Generic Approach for Dynamic Disaster Management System Component

Marina Tropmann-Frick
Christian-Albrechts-University Kiel
Department of Computer Science
Christian-Albrechts-Platz 4, D-24118
Kiel, Germany
mtr@is.informatik.uni-kiel.de

Thomas Ziebermayr
Software Competence Center Hagenberg
Softwarepark 21, A-4232
Hagenberg, Austria
thomas.ziebermayr@scch.at

Abstract—Software support for disaster management requires approaches for flexible and generic processes. The “mini story approach” provides those properties and is therefore an appropriate concept to address the challenges of disaster management.

This paper focuses on the implementation of the mini story approach in the context of disaster management. It presents the concept, design and the mapping strategy as well as implementation details of the Dynamic Disaster Management System Component. This includes details on required design decisions and selection of appropriate technologies. The resulting Dynamic Disaster Management System Component is able to provide the required flexibility for task handling and the management processes in case of disaster.

I. INTRODUCTION

Most actions during a disaster response are not predictable and cannot be planned completely beforehand. So the corresponding processes cannot be prespecified and handled in a standard way. We introduce generic workflows for coordination of disaster management processes. They allow accurate, fast and dynamic activity guidance and information coordination in complex situations.

Generic workflows are flexible and adaptable workflows belonging to the area of process-aware information systems [1]. Process-aware information systems are going to be used in applications which demand higher flexibility [2]–[8] such as disaster management systems.

Workflow management systems are mostly used to organize and control business processes [9]. They are especially applicable for structured processes with sequential or parallel activities which require coordinated processing and involve several actors with different roles. Typical workflow management systems can mainly be used in a static environment with clearly defined organizational structures, completely given business processes and full control. Workflows for such business processes are completely predefined at process design time. Exceptions are part of the workflow. Deviations are often described as separate workflows. They must however be known at modeling time [10], [11]

Due to the highly dynamic situation changes in disaster management static approaches are quite ineligible. Although existing dynamic approaches can deal with a certain degree of flexibility, they may fail because of the huge number of parameters and case variations that must be considered [1].

In this paper we present the structure and main components of our generic approach and discuss in section IV our component design for the implementation of Dynamic Disaster Management System Component.

II. DISASTER MANAGEMENT

Disaster Management can be defined as the organization and management of resources and responsibilities for dealing with all humanitarian aspects of emergencies, in particular preparedness, response and recovery in order to lessen the impact of disasters [12], [13].

Disaster management is one of the challenging, complex and critical application areas dealing with hyper dynamic situation changes, high velocity, voluminous data and organizational heterogeneity. Successful management of disaster response requires flexible and adaptable solution techniques including very accurate, fast and dynamic activity guidance for supporting of process coordination, decision making and information logistics in real-time.

Effective disaster management depends on the informed participation of all stakeholders. In the situations that are categorized as disasters management and controlling depend on a large variety of parameters and on organizations handling the situation and services. There are always many different teams, services and information sources involved that must be integrated, e.g. police, fire brigade, medical or meteorological services. In some cases it is necessary to involve also actors with specific capabilities or knowledge about context, location or jurisdiction, e.g. the municipality, representatives from industrial complexes, divers, or local safety and security staff. Most disaster events can be divide into the phases depicted in Figure 1.

We concentrate in our work on the preparation phase and the response phase of disaster management. Those phases comprise the activities during and immediately after the current disaster event, such as saving of human life, protection of important constructions, supply of goods and services, as well as the protection of the environment. At this point disaster
management relies heavily on quick information flow and efficient decisions of disaster response teams.

III. GENERIC APPROACH

In the last section we described many critical issues for disaster management. Disaster management require intensive communication, coordination and immediate response to the changing environment. Most actions during a disaster response are not predictable and cannot be planned completely beforehand. So the corresponding processes cannot be prespecified and handled in a standard way. Therefore we apply the generic approach for coordination of disaster management processes. Our approach is based on the idea of genericity. It allows us to construct an abstract generic workflow which can be adapted to dynamic changes during the refinement process at runtime. This work extends our research on structure and basic components of generic workflows we already introduced in [14], [15] and [16].

