Application of Generic Workflows for Disaster Management

Bernhard THALHEIM\(^a\), Marina TROPMANN-FRICK\(^{a,1}\) and Thomas ZIEBERMAYR\(^b\)

\(^a\) Christian Albrechts University Kiel, Department of Computer Science, Christian-Albrechts-Platz 4, D-24118 Kiel, Germany
\(^b\) Software Competence Center Hagenberg GmbH, Softwarepark 21, A-4232 Hagenberg, Austria

Abstract.
Workflow management systems provide support for structured processes and help to follow the defined business process. Although their importance has been proved by various applications over the last decades they are not appropriate for all use cases. Such workflow management systems are only applicable for domains where the process is well structured and static. In various domains it is essential that the workflow is adapted to the current situation. In this case the traditional workflow systems are not applicable. A flexible approach is required.

Refinement of specifications is inherently connected to the development of information systems. Throughout the development process models are refined towards the implementation. Especially the coherence of the models developed throughout this process is important. Concepts for adaptation have been developed in the area of functions. The application of this methodology in combination with the abstraction of workflows based on the concept of word fields allows to solve the adaptation problem for workflow applications.

This concept of generic workflows addresses the required adaptation and provides mechanisms to describe generic workflows and refine them during runtime to specific workflows adapted to the current situation. The chosen conservative approach is based on proven methods and provides a robust approach for workflow adaptation. And so it allows us to handle the highly dynamic characteristics of disaster management.

Keywords. Generic workflows, disaster management, workflow management systems, dynamic processes, refinement methods

1. Introduction

Workflow management systems are used to improve and optimize business processes in business software since at least two decades [5,9,15]. Workflow management systems are especially applicable for structured processes with sequential or parallel activities which require coordinated processing and involve several people with different roles.

\(^1\)Corresponding Author: Marina Tropmann-Frick, Christian Albrechts University Kiel, Department of Computer Science, Christian-Albrechts-Platz 4, D-24118 Kiel, Germany, email: mtr@is.informatik.uni-kiel.de
A workflow is a finite set of connected activities which may produce or consume events. The workflow itself is started or interrupted based on events. In common workflows are business processes with several involved roles and persons supported by a software system.

Workflow management comprises all tasks required for modeling, simulating, executing and steering business processes. Workflow management tools support those tasks by software tools like a modeling tool or workflow interpreter. The main support is provided by the workflow engine which interprets the workflow model and coordinates the involved parties and their tasks.

Workflow management systems are process oriented systems which consist of a build time component to define workflows and a runtime component to run a workflow. Such systems are different to document management systems which are used to produce, provide, to forward and to archive documents in the context of organizational processes.

Workflow management helps to optimize the usage of resources and provide a view on the companies business processes which help to make the right management decisions. Profit can be increased by consequently align the business processes to the customers needs. So a workflow management system is the basis for a controlling system that allow to steer the success of the company [10].

Actual workflow management systems can be used in a static environment where on the one hand the organizational structure is clearly defined including responsibilities, deputies and controlling. And on the other hand with defined business processes and process owners which are followed with a high discipline and without exceptions. Such business process are well supported by standards like EPK, BPMN and protocols like SWAMP [11,29]. Such processes are developed with the existing development methods in a standard way.

But when business processes, tasks and the course of action are dynamic, current approaches cannot be applied. Until now workflows are completed during process design. Possible exceptions have to be described as part of the workflow. Deviations are often described as separate workflows and only those parts which are known at modeling time [28,15]. During workflow processing the user has to adapt to the defined workflows because it is required to strictly follow the workflow model. No exceptions or variations are allowed outside the model.

Within the scope of workflow management there are already several approaches existing for the adaptivity, flexibility and handling of dynamic changes [8,17,18,25,21,23]. In [32] the author describes two types of changes within a workflow management system: (1) ad-hoc changes and (2) evolutionary changes. Ad-hoc changes specify an opportunity to provide customer specific solutions or to handle rare events, the process is then adapted for a single case or a limited group of cases. Evolutionary changes involve the results of reengineering efforts. The process is then changed to improve responsiveness to the customer or to improve the efficiency. In this approach the author adopt the notion of process families to construct generic workflow process models. But the understanding of genericity there is distinguished from our approach in several respects and the presented solution is restricted by the adoption of techniques for a special workflow management system as are also many other approaches in this area, e.g. [25,23].
Disaster management is highly dynamic and it is not possible to predefine every exception or variation. During disaster response highly dynamic measures are applied which are possibly also applied in other disasters, but which have to be adapted to the actual disaster scenario. Therefore the application of actual workflow management systems for disaster management is not possible. Currently teams for disaster response are trained to apply patterns for disaster response developed based on experiences gained during previous disasters. Exceptions and variations are explicitly allowed and disaster management is possible due to abstract processes. So a pattern on how to respond to disasters is specified which is only adapted to the actual situation. This leads us to the concept of generic workflows.

2. Fundamentals of Generic Workflows

This section should help to understand the scientific and technical basics of generic workflows. The concepts and theories were developed as basis for software development and to solve practical problems.

2.1. Generic Functions

In the first decades of the development of information systems the specification of the functionality was not important. The provided mechanisms for ad-hoc specification of queries was sufficient for the most use cases. The success of information systems led to the application of such systems in domains where the required functionality was rather static. Thus the specification of the functionality and its structure was required at the beginning of the software development. Various languages where introduced, e.g. Event Process Chains (EPC) [31] and workflow modeling languages [33,16]. Those languages at least supported a conceptional specification of the required functionality. In the most cases those languages where used only to define the conceptional specification of the functionality. A layered specification of the functionality dependent on the users point of view, a conceptual view and dependent on the developers point of view similar to a layered architecture was not developed. Those different views would support the refinement from the conceptional specification to the development view.

Modern information system applications have challenges which can be addressed by a theory, technique and pragmatic of generic functions:

*Three layers in functional specification:*

The functionality have to be specified for different views on the application. For the presentation and conceptual view but also for the development view.

*Abstract description of business process functionality and applications:*

Functions, both for the user and the developer, must have the same semantic or have to be defined in an abstract semantic way which allows a similar use, regardless of the chosen abstraction.
**Mapping of abstract specifications on conceptual and procedural specifications:**

To support the three abstraction levels comprehensively it is necessary to support refinement of functionality in different release states. This includes refinement of of functionality which is not entirely defined.

**Improving flexibility of the functionality specification:**

Functions can be implemented in different ways dependent on the application, the user, the usage history and the system context. Therefore it is required to provide this flexibility also for the specification of the functionality.

**Development of mechanisms for adaptation of functionality:**

All these facets: views, abstraction levels, refinements and adaptations requires automation support. Therefore appropriate mechanisms for adaptation must also be developed which support those facets.

The concept of *generic functions* was introduced 2006 at the Entity-Relationship conference\(^2\) [2] and was central part of the dissertation from Bienemann [1]. The prototypical implementation provided by Bienemann is based on government and binding introduced by Chomsky [6]. Chomsky proposed a universal theory of languages. It bases on observations of multi language communication. Speaking follows a two phase approach.

A universal grammar can also be defined in two phases. Basic concepts are here the atomic units of the syntax. Lexical items are combined together to a D-structure that might be a collection of concept fields [7]. In the first phase required concepts for communication are gained from a D-structure. For filtering \(\alpha\)-rules are used. In the second phase the expression is transformed to S(entence)-structure using \(\beta\)-rules. S-structures depend on the target language and might also differ dependent on transforming to text or to spoken word. The abstract transformation is depicted in **Figure 1**.

![Figure 1. Stepwise generation of an expression using the Government and Binding Theory](image)

This concept can be utilized for the stepwise generation of functions in a system. In the first step \(\alpha\)-rules are used to transform generic concepts to a function class as depicted in **Figure 2**. Available information and steering rules are used for this transformation.

---

\(^2\)It has been applied successfully for the development of web information systems (governmental portals)
In the second step $\beta$-rules are applied to develop the specific function from the function class. How available information and specific rules are used to select an adequate function is depicted in Figure 3.

Generic functions are functions $F = (Dom, \phi, F, \psi, Rng)$ with free configuration parameters, with predicates $(\phi, \psi)$ for the image area $Rng$ and the initial area $Dom$. A derived function is generated from $F = \theta(F_1, ..., F_m)$ based on expression and instantiation of these parameters and on instantiation of the predicates.

A typical example of a generic function is the Search-function. This function combines seven different kinds of search:

- Search by a query with knowledge about data structure and data reachability
- Search in known data set
- Search masks with search expressions
- Browsing and tracing
- Search for associations and digging in the deep
- Browsing, refining search
- Zapping

This different types of search are also dependent on the context, the supported functions and the user. The context encompasses available data, data associations, support for storing search result and the search history. Typical functions for search support are...
functions for browsing, for zapping, functions to remember searches, functions for information about changes and functions that support the definition of queries. Typical query support are functions to lead the user, that provide an overview, showing the way back, for browsing and for zapping.

Refinement of a generic function $F$ to a function $(S, F, \Sigma, s_0)$ is characterized by three dimensions as depicted in Figure 4 from [1].

1. Instantiation of parameters, embedding in the appropriate data space and its structure $S$, dynamic consistency conditions $\Sigma$ and a starting point $s_0$ in the data space.
2. Enhancement by the context of the application, available data, applicable system support, comunity of practice and usage history.
3. Refinement of functions to specific functions using refinement as specified in computer science theory.

![Figure 4. Dimensions of generic Functions](image-url)

2.2. Word Fields and Generic Functions

The specification of structures as well as the specification of functionality is inherently dependent from the lexical semantics of words or concepts. A word like *product* or *person* is linked to a basic semantic concept of a database type, its basic properties and mechanisms for processing.

This semantic of the used words is refined stepwise in the process of specification to the specification of types with defined structure, views on data of this types, usage and storage of the types.

This approach has not only proved to be very powerful but is also widely accepted and applicated intuitively in all application domains. Therefore it is possible to define general abstraction layers for information systems.
These layers are:

- Use Case layer
- Business Process layer
- Concept layer (Architecture)
- Implementation layer

Although these abstraction layers are widely used and accepted (at least by the software development process), a specification along this layers is currently not entirely supported.

Lexicons of natural languages can be characterized by word fields. A word field groups words syntactically as well as semantically and pragmatically to word classes. A simple word field is *say-speak-talk* which groups different kinds of a communication type. This word field encompasses several facets:

- **inform**: shout, cry, whisper, assert, announce
- **mention**: note, bring something up, touch on something
- **ask**: call for something, query, seek
- **request**: asking for, propose, order
- **answer**: reply, respond,
- **repeat**: echo, iterate, recur, replicate

   Word fields combines phonetic, morphological, syntax, semantic and pragmatic as sketched in **Figure 5**. This structure enhances the frame of word fields used in the area of linguistic to address required aspects of software systems.

The description of processes follows the morphology. The theory of word fields can be extended to concept fields by conjugation and instantiation of the following parameters:

- Actor profile and portfolio
- Repetition profile
- Time profile
- Deontic mode with imperative, conjugative and indicative characteristics
- Action and process direction to specify the relation between actor and process

A concept field is a generic concept which is used for the instantiation of a concept using a set of properties.

In order to construct generic workflows by means of the introduced concepts it is necessary to examine the required specification aspects and issues. In the main there are three areas to be covered, which have already been presented as the three dimensions of generic Functions: instantiation/composition, refinement and context embedding/context injection. The context injection part goes beyond the scope of this paper and is only touched in the next sections. Refinement and composition are essential for the specification of generic workflows. The following section provides therefore a detailed description of these concepts.
3. Construction of Generic Workflows

In this section we describe how the previously explained theories can be applied to the concept of generic workflows.

Generic functions can be used as atomic activities within generic workflows. The specification can be defined similar to workflow specification languages like BPMN [11] although it is necessary to improve the languages towards formal correct and unambiguous languages [4]. Therefore the specification of generic workflows can be implemented as follows.

3.1. Components and Steering Mechanisms of Generic Workflows

An **atomic generic workflow** is a generic function.

**The simple control statements** are

- *The sequential execution*: generic workflows are processed sequentially. The semantic of the first generic workflow complements the semantic of the second generic workflow. Conflicting combination of semantics of the generic workflows leads then to an empty workflow.
- **Parallel branch**: generic workflows are executed in parallel and the generic workflow terminates when all branches terminate.
- **Exclusive selection of branch**: exactly one branch can be non-deterministically selected for processing.
- **Synchronisation**: allows execution in parallel branches with a synchronisation condition.
- **Simple Merge**: two alternative branches can be brought together.

**Extended branch and synchronisation operations** are
- **Multi branch**: various execution paths can be selected.
- **Multi merge**: various branches can be merged.
- **Discriminator**: different execution paths without synchronisation can be merged. Generic workflow parts are only executed once.
- **n-out-of-m assembly**: different execution paths with partial synchronisation can be merged. Generic workflow parts are only executed once.
- **Synchronised assembly**: different branches with full synchronisation can be merged. Generic Workflow parts are only executed once.

**Structural statements** are
- **Repetition**: generic workflows can be executed any number of times.
- **Implicit termination**: stops a generic workflow.

**Data related operations** are
- **Static operations**: which are executed according conditions which can be checked during compile time.
- **Static operations**: which are executed according conditions which can be only checked during runtime.
- **Operations with defined runtime assertions**: allow to predefine the creation of a limited set of repetitions.
- **Operations with conditions for synchronisation**: which allow to execute unlimited alternatives in parallel with synchronisation when operations are ending.

**State based operations** are
- **The delayed selection**: where all alternatives are executed and the selection of the appropriate alternative is done after execution.
- **The related parallel execution**: alternatives are executed sequentially in random order.
- **The milestone oriented operation**: where an activity is executed until a milestone is reached.

**Termination statements** are
- **Termination operation**: a generic function is terminated.
- **Case termination**: a case is terminated.

This algebra can be used to construct generic workflows. It introduces a strict hierarchical specification. This prevents the specification of some of the expression discussed in workflow literature.
This algebra enables us to avoid typical problems of languages for generic workflows. Such problems can be relatively easy solved, if we look at the semantic of results. In this case both programs are represented by AND-AND programs. If we look at the execution semantic both programs are different. Even more difficult are workflow semantics, where synchronisation might happen both at the end and at the beginning of a branch. In this case even the branch gets a different semantic.

This is the reason we prefer a strict semantic for generic workflows. This restriction supports refinement but also supports solving conflicts at the data layer by limitation.

A workflow can be seen as a course of action through process layers. Each step in a workflow can be related to a natural number. In cycles the natural number for the cycle as well as the step number is used. These numbering is possible because we only allow hierarchical structures (Fitch structures) [12]. Such a numbering can be defined dense so that every number \( n \) has a predecessor \( n - 1 \) (for \( n > 1 \)) and the successor \( n + 1 \) (for \( n < \text{max} \)) exists. A process layer is defined by the process layer number. Process layers represent the stepwise and where necessary parallel execution of a workflow.

### 3.2. Methods for Refinement

The refinement of processes and calculations is a repeatedly occurring and difficult task in information science. Additionally it is required to consider the current context of the process at runtime. The approach we use here is to utilize Abstract State Machines (ASM) [3] for the refinement of workflows. With a refinement of a workflow the behavior of the general abstract workflow should not change, at least for the viewer.

This allows us to specify the refinement mechanism by the set of observed states and observed events. A workflow is a refinement of an abstract workflow if the behavior of the observed states and events is a refinement and if all processes of the abstract workflow are also part of the refined workflow.

For this kind of refinement the \( Y \)-refinement methodology can be applied. Both, the generic and the specific workflow and their context are input for refinement. So these three inputs are reflected by the commonly refined workflow.

Processes can be defined here by generic functions. Therefore the \( Y \)-Refinement is either a refinement of the generic functions or a refinement of the workflows. In case of generic functions the \( Y \)-refinement corresponds to context injection for generic functions. In case of refinement of workflows the \( Y \)-refinement is usually more complex without the possibility to find a common solution.

But it is possible to develop a conservative solution. For that we use the structure of the workflow as starting point and limit the refinement options as follows:

- **Simple control statements** like the sequential execution, parallel branch, exclusive selection of branch, synchronisation and simple merge are refined by replacing an activity of the workflow by a workflow in case of a sequential execution, by an exclusive extension in case of parallel branches and by extension for simple merge.

- **Extended branch and synchronisation operations** are not refined.

- Refinement of **structural statements** is not possible.
Data related operations can be refined by the refinement of data structures where the abstract data structures can be represented by views.

- Refinement of state based operations is not possible.
- Refinement of termination statements is not possible.

This refinement is supported by the previously described government-and-binding technique in a way that implementation support can be developed.

3.3. Context Injection

The context of workflows is defined by the current application, available data, available system support, community of practice and its usage history. The context of the specific workflow is additionally influenced by the history of the workflow. So it is possible to inject the context in the workflow stepwise along the execution of the generic functions. But it is important to use always the same context.

The formal aspects of context injection are presented in the context modelling part of [24]. The authors examine there the evolution of the context for actors, scenarios, systems and over time, and model the relation between different contexts by lifting relations. So can the properties, that are valid for a certain context, be lifted to another context. This transfer can be based on local model semantics.

3.4. Development of Current (Specific) Workflows from Generic Workflows

We use for the development of current workflows from generic workflow the previously described refinement methods as well as the definition of generic functions and a methodology for stepwise, conservative extension of the context and of the instantiation. The development of the workflow is completed when all parameters are bind to values.

This development of the workflow can be tested already at specification time for all essential scenarios for consistency. Scenarios are defined by the storyboarding approach [30]. The scenarios are used to derive test cases for workflows. Those test cases are used for tests of fully developed workflows. This can be done during specification of the workflow to avoid influence on workflows during runtime. So the following four development methods are used by our approach for the development of current workflows from generic workflows:

- Development by refinement: The workflow is refined conservatively.
- Development by development of generic functions: The generic functions are developed in the same order as they occur in the workflow. Relations of parameters, during instantiation and in the context are handed over using attribute grammar.
- Development by context extension: The context will be stepwise injected in the generic functions and the workflow steering operations.
- Development by instantiation: All generic functions are instantiated along the process. Relations defined for instantiation are handed over using methods of attribute grammar.

If a generic workflow is consistent in the scenarios, refinement methods can be applied during runtime to develop stepwise a workflow from a generic workflow. This development is therefore conservative.
4. Concept of Generic Workflows

This section describes our approach of the conception for generic workflows. We explore here in particular generic workflows, current workflows and workflow fields that are essential for our goal to handle the highly dynamic characteristics of disaster management.

Generic workflows are abstract, configurable and adaptable workflows which are used to derive the current (developed) workflow instance. The current workflow considers the current situation, the current requirements and the current data available.

In Figure 6 a generic workflow is depicted which is the basis to derive a current workflow. This current workflow is developed within the border of the generic workflow and uses all services of the generic workflow.

![Figure 6. Generic and Instantiated Workflow](image.png)

A generic workflow consists of generic functions. During instantiation a current workflow is derived with the help of adapters from a generic workflow. This concept allows flexible adaption of workflows to the current situation.

4.1. Generic Workflows, Current Workflows and Workflow Fields

According to the theory of concept fields we distinct between:

- **generic workflows**, which are the basis for the derivation of workflows,
- a **workflow**, which is a current workflow developed from a generic workflow,
- and **workflow fields**, which are used to define the frame for generic workflows.

A generic workflow might consist of one or more principal forms. A workflow field consists of:

- a set of principal forms,
- a set of dynamic integrity constraints which must be met by workflows of a workflow field,
- a set of development forms which associates the workflow field with other workflow fields,
- a set of flection to derive workflows from workflow fields.

We often assume for abstraction that a workflow field only has one principal form and that a generic workflow only consists of workflows from one workflow field. A generic workflow not necessarily contains all possible workflows of a field, it might consists only of some current workflows.

We have applied this distinction for the implementation of an e-learning Web site. This site allows the development of a course based on meta information and context.
information on activities, actors and data as well as the history of activities. This allows to understand a learning field as an abstract course

- where the principal form is defined as an expression of course elements;
- which is extended based on derivation rules to a complex learning module in a way that the student get the required learning elements;
- and where flection allows to develop variants and exceptions. Flection rules allow an extension
  - by actor profile and actor portfolio
  - by a repetition profile
  - by a time profile
  - by a deontic mode and
  - by kinds of action and by direction of action.

This extension meets the requirements of the application in disaster scenarios. Disaster scenarios have specific properties beyond software applications. They are characterized by

- highly parallel processes and therefore problems in handling the complexity of this parallelism;
- processes which requires the feedback about idle times;
- organizations which are often controlled from outside.

So we generalize the morphology of story lines and introduce abstract workflow fields:

- The opening field is characterized by
  - the depiction of the context which dominates when the workflow field is associated with other fields
  - the depiction of actors
  - the depiction of the situation and
  - the association with view types for input, retrieval and output data.

- The initial position field is used for meta specification and supports embedding the geographic and time context together with motivation and causes.

- The activity step field is specified by
  - connection parameters
  - supporting elements and context conditions
  - actor roles
  - view types.

- The delivery field supports handing over objects from one actor view to the view of another actor. Additionally the specification of the context and agreement of how to handover the object is possible.

- The working field supports processing view data and sending them to other actors and also their insertion into the system.

Beside those basic fields the specification of construction field that allow to combine fields is possible:
• The branching field supports the branch of workflows in parallel synchronized workflows.
• The repetition field provides the frame for repeated execution of a workflow.
• The bulk field binds parameter relevant for the workflow field.

We have harmonized this theory of workflow fields with composition operations of workflows to support the development of complex workflow fields. Workflow fields support an abstract depiction of workflows.

Generic workflows are analogous to generic operations like insert, delete and update. Those operations are instantiated based on type information for which they are used. Generic workflows can be developed to concrete specific workflows similar to the instantiation of generic operations. Parameters used might be interdependent. We distinct the following types of generic workflows:

• **Developable workflows** are generic workflows with a generic runtime workflow where the instantiable parameters have no relations to other parameters. They can be fully developed during runtime. A typical example for this type of workflow is a workflow for working groups where each group member has the same tasks.

• **Parallelized workflows** are generic workflows which store intermediate results and provide to do lists that can be set during runtime of the workflow. Those to do’s might contain references to other workflows e.g. shared resources.

• **Multiple choice workflows** are generic workflows which allow variants for roles, selection of data and which therefore allows bundling with other workflows.

• **Transaction based meta workflows** are generic workflows with integrated resource and role management, compensation fields and therefore integrated transactional semantic.

A developed workflow is a complete instantiated workflow. All parameters of the generic workflow have defined values. The relation between generic workflow and developed workflow is depicted in Figure 6. According to this picture a developed workflow is an iteration or a specific instance of a generic workflow.

### 4.2. Coherence and Inner Correlation in Generic Workflows

It is admitted that the development of workflows is a complex process. Business processes should address the requirements of the given application but should also optimize resource usage (e.g. duration and storage capacity). The complexity of the processes often requires complex software. In a lot of cases even at project start it is not entirely clear which tasks should be addressed by the workflow application. Clients often have a vague picture of the required functionality. Additionally requirements change even during software development. Business processes are usually used a long period of time. Therefore they are modified and even implemented based on different hardware and software. This leads to complex software with an unmanageable high number of dependencies preventing maintenance and adaptability. An example for such systems are systems which bases on SAP R/3. SAP R/3 is able to support nearly every information or data need in a company. But it is inherently complex and every small adaptation is time consuming and requires a team of experts.
During workflow development models are used for intermediation. The model specifies the relevant aspects of the process but is also used as guide for development of the supporting system. So the model should support the specification of the process in an understandable way, but have to be also detailed enough for software development. The model should also help the developer to understand what the software should do. Models are therefore vital part of software development.

Workflows should consider various aspects of a software application at an adequate detail level. So it is possible to describe all aspects of the workflow application within the workflow model. Another possibility is to describe those aspects in a set of different but related models. In the early stages of information technology the one model approach was applied. But the failure using this approach leaded to the application of the multi model approach.

Those weak related specifications are grouped to model ensembles. Modifications may be applied to each model of the ensemble. But the developer is responsible for the management of those modifications and their impact. As in the most cases a team of developers implements software, it is not possible for them to handle the relations for every change. So in [1] the approach of model development is extended by adjustment of models. The adjustment of the models in the ensemble is based on integrity constraints which must be met by the model ensemble. This model ensemble is called model suite. The model suite defines explicit relations between its models similar to relations defined between elements of an architecture which help to implement the models.

The different models of a model suite represent in the most cases the different abstraction levels used during the development process of a workflow. Some of the models are used for communication with the customer and the user, others are used for the documentation of the products that will be implemented.

Therefore the relationship of the models (coherence) is one of the topics we have to address in the concept of generic workflows. The following aspects are the most important and should be addressed:

- Means to characterize coherence in a model suite have to be defined which allow to describe different levels of coherence dependent on development status.
- Criteria to assess semantic coherence applicable to incomplete models are required.
- Methods to define and maintain coherence within a model suite.
- Methods should be integrated in workflow development tools.

4.3. Adaptation of Generic Workflows to the Current (Specific) Situation

Workflows describe the business processes implemented in a software application.

The developed theory and technology of generic workflows supports the utilization of rule based transformation methods for the development of a workflow from a generic workflow. Such methods are known for the usage in computer analysis and computer algebra. Rules are used for the refinement of generic workflows to specific workflows. We have applied this method already for the preparation of course material of an e-learning web site. This method allowed us to adapt course material to already finished learning steps, to the availability of learning material, to the learning history, to the profile of the student, to the problem fields of the student and to add related material. Based on this method it is possible to provide appropriate course material.
The development of the workflow (refinement) is based on properties of the current situation. This allows to inject the application context.

One development path might be incompatible with other paths. Therefore it is necessary to check coherence rules for refinement of the workflow from a generic workflow. An abstract solution is not available for this problem. But it is possible to utilize test scenarios to check for conflicts during specification of the workflow. If such conflicts exist, the workflow needs to be revised.

4.4. Controlling Data Quality

The response to disasters from a software technical point of view strongly depends on data and their quality. Therefore it is important to consider for such application cases the database explicitly. Dependency on data especially occur when large data is used or insufficient, unreliable or have limited quality due to other reasons. Therefore it is necessary to consider those properties during development of the workflow from a generic workflow e.g. data errors occurring in sensors.

This leads to a couple of challenges:

1. Development of a method to handle inexact measurement and data orientation.
2. Development of a method to improve data quality and to improve data density if required.
3. Development of a method to evaluate whether a data set is sufficient for a workflow or not.

The improvement of data quality is based on algorithms which support overloaded data, data sets and colored data. Dependent on data viability appropriate methods for data refinement respectively statements on inapplicability, inappropriate date and incoherence are used. The conditioning of data for intelligent data observation in disaster management comprises methods for data validation, data merging, object identification, analysis of multirelational data and methods for data quality classification. In this area also known methods to assess accuracy of measurement [19,27,14].

The filling data gaps is implemented based on different methods dependent from data type and dimension [22,13].

5. Conclusion and Future Work

In some cases the application of static workflows is inappropriate as the context changes before the specified workflow is instantiated. Therefore the support of adaptable workflows is required in such use cases. In this paper we propose an approach that bases on the concept of generic functions and combines it with the word field theory. In combination with the consideration of the coherence of the workflow models this approach provides a robust and powerful approach to define generic workflows and to refine them to current workflows.

Our approach of generic workflows differ from the approaches of dynamic workflows [26] in many ways. The elements and structures of dynamic workflows are mostly predefined and are put together dynamically at runtime. Our presented approach of generic workflows supports adaptation to the highly dynamic disaster management, there it is not possible to predefine every situation variation or occurring exception.
The concept is theoretically elