.
2. Design of an experiment for evaluating tool support addressing the research questions stated in chapter 2. We conducted several experiments to validate the design of the experiment. Further experiments have to be conducted to validate the results with respect to tool support.

The first approach allowed us to explore how the concepts could suit to the work context of the disaster managers. The results indicate that the concepts can be useful for the disaster managers and helped us to find the scope of our approach with respect to its deployment. Further interviews with different organizations need to be conducted to validate the results.

Given the results of our three conducted experiments, we conclude that the experiment design can provide meaningful results for evaluating tools with respect to the research questions under the premise that they are repeated several times. More experiments have to be conducted to analyze the advantages and limitations of tools used for coordination.

Chapitre 8

Conclusion and Perspectives

We investigated in this thesis problems related to the **coordination of activities by people of different organizations in dynamic situations**. The disaster response management domain has been a critical example for this. Coordination is crucial to avoid inaction, double efforts or conflicting actions. It is driven by humans based on their judgment of the situation. Current tools used by disaster managers of the SoKNOS project [DPZM09], such as phone, fax or e-mail, show limitations. All the information what has been done, what is currently going on and what are the next steps is hidden in unrelated messages. Furthermore, the dynamic situation leads to shifting goals of the different organizations with different priorities for what needs to be done. This has to consider the relations between activities. Different organizations are autonomous and thus sharing of information about their activities with other organizations is restricted due to privacy, regulatory, strategic or other reasons.

Our analysis of the state of the art revealed that current technology is limited either with respect to coordination of activities in dynamic situations or coordination by people of different organizations. Furthermore, an integration of these aspects has drawn only little attention. We supported this also with evidence from a process modeling effort together with end users in the SoKNOS project, such as fire brigade and police.

8.1 Contributions

We designed in this thesis a process-based information system to support coordination of activities by people of different organizations. Such an approach emphasizes the explicit definition of relations between activities. This has to take into account autonomous organizations. The approach is motivated by a realistic scenario in disaster response management and discussions with domain experts, such as police and fire brigade. We addressed the research problems with the following contributions :

– A **framework for coordination of activities with temporal dependencies**. Activities, dependencies and governance roles can be modeled. We allow a richer set of dependencies than the state of the art, but still support simple temporal dependencies. They can be visualized according to a timeline to highlight what has been done, what is currently done and what are the next steps. We do not require

specifying the model completely, but it can be simply extended as the situation evolves. A model consisting of activities and temporal dependencies can be verified for correctness in predictable time. This enables correct ad-hoc modeling of activities and dependencies. Deviations from what is defined in the model and how activities are executed can be detected and displayed to the user. This highlights the impact of shifting goals in dynamic situations leading to a reassessment of activities and their relations by different stakeholders. The user can then communicate with the stakeholders of the involved activities to resolve any issues.

– An **extension of the framework to the inter-organizational level**. Activities can be shared between people and replicated in the models defined in different activity workspaces of different organizations. Sharing and integration is voluntary. Thus the organizations can keep their autonomy and they can keep control over privacy, regulations as well as disclosure of strategic intent. We presented a protocol for verifying an activity workspace containing shared activities. We described how state changes of shared activities can be replicated optimistically to different activity workspaces. We address two conflicts leading to diverging views on the activities and dependencies. We explained how they can be detected and handled to ensure eventually a converging view. Thus, organizations can ensure a partial shared common view on the shared activities and dependencies. They can be warned about diverging views of the situation. This is beneficial for coordination and difficult to achieve with current tools used for coordination in a disaster response as presented in chapter 2 or with the information systems presented in chapter 3.

The concepts have been **implemented as a proof-of-concept** in a Java library and partly as an **extension to the distributed collaboration service Google Wave**. The extension of the collaboration service shows how the concepts unfold in the context of other tools that can be used for disaster response management, such as collaborative text-editing, maps, images or video. Furthermore, it provides a distributed infrastructure as required by our concepts. Each organization can have their own server under their control. Additionally, it offers some functionality described by our concepts out-of-the-box, such as optimistic propagation of changes to different servers. However, still our concepts are necessary for detecting and handling conflicts as well as the functionality of the framework. The implementation shows that they are technically feasible.

The framework and the extension to the distributed inter-organizational level **have been commented by four disaster managers**. They described how the concepts could suit to their work context in disaster response management. We **designed an experiment** to evaluate tool support addressing the research problems stated. The goal of the experiment design is to gather repeatable results in a controlled environment about different tools addressing the research questions. The results expected by an experiment are complementary to those obtained, for example, in disaster exercises or in an enterprise work context, because they are limited with respect to repeatability, they usually have different goals than the research goal stated here and involve a lot of complex interacting factors. More precisely, we believe that the results of the experiments can inform the evaluation of tool support in disaster exercises or in an enterprise context. We tested the experiment design together with students and different tools including our prototype. We successfully repeated the experiment several times. Thus, we can obtain in future experi-

ments results with respect to validation of the concepts implemented in the prototype.

8.2 Perspectives for Future Research

During our interviews with domain experts, implementation efforts and experiments with students, we had the opportunity to evaluate different parts of our concepts. We identified based on this some opportunities for further research in this thesis.

Framework for Coordination of Activities in Dynamic Situations

We focused in our framework on temporal dependencies. It would be interesting to explore if there are other types of dependencies that can be useful for coordination in dynamic scenarios (e.g. spatial dependencies [Gue89] or resource dependencies). At the moment, the integration of our concepts into Google Wave only allows to deal with them implicitly by describing in "Activity-Waves" the required resources, material or maps of the situation. However, deviations from model and how resources are deployed are not directly highlighted to the user. This research direction would also need to answer how relevant it is to define these types of dependencies in a more structured way as proposed in our framework and how relevant are they for coordination on the inter-organizational level.

Inter-Organizational Coordination

We explained in chapter 5 how activities can be shared between people of different organizations and how a converged view on activities as well as dependencies can be ensured. However, this does not mean that everybody has the same understanding of activities and understands the context in which they are executed. This means that there needs to be an approach to establish such a context, e.g. a generic common understanding of each other's processes. We have already started to integrate reference process models and our prototype extension [FWC+11]. These reference process models have been developed together with domain experts [Tuf06, Ble10] in humanitarian supply chain management operations. They describe activities as well as roles and responsibilities. The integration in our prototype may lead to an improved coordination, because they describe standardized knowledge of the domain codified in processes. They support a common ground between different organizations. They can also help to establish the social network for sharing activities based on defined roles and responsibilities. Finally, these reference process models may also be used to guide users when coordinating by proposing them activities to be executed next or with whom to share activities. This may be also useful in the case of handling concurrent state changes to show to the user, which follow-up actions are not relevant anymore, because the state of an activity has been modified to ensure a converging view (e.g. from the state "Fail" to the state "Cancel").

Furthermore, we want to explore in more detail how unused shared activities can be deleted or activity workspaces can be removed. For example, in the experiments the students changed the state of shared activities to "Obsolete" to mark them as deleted. However, they were not physically deleted. An activity workspace in the prototype could

be somehow deleted, because a "Wave" can be marked as deleted and is then deleted physically by the server after a certain time. However, there might be still shared activities in this workspace that are not in their end state and thus may still receive state changes. A simple solution would be that the other workspaces do not send state changes after they could not reach a workspace. If this workspace has not been deleted then it would synchronize with the other workspaces after a certain time. It should be avoided in this case to reuse unique identifiers to avoid conflicts of workspaces or activities with the same identifier.

Implementation

An interesting aspect of the implementation is the visualization of the process over time. At the moment, the graphical capabilities are limited with respect to that and manual effort by the user is required to visualize the activities over time. We described already two different forms of representing activities and dependencies to the user. We plan to extend algorithms developed for visualization of information in graphs for more advanced visualizations [HMM00]. The idea is that the user can enter the process in the table-based notation and different visualizations can be automatically generated from this information. The most obvious visualization would be to display activities along a timeline to highlight what has been done, what is currently going on and what are the next steps.

We also discovered that the state of activities can be enriched with further information, such as a text to describe why an activity has failed. This may be done by extending the graphical modeling tool "Gadget", so that this information can be entered and displayed. Furthermore, filtering mechanisms can be useful (e.g. hiding activities that are in their end state). The graphical modeling tool can also be extended to support modeling of different activity types. Another aspect is to improve the performance by implementing it as an extension to the Open Wave Federation Protocol based on the open-sourced version of Google Wave called Apache Wave [Apa].

Evaluation

We want to present the approach to more domain experts of different organizations to validate our initial results. Ideally this should be done on the international level, because different backgrounds and cultures can provide different views on our approach. Furthermore, we want to extend the interviews to other types of organizations besides fire brigades. Thus, we expect more generalized findings related to our concepts with respect to the domain. The findings can also be used to improve the implementation. Finally, more experiments are required to validate the concepts implemented in the prototype and to explore further aspects of the prototype. They can support complementary evaluation in disaster exercises or in an enterprise setting.

Summary

Similar to all technologies, people using the technologies presented here need to have an appropriate training and understanding about how to leverage them. However, we have first indicators from our interviews and student experiments that the concepts can easily

be understood by non-experts. Our preliminary research results in interviews with domain experts and the student experiments led us to the conclusion that the concepts in this thesis are highly relevant not only for disaster response management, but also for other domains with similar characteristics. Thus further research would be beneficial.

We expect that our contributions are also useful for other scenarios involving dynamic inter-organizational processes in business networks, for responding to complex enterprise incidents, such as security response or complex support processes, or for distributed development projects. The proposed algorithms and data structures seem to complement the emerging paradigm of unstructured processes [OR09] or artifact-centric processes / case management [vdAWG05, dM09a, BCK+07]. We particularly expect that the concepts presented here can contribute to the inter-organizational dimension of those paradigms.

Annexe A

Case Study Prototype

We illustrate in this chapter, a case study demonstrating some functionality of the prototype. The case study is based on the motivational example in chapter two. It is divided into the following steps :

1. Modeling of activities
2. Modeling of dependencies
3. Sharing of activities
4. Concurrent state changes of the same shared activity
5. State change of shared activities and dependency violation
6. Verification

Each step is explained and illustrate with screenshots in the following sections. We describe the case study using the graphical modeling tool with the graphical notation (cf. chapter 6).

A.1 Modeling of Activities

We illustrate modeling of activities in the prototype in Figure A.1. It shows the "AW-Wave" of Bob, the military commander. He is responsible for protecting a chemistry plant from a flood. Bob has modeled four response activities : "Protect Chemistry Plant from Flood", "Fill Sandbags", "Transport Sandbags" and "Build Dam". They are all in state "Plan".

A.2 Modeling of Dependencies

Bob decides to model dependencies between the activities. He wants to define that only when the strategic activity "Protect Chemistry Plant from Flood" is executed all operational activities, such as filling sandbags, transporting sandbags and build dam can be executed. Figure A.2 illustrates this. Bob has defined a dependency "contains" between the state "Execute" of the strategic activity "Protect Chemistry Plant from Flood" and the state "Execute" of the operational activities "Fill Sandbags", "Transport Sandbags" and "Build Dam".

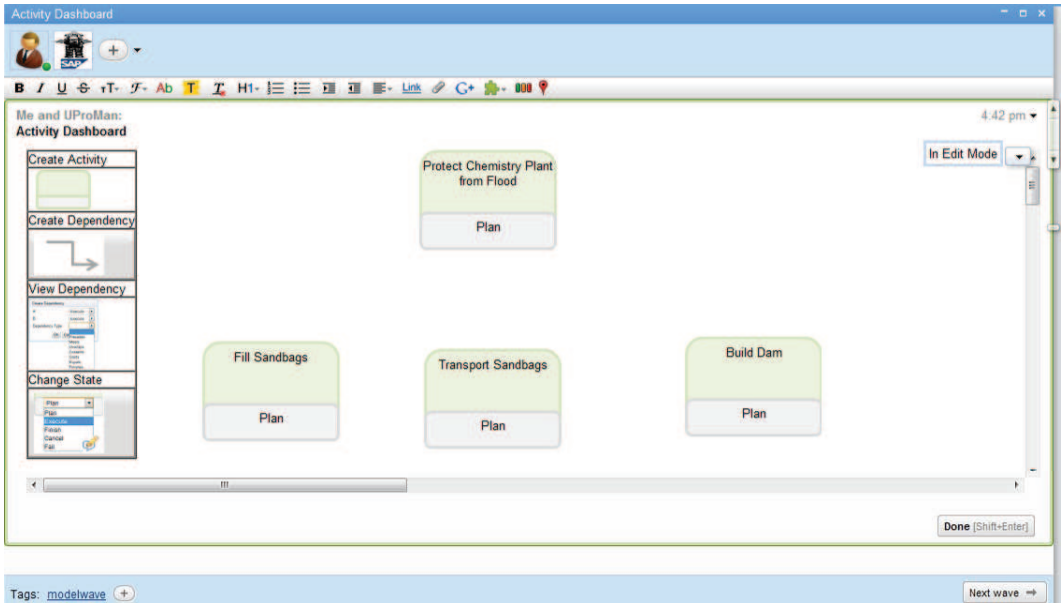


FIGURE A.1 – Screenshot : modeling activities

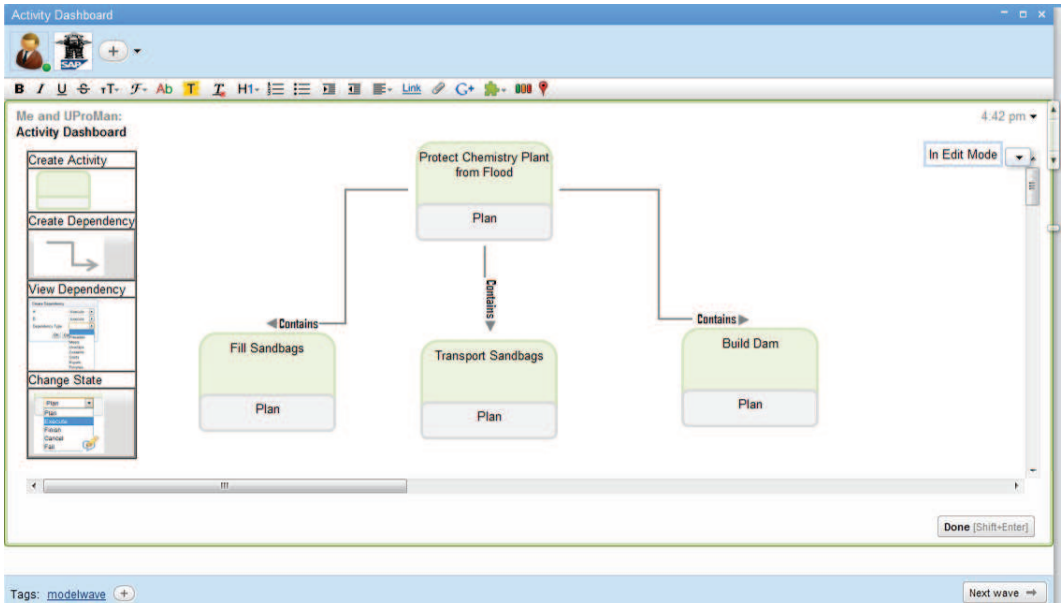


FIGURE A.2 – Screenshot : modeling dependencies

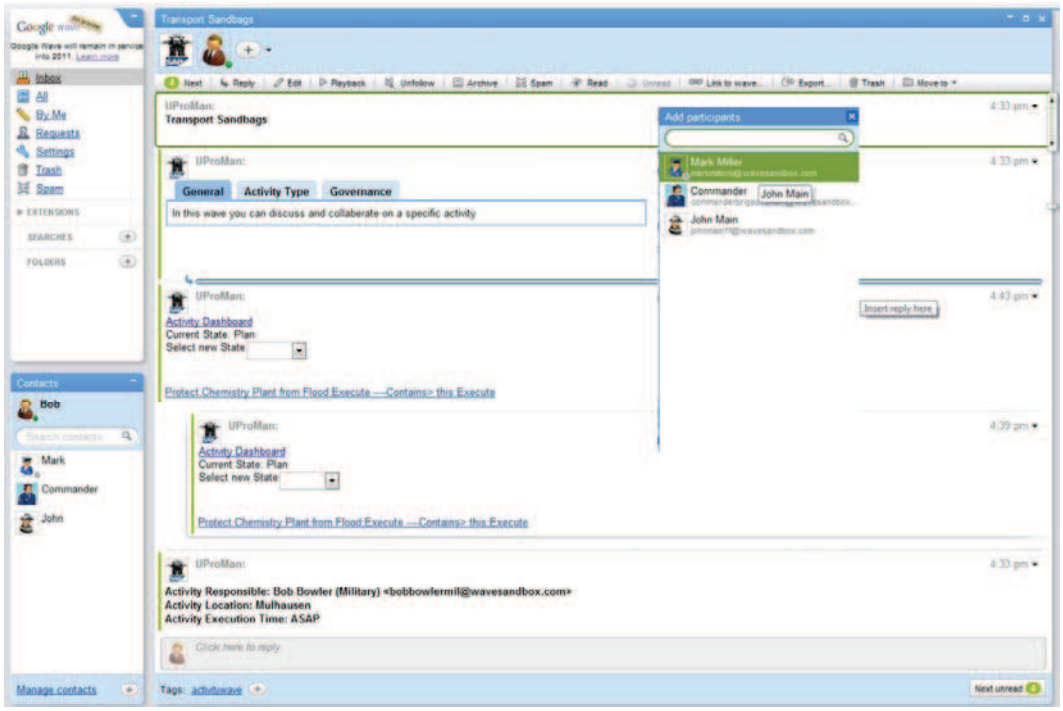


FIGURE A.3 – Screenshot : sharing activities

A.3 Sharing of Activities

As described in the use case in chapter two, the fire fighter need sandbags from the military. Bob knows the fire fighter commander John from previous disaster exercises. John is responsible for protecting the residential area from a flood. He needs also to build a dam with sandbags. Bob shares the activity “Transport Sandbags” with John to keep him informed about the status of this activity. He enters the “Activity-Wave” of the activity “Transport Sandbags” and adds John to this “‘Wave” (see Figure A.3).

Figure A.4 illustrates the “AW-Wave” of John. He has already modeled the activity “Protect Residential area from Flood”. On the bottom left, we can see that the shared activity “Transport Sandbags” is shown in a list and can be integrated into the model by John.

John decides to integrate the shared activity “Transport Sandbags” in his “AW-Wave”. Figure A.5 shows two activities. The one previously modeled by John and the shared activity. John establishes a dependency “contains” between the state “Execute” of the strategic activity “‘Protect Residential Area from Flood” and the shared activity “Transport Sandbags”.

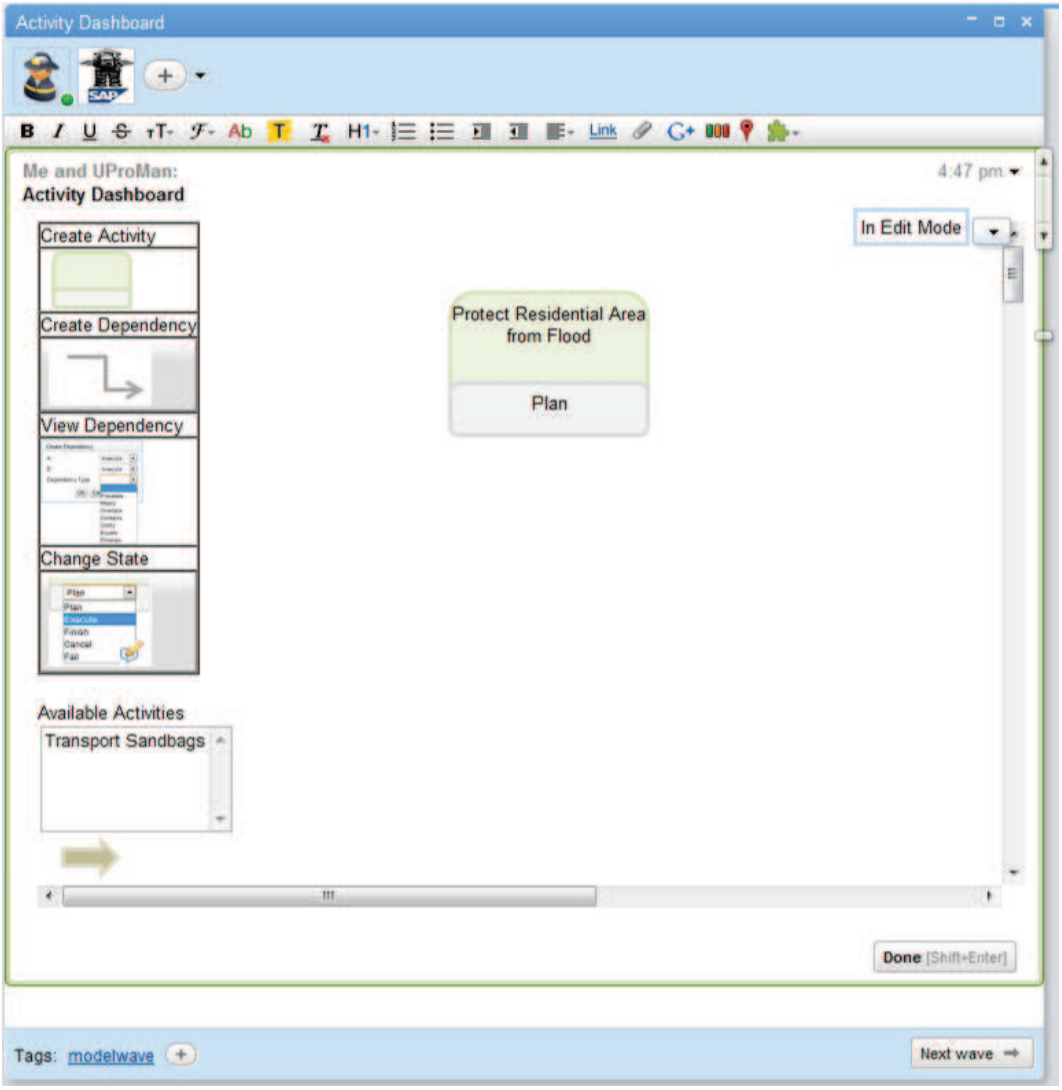


FIGURE A.4 – Screenshot : shared activity can be integrated into model

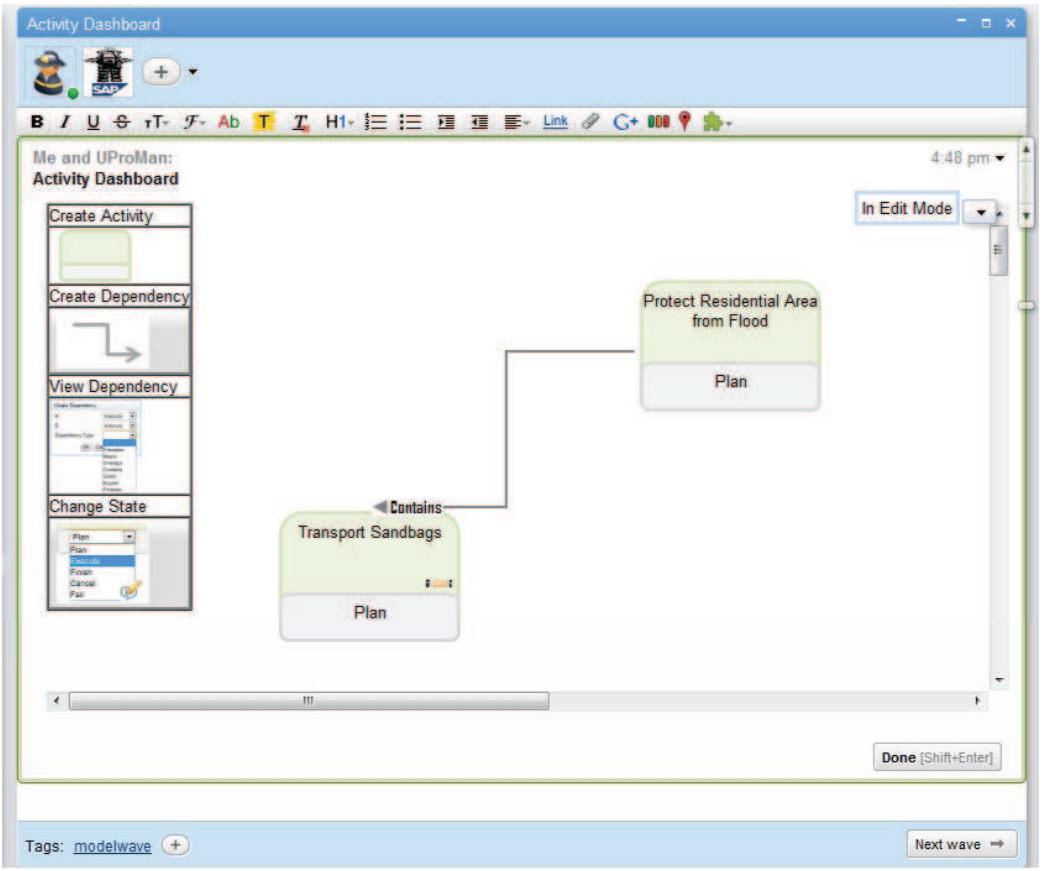


FIGURE A.5 – Screenshot : shared activity is integrated into model

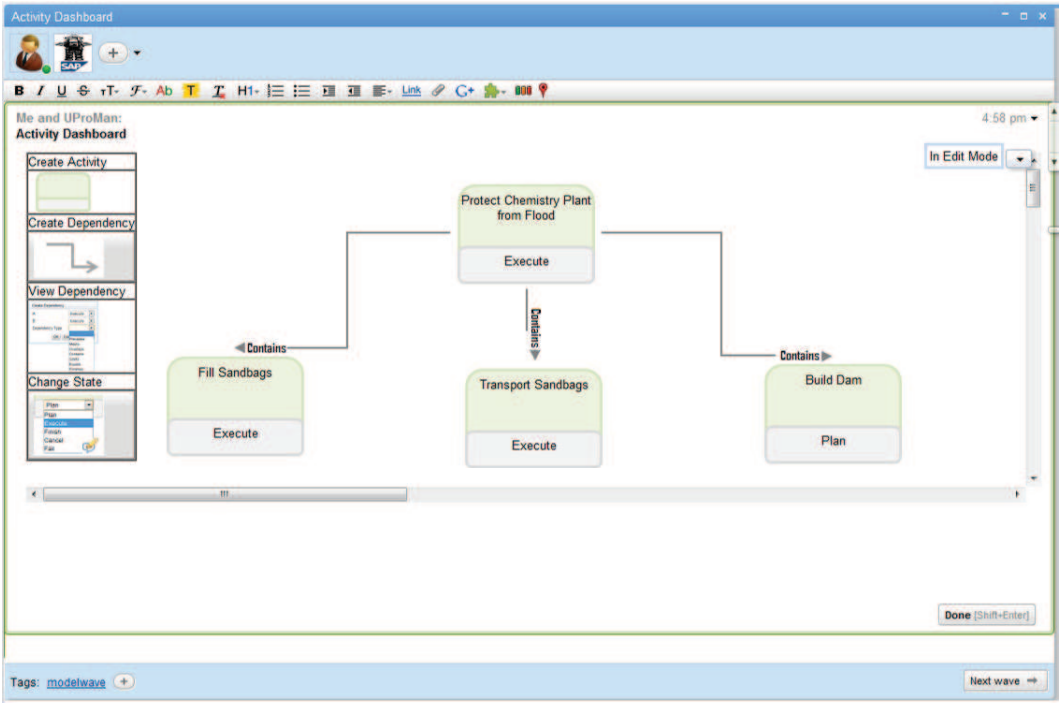


FIGURE A.6 – Screenshot : state change of shared activity

A.4 State Change of Shared Activities and Dependency Violation

Bob decides to start execution of the strategic activity "Protect Chemistry Plant from Flood". He changes the state by clicking on the activity and selecting state change. He changes the state to "Execute". He also changes the activity "Fill Sandbags" and the shared activity "Transport Sandbags" to the state "Execute". Figure A.6 shows his model and the three activities in state "Execute". The activity "Build Dam" remains in state "Plan".

Figure A.7 shows the "AW-Wave" of John, the fire fighter commander. His strategic activity for protecting the residential area from a flood is still in state "Plan", but the activity "Transport Sandbags" has already been changed to state "Execute" by Bob. This leads to a violation of a dependency and this is highlighted to John with a message and the violated dependency is colored red. John can now, for example, enter the "Activity Wave" of "Transport Sandbags" to discuss with Bob about this, change his strategic activity "Protect Residential Area from Flood" into state "Execute" or wait.

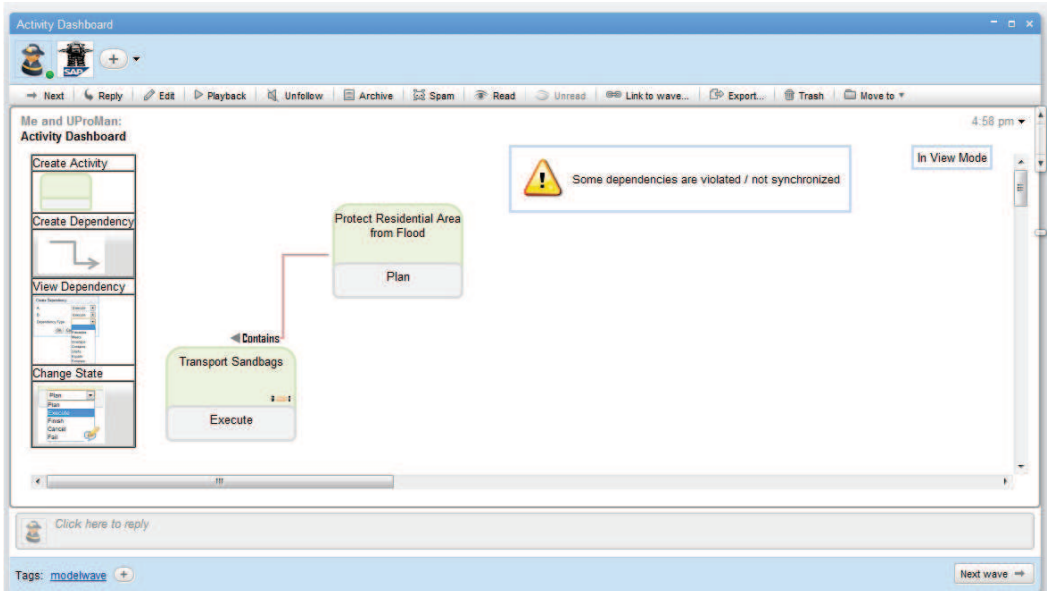


FIGURE A.7 – Screenshot : state change of shared activity leads to dependency violation in another activity workspace

A.5 Concurrent State Changes of the Same Shared Activity

Bob now changes the shared activity “Transport Sandbags” into state “Fail”, because the highway to the residential area is flooded. Concurrently, John changes this shared activity to “Cancel”, because the residential area is going to be flooded and another solution needs to be found. We illustrate in Figure A.8 the “AW-Wave” of Bob. He sees that a concurrent state change of the shared activity “Transport Sandbags” has happen. The resolution has lead to a common state “Cancel” for the shared activity “Transport Sandbags”. He has currently a dialog box open, where the two concurrent state changes “Fail” and “Cancel” are listed.

A.6 Verification

We demonstrate in this section the verification capabilities of the prototype. Mark, the police commander, responsible for the evacuation of the residential area in case of a flood, has modeled the model in his “AW-Wave” illustrated Figure A.7. Five activities are modeled there : “Protect Residential Area from Flood”, “Fill Sandbags”, “Transport Sandbags”, “Build Dam”, “Evacuate Residential Area” and “Warn people”. Three dependency “contains” are established between the activity “Protect Residential Area from Flood” in state “Execute” and the corresponding operational activities “‘Fill Sandbags”, “Transport Sandbags” and “Build Dam” in state “Execute”. Another dependency “contains” has been established between the activity “Evacuate Residential Area” in state “Execute” and the activity “Warn people” in state “Execute”. A dependency “precedes” has been established

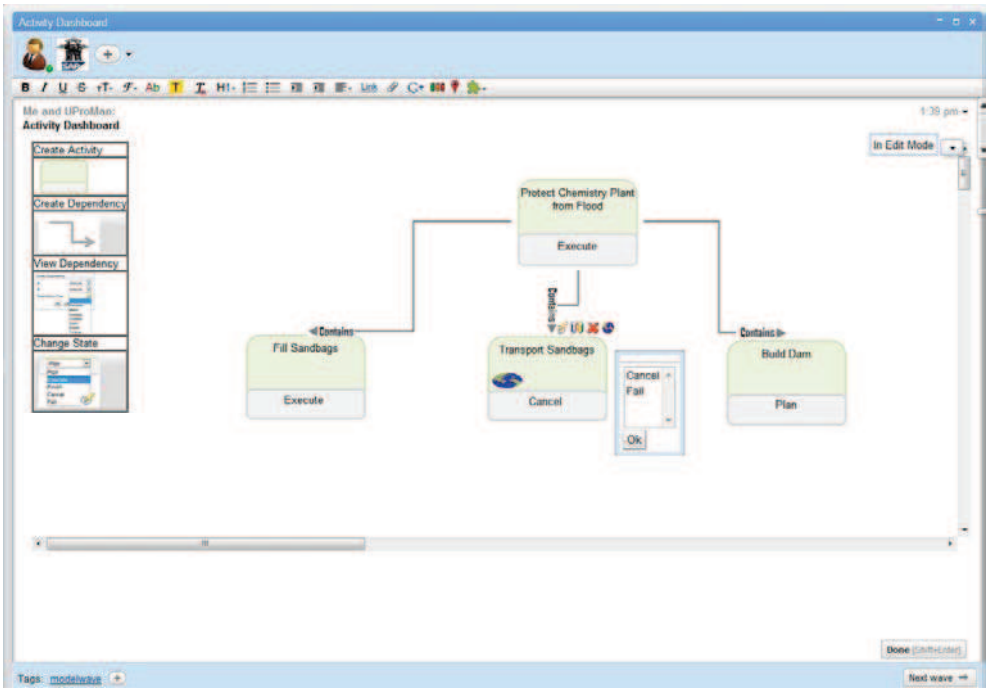


FIGURE A.8 – Screenshot : concurrent state change of the same shared activity

between the state "Fail" of activity "Build Dam" and the state "Execute" of the activity "Build Dam".

Mark now decides that the evacuation of the residential area should be executed before protection measures from the flood are executed. He models a dependency "precedes" between the state "Execute" of the activity "Evacuate Residential Area" and the state "Execute" of the activity "Protect Residential Area from Flood". This leads to an inconsistent model and is shown to Mark with a warning message. Mark can now remove this dependency from the model to have a consistent model or remove other dependencies (e.g. the one between the activities "Build Dam" and "Warn People").

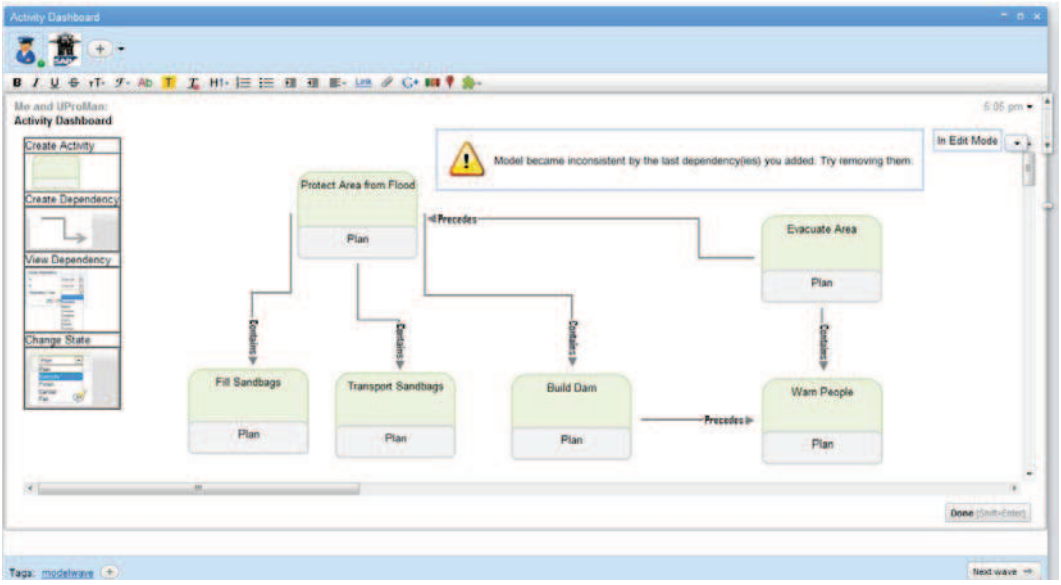


FIGURE A.9 – Screenshot : verification of a model in an activity workspace

Annexe B

Allen's Path Consistency Method

We present in this chapter some minor changes to Allen's path consistency method [All83]. They do not change the concept, but will help to correctly implement it.

The basic algorithm is described in Algorithm 7. We use the following notation in the algorithm :

- CN describes a constraint network
- n is the number of nodes in CN
- C_{ij} describes a constraint between node i and node j in CN
- P_{ij} describes the position of the constraint between node i and node j in CN
- S_{ij} describes a temporary variable containing an updated constraint between node i and node j
- $r_a \in C_{ij}$ is a temporal interval relationship defined in the constraint C_{ij}
- $T(r_a, r_b)$ is a entry in the transitivity table (cf. Table B.2) between the temporal interval relationships r_a and r_b. For example, $T(precedes, precedes) = precedes$.

The idea behind this algorithm is as follows : Based on the constraint c_{ab} between the nodes A and B and the constraint c_{bc} between the nodes B and C, we can derive the possible constraint cp_{ac} between the nodes A and C as they are defined in Allen's proposed transitivity table (cf. Table B.2, which is processed by Algorithm 8). If there is not at least one common temporal interval relationship in the possible constraint cp_{ac} and the existing constraint c_{ac} between the nodes A and C then the temporal constraint network is inconsistent (i.e. there is an empty solution). Allen's original algorithm did not check for satisfiability, but only tries to find a possible solution to the constraint network. Thus, we added conditions that check if an empty solution is generated.

The corrections we made to this algorithm basically ensure that only changed constraints are re-evaluated by the path consistency method.

We changed the following entries of the transitivity table (cf. Table B.2 and Table B.1 for its notation) so that the path consistency returns correct results :

1. Contains (row) → During (column) : all relationships (in the original : overlaps, overlapped by, during, contains, equals)
2. Overlaps (row) → Overlapped By (column) : overlaps, overlapped by, during, contains, equals, starts, started by, finishes, finished by (in the original : overlaps, overlapped by, during, contains, equals)

Algorithm 7: Verifying Satisfiability of a Temporal Constraint Network (Based on [All83])

input : A constraint network CN
output: true=CN is satisfiable or false=CN is not satisfiable

// start processing of the constraint at position 0,0 in the constraint network
NodeList.add(P_{00})
while NodeList.*size*>*0* **do**
  $P_{ik} \leftarrow$ NodeList.fetch
  for $j \leftarrow 0$ **to** n **do**
    $S_{jk} \leftarrow C_{jk} \cap$ `deriveConstraints` (C_{jk},C_{ik})
    if $S_{jk} = \{\}$ **then**
      // no valid constraint possible
      `return` false
    // the following line has been corrected compared to the original algorithm
    if $S_{jk} \subset C_{jk}$ **then**
      // constraint has been updated
      NodeList.add(P_{jk})
      $C_{jk} \leftarrow S_{jk}$
    $S_{ij} \leftarrow C_{ij} \cap$ `deriveConstraints` (C_{ik},C_{kj})
    if $S_{ij} = \{\}$ **then**
      // no valid constraint possible
      `return` false
    // the following line has been corrected compared to the original algorithm
    if $S_{ij} \subset C_{ij}$ **then**
      // constraint has been updated
      NodeList.add(P_{ij})
      $C_{ij} \leftarrow S_{ij}$
`return` true

Algorithm 8: deriveConstraints(C_1,C_2) - Derives the possible constraint C_{12} given existing constraints C_1 and C_2 from the transitivity table (Based on [All83])

input : C_1
input : C_2
output: C_{12}

$C_{12} = \{\}$
for *each* $r_a \in C_1$ **do**
  for *each* $r_b \in C_2$ **do**
    $C_{12} = C_{12} \cup T(r_a, r_b)$
`return` C_{12}

3. Overlapped By (row) → Overlaps (column) : overlaps, overlapped by, during, contains, equals, starts, started by, finishes, finished by (in the original : overlaps, overlapped

TABLE B.1 – Constraint notation and abbreviation

Constraint Name	**Inverse Name**	**Abbreviation**	**Abbreviation Inverse**
Precedes	Preceded by	p	p^{-1}
Meets	Met by	m	m^{-1}
Starts	Started by	s	s^{-1}
Overlaps	Overlapped by	o	o^{-1}
Finishes	Finished by	f	f^{-1}
During	Contains	d	d^{-1}
Equals	Equals	e	e

by, during, contains, equals)

We found out about the incorrect table entries when testing our implemented algorithms. For example, the following consistent constraint network would have been detected as inconsistent with the old table. We assume three nodes "A", "B" and "C". There is a constraint "starts" between node "A" and node "B". Another constraint "contains" is defined between the node "B" and node "C". This is not due to the general approach, but due to some minor mistakes in the entries of the table.

An example for Detecting Inconsistencies in a Temporal Constraint Network Using the Path Consistency Algorithm.

We illustrate the path consistency algorithm using the example constraint network illustrated in Figure B.1 and its matrix version in Table B.3. We take as an example the constraint "o" between node "A" and node "B" as well as the constraint "o" between node "B" and node "C". We look up in the transitivity table (cf. Table B.2) what constraint is possible between node "A" and "C" in this case : "$p\ o\ m$". Since there is only one basic constraint, we need not to do further table lookups. In the next step, we have to intersect the possible constraint with the existing constraint o^{-1} between node "A" and "C" : $\{pom\} \cap \{o^{-1}\} = \{\}$. We see that this leads to an empty set and this means that the constraint network is inconsistent.

TABLE B.2 – Corrected transitivity table (based on [All83])

N_1 and N_2/N_2 and N_3	p	p^{-1}	d	d^{-1}	o	o^{-1}	m	m^{-1}	s	s^{-1}	f	f^{-1}	e
precedes (p)	p	all	$p\ o\ m\ d\ s$	p	p	$p\ o\ m\ d\ s$	p	$p\ o\ m\ d\ s$	p	p	$p\ o\ m\ d\ s$	p	p
preceded by (p^{-1})	all	p^{-1}	$p^{-1}\ o^{-1}\ m^{-1}\ d\ f$	p^{-1}	$p^{-1}\ o^{-1}\ m^{-1}\ d\ f$	p^{-1}	$p^{-1}\ o^{-1}\ m^{-1}\ d\ f$	p^{-1}	$p^{-1}\ o^{-1}\ m^{-1}\ d\ f$	p^{-1}	p^{-1}	p^{-1}	p^{-1}
during (d)	p	p^{-1}	d	all	$p\ o\ m\ d\ s$	$p^{-1}\ o^{-1}\ m^{-1}\ d\ f$	p	p^{-1}	d	$p^{-1}\ o^{-1}\ m^{-1}\ d\ f$	d	$p\ o\ m\ d\ s$	d
contains (d^{-1})	$p\ o\ m\ d^{-1}\ f^{-1}$	$p^{-1}\ o^{-1}\ d^{-1}\ m^{-1}\ s^{-1}$	all	d^{-1}	$o\ d^{-1}\ f^{-1}$	$o^{-1}\ d^{-1}\ s^{-1}$	$o\ d^{-1}\ f^{-1}$	$o^{-1}\ d^{-1}\ s^{-1}$	$d^{-1}\ f^{-1}\ o$	d^{-1}	$p^{-1}\ s^{-1}\ o^{-1}$	d^{-1}	d^{-1}
overlaps (o)	p	$p^{-1}\ o^{-1}\ d^{-1}\ m^{-1}\ s^{-1}$	$o\ d\ s$	$p\ o\ m\ d^{-1}\ f^{-1}$	$p\ o\ m$	$o\ o^{-1}\ d\ d^{-1}\ e\ s\ s^{-1}\ f\ f^{-1}$	p	$o^{-1}\ d^{-1}\ s^{-1}$	o	$d^{-1}\ f^{-1}\ o$	$d\ s\ o$	$p\ o\ m$	o
overlapped by (o^{-1})	$p\ o\ m\ d^{-1}\ f^{-1}$	p^{-1}	$o^{-1}\ d\ f$	$p^{-1}\ o^{-1}\ m^{-1}\ d^{-1}\ s^{-1}$	$o\ o^{-1}\ d\ d^{-1}\ e\ s\ s^{-1}\ f\ f^{-1}$	$p^{-1}\ o^{-1}\ m^{-1}$	$o\ d^{-1}\ f^{-1}$	p	$o^{-1}\ d\ f$	$o^{-1}\ p^{-1}\ m^{-1}$	o^{-1}	$o^{-1}\ d^{-1}\ s^{-1}$	o^{-1}
meets (m)	p	$p^{-1}\ o^{-1}\ m^{-1}\ d^{-1}\ s^{-1}$	$o\ d\ s$	p	p	$o\ d\ s$	p	$f\ f^{-1}\ e$	m	m	$d\ s\ o$	p	m
met by (m^{-1})	$p\ o\ m\ d^{-1}\ f^{-1}$	p^{-1}	$o^{-1}\ d\ f$	p^{-1}	$o^{-1}\ d\ f$	p^{-1}	$s\ s^{-1}\ e$	p^{-1}	$d\ f\ o^{-1}$	p^{-1}	m^{-1}	m^{-1}	m^{-1}

N_1 and N_2/N_2 and N_3	p	p^{-1}	d	d^{-1}	o	o^{-1}	m	m^{-1}	s	s^{-1}	f	f^{-1}	e
starts (s)	p	p^{-1}	d	p o m d^{-1} f^{-1}	p o m	o^{-1} d f	p	m^{-1}	s	s s^{-1} e	d	p m o	s
started by (s^{-1})	p o m d^{-1} f^{-1}	p^{-1}	o^{-1} d f	d^{-1}	o d^{-1} f^{-1}	o^{-1}	o d^{-1} f^{-1}	m^{-1}	s s^{-1} e	s^{-1}	o^{-1}	d^{-1}	s^{-1}
finishes (f)	p	p^{-1}	d	p^{-1} o^{-1} m^{-1} d^{-1} s^{-1}	o d s	p^{-1} o^{-1} m^{-1}	m	p^{-1}	d	p^{-1} o^{-1} m^{-1}	f	f f^{-1} e	f
finished by (f^{-1})	p	p^{-1} o^{-1} m^{-1} d^{-1} s^{-1}	o d s	d^{-1}	o	o^{-1} d^{-1} s^{-1}	m	s^{-1} o^{-1} d^{-1}	o	d^{-1}	f f^{-1} e	f^{-1}	f^{-1}
equals (e)	p	p^{-1}	d	d^{-1}	o	o^{-1}	m	m^{-1}	s	s^{-1}	f	f^{-1}	e

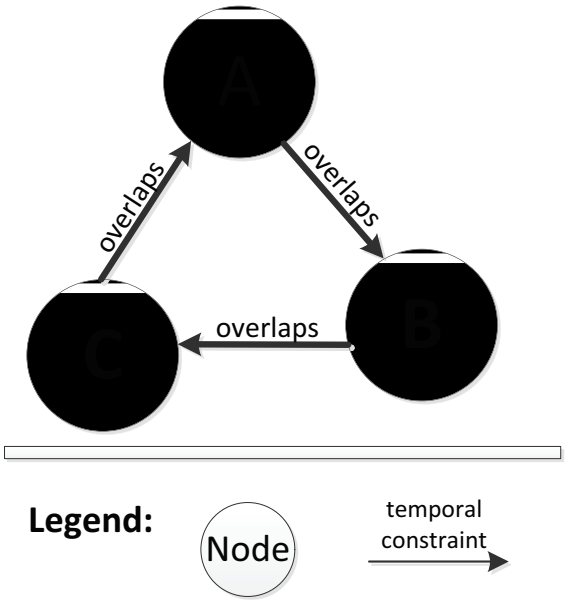


FIGURE B.1 – Example for an inconsistent constraint network

TABLE B.3 – Matrix notation of inconsistent constraint network

	A	**B**	**C**
A	e	o	o^{-1}
B	o^{-1}	e	o
C	o	o^{-1}	e

Annexe C

Dependency State Machines

We illustrate in this chapter the dependency state machines (cf. chapter 4) representing temporal dependencies. Figure C.1 describes the notation used in the subsequent figures.

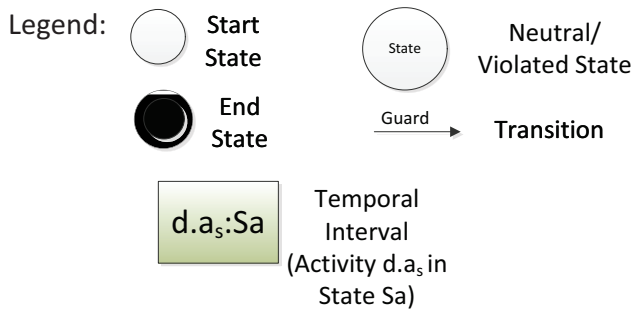


FIGURE C.1 – Dependency State Machine Notation

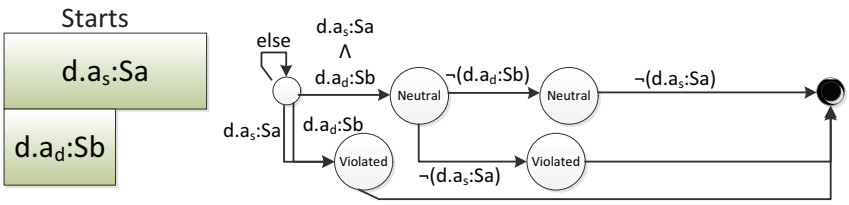


FIGURE C.2 – Dependency State Machine : Starts

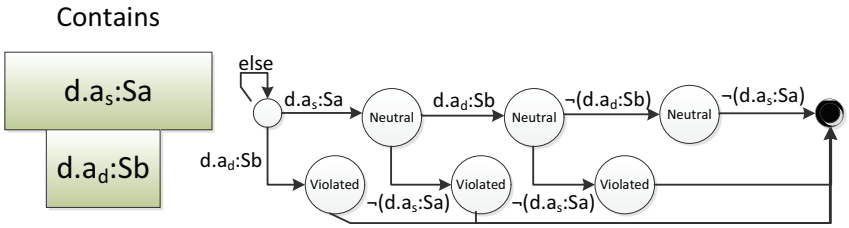


FIGURE C.3 – Dependency State Machine : Contains

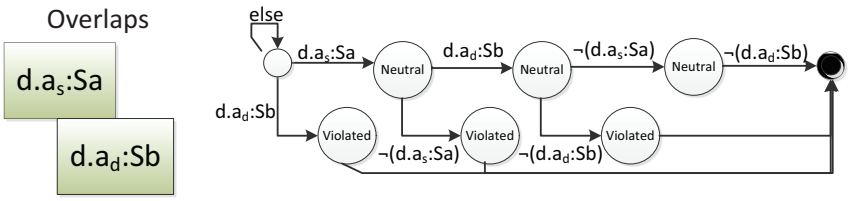


FIGURE C.4 – Dependency State Machine : Overlaps

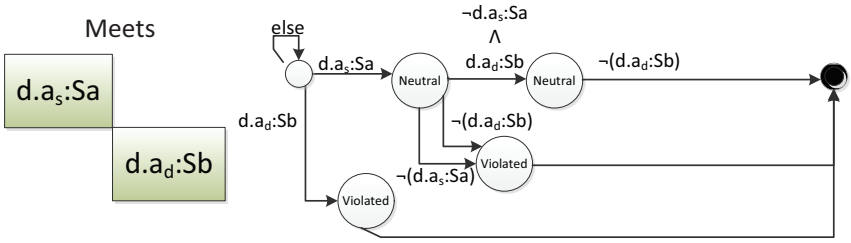


FIGURE C.5 – Dependency State Machine : Meets

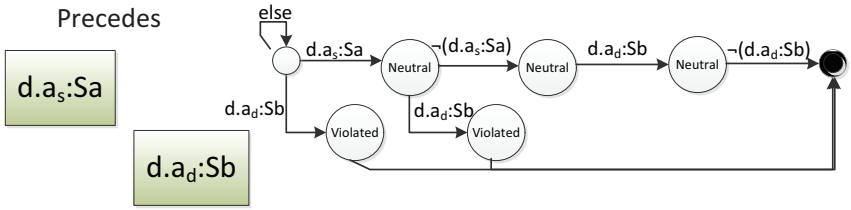


FIGURE C.6 – Dependency State Machine : Precedes

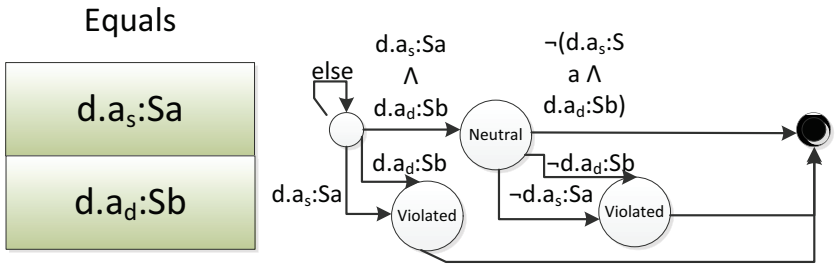


FIGURE C.7 – Dependency State Machine : Equals

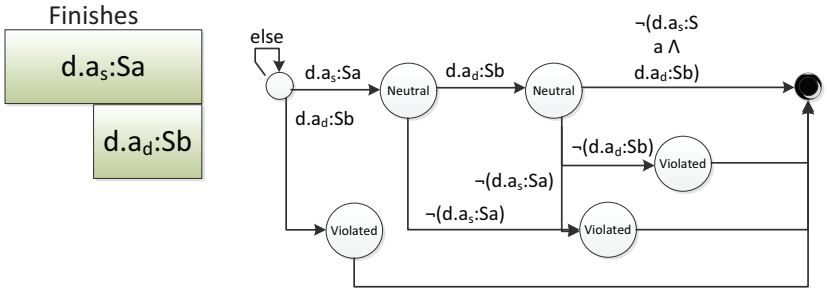


FIGURE C.8 – Dependency State Machine : Finishes

Annexe D

Acronyms

AW Activity Workspace
BPM Business Process Management
BPMN Business Process Modeling Notation
BPMS Business Process Management System
CSCW Computer Supported Cooperative Work
DSM Dependency State Machine
ECA Event Condition Action
LTL Linear Temporal Logic
OMG Object Management Group
ID Identifier
SAM Status and Action Model
SoKNOS Service-orientierte ArchiteKturen zur Unterstützung von Netzwerken im Rahmen Oeffentlicher Sicherheit - Service-Oriented ArchiteCtures Supporting Networks of Public Security
US United States

Annexe E

Résumé de la Thèse

E.1 Introduction

La gestion de la réponse à une catastrophe ou à une crise d'ampleur importante fait en général appel à l'intervention d'une multitude d'organisations et d'individus. Des cellules de crise sont mises en place, des équipes issues de différentes organisations sont déployées, des personnes sont alertées et des services se reconfigurent pour faire face à l'évènement et à ses conséquences. Une des caractéristiques de ces situations est la mise en œuvre d'un nombre conséquent d'actions à exécuter par différents types d'intervenant ou groupes d'intervenants dans le but de circonvenir les conséquences de la crise. Bien que de nombreux outils et supports faisant appels aux technologies de l'information aient déjà été développés et déployés, il ne semble pas exister, à notre connaissance, de système capable de représenter, de contrôler et de coordonner l'ensemble de ces actions dans un contexte multi-organisationnel. Une étude préliminaire que nous avons menée, nous a permis d'identifier un ensemble de besoins que devrait couvrir un tel système pour progresser dans cette voie. Dans cette thèse, nous nous intéressons justement aux caractéristiques de ces actions, aux moyens de les modéliser, de contrôler leur exécution, et de distribuer leur exécution entre les différents acteurs Dans un premier temps nous détaillerons à partir d'un exemple simplifié le type de scénario qui nous intéresse. Ensuite nous détaillerons le résultat de notre étude préliminaire sur cette question de coordination. Elle se concrétise principalement par un ensemble de besoins, concernant principalement la gestion de la coordination dans un processus de gestion de crise. Nous verrons en particulier qu'un des problèmes qu'on retrouve dans la gestion de crise concerne l'évolution des objectifs à atteindre en fonction d'information de contexte ou d'évènements nouveaux qui peuvent changer la perspective. Nous verrons également que c'est du coté de la coordination entre organisations que se situent les enjeux les plus importants. Nous décrirons ensuite une proposition de modèle pour la coordination distribuée d'activité dans un contexte de crise. Ce modèle est défini de manière à pouvoir décrire des dépendances entre activités et à montrer aux utilisateurs les violations des dépendances temporelles. Pour finir nous présenterons les résultats préliminaires de son évaluation.

E.2 Un exemple simplifié

Pour illustrer notre propos, nous nous basons sur un exemple simplifié d'une situation réelle : le cas des inondations en Allemagne qui a été développé pour le projet SOKNOS [DPZM09]. Ce cas a été construit à partir d'interactions et d'interviews avec différents utilisateurs (police, pompiers). L'inondation dont il est question menace plusieurs régions d'Allemagne. Nous nous concentrons sur une situation particulière où l'inondation menace une usine chimique et une zone résidentielle. Si l'usine est inondée, il faudra évacuer la population de la zone résidentielle. La préparation au risque d'inondation de l'usine est faible, le risque ayant été sous-estimé. Des plans spécifiques doivent être mis en œuvre. Le scénario inclut l'échec de la protection de l'usine, une pollution de l'air et la nécessité d'évacuer les populations. La situation devient rapidement chaotique. La Figure E.1 donne une idée de la situation générale du cas et des activités exécutées par les différentes organisations.

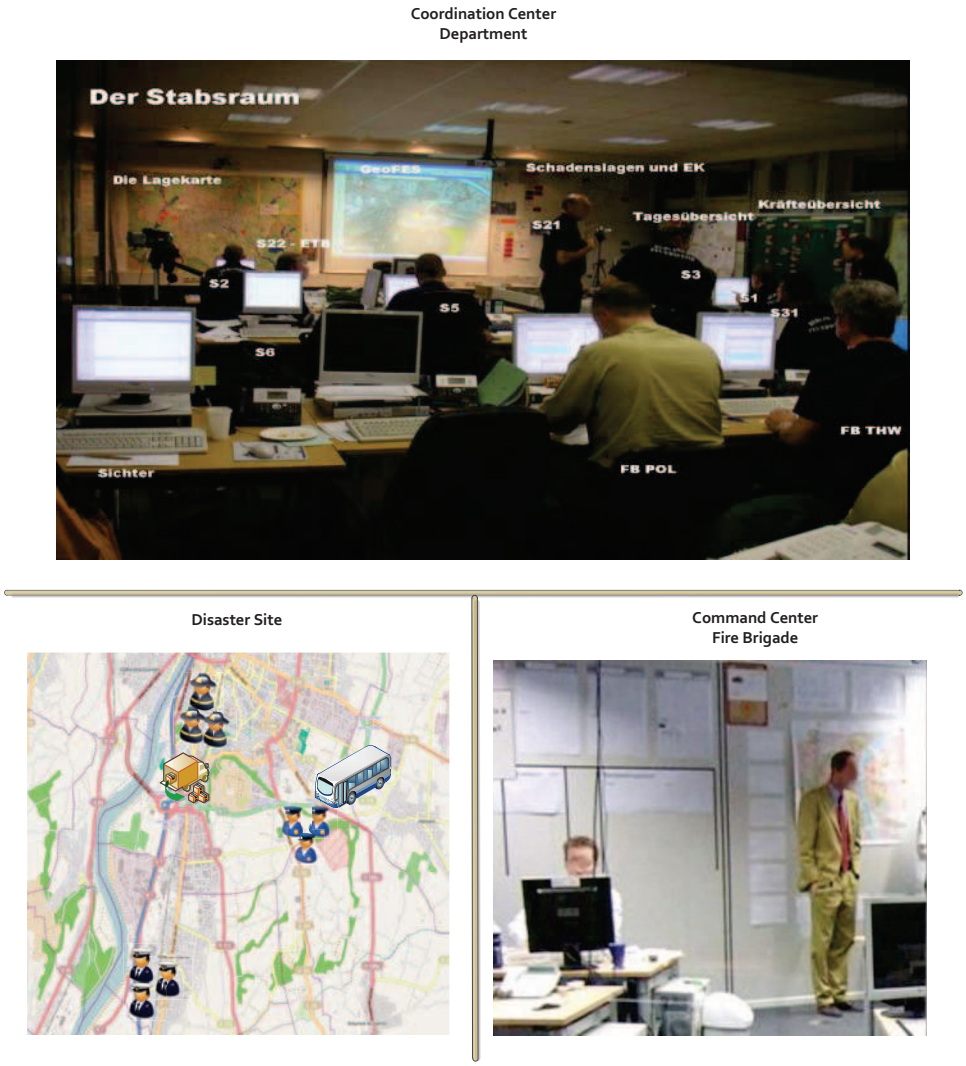


FIGURE E.1 – Exemple simplifié

E.3 Identification des besoins

Sur la base de cette exemple et à partir d'études de cas et d'interviews, nous avons identifiés un certain nombre de besoins pour la gestion d'activités dans un contexte de crise mettant en jeu plusieurs organisations.

1. La gestion des activités : il doit être possible de représenter les activités et leurs dépendances. Il doit également être possible de contrôler leur exécution. Les activités sont de différentes natures (prise de décision, métier, planning, etc.). Elles ont des cycles de vie différents. Les activités et leurs dépendances peuvent changer en fonction des évènements.
2. Les objectifs poursuivis par chacun des intervenants peut varier dans le temps.
3. Conscience, partage, "sensemaking" : un des problèmes importants concerne la connaissance que les acteurs peuvent avoir de l'exécution d'activités par les autres. Ils doivent connaitre l'existence des activités qui se déroulent (partage), ils doivent pouvoir y ajouter des dépendances, ils doivent pouvoir construire une conscience de l'action en cours.
4. Même si des cellules de crise se mettent en place et ont pour rôle de coordonner l'action, chaque intervenant ou groupe d'intervenants dispose en général de sa propre infrastructure de gestion de la l'information et de la communication. Il n'existe pas de système central gérant l'ensemble des activités en cours. Chaque organisation est maitresse de ce qu'elle partage et avec qui.
5. La question de la gouvernance et des responsabilités est centrale dans la gestion de crise. Les rôles doivent être clairs pour chaque action à l'intérieur d'une organisation ou en relation avec une autre organisation. Si la gouvernance des activités n'est pas claire, les actions sont mal exécutées, voire pas exécutées du tout. Nous considérons pour les activités quatre rôles : responsable, comptable, consulté, informé (Matrice RACI).
6. Sécurité et confiance : la sécurité et la confiance sont deux dimensions majeures d'une bonne coordination pour les utilisateurs finaux. Le partage d'information se fait en priorité sur une base de confiance. Il doit également être sécurisé. Ces besoins ont bien sûr été identifiés de façon empirique et il est possible d'en extraire d'autres de l'étude conduite. Nous considérerons dans la suite de ce travail qu'ils forment un ensemble d'hypothèses qui vont nous conduire à la conception d'un modèle permettant d'assister les participants à une crise et qu'ils resteront à valider de manière expérimentale.

E.4 Etat de l'art

Différents types de modèles de processus ont été identifiés dans la littérature [McC92]. Ils sont cependant tous basés sur l'idée d'activité atomique, organisées de façon séquentielle. La plupart de ces modèles ont été considérés pour modéliser et supporter des processus de gestion de crise, que ce soit pour la phase de préparation, la phase de gestion ou la phase de restauration. Les organisations de secours par exemple disposent souvent de

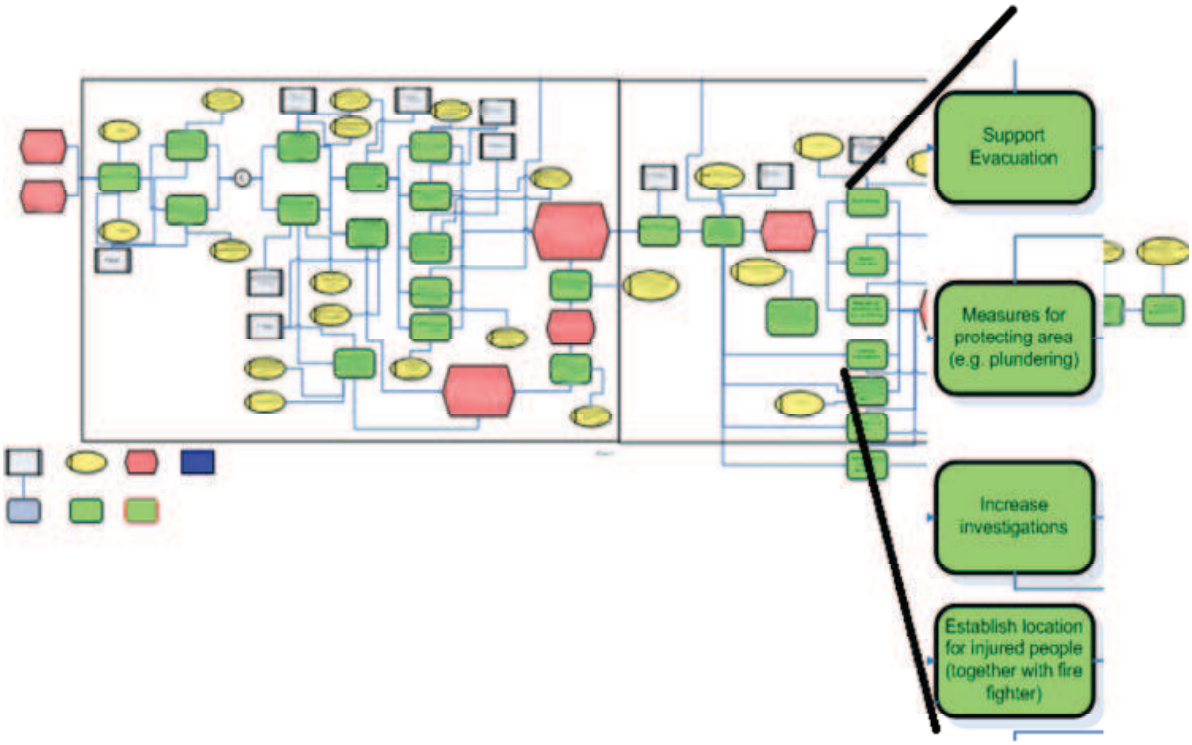


FIGURE E.2 – Exemple de modèle d'un processus de réponse pour répondre à un accident de train avec des matières dangereuses (du point de vue de la police)

processus pour la gestion de leurs ressources ou l'entrainement. Cependant, les activités de réponse à une situation de crise ne semblent pas pouvoir s'inscrire simplement dans ces modèles. Les interviews que nous avons menés et les exercices de modélisation que nous avons conduits avec des utilisateurs (policiers et pompiers du projet SOKNOS) ont montrés l'inadéquation des modèles de gestion de procédés existant.

Les schémas résultants sont trop compliqués et perdent beaucoup de leur intérêt du fait que la plupart des activités s'exécutent en parallèle. Lorsque nous avons voulu intégrer des aspects de gouvernance (autorité, responsabilité), la complexité s'est encore accrue. La difficulté principale est due au fait qu'il est difficile de définir des processus isolés comme cela peut être le cas dans les entreprises. Nous nous sommes également heurtés à la difficulté à représenter des dépendances particulières entre activités, plus spécifique que la simple dépendance qui fait dépendre le démarrage d'une activité de la fin d'une ou plusieurs autres. Pour les utilisateurs interrogés, le modèle n'était pas adapté. Le manque d'expertise n'était pas en cause. Des spécialistes de la modélisation étaient disponibles pour assister les professionnels de la crise. Ceci confirme d'autres travaux [Den06, FW09, GSBC00], qui ont proposés d'autres approches. Notre analyse des plans génériques ou spécifiques de réponse à des catastrophes nous ont permis d'arriver à nouveau à ces conclusions. Il n'y a pas de transformation simple de ces plans dans des

processus métiers. L'étude des besoins que nous avons faite et sa confrontation avec l'état de l'art dans le domaine de la gestion de procédé est sans appel. Les modèles de procédés classiques sont peu ou pas adaptés à la gestion des activités en situation de crise. Un procédé métier est un procédé qui peut être modélisé en grande partie en avance, et qui est susceptible d'un grand nombre d'exécutions. Chaque contexte de crise est différent et son traitement est basé en grande partie sur des plans difficilement formalisables. De nombreux travaux ont été conduits pour augmenter la flexibilité de la modélisation (e.g. [Rin04, DR09, Gri01, HEN01, Hut04, MMWFF92, Gaa10, BP07, MMBA99, RW07, GSBC00, dLMDG07, dL09, RJVzMA07, WR09, BGH+07, GS07, vdAWG05, MRH07, Van09, VRvdA11, HWK09, SCG96, Pes08, ZPG10, DHL90, Kri95, DHMZ96, WPT06, MGR04, MRvdA+09, MRvdA+10, RRS09, PBB10, WA05, Cug98, SSO01, Ada07]) et/ou de l'exécution des procédés y compris dans des contextes inter-organisationnels (e.g. [GH98, MM00, vdAW01, CDT06, MM05, SO04, FYG09, SYY06]). D'après notre étude préliminaire et les interviews, ces systèmes conduisent à des modèles trop complexes. Malgré les travaux menés durant ces dix dernières années, les modèles existants ont également du mal à prendre en compte les situations imprévues, ce qui est la norme lors de crises (échec d'activités, changement de plans). Un système supportant la coordination d'activités au cours d'une crise doit permettre une modélisation simple et flexible dans un contexte dynamique. Il doit également supporter de fonctionner dans un contexte multi-organisationnel. De ce point de vue également, les propositions existantes sur les processus inter organisationnels ne sont pas adaptées en raison des modèles sous-jacent et de la difficulté à gérer les connections entre organisations. Par exemple, [dLMDG07, FW09] proposent de modéliser différents scenarios à l'avance puis de composer ces scénarios en fonction de l'évolution de la crise. Cette phase de modélisation à l'avance se prête bien à des processus de gestion de l'urgence qui peut être plus routinière. Elle ne couvre pas les besoins que nous avons identifiés. D'autres travaux ont introduits des contraintes temporelles dans les modèles de procédés [LSPG06]. Ces contraintes considèrent plutôt des relations temporelles absolues et s'appliquent plutôt à des activités séquentielles. Globalement l'étude de l'état de l'art et nos interviews avec les professionnels de la crise montrent que les modèles classiques de coordination des activités ne se prêtent pas bien aux problèmes qui peuvent être rencontrés dans une situation de crise. Même si des solutions ad-hoc peuvent être utilisées pour certaines situations particulières, prévisibles et donc modélisables à l'avance, il est nécessaire de réfléchir à ce que pourrait être un modèle coordination global qui pourrait être adapté à des situations changeantes dans un contexte multi-organisationnel. Nous avons choisi comme nous allons le montrer par la suite, de nous baser sur une approche centrée activités, sans s'imposer les limites d'un ensemble de modèles de processus sans relations. C'est ce que nous allons décrire dans les sections suivantes.

E.5 Framework

Le modèle que nous proposons cherche à rester simple tout en permettant une expressivité suffisante. La simplicité est une condition importante selon nous pour une mise en œuvre efficace et rapide. Le système doit cependant rester expressif pour apporter une in-

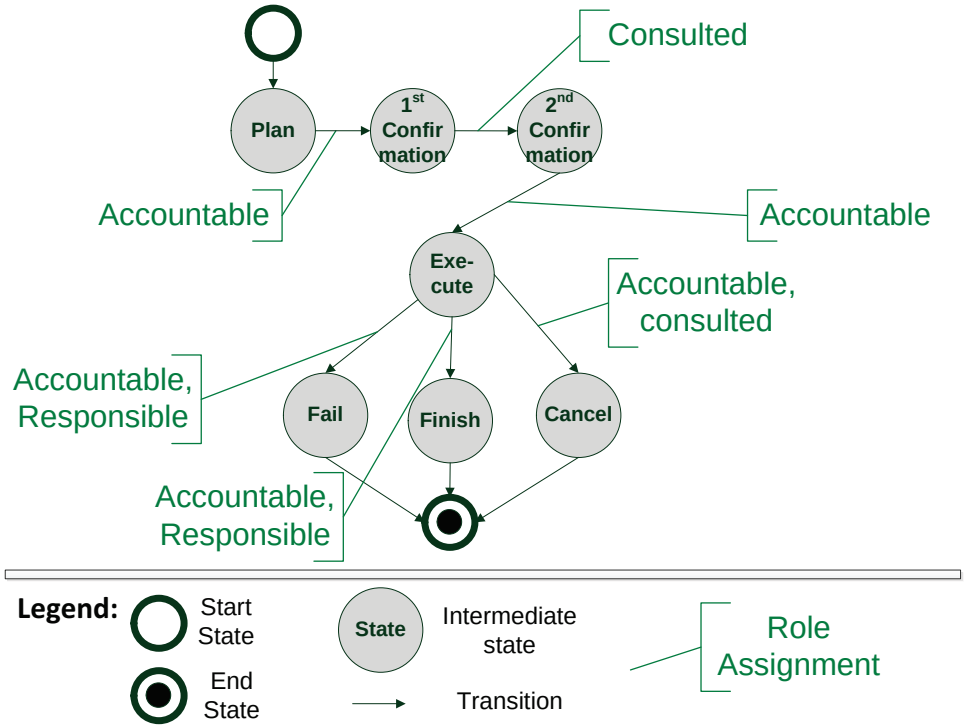


FIGURE E.3 – Exemple pour un type d'activité avec des rôles de gouvernance pour l'approbation : Opération complex sur le terrain

formation suffisante et utile aux intervenants. Notre modèle se base sur les notions de type d'activité, d'activité et de dépendances temporelles. Un type d'activité permet de rendre compte des différentes formes que peuvent prendre des activités de gestion de crise. Les types d'activités varieront principalement par leur cycle de vie et par leur gouvernance. Le cycle de vie d'une activité correspond au processus particulier de son exécution.

Une activité de type décisionnel aura un processus différent d'une activité de type opérationnelle délégable. Ce cycle de vie se décrit comme un diagramme d'état ayant un état initial et un état final. Les transitions entre les états sont contrôlées par les rôles de gouvernance attachées au type. Ces rôles sont pour l'instant prédéfinis et correspondent à la matrice RACI (Responsable, Autorité, Consulté, Informé). L'exemple de la Figure E.3 montre un exemple complexe d'un type d'activité. Trois rôles sont concernés qui peuvent activer les transitions de l'automate. Les rôles seront ensuite attachés à des personnes particulières ou lors de transfert à des organisations, charge à elle de faire leur propre affectation. L'intérêt de disposer de rôles de gouvernance générique, adaptables à différentes organisations est de permettre d'éviter les questions d'alignement des rôles entre ces organisations. Les rôles consultés et informés sont des rôles techniques qui permettront de donner accès à différentes personnes à l'état de l'activité. Ces rôles ne permettent en général pas de changer l'état de l'activité.

Les activités ont un type. Elles sont créées au fur et à mesure du déroulement de l'action par les intervenants dans la gestion de crise. Elles peuvent être préparées à l'avance ou créées en fonction des besoins de façon ad-hoc. Les rôles de gouvernance correspondants

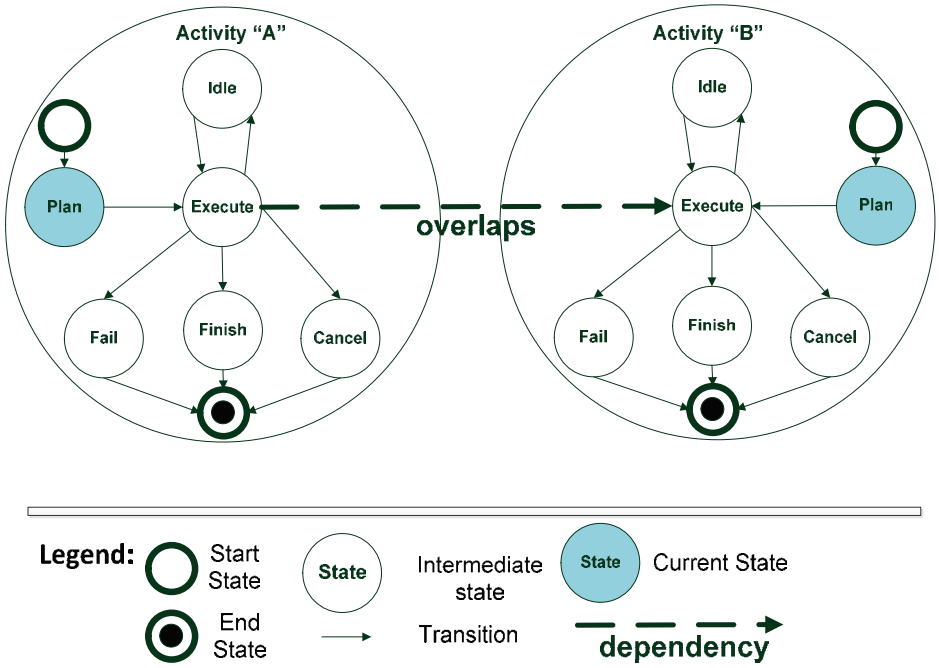


FIGURE E.4 – Exemple d'un modèle contenant deux activités avec une dépendance temporelle

à ce type sont assignés à des participants de l'organisation ou comme nous le verrons plus tard à d'autres organisations. L'autorité a le droit, qu'elle peut déléguer, d'assigner des participants à un rôle. Une activité a un état, dépendant de son cycle de vie. Les transitions sont gouvernées par les règles de gouvernance attachées au type. Ainsi, il existe un continuum entre la modélisation du processus de la crise, son exécution et les possibilités de monitoring de cette exécution.

Il est ensuite possible de décrire les dépendances existantes entre les activités. Ces dépendances sont des dépendances temporelles sur les états de l'activité. La possibilité de décrire les dépendances sur les états offre une plus grande souplesse dans leur définition. Les dépendances vont s'exprimer sur le moment ou une activité arrive dans un état et celui où elle quitte cet état. Elles auront plusieurs fonctions comme nous le verrons plus tard. Pour décrire les dépendances, nous nous basons sur les intervalles de Allen [All83]. Outre les dépendances classiques de terminaison/démarrage, il est possible d'exprimer des dépendances plus complexes comme par exemple la synchronisation de l'exécution de deux activités ou de leur échec.

Dans l'exemple de la Figure E.4, une dépendance d'overlapping est déclarée entre deux activités. L'activité de droite doit s'exécuter pendant que l'activité de gauche s'exécute. Plusieurs politiques de réactions à la violation de la contrainte peuvent être mise en œuvre. Une violation possible de la contrainte concerne le cas où l'activité de droite commence son exécution avant celle de gauche. La violation est détectée et elle peut simplement être notifiée. Dans le second cas, la violation est vérifiée et une action est entreprise pour revenir à un état cohérent. Ici, l'activité de gauche est automatiquement passée à l'état d'exécution. Dans le dernier cas, la réalisation de l'action est annulée pour

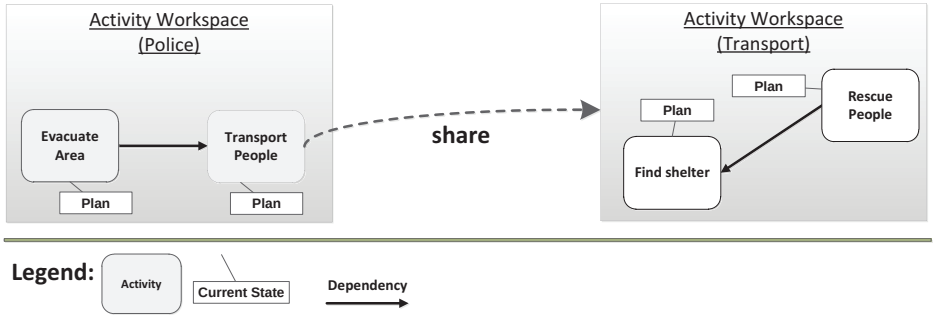


FIGURE E.5 – Exemple : Transmettre une activité à une autre organisation

empêcher la violation de la contrainte. L'activité ne peut pas être exécutée. Ces différentes politiques permettent de conserver la flexibilité nécessaire au traitement de la crise, tout en fournissant quand c'est nécessaire des moyens de contrôle. La plupart des activités n'étant pas automatique ou contrôlable, comme cela peut être le cas dans un processus métier classique, il est important d'arriver à garder une synchronisation entre ce qui se passe réellement et l'état reflété par le système. Ces politiques doivent permettre de faciliter cette synchronisation.

E.6 Distribution des activités

Nous nous plaçons dans un cadre distribué sans coordination centrale. Il nous semble qu'une approche centralisée et la possibilité d'un point individuel de défaillance est un risque trop important pour un système de gestion de crise. Nous considérons que chaque organisation ou partie d'organisation est capable de gérer un ensemble propre d'activités. La coordination est assurée par partage ou échange d'activités. Une organisation peut transmettre des activités à une autre organisation avec des règles de gouvernance adaptées (cf. Figure E.5).

Dans l'exemple, l'autorité (la police) a créé dans son espace deux activités et leur dépendance. L'activité de transport est ensuite transmise à une organisation qui en sera responsable. L'activité est copiée dans l'espace correspondant de ce nouvel acteur. La transmission se fait en utilisant le réseau connu de l'autorité. Les règles de gouvernance permettront ensuite au responsable de déléguer tout ou partie de son activité. Le fait de distribuer des activités entre plusieurs acteurs pose bien la question de la synchronisation de l'état de ces activités. La décision de l'état réel et des conséquences de cette décision devront être prises par l'acteur qui a autorité sur l'activité. Si par exemple, le responsable a décidé que l'activité avait échouée alors que l'autorité l'a placée dans un état d'attente et que les deux organisations sont déconnectées temporairement, à la resynchronisation, c'est l'autorité qui décidera du nouvel état en cas de conflit. Nous avons travaillé cependant sur des protocoles de réplication optimiste pour automatiser cette tâche.

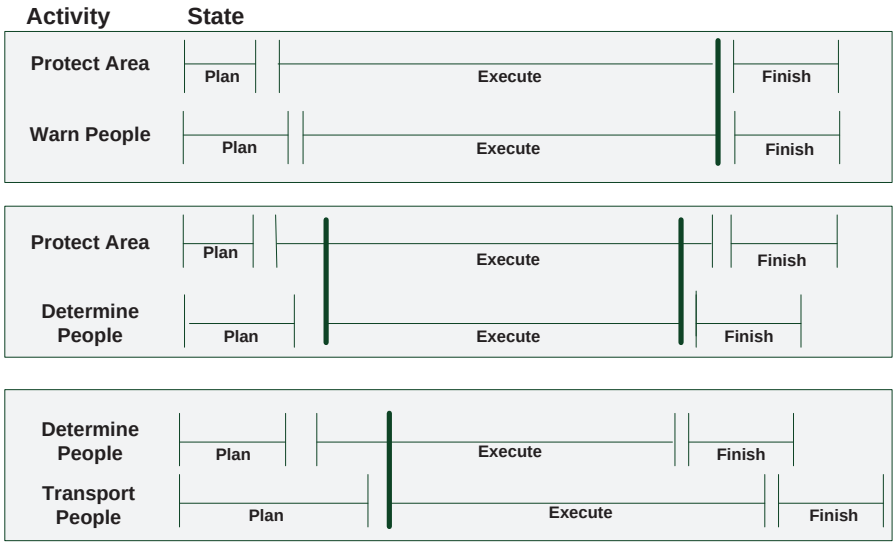


FIGURE E.6 – Exemple : dépendances exprimées entre des activités

E.7 Exemple

Dans cette section, nous allons décrire comment le système pourra être utilisé dans un exercice ou lors d'une crise réelle. Nous ne considérons pas ici les aspects de distribution. Au début de la crise, les utilisateurs de la cellule de crise pourront charger un ensemble d'activités et de dépendances définies à l'avance à partir des plans existants. Il est classique qu'au départ de la crise, les répondants s'appuient sur des plans et il n'est pas imaginable que toutes les activités soient créées au départ. A ce moment, ils peuvent également ajouter des activités et de nouvelles dépendances en fonction des besoins de la situation. Quatre activités ont été sélectionnées de notre premier scenario pour illustrer la mise en œuvre des dépendances temporelles : "prévenir les habitants", "protéger la zone", "identifier les habitants", "transporter les habitants". "Prévenir les habitants" concerne les habitants de la zone résidentielle proche de l'inondation qui vont devoir être évacués. "Protéger la zone" correspond à la construction d'une digue pour protéger la zone résidentielle. "Identifier les habitants" nécessite de dénombrer et de rechercher les habitants de la zone. "Transporter les habitants" est l'activité effective d'évacuation par des transports routiers. Nous considérons que toutes les activités sont du même type. Elles sont d'abord planifiées puis exécutées, puis terminées. Elles peuvent être annulées ou échouer. Le rôle responsable est en charge de l'exécution et de la terminaison. Une activité est dans un état donné pour une certaine durée que nous avons choisi de ne pas spécifier de façon absolue. Nous privilégions les dépendances relatives entre activité, même si des dépendances de temps absolues peuvent être utilisées parfois.

Dans la Figure E.6, il y a trois dépendances exprimées entre des activités. Toutes ces dépendances sont sur l'état "Execute". La première boite montre le comportement attendu et la dépendance entre l'activité de protection et celle d'information. Si la protection de la zone est terminée, il faut arrêter l'information des citoyens. La seconde boite décrit la

dépendance entre la protection de la zone et l'identification et la recherche des habitants. Cette activité s'exécute pendant l'exécution de la première. Elle démarre après et doit se terminer avant. C'est la dépendance temporelle "Contains". La troisième activité décrit la dépendance entre les activités d'identification et de transport. L'activité de transport commence après l'activité d'identification et peut se poursuivre après. Représenter et contrôler l'exécution de ce scenario avec des modèles de processus est quasiment impossible ou très compliqué. Le modèle sera large, rendant sa compréhension difficile et le risque est grand que le système ne soit pas réellement utilisé.

Les utilisateurs sur le terrain ou dans les centres de gestion de crise pourront créer des activités, les échanger avec d'autres organisations et créer des dépendances. Une personne sera en général chargée de la maintenance de l'information. Certains n'auront qu'à consulter les activités qui leur sont assignées, et en changer l'état en fonction des circonstances. Les conséquences de ces changements d'état pourront être exploitées directement voire automatiquement par le système. Les déviations par rapport au plan en cas de retard relatif, d'échec ou d'annulation pourront être visualisées et diffusées rapidement pour en tirer les conséquences et éventuellement activer les plans alternatifs déjà prévu selon les situations. Si, par exemple, un pompier est blessé dans l'exemple de la Figure E.1, la construction de la digue pourra être temporairement interrompue pour porter secours. C'est une nouvelle activité qui pourra être transmise directement par le système ou par tout autre moyen au centre de commandement. D'autres activités de granularité plus fines pourront être insérées dans le diagramme ("Envoyer une ambulance", "Envoyer une équipe supplémentaire").

Ceci illustre la possibilité du système de gérer également une coordination verticale, du terrain vers les centres de commandement. Les équipes sur le terrain ont une meilleure idée de la situation réelle et elles peuvent juger parfois plus surement des activités à mener en priorité et de leurs dépendances. Ces mécanismes peuvent être également déployés horizontalement entre les équipes des différentes organisations sur le terrain. Les notions sous-jacentes sont suffisamment simples pour qu'on puisse espérer à ce niveau une compréhension mutuelle sans apprentissage lourd. Le fait de disposer d'un modèle peu complexe nous permet d'envisager une diffusion relativement aisée. Ensuite vient le problème du contenu même des activités mais des études ont montré que la compréhension mutuelle entre pompiers et policiers par exemple est rarement un gros problème en cas de crise. (cf. [Wac00]).

E.8 Evaluation

Il y a deux types d'évaluation possible pour ce système. Une évaluation analytique et une évaluation empirique. L'évaluation analytique consiste à s'assurer que le modèle sera correctement interprété et qu'il n'y aura pas de dépendances cycliques créées entre des activités. Nous n'entrerons pas dans les détails ici sur les travaux de formalisation et de validation qui sont détaillés dans le chapitre 4.

Pour l'évaluation, dans une première phase nous avons utilisé le modèle dans le cadre d'interviews avec des spécialistes des procédures de lutte contre les feux. La terminologie que nous utilisons est alignée avec la leur. Nous avons ainsi eu une première confirmation

de l'utilité potentielle de notre modèle et en particulier du système liant activités et dépendances (cf. chapitre 7).

L'importance de la dimension inter organisationnelle a également été confirmée. La recommandation principale concerne la possibilité de fournir plusieurs vues sur les modèles au cours de l'exécution. L'idée serait de pouvoir intégrer les données concernant les activités aux systèmes de visualisation existant. Sur la base du même scénario, nous avons pu mener une modélisation satisfaisante des activités, n'ajoutant pas de complexité particulière à l'évènement.

Par ailleurs, nous avons conçu une expérience qui doit permettre de reproduire les problèmes de conflits et de coordination dans le cadre d'une coordination entre plusieurs participants travaillant à distance. Cette expérience fait intervenir plusieurs équipes (au moins 6) pour leur faire construire deux objets en LEGO®, l'un dépendant de l'autre. L'utilisation des LEGO® rend le travail à réaliser concret et contraint tout en restant facile d'accès. Les ressources sont limitées. La séparation des tâches et la nécessité de transporter des LEGOS® [6] d'une équipe à une autre produit assez facilement des erreurs, des incompréhensions et des problèmes entre des équipes qui ne peuvent communiquer qu'à travers des outils informatiques. Les premières exécutions de cette expérience ont été très encourageantes et ont bien montré les problèmes des outils classiques. Par manque de ressources, nous n'avons pas pu les mener de façon assez extensive ni sur une version opérationnelle de l'outil pour en tirer des informations vraiment exploitable mais nous considérons cependant cette expérience comme une contribution qui doit permettre de poursuivre l'étude de la coordination médiée par des ordinateurs.

E.9 Déploiement et architecture

L'architecture adoptée pour le déploiement et l'exécution de ce système est une architecture distribuée, proche d'un système pair à pair mais avec une réplication contrôlée. Chaque nœud doit pouvoir fonctionner de manière autonome et même être déconnecté. Les mises à jours doivent pouvoir se faire en utilisant tous les canaux de communications disponibles au cours d'une crise, y compris manuellement, par téléphone. Nous avons adopté Google Wave™ [Fer09, TP10] comme plateforme pour développer notre prototype. Ceci permet une déploiement de notre système dans une fédération de serveurs en bénéficiant d'un ensemble d'autres services utiles à la communication et à l'échange de documents entre partenaires. Nous sommes particulièrement conscients de la nécessité d'avoir une plateforme qui puisse s'intégrer avec les outils existants, qui soit dynamique et dont les modes de communications puissent être aussi divers que possible. La possibilité de proposer différentes perspectives de visualisation fait également partie de notre implantation. Les aléas de la vie de Google Wave ont malheureusement ralentit la production d'un prototype réellement expérimentable. Google Wave™ a été publié en open source sous le nom d'Apache Wave [Apa].

6. LEGO® is a trademark of the LEGO Group of companies which does not sponsor, authorize or endorse this thesis

E.10 Conclusion et travaux futures

Nous avons décrit dans cette thèse un support pour la coordination d'activités par des personnes d'organisations différentes. Ce système a été construit pour répondre aux problèmes qui se posent lorsqu'on veut gérer les évènements qui se produisent lors d'une crise. Ces problèmes, qui nous intéressent, sont ceux que rencontrent les organisations qui participent à la résolution d'une crise comme les pompiers ou la police.

Nos contributions concerne un modèle pour la coordination d'activités permettant d'exprimer des dépendances temporelles. Des activités, des dépendances et des règles de gouvernance peuvent être modélisés de façon dynamique. La cohérence du modèle peut être vérifiée dynamiquement. Le modèle peut être exécuté et il est possible d'avoir des incohérences durant l'exécution. Ces incohérences sont présentées aux utilisateurs qui peuvent décider des actions à mener pour les résoudre.

Un enjeu important était de permettre de gérer cette coordination entre plusieurs organisations. Notre modèle permet de connecter des organisations puis permet à ces organisations de partagers des activités et de gérer les dépendances entre elles. La distribution entraine de nouvelles questions liée à la vérification globale du modèle dans ce cadre et aux conflits qui peuvent se produire en cas de changements concurrents d'états des applications. Nous avons proposé des solutions techniques à ces problèmes adaptées au contexte. Le principe n'est pas de trouver des solutions automatisées mais de rendre conscient des groupes d'utilisateurs distribués des problèmes qui peuvent exister entre leur vision de la situation. La conclusion de ces entretiens a confirmé et validé notre approche permettant de supporter la coordination inter organisationnelle et le fait que le système ne pouvait pas prendre de décision à la place des agents engagés dans la gestion de la crise.

Faire des expériences sur le terrain est difficile dans ce contexte. Nous avons donc conçu une expérience qui permet d'évaluer des outils de coordination et de communication dans des conditions qui peuvent reproduire les problèmes qu'on voulait prendre en compte. Cette expérience a été conçue et testée mais pas suffisamment pour en tirer des conclusions sur notre modèle. Les perspectives de ce travail sont maintenant nombreuses. Il faudrait d'abord prendre le temps de compléter l'évaluation en exécutant l'expérience un nombre de fois plus important et avec d'autres outils ou d'autres systèmes pour permettre des comparaisons.

Durant la thèse nous avons été plusieurs fois sollicité pour essayer d'adapter ce modèle à d'autre domaine que la crise. Il y a un fort mouvement autour de la gestion adaptative de cas et un fort besoin de gestion de processus plus dynamique et inter organisationnel. Notre approche si elle était validé pour la crise pourrait certainement être adaptée à d'autres domaines. La validation d'un tel système peut se faire techniquement. Elle doit aussi se faire empiriquement ou expérimentalement.

Bibliographie

[Ada07] Michael Adams. *Facilitating Dynamic Flexibility and Exception Handling for Workflows*. PhD thesis, Queensland University of Technology, 2007.

[AKD00] Anish Arora, Sandeep Kulkarni, and Murat Demirbas. Resettable vector clocks. In *19th Annual ACM Symposium on Principles of Distributed Computing*, Portland, Oregon, USA, 2000.

[All83] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11) :832–843, 1983.

[Apa] Apache. Incubator - apache wave, `http://incubator.apache.org/wave/`, retrieved 18.04.2011.

[ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4) :335–371, 2004.

[Bar97] Jakob E. Bardram. Plans as situated action : An activity theory approach to workflow systems. In *5th European Conference on Computer Supported Cooperative Work (ECSCW)*, Lancaster, UK, 1997.

[Bar98] Jacob E. Bardram. *Collaboration, Coordination, and Computer Support*. PhD thesis, Department of Computer Science, University of Aarhus, 1998.

[Bar00] Jakob E. Bardram. Temporal coordination. *Computer Supported Cooperative Work*, 9 :157–187, 2000.

[BBS95] John Bowers, Graham Button, and Wes Sharrock. Workflow from within and without : Technology and cooperative work on the print industry shopfloor. In *Fourth European Conference on Computer-Supported Cooperative Work*, Stockholm, Sweden, 1995.

[BBSBMD+06] N. Bellamine-Ben Saoud, T. Ben Mena, J. Dugdale, B. Pavard, and M. Ben Ahmed. Assessing large scale emergency rescue plans : an agent based approach. *Special Issue on Emergency Management Systems. International Journal of Intelligent Control and Systems*, 11(4) :260–271, 2006.

[BCK+07] K. Bhattacharya, N.S. Caswell, S. Kumaran, A. Nigam, and F.Y. Wu. Artifact-centered operational modeling : Lessons from customer engagements. *IBM Systems Journal*, 46(4) :703–721, 2007.

[BCSR07] Chris Baber, James Cross, Paul Smith, and Dengel Robinson. Supporting implicit coordination between distributed teams in disaster management. In *1st International Conference on Mobile Information Technology for Emergency Response*, Sankt Augustin, Germany, 2007.

[Beg98] James Michael Allen Begole. *Flexible Collaboration Transparency : Supporting Worker Independence in Replicated Application-Sharing Systems*. PhD thesis, Virginia Polytechnic Institute and State University, 1998.

[BGH+07] Kaml Bhattacharya, Cagdas Gerede, Richard Hull, Rong Liu, and Su Kianwen. Towards formal analysis of artifact-centric business process models. In *5th International Conference on Business Process Management*, Brisbane, Australia, 2007.

[Bio08] B.J. Biornstadt. *A Workflow Approach to Stream Processing*. PhD thesis, ETH Zürich, 2008.

[Ble10] Alexander Blecken. *Humanitarian Logistics : Modelling Supply Chain Processes of Humanitarian Organisations*. Haupt Verlag, 2010.

[BLJ10] Nitesh Bharosa, KinKyu Lee, and Marijn Janssen. Challenges and obstacles in sharing and coordinating information during a multi-agency disaster response : Propositions from field exercises. *Information System Frontiers*, 12(1) :49–65, 2010.

[BM83] Bernard Berthomieu and Miguel Menasche. An enumerative approach for analyzing time petri nets. In *IFIP Congress*, Paris, France, 1983.

[BM08] Udo Bartlang and Jörg P. Müller. Dhtflex : A flexible approach to enable efficient atomic data management tailored for structured peer-to-peer overlays. In *3rd International Conference on Internet and Web Applications and Services*, Athens, Greece, 2008.

[Bor92] Benton H. Borum. Crisis action procedures at the joint chiefs of staff / unified command level. Technical Report 17013-5050, U.S. Army War College, Carlisle Barracks, PA, 1992.

[BP07] B.J. Biörnstadt and C. Pautasso. Let it flow : Building mashups with data processing pipelines. In *ICSOC Workshops*, Vienna, Austria, 2007.

[bpm] Object management group (omg) : Business process model and notation (bpmn) - version 2.0, formal/2011-01-03, `http://www.omg.org/spec/BPMN/2.0/`.

[BR10] Martin Becker, Jörg Kugeler and Michael Rosemann, editors. *Process Management : A Guide for the Design of Business Processes*. Springer, 2010.

[BS00] Uwe M. Borghoff and Johann H. Schlichter. *Computer-Supported Cooperative Work : Introduction to Distributed Applications*. Springer, 2000.

[BVL04] Peter Bürgi, Bart Victor, and Jody Lentz. Modeling how their business really works prepares managers for sudden change. *Strategy & Leadership*, 32(2) :28–35, 2004.

[CDT06] Issam Chebbi, Schahram Dustdar, and Samir Tata. The view-based approach to dynamic inter-organizational workflow cooperation. *Data & Knowledge Engineering*, 56 :139–173, 2006.

[Cho08] Sang Ok Choi. Emergency management : Implications from a strategic management perspective. *Journal of Homeland Security and Emergency Management*, 5(1) :1, 2008.

[CK06] Louise K. Comfort and Naim Kapucu. Inter-organizational coordination in extreme events : The world trade center attacks, september 11, 2001. *Natural Hazards*, 39(2) :309–327, 2006.

[CL85] K. Mani Chandy and Leslie Lamport. Distributed snapshots : Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1) :63–75, 1985.

[CMF+09] Lorenzo Cantoni, Elena Marchiori, Marco Faré, Luca Botturi, and Davide Bolchini. A systematic methodology to use lego bricks in web communication design. In *27th ACM International Conference on Design of Communication*, Bloomington, Indiana, USA, 2009.

[CO46] CO. U.s. strategic bombing survey - the effects of the atomic bombings of hiroshima and nagasaki. Technical Report 19 June 1946, Chairman's Office, 1946.

[Coh00] Norman H. Cohen. A java framework for mobile data synchronization. In *7th International Conference on Cooperative Information Systems*, Eilat, Israel, 2000.

[CSRU08] Rui Chen, Raj Sharman, H. Raghav Rao, and Shambhu J. Upadhyaya. Coordination in emergency response management. *Communications of the ACM*, 51(5), 2008.

[Cug98] G. Cugola. Tolerating deviations in process support systems via flexible enactment of process models. *IEEE Transactions on Software Engineering*, 24(11) :982–1001, 1998.

[DÖ97] Dietrich Dörner. *The Logic of Failure : Recognizing And Avoiding Error in Complex Situations*. Basic Books, 1997.

[DÖ03] Dietrich Dörner. *Die Logik des Mißlingen : Strategisches Denken in komplexen Situationen*. rororo, 2003.

[DA79] Russel R. Dynes and B.E. Aguirre. Organizational adaptation to crises : Mechanisms of coordination and structural change. *Disasters*, 3(1) :71–74, 1979.

[DBBSPP09] J. Dugdale, N. Bellamine-Ben Saoud, B. Pavard, and N. Pallamin. *Information Systems for Emergency Management*, chapter Simulation and Emergency Management, pages 229–253. Sharp, 2009.

[Dek03] Sidney Dekker. Failure to adapt or adaptations that fail : contrasting models on procedures and safety. *Applied Ergonomics*, 34 :233–238, 2003.

[Den06] Peter J. Denning. Infoglut. *Communications of the ACM*, 49(7) :15–19, 2006.

[DHL90] U. Dayal, M. Hsu, and R. Ladin. Organizing long-running activities with triggers and transactions. In *ACM SIGMOD Conference on Management of Data*, Atlantic City, New Jersey, United States, 1990.

[DHL01] Umeshwar Dayal, Meichun Hsu, and Rivka Ladin. Business process coordination : State of the art, trends and open issues. In *27th International Conference on Very Large Data Bases*, Roma, Italy, 2001.

[DHMZ96] Paul Dourish, Jim Holmes, Pernille MacLean, Allan an Maqvardsen, and Alex Zbyslaw. Freeflow : Mediating between representation and action in workflow systems. In *ACM Conference on Computer Supported Cooperative Work (CSCW)*, Boston, Massachusetts, USA, 1996.

[DKU+08] Jörg Dörr, Daniel Kerkow, Özgür Ünalan, Sebastian Adam, Eva Peslova, Martin Treiblmayr, Thorsten May, Thomas Ziegert, Anette Balden, Verena Such, Katharina Goering, and Anoshirwan Soltani. Soknos - deliverable d1.2 : Beschreibung des konkreten anwendungsfalls innerhalb des katastrophenszenarios. Technical Report D1.2, Service-Orientierte ArchiteKturen zur Unterstützung von Netzwerken im Rahmen Oeffentlicher Sicherheit (SoKNOS), Projekt Förderkennzeichen 01|S07009A SoKNOS, Bundesministerium für Bildung und Forschung (BmBF), 2008.

[dL09] Massimiliano de Leoni. *Adaptive Process Management in Highly Dynamic and Pervasive Scenarios*. PhD thesis, Sapienza - Universita di Roma, 2009.

[dLMDG07] Massimiliano de Leoni, Massimo Mecella, and Guiseppe De Giacomo. Highly dynamic adaptation in process management systems through execution monitoring. In *5th International Conference on Business Process Management (BPM'2007)*, Brisbane, Australia, 2007.

[dM09a] Henk de Man. Case management : A review of modeling approaches. *BPTrends*, January :1–17, 2009.

[dM09b] Henk de Man. Case management : Cordys approach. *BPTrends*, February :1–13, 2009.

[DPZM09] Sebastian Döweling, Florian Probst, Thomas Ziegert, and Knut Manske. Soknos - an interactive visual emergency management framework. In *Workshop on Geographical Information Processing and Visual Analytics for Environmental Security*, Trento, Italy, 2009.

[DR09] Peter Dadam and Manfred Reichert. The adept project : A decade of research and development for robust and flexible process support. *Computer Science - Research and Development*, 23(2) :81–97, 2009.

[Dra03] T. E. Drabek. *Strategies for Coordinating Disaster Responses*. Institute of Behavior Sciences, University of Colorado, 2003.

[Dus04] Schahram Dustdar. Caramba - a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distributed and Parallel Databases*, 15(1) :45–66, 2004.

[DvdAtH05] Marlon Dumas, Wil M. P. van der Aalst, and Arthur H.M. ter Hofstede. *Process-Aware Information Systems*, chapter Introduction, pages 3–21. Wiley, 2005.

[EAWJ02] E.N. Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys*, 34(3) :375–408, 2002.

[EKR95] Clarence Ellis, Karim Keddara, and Grzegorz Rozenberg. Dynamic change within workflow systems. In *Conference on Organizational Computing Systems*, Milpitas, California, USA, 1995.

[Erl05] Thomas Erl. *Service-Oriented Architecture : Concepts, Technology & Design*. Prentice Hall/Pearson, 2005.

[fac] Facebook, `http://www.facebook.com`, retrieved 06.05.2011.

[FBZ09] Simon French, Clare Bayley, and Nan Zhang. Web-based group decision support for crisis management. *International Journal of Information Systems for Crisis Response Management*, 1(1) :41–53, 2009.

[FC10] Jörn Franke and François Charoy. Design of a collaborative disaster response process management system. In *9th International Conference on the Design of Cooperative Systems (COOP'2010)*, Aix-En-Provence, France, 2010.

[FCEK10] Jörn Franke, François Charoy, and Paul El Khoury. Collaborative coordination of activities with temporal dependencies. In *18th International Conference on Cooperative Information Systems (COOPIS'2010) / OnTheMove (OTM) Conferences*, Heraklion, Crete, Greece, 2010.

[FCU10a] Jörn Franke, François Charoy, and Cédric Ulmer. Coordination and situation awareness for inter-organizational disaster response. In *10th IEEE International Conference on Technologies for Homeland Security (HST'2010)*, Boston, MA, USA, 2010.

[FCU10b] Jörn Franke, François Charoy, and Cédric Ulmer. A model for temporal coordination of disaster response activities. In *7th International Conference on Information Systems for Crisis Response and Management (ISCRAM'2010)*, Seattle, Washington, USA, 2010.

[FCU10c] Jörn Franke, François Charoy, and Cédric Ulmer. Un modèle centré activité distribué pour la coordination des acteurs de la crise. In *Workshop Interdisciplinaire sur la Sécurité Globale (WISG'2010)*, Troyes, France, 2010.

[FCU11] Jörn Franke, François Charoy, and Cédric Ulmer. Handling conflicts in autonomous coordination of distributed collaborative activities. In *20th IEEE International Conference on Collaboration Technologies and Infrastructures (WETICE'2011)*, Paris, France, 2011.

[Fer09] Andres Ferrate. *Getting Started with Google Wave*. O'Reilly Media, Inc., 2009.

[FFJ+09] Dirk Fahland, Cédric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In *7th International Conference on Business Process Management*, Ulm, Germany, 2009.

[FFU10] Jörn Franke, Charoy François, and Cédric Ulmer. Pervasive emergency response process management system. In *First Annual Workshop on Pervasive Networks for Emergency Management (PERNEM) in Proceedings of 8th IEEE International Conference on Pervasive Computing and Communciations (PERCOM)*, Mannheim, Germany, 2010.

[FW09] D. Fahland and H. Woith. Towards process models for disaster response. In *Process Management for Highly Dynamic and Pervasive Scenarios*, Milano, Italy, 2009.

[FWC+11] Jörn Franke, Adam Widera, François Charoy, B. Hellingrath, and Cédric Ulmer. Reference process models and systems for inter-organizational ad-hoc coordination - supply chain management in humanitarian operations. In *8th International Conference on Information Systems for Crisis Response and Management (ISCRAM'2011)*, Lisbon, Portugal, 2011.

[fwd] Feuerwehr dienst-vorschrift (fwdv) 100, führung und leitung im einsatz, stand 10. märz 1999, beschlossene fassung des aw vom 10.03.1999.

[FYG09] Walid Fdhila, Ustun Yildiz, and Claude Godart. A flexible approach for automatic process decentralizations using dependency tables. In *7th IEEE International Conference on Web Services*, Los Angeles, CA, 2009.

[Gaa10] Khaled Gaaloul. *A Secure Framework for Dynamic Task Delegation in Workflow Management Systems*. PhD thesis, École Doctorale IAEM Lorraine, LORIA, Université Henri Poincaré - Nancy 1, 2010.

[GAHL00] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig. Crossflow : Cross-organizational workflow management in dynamic virtual enterprises. *International Journal of Computer Systems Science & Engineering*, 15(5) :277–290, 2000.

[GC10] Anna Gryszkiewicz and Fang Chen. Design requirements for information sharing in a crisis management command and control centre. In *7th International ISCRAM Conference*, Seattle, Washington, USA, 2010.

[GCG01] D. Grigori, F. Charoy, and C. Godart. Anticipation to enhance flexibility of worflow execution. In *Database and Expert Systems Applications (DEXA'2001)*, Munich, Germany, 2001.

[GG10] Nawal Guermouche and Claude Godart. Timed conversational protocol based approach for web services analysis. In *International Conference on Service Oriented Computing (ICSOC)*, San Francisco, California, USA, 2010.

[GH98] John C. Grundy and John G. Hosking. Serendipity : integrated environment support for process modelling, enactment and work coordination. *Automated Software Engineering*, 5(1) :27–60, 1998.

[GH05] Vincent Gauthereau and Erik Hollnagel. Planning, control, and adaptation : A case study. *European Management Journal*, 23(1) :118–131, 2005.

[GHSHM03] Nancy K. Grant, David H. Hoover, Anne-Marie Scarisbrick-Hauser, and Stacy L. Muffet. *Beyond September 11th - An Account of Post-Disaster Research*, chapter The Crash of United Flight 93 in Shanksville, Pennsylvania, pages 83–108. Nautral Hazards Research and Applications Information Center, University of Colorado, 2003.

[GM94] Saul Greenberg and David Marwood. Real time groupware as a distributed system : Concurrency control and its effect on the interface. In *ACM CSCW Conference on Computer Supported Cooperative Work*, Chapel Hill, NC, USA, 1994.

[Gon08] Rafael A. Gonzales. Coordination and its ict support in crisis response : confronting the information-processing view of coordination with a case study. In *41st Hawaii International Conference on System Sciences*, Waikoloa, Big Island, Hawaii, 2008.

[gra] Sap gravity, `http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/17826`, retrieved 11.02.2011.

[Gri00] Rebecca E. Grinter. Workflow systems : Occasions for success and failure. *Computer Supported Cooperative Work*, 9(2) :189–214, 2000.

[Gri01] Daniela Grigori. *Eléments de flexibilité des systèmes de workflow pour la définition et l'exécution de procédés coopératifs*. PhD thesis, LORIA, Université Henri Poincaré - Nancy 1, 2001.

[GS07] Gagdas E. Gerede and Jianwen Su. Specification and verification of artifact behaviors in business process models. In *5th international Conference on Service-Oriented Computing (ICSOC'2007)*, Vienna, Austria, 2007.

[GSBC00] D. Georgakopoulos, H. Schuster, D. Baker, and A. Cichocki. Managing escalation of collaboration processes in crisis mitigation situations. In *16th International Conference on Data Engineering*, San Diego, California, USA, 2000.

[Gue89] Hans Werner Guesgen. Spatial reasoning based on allen's temporal logic. Technical Report TR-89-049, International Computer Science Institute, Berkeley, CA, USA, 1989.

[GZCG10] Khaled Gaaloul, Ehtesham Zahoor, François Charoy, and Claude Godart. Dynamic authorisation policies for event-based task delegation. In *22nd International Conference on Advanced Information Systems Engineering*, Hammamet, Tunisia, 2010.

[Ham10] Michael Hammer. *Handbook on Business Process Management 1*, chapter What is Business Process Management ?, pages 3–16. Springer, 2010.

[HEN01] C. Huth, I. Erdmann, and L. Nastansky. Groupprocess : using process knowledge from the participative design and practical operation of ad hoc processes for the design of structured workflows. In *34th Annual Hawai International Conference on System Sciences*, Maui, Hawaii, USA, 2001.

[HMM00] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization : A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1) :24–43, 2000.

[HNSY92] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yoine. Symbolic model checking for real-time systems. In *7th Annual IEEE Symposium on Logic in Computer Science*, Santa Cruz, CA, USA, 1992.

[htm] W3c, html5 - a vocabulary and associated apis for html and xhtml, editor's draft 15 april 2011, `http://dev.w3.org/html5/spec/spec.html`, retrieved 18.04.2011.

[Hut04] Carsten Huth. *Groupware-basiertes Ad-Hoc-Workflow-Management : Das GroupProcess-System*. PhD thesis, Fakultät für Wirtschaftswissenschaften, Fachgebiet Wirtschaftsinformatik, Universität Paderborn, 2004.

[HWK09] Joerg Hoffmann, Ingo Weber, and Frank Kraft. Planning@sap : An application in business process management. In *2nd International Scheduling and Planning Applications woRKshop*, Thessaloniki, Greece, 2009.

[HWK10] Jörg Hoffmann, Ingo Weber, and Frank Michael Kraft. Sap speaks pddl. In *24th AAAI Conference on Artificial Intelligence*, Atlanta, Georgia, USA, 2010.

[HY97] Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In *Third International Conference on Principles and Practice of Constraint Programming*, Schloss Hagenberg, Austria, 1997.

[IfCIS03] Public Entity Risk Institute and Institute for Civil Infrastructure Systems, editors. *Beyond September 11th - An Account of Post-Disaster Research*, volume Natural Hazards Research and Applications Information Center. Nautral Hazards Research and Applications Information Center, University of Colorado, 2003.

[IROM06] Abdessamad Imine, Michaël Rusinowitch, Gérald Oster, and Pascal Molli. Formal design and verification of operational transformation algorithms for copies convergence. *Theoretical Computer Science*, 351(2) :167–183, 2006.

[Jon88] Christopher V. Jones. The three-dimensional gantt chart. *Operations Research*, 36(6) :891–903, 1988.

[KHW93] Thomas Kreifelts, Elke Hinrichs, and Gerd Woetzel. Sharing to-do lists with a distributed task manager. In *Third European Conference on Computer-Supported Cooperative Work*, Milano, Italy, 1993.

[KJJ03] A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations : the maximal tractable subalgebras of allen's interval algebra. *Journal of the ACM*, 50(5) :591–640, 2003.

[KK00] Eleanna Kafeza and Kamalakar Karlapalem. Gaining control over time in workflow management applications. In *Database and Expert Systems Applications (DEXA'2000)*, London, UK, 2000.

[Kle99] Gary Klein. *Sources of Power : How People Make Decisions*. MIT Press, 1999.

[KMR00] E. Kindler, A. Martens, and W. Reisig. *Business Process Management : Models, Techniques, and Empirical Studies*, chapter Inter-Operability of Workflow Applications : Local Criteria for Global Soundness, pages 235–253. Springer, 2000.

[Knu76] Donald E. Knuth. Big omicron and big omega and big theta. *ACM SIGACT News*, 8(2) :18–24, 1976.

[Kre10] Martin Kreichgauer. processwave.org editor architecture and deployment. Technical Report Bachelor Thesis, Hasso Plattner Institut, 2010.

[Kri95] Narayanan Krishnakumar. Managing heterogeneous multi-system tasks to support enterprise-wide operations. *Distributed and Parallel Databases*, 3(1) :155–186, 1995.

[KRSR98] G. Kappel, S. Rausch-Schott, and W. Retscheitzegger. Coordination in workflow management systems - a rule based approach. In *Coordination Technology for Collaborative Applications - Organizations, Processes, and Agents*, 1998.

[Kum92] Vipin Kumar. Algorithms for constraint satisfaction problems : A survey. *AI Magazine*, 13(1) :32–44, 1992.

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7) :558–565, 1978.

[Lam98] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2) :133–169, 1998.

[Lan05] Tobias Landes. Dynamic vector clocks for consistent ordering of events in dynamic distributed applications. Technical report, Institut für Informatik, Technische Universität München, 2005.

[LD98] R.G. Lee and B.G. Dale. Business process management : A review and evaluation. *Business Process Management Journal*, 4(3) :214–225, 1998.

[LN07] Jonas Landgren and Urban Nulden. A study of emergency response work : Patterns of mobile phone interaction. In *SIGCHI Conference on Human Factors in Computing Systems*, San Jose, CA, USA, 2007.

[Loe09] Jon Loeliger. *Version Control with Git : Powerful Tools and Techniques for Collaborative Software Development*. O'Reilly Media, 2009.

[LR92] Peter B. Ladkin and Alexander Reinefeld. Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57(1) :105–124, 1992.

[LS87] L. Lynch and S.K. Sarin. Discarding obsolete information in a replicated data base system. *IEEE Trans. Software Eng.*, 13(1) :39–46, 1987.

[LSPG06] Ruopeng Lu, Shazia Sadiq, Vineet Padmanabhan, and Guido Governatori. Using a temporal constraint network for business process execution. In *17th Australasian Database Conference*, Hobart, Tasmania, Australia, 2006.

[LUW10] Frank Leymann, Tobias Unger, and Sebastian Wagner. On designing a people-oriented constraint-based workflow language. In *2nd Central-European Workshop on Services and their Composition*, Berlin, Germany, 2010.

[LWZ08] C. Lutz, F. Wolter, and M. Zakharyashev. Temporal description logics : A survey. In *15th International Symposium on Temporal Representation and Reasoning*, Montreal, Quebec, Canada, 2008.

[Mat89] Friedemann Mattern. Virtual time and global clocks in distributed systems. In *Workshop on Parallel and Distributed Algorithms*, 1989.

[MC94] Thomas W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1) :87–119, 1994.

[McC92] S. McCready. There is more than one kind of workflow software. *Computerworld*, 2(November), 1992.

[MDvdW10] Willem J. Muhren, D. Durbié, and B. van de Walle. Exploring decision-relevant information pooling by humanitarian disaster response teams. In *7th International Conference on Information Systems for Crisis Response and Management (ISCRAM'2010)*, Seattle, Washington, USA, 2010.

[MGR04] R. Müller, U. Greiner, and E. Rahm. Agentwork : A workflow system supporting rule-based workflow adaptation. *Data & Knowledge Engineering*, 51 :223–256, 2004.

[MM00] John Murdoch and John A. McDermid. Modelling engineering design processes with role activity diagrams. *Journal of Integrated Design and Process Science*, 4(2) :45–65, 2000.

[MM05] F. Montagut and R. Molva. Enabling pervasive execution of workflows. In *1st IEEE International Conference on Collaborative Computing (CollaborateCom)*, San Jose, CA, USA, 2005.

[MMBA99] Hing-Yin Mak, Andrew P. Mallard, Tung Bui, and Grace Au. Building online crisis management support using workflow systems. *Decision Support Systems*, 25(3) :209–224, 1999.

[MMWFF92] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. The action workflow approach to workflow management technology. In *ACM Conference on Computer-supported Cooperative Work (CSCW)*, Toronto, Canada Toronto, Canada Toronto, Canada, 1992.

[Mon07] Frédéric Montagut. *Pervasive Workflows - Architecture, Reliability and Security*. PhD thesis, Computer Science and Networks, Ecole Nationale Supérieure des Télécommunications, 2007.

[MPBW07] Laura G. Militello, Emily S. Patterson, Lynn Bowman, and Robert Wears. Information flow during crisis management : Challenges to coordination in the emergency operations center. *Cogn Tech Work*, 9(1) :25–31, 2007.

[MR09] Jim Marino and Michael Rowley. *Understanding SCA (Service Component Architecture)*. Addison-Wesley Longman, 2009.

[MRH07] Dominic Müller, Manfred Reichert, and Joachim Herbst. Data-driven modeling and coordination of large process structures. In *15th International Conference on Cooperative Information Systems (CoopIS)*, Vilamoura, Algarve, Portugal, 2007.

[MRvdA+09] R.S. Mans, N.C. Russell, W.M.P. van der Aalst, A.J. Moleman, and P.J.M. Bakker. Proclets in healthcare. Technical Report BPM-09-06, BPM Center, bpmcenter.org, 2009.

[MRvdA+10] R.S. Mans, N.C. Russel, W.M.P. van der Aalst, A.J. Moleman, and P.J.M. Bakker. Inter-workflow support. Technical Report BPM-10-09, BPM Center, BPMcenter.org, 2010.

[Mur89] T. Murata. Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, 77(4) :541–580, 1989.

[MWR08] Bela Mutschler, Barbara Weber, and Manfred Reichert. Workflow management versus case handling : Results from a controlled software experiment. In *23rd Annual ACM Symposium on Applied Computing*, Fortaleza, Ceará, Brazil, 2008.

[Nat10] United Nations. Haiti earthquake - flash appeal. Technical report, United Nations, 2010.

[NB95] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations : A maximal tractable subclass of allen's interval algebra. *Journal of the ACM*, 42(1) :43–66, 1995.

[NCDL95] David A. Nichols, Pavel Curtis, Michael Dixon, and John Lamping. High-latency, low-bandwidth windowing in the jupiter collaboration system. In *8th Annual ACM symposium on User Interface and Software Technology*, Pittsburgh, Pennsylvania, United States, 1995.

[Neb97] Bernhard Nebel. Solving hard qualitative temporal reasoning problems : Evaluating the efficiency of using the ord-horn class. *Constraints*, 1(3) :175–190, 1997.

[OAWtH10] Chun Ouyang, Michael Adams, Moe Thandar Wynn, and Arthur H.M. ter Hofstede. *Handbook on Business Process Management 1*, chapter Workflow Management, pages 387–418. Springer, 2010.

[Ola10] Gisli Rafn Olafsson. Effective coordination of disaster response - the international perspective. In *7th International Conference on Information Systems for Crisis Response and Management*, Seattle, Washington, USA, 2010.

[OMMD10] Gérald Oster, Rubén Mondéjar, Pascal Molli, and Sergiu Dumitriu. Building a collaborative peer-to-peer wiki system on a structured overlay. *Computer Networks*, 54(12) :1939–1952, 2010.

[ope] Openstreetmap contributors, cc-by-sa, `http://www.openstreetmap.org/`, retrieved 26.05.2011.

[OR09] Elise Olding and Carol Rozwell. Expand your bpm horizons by exploring unstructured processes. Technical Report G00172387, Gartner, 2009.

[O'S09] Bryan O'Sullivan. Making sense of revision-control systems. *Communications of the ACM*, 52(9) :56–62, 2009.

[PBB10] Yong Peng, Y. Badr, and F. Biennier. Designing data-driven collaboration in service systems. In *4th International Conference on New Trends in Information Science and Service Science*, Gyeongju, Korea, 2010.

[Pes08] Maja Pesic. *Constraint-Based Workflow Management Systems : Shifting Control to Users*. PhD thesis, Technische Universiteit Eindhoven, 2008.

[PM02] Lori Peek and Dennis Mileti. *Handbook of Environmental Psychology*, chapter The History and Future of Disaster Research, pages 511–524. Wiley, 2002.

[pro] Processwave.org, `http://www.processwave.org`, retrived 11.01.2011.

[QD77] E.L. Quarantelli and Russel R. Dynes. Response to social crisis and disaster. *Annual Review of Sociology*, 3 :23–49, 1977.

[Qua83] E.L. Quarantelli. Emergent behavior at the emergency - time periods of disasters. Technical report, Disaster Research Center, University of Delaware, 1983.

[Qua86] Enrico L. Quarantelli. Organizational behavior in disasters and implications for disaster planning. Technical Report FEMA 104 / July 1986, FEMA, 1986.

[Qua91] E.L. Quarantelli. *Managing Natural Disasters and the Environment*, chapter Disaster Response : Generic or Agent-Specific, pages 97–105. The World Bank, 1991.

[Qua05] E.L. Quarantelli. Catastrophes are different from disasters : Some implications for crisis planning and managing drawn from katrina. Technical report, Disaster Research Center (DRC), University of Delaware, 2005.

[Rah10] Mohammad Ashiqur Rahaman. *Document-based agile workflows : Models, interoperability and security.* PhD thesis, EURECOM/TELECOM ParisTech, 2010.

[RB91] Tom Rodden and Gordon Blair. Cscw and distributed systems : the problem of control. In *2nd European Conference on Computer-Supported Cooperative Work (ECSCW'1991)*, Amsterdam, The Netherlands, 1991.

[RB07] Manfred Reichert and Thomas Bauer. Supporting ad-hoc changes in distributed workflow management systems. In *OTM Conference/Cooperative Information Systems (CoopIS)*, Vilamoura, Portugal, 2007.

[RD07] Jan Recker and Alexander Dreiling. Does it matter which process modelling language we teach or use ? an experimental study on understanding process modelling languages without formal education. In *18th Australasian Conference on Information Systems*, Toowoomba, Queensland, Australia, 2007.

[Rec08] Jan Recker. *Understanding Process Modelling Grammer Continuance - A Study of the Consequences of Representational Capabilities.* PhD thesis, School of Information Systems, Queensland University of Technology, Brisbane, Australia, 2008.

[Rei00] Manfred Reichert. *Dynamische Ablaufänderungen in Workflow-Management-Systemen.* PhD thesis, Fakultät für Informatik, Universität Ulm, 2000.

[Rin04] Stefanie Rinderle. *Schema Evolution in Process Management Systems.* PhD thesis, Abt. Datenbanken und Informationssysteme, Universität Ulm, 2004.

[RJVzMA07] Hajo A. Reijers, Monique Jansen-Vullers, Michael zur Muehlen, and Winfried Appl. Workflow management systems + swarm intelligence = dynamic task assignment for emergency management applications. In *5th International Conference on Business Process Management*, Brisbane, Australia, 2007.

[RMR00] Alberto B. Raposo, Léo P. Magalhaes, and Ivan L.M. Ricarte. Petri nets based coordination mechanisms for multi-workflow environments. *International Journal of Computer Systems Science & Engineering*, 15(5) :315–326, 2000.

[Rot00] Rotkreuzgemeinschaften. Drk-dienstvorschrift 100 - führung und leitung im einsatz, 2000.

[RRS09] Mohammad Ashiqur Rahaman, Yves Roudier, and Andreas Schaad. Document-based dynamic workflows : Towards flexible and stateful services. In *IEEE International Conference on Services Computing*, Bangalore, India, 2009.

[RS96] M. Raynal and M. Singhal. 1996. *Computer*, 29(2) :49–56, 1996.

[RW07] U. Rüppel and A. Wagenknecht. Improving emergency management by formal dynamic process-modelling. In *24th Conference on Information Technology in Construction*, Maribor, Slovenia, 2007.

[RWR06] Stefanie Rinderle, Andreas Wombacher, and Manfred Reichert. Evolution of process choreographies in dychor. In *OTM Conferences / 14th International Conference on Cooperative Information Systems*, Montpellier, France, 2006.

[Sac03] Freistaat Sachsen. Bericht der sächsischen staatsregierung zur hochwasserkatastrophe im august 2002. Technical report, Freistaat Sachsen, 2003.

[San09] Dan Sanderson. *Programming Google App Engine*. O'Reilly Media, 2009.

[SAP10] SAP. Sap research report 2009/2010. Technical report, SAP Research, 2010.

[SAST09] Peter Saint-Andre, Kevin Smith, and Remko Troncon. *XMPP : The Definitive Guide : Building Real-Time Applications with Jabber Technologies*. O'Reilly, 2009.

[SC99] Hala Skaf and Claude Charoy, Francois an Godart. Maintaining shared workspaces consistency during software development. *International Journal of Software Engineering and Knowledge Engineering*, 9(5) :623–642, 1999.

[SC02] Chengzheng Sun and David Chen. Consistency maintenance in real-time collaborative graphics editing systems. *ACM Transactions on Computer-Human Interaction*, 9(1) :1–41, 2002.

[SCG96] Hala Skaf, Francois Charoy, and Claude Godart. Maintaining consistency of cooperative software development activities. In *6th International Workshop on Foundations of Models and Languages for Data and Objects*, Schloss Dagstuhl, Germany, 1996.

[Sch99] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach : A tutorial. *ACM Computing Surveys*, 22(4) :299–319, 1999.

[Sch00] August-Wilhelm Scheer. *ARIS - Business Process Modeling*. Springer, 3rd edition, 2000.

[Sch08] Sabine Friederike Schulz. *Disaster Relief Logistics : Benefits of and Impediments to Horizontal Cooperation between Humanitarian Organizations*. PhD thesis, Technische Universität Berlin, 2008.

[Sch11] Bruce Scheier. Detecting cheaters. *IEEE Security & Privacy*, 9(2) :96–96, 2011.

[Sed03] Robert Sedgewick. *Algorithms in Java*. Addison-Wesley, 3rd edition, 2003.

[ser] Java community process (jcp), jsr 315 : Java (tm) servlet 3.0 specification, `http://www.jcp.org/en/jsr/detail?id=315`, retrieved 18.04.2011.

[SFA11] Cláudio Sapateiro, Antonio Ferreira, and Pedro Antunes. Evaluating the use of mobile devices in critical incidents response : A microworld approach. In *20th IEEE International Conference on Collaboration Technologies and Infrastructures (WETICE'2011)*, Paris, France, 2011.

[SG11] Klaus-Peter Schulz and Silke Geithner. The development of shared understandings and innovation through metaphorical methods such as lego serious play (tm). In *International Conference on Organizational Learning, Knowledge and Capabilities*, Hull, UK, 2011.

[Shi11] Shigeki Shibahara. The 2011 tohoku earthquake and devastating tsunami. *The Tohoku Journal of Experimental Medicine*, 223 :305–307, 2011.

[SO04] Karsten A. Schulz and Maria E. Orlowska. Facilitating cross-organisational workflows with a workflow view approach. *Data & Knowledge Engineering*, 51 :109–148, 2004.

[SO08] M. Statler and D. Oliver. *The Oxford Handbook on Organizational Decision-Making*, chapter Facilitating Serious Play, pages 475–494. Oxford University Press, 2008.

[SQ85] Robert A. Stallings and E.L. Quarantelli. Emergent citizen groups and emergency management. *Public Administrative Review*, 45(Special Issue : Emergency Management : A Challenge for Public Administration) :93–100, 1985.

[SSO01] Shazia Sadiq, Wasim Sadiq, and Maria Orlowska. Pockets of flexibility in workflow specification. In *20th International Conference on Conceptual Modeling*, Yokohama, Japan, 2001.

[Suc87] L.A. Suchman. *Plans and Situated Actions*. Cambridge University Press, 1987.

[SYY06] Jun Shen, Jun Yan, and Yun Yang. Swindew-s : extending p2p workflow systems for adaptive composite web services. In *Australian Software Engineering Conference*, Sydney, Australia, 2006.

[TAH+08] Omar Tahir, Eric Andonoff, Chihab Hanachi, Christophe Silbertin-Blanc, Frédéric Benaben, Vincent Chapurlat, and Thomas Lambolais. A collaborative information system architecture for process-based crisis management. In *12th International Conference on Knowledge-Based Intelligent Information Engineering Systems*, Zagreb, Croatia, 2008.

[Tar72] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2) :146–160, 1972.

[TBP09] Sébastian Truptil, Frédéric Bénaben, and Hervé Pingaud. Collaborative process design for mediation information system engineering. In *6th International Conference on Information Systems for Crisis Response and Management (ISCRAM'2009)*, Gothenburg, Sweden, 2009.

[TBP10] Sébastian Truptil, Frédéric Bénaben, and Hervé Pingaud. A mediation information system to help to coordinate the response to a crisis. In *Collaborative Networks for a Sustainable World, 11th IFIP WG 5.5 Working Conference on Virtual Enterprises*, St. Etienne, France, 2010.

[TI06] Nicholas E. Taylor and Zachary G. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *ACM SIGMOD International Conference on Management of Data*, Chicago, Illinois, USA, 2006.

[tic] Ticgit, `https://github.com/schacon/ticgit/wiki/`, retrieved 05.04.2011.

[TP10] Gina Trapani and Adam Pash. *The Complete Guide to Google Wave*. 3ones, Inc., 2010.

[Tuf06] Philippe Tufinkgi. *Logistik im Kontext internationaler Katastrophenhilfe - Entwicklung eines logistischen Referenzmodells für Katastrophenfälle*. Haupt Verlag, 2006.

[twi] Twitter, `http://www.twitter.com`, retrieved 06.05.2011.

[Unk06] Unknown. The federal response to hurricane katrina - lessons learned. Technical report, The White House, 2006.

[Van09] I. Vanderfeesten. *Product-Based Design and Support of Workflow Processes*. PhD thesis, Eindhoven University of Technology, 2009.

[vBM96] Peter van Beek and Dennis W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4 :1–18, 1996.

[vBR10] Jan vom Brocke and Michael Rosemann. *Handbook on Business Process Management 1*, chapter The Six Core Elements of Business Process Management, pages 107–122. Springer, 2010.

[vdA00] Wil van der Aalst. Loosely coupled interorganizational workflows : modeling and analyzing workflows crossing organizational boundaries. *Information & Management*, 37(2) :67–75, 2000.

[vdABEW01] W. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Proclets : A framework for lightweight interacting workflow processes. *International Journal of Cooperative Information Systems (IJCIS)*, 10(4) :443–481, 2001.

[vdAtHW03] W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske. Business process management : A survey. In *1st International Conference on Business Process Management*, Eindhoven, The Netherlands, 2003.

[vdAW01] W.M.P. van der Aalst and Mathias Weske. The p2p approach to interorganizational workflows. In *13th International Conference on Advanced Information Systems*, Interlaken, Switzerland, 2001.

[vdAWG05] W.M.P. van der Aalst, Mathias Weske, and Dolf Grünbauer. Case handling : A new paradigm for business process support. *Data & Knowledge Engineering*, 53(2) :129–162, 2005.

[vGS03] Rob van Glabbeek and David G. Stork. Query nets : Interacting workflow modules that ensure global termination. In *1st International Conference on Business Process Management*, Eindhoven, The Netherlands, 2003.

[VHSP10] Sarah Vieweg, Amanda L. Hughes, Kate Starbird, and Leysia Palen. Microblogging during two natural hazards events : what twitter may contribute to situational awareness. In *28th International Conference on Human Factors in Computer Systems (CHI'2010)*, Atlanta, Georgia, USA, 2010.

[Vid10] Renaud Vidal. The france-usa hro project - enhancing reliability in incident management. Technical Report 07-01-2010 (Working Paper), Centre d'Etude et de Recherche en Gestion Aix Marseille (CERGAM), 2010.

[VRvdA11] Irene Vanderfeesten, Hajo A. Reijers, and W.M.P. van der Aalst. Product-based workflow support. *Information Systems*, 36(2) :517–535, 2011.

[WA05] Jianrui Wang and Kumar Akhil. A framework for document-driven workflow systems. In *3rd International Conference on Business Process Management*, Nancy, France, 2005.

[Wac00] Tricia Wachtendorf. Interaction between canadian and american governmental and non-governmental organizations during the red river flood of 1997. Technical report, Disaster Research Center, University of Delaware, 2000.

[WdB85] William A. Wallace and Frank de Balogh. Decision support systems for disaster management. *Public Administrative Review*, 45(Special Issue : Emergency Managment : A Challenge for Public Administration) :134–146, 1985.

[Wei93] Karl E. Weick. The collapse of sensemaking in organizations : The mann gulch disaster. *Administrative Science Quarterly*, 38(4) :628–652, 1993.

[Wei00] Karl Weick. *Making Sense of the Organization*. Blackwell, 2000.

[WPT06] Gerhard Wickler, Stephen Potter, and Austin Tate. Using i-x process panels as intelligent to-do lists for agent coordination in emergency response. *International Journal of Intelligent Control and Systems (IJICS)*, 11(4), 2006.

[WR09] W. Wang, Jiacun an Tepfenhart and D. Rosca. Emergency response workflow resource requirements modeling and analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C : Applications and Reviews*, 39(3) :270–283, 2009.

[WRZW09] Barbara Weber, Hajo A. Reijers, Stefan Zugal, and Werner Wild. The declarative approach to business process execution : An empirical test. In *21st International Conference on Advanced Information Systems*, Amsterdam, The Netherlands, 2009.

[WW06] Minhong Wang and Huaiqing Wang. From process logic to business logic - a cognitive approach to business process management. *Information & Management*, 43(2) :179–193, 2006.

[ZPG10] Ehtesham Zahoor, Olivier Perrin, and Claude Godart. Disc-set : Handling temporal and security aspects in the web services composition. In *8th IEEE European Conference on Web Services*, Ayia Napa, Cyprus, 2010.

Résumé

De nombreuses catastrophes de diverses envergures frappent régulièrement des populations partout dans le monde. Parmi les exemples marquant on peut citer l'ouragan Katrina en 2005, le tremblement de terre en Haïti en 2010 ou plus récemment le Tsunami au Japon et la catastrophe de Fukujima qui a suivie. Au cours de ces catastrophes, plusieurs centaines d'organisations, comme la police, les pompiers ou les organisations d'aide humanitaire, interviennent pour sauver les gens et aider à revenir à une vie normale. Ces organisations ont besoin de se coordonner pour faire face à une situation dynamique avec des ressources limitées et une vision partielle de la situation. L'évolution de la situation entraîne souvent des changements d'objectif et de plan. Un des problèmes typique est d'obtenir un aperçu sur les relations entre ce qui a été fait, ce qui se passe actuellement et quelles sont les prochaines étapes. Ce problème est particulièrement difficile sur le plan inter-organisationnel : Chaque organisation coordonne la réponse de sa propre perspective et s'appuie sur les informations fournies par d'autres organisations.

Notre objectif dans cette thèse est d'étudier comment supporter la coordination des activités par des personnes de différentes organisations dans une situation dynamique par un système d'information. L'idée de base est de tirer profit d'une approche basée sur les processus, où les activités et leurs relations sont rendues explicites. Nous présentons un cadre pour la coordination des activités dans des situations dynamiques. Il permet la modélisation ad hoc des relations entre ce qui a été fait, ce qui se passe actuellement et quelles sont les prochaines étapes. Les écarts par rapport au modèle et comment les activités ont été réalisées sont affichées à l'utilisateur pour mettre en évidence l'impact de l'évolution des objectifs.

Nous étendons ce cadre au niveau inter-organisationnel. Certaines activités peuvent être partagées entre différentes organisations. Tout n'est pas partagé entre tout le monde pour tenir compte du respect de la vie privée, de la réglementation, des raisons stratégiques ou autres. Les activités partagées sont reproduites dans les espaces de travail de ces organisations. Nous décrivons comment des vues divergentes sur les activités et leurs relations peuvent être détectées et traitées afin de revenir éventuellement à une vue convergente.

Les concepts sont mis en œuvre comme une extension d'un service de collaboration distribuée ouvert. Ils ont été évalués par des gestionnaires de catastrophes expérimentés. Par ailleurs, nous avons conçu une expérience visant à évaluer l'utilisation d'outils pour aborder ces question. Nous avons effectué plusieurs expériences pour valider cette expérience. D'autres expériences pourront fournir une validation plus complété du modèle proposé dans cette thèse.

Mots-clés: coordination, activité, dynamique, situation, inter-organisationnel

Abstract

Recently we have seen several large scale disasters affecting humans all over the world. Examples are Hurricane Katrina in 2005, the Haiti earthquake in 2010 or the September 11/2001 terrorist attacks on the world trade center. During these disasters, several hundred organizations, such as police, fire brigade or humanitarian aid organizations, respond with the goal to save people and support them to live a normal life again. They need to coordinate to deal with scarce resources, different skills and capabilities. People in these organizations drive coordination based on their judgment of the situation. The situation can be dynamic : it evolves in sometimes unexpected ways, goals shift and priorities of the organizations change. Typical problems are to get an overview on the relations between what has been done, what is currently going on and what are the next steps. This problem is specially challenging on the inter-organizational level : Each organization coordinates the response from its own perspective and relies on the information provided by other organizations.

We aim in this dissertation at supporting coordination of activities by people of different organizations in a dynamic situation by an information system. The disaster response is a critical example for this. The basic idea is to leverage a process-based approach, where activities and their relations are made explicit. We present a framework for coordination of activities in dynamic situations. It allows ad-hoc modeling of the relations between what has been done, what is currently going on and what are the next steps. A model can be verified for correctness in predictable and acceptable time. Deviations from the model and how activities have been performed are displayed to the user to highlight the impact of shifting goals.

We extend this framework to the inter-organizational level. Selected activities can be shared by people with selected organizations. This means not everything is shared between everybody to take into account privacy, regulatory, strategic or other reasons. Shared activities are replicated in the workspaces of these organizations. We describe how diverging views on replicated activities and their relations can be detected and handled to ensure eventually a converging view.

The concepts are implemented as an extension to an open distributed collaboration service. They are also commented by experienced disaster managers. Furthermore, we design an experiment to evaluate tool support addressing the research questions. We conducted several experiments to validate the design of the experiment. Further experiments can provide validation of the concepts implemented as a prototype in this thesis.

Keywords: coordination, activity, dynamic, situation, inter-organizational