A. Genericity

The notion of genericity is not new. According to [17], genericity can be described as a quality to be not specific, typifying, applied to or characteristic of all members of a genus, species, class or group. In our everyday life we come almost permanently upon generic activities. In science, particularly in computer science genericity is also widely used. A good example is the usage of generic algorithms in context of generic programming, e.g. [18].

We understand genericity as a capability to describe a group or class of objects on a certain abstraction level. This allows higher adaptation and flexibility.

B. Generic Components

In our previous work [14], [15] and [16] we already described in detail the construction and components of generic workflows. Therefore for the sake of completeness we give a short overview about the most important parts and refer for more details to our previous work.

1) Generic Functions: The concept of generic functions provided by Bienemann [19] is based on government and binding (GB) approach that was introduced by Chomsky [20]. Chomsky proposed a universal theory of languages. Basic concepts of the theory are the atomic units of the syntax [21], [22]. Consider functions \(F, F_1, ..., F_n\) of a chosen function algebra. Generic functions are functions

\[
F = (\text{Dom}, \phi, F, \psi, \text{Rng}),
\]

with free configuration parameters, with predicates \(\phi\) for the domain \(\text{Dom}\) and \(\psi\) for the range \(\text{Rng}\) of \(F\). A derived function is generated from \(F = \theta(F_1, ..., F_n)\) based on expression and instantiation of the configuration parameters and on instantiation of the predicates; \(\theta\) is here an \(n\)-ary operator \(\theta \in \text{op}^F\) (set of function manipulation operators [19]). In [15] we described the approach of generic functions more detailed.

Generic functions are basic elements for generic workflows. They represent the atomic activities within a generic workflow and are indecomposable.

2) Mini Stories: As we described above, the generic functions are atomic components of generic workflows. The first level of abstraction for our generic approach are semantically logical units - mini stories [14]. This are also atomic components, but in an abstract semantic way. Mini stories represent abstract collections of generic functions which can be dynamically composed at runtime based on parameter initialization. We define a mini story as a quadruple [16]

\[
\mathcal{M} = (\mathcal{F}, \mathcal{T}, \mathcal{S}, P),
\]

where \(\mathcal{F}\) is a set of generic functions as defined above. \(\mathcal{T}\) is a set of transitions within a mini story. \(\mathcal{S}\) is a set of parameters defining the current state of the mini story. And \(P\) is a priority function.

Transitions are given as tuples of the form \(T_{ij} = (F_i, F_j) \in \mathcal{T}\) and can be represented as edges of a directed graph with node set \(\mathcal{F}\).

The priority function \(P\) is defined as follows:

\[
P : (\mathcal{F}, \mathcal{S}) \rightarrow \mathbb{R}_{\geq 0},
\]

and assigns each function \(F_i \in \mathcal{F}\) depending on the current state \(S_j \in \mathcal{S}\) a priority. The function with the highest priority is the best for current situation.

The instantiation of a mini story is performed at runtime depending on the current state (and influencing parameters) during the execution of the priority function.

3) Generic Workflows: As the next level of abstraction for our generic approach we define generic workflows as collections of semantically indecomposable mini stories. In order to specify a generic workflow the composition rules for mini stories are needed. These rules describe the conditions for composition of mini stories within a generic workflow. For example there can be rules that require the execution of some specific mini story after another or even as a successor of another specific mini story. There can also be rules defined for the prohibition of mini story execution in a specific order. Mini story composition is a complex issue, that can be characterized by the following three general aspects:
The order of mini story execution is partly given by the execution order of generic functions. Depending on the priority function generic functions for the next execution step are selected. Therefore only those mini stories can be executed as next, which contain the selected generic functions and optimally start with one of them.

On the other hand some rules for the composition are given by the context, where the execution takes place. The context of disaster management discussed in this paper possesses specific requirements and conditions. For the most disaster categories there are hazard maps, contingency plans or other guidelines existing (e.g. from the natural hazard management or municipality), which partly determine the order of mini story execution.

The next important part are the influencing parameters. They can be characterized as configuration or control parameters. Some parameters belong only to one mini story and get their values allocated during the instantiation. Another parameters can be shared between various mini stories. As a consequence the instantiation of one mini story reduces the set of mini stories suitable for the next step and sets inevitably limitations for the instantiation of the following mini stories.

IV. COMPONENT DESIGN AND IMPLEMENTATION TECHNOLOGY

Given the mini story approach as solution to the requirements for dynamic and generic needs of disaster management and disaster processes the task was to design a component which implements the mini story approach for the usage in disaster management. At the beginning of the design phase two possibilities to implement the mini story approach where identified:

- Implementation of a full functional process engine which is able to interpret, instantiate and execute processes based on the mini story approach.
- Mapping of the approach to available implementations of process engines utilizing their functionality to implement the mini story approach.

Currently numerous process engine implementations are available. They provide rich functionality and are able to run processes and manage their instances including task assignment. Tool support also encompasses tools for modelling the processes and client implementations which provide the required interface for task selection. In addition interfaces for mobile applications are available. Implementing this functionality from scratch requires enormous effort with even less maturity of the implementation. Although this brings some disadvantages we decided to utilize available implementations and to map the mini story approach to this environment. The details of this mapping are described in the following.

A. Mapping of Generic Workflows

At first a modelling language is required to define the mini stories. As described earlier mini stories consist of generic functions which are connected to a process, the mini story. This can be modelled by a modelling language that supports sequence flows and tasks or activities. Some modelling languages are available that supports the modelling of this kind of processes like UML activity diagrams [23]. For this application BPMN 2.0 [24] was chosen to model mini stories for various reasons:

- Industry standard for business process modelling
- Rich tool support for modelling and interpretation
- Intuitive model of control flow based process models
- Modelling of mini stories with BPMN 2.0 possible

But choosing BPMN 2.0 or even any other standard process modelling language has some disadvantages concerning the features of the mini story approach. As described earlier the mini story approach supports the combination of mini stories to a process during the adaptation phase. Mini stories provide the capsule for processes with tasks that belong together from a domain perspective. But in general the selection comes down to the selection of functions which can be combined. So the approach supports to use any function as an entry point for a mini story and any function as an exit point. Using a process modelling language like BPMN 2.0 does not support this feature as during modelling the possible entry and exit points (start and end events) are defined. Other entries or exits are not supported. So mapping of the approach requires to define mini stories as atomic units with defined entry and exit points which cannot be changed during instantiation or runtime.

BPMN 2.0 not only provides means for process modelling, the standard also describes how a workflow interpreter has to interpret process models defined according to the BPMN 2.0 standard. Although the standard leaves some decisions on how to interpret certain modelling constructs open (as discussed in numerous publications and also in [25]), various process engine implementations are available which in common do the same. Even some open source implementations are available which provide a sufficient level of maturity. So the logical next step is to choose a workflow interpreter for the interpretation of mini stories modelled based on BPMN 2.0. The selection of the engine and its integration is described in the next subsection.

This workflow engine will be used to execute mini stories. Mini stories are atomic due to the limitations of BPMN 2.0 and therefore the mini story will start with one of the defined entry points. If during the process execution changes to the environment or the context make the process obsolete or adaptation to the process is necessary the execution will be stopped or aborted. For the management of starting, stopping or aborting mini stories a component has to be provided which is able to:

- Select mini stories based on the current situation and according to the rules defined for the mapping to mini stories and according to the rules defined for combining mini stories.
- Control the process engine including starting of processes, stopping of processes
- Storing the history of selected, executed, finished, stopped
and aborted mini stories. 
• Provide an overview over running and finished mini stories.

This component (Dynamic Workflow Component, (DMC)) implements the main functionality of the mini story approach, building of custom processes adapted to the current situation in disaster management based on generic mini stories. It provides the core functionality and is the abstraction level to the process engine used for task processing and control of sequence flows in mini stories. The concrete architecture and implementation of the DMC will be presented in the next subsection.

B. Component Design

The dynamic disaster process component consists of two sub-components and a storage for the required data as sketched in Figure 2.

![Component Architecture](image)

Fig. 2. Component Architecture

The DMC provides two interfaces, one waiting for situations indicating a disaster or at least requiring measures defined as mini stories. And another interface that supports querying an overview on the current situation in a certain region and the running and finished processes within a certain duration. The DMC uses a database where the mini stories are stored, the rules for combining them can be queried and where information on how to map situations to mini stories can be found. Tools for modelling mini stories directly will provide the process models stored in the database.

As described in the mapping section the Dynamic Disaster Process Component utilizes a workflow engine for running the selected mini stories. For the concrete implementation we have chosen the JBoss workflow engine as it provides the required functionality and also supports for rule handling which helps in the process definition and execution. We have not conducted a deep evaluation of process engines as this is not the main issue here.

The DMC uses the JBoss workflow engine for the execution of mini stories and is able to control the execution by the provided interface of the JBoss workflow engine.

The workflow engine provides an interface for task handling where applications used for task management during the disaster can ask for open tasks and also inform about task completion or exceptions during task processing. The workflow engine uses a database for persistence of process states and context information.

When a situation event occurs it provides information about the situation itself and its parameters like coordinates, flood height or other measurements or predictions describing the situation in detail. This information will be used to select appropriate mini stories. But this information will be provided to the mini story as context information within the process engine. So during process execution decisions based on the context information are possible or this context information can be provided as information to the people processing the tasks of the mini story.

When processes are finished the workflow engine informs the DMC about successful process execution. The DMC is also informed in case of exceptions during task processing or any other unforeseen problems. This enables the DMC to provide this information to others and also to react on that by starting new mini stories dealing with those issues or mini stories which base on the already finished mini story.

The implementation of the dynamic disaster process component has shown that some experience is required to control the JBoss workflow engine. Due to the rich functionality the appropriate handling is not very easy and as the current version is very new and documentation of open source implementations is not the same than for commercial products it is hard to learn that. But all in all the integration of the JBoss workflow engine provides the required functionality and therefore is an important part of the mini story implementation.

C. Evaluation

In this subsection we will have a closer look whether the proposed mapping of the approach to existing technologies addresses the requirements on disaster management. In this section it is not evaluated whether the mini story approach is the right solution, this was already discussed in other publications ([15], [16], [14]).

The goal is to provide a dynamic solution for disaster management which is able to provide the generic and dynamic features of the mini story approach. Utilizing BPMN 2.0 for modelling mini stories and using an existing workflow engine supports this required features and enables using existing tools and knowledge. The combination with the dynamic selection and adaptation component combines the missing functionality with already available workflow engines. Although the proposed mapping was implemented based on JBoss, any other workflow engine providing similar functionality for process instantiation, management and task processing can be used as workflow engine. This is especially supported by a component architecture which defines an abstraction layer between...
process engine and the process management component.

This shows that despite the missing functionality for choosing arbitrary entry and exit points in and out of mini stories, that the approach addresses all challenges of disaster management and provides a solution ready to use in a real environment.

V. Conclusion

In this contribution we discuss our generic approach for disaster management and show that our solution can address crucial challenges of permanently changing disastrous situation. We present how this approach can be implemented based on existing technologies adding the required genericity and flexibility to address the challenges of disaster management. Utilizing existing technologies helps on the one hand to provide a mature solution with manageable effort and on the other improves the usability through well known modelling languages and tools. Although this approach prevents the availability of the full feature set of the mini story approach, the advantages outweigh the disadvantages.

Our solution is going to be used in the EU-project INDYCO [26]. The main goal of this collaborative project is the development of an INIntegrated DYNAMIC decision support system COMponent for disaster management systems. The dynamic process component belongs to the core functionality of the provided component.

Acknowledgment

The authors would like to thank all partners from the INDYCO project [26] for their support and contribution to the project.

The publication of this paper has been supported by the project Vertical Model Integration (VMI) 4.0. The project VMI 4.0 is supported within the program “Regionale Wettbewerbsfähigkeit OÖ 2007-2013” by the European Fund for Regional Development as well as the State of Upper Austria.

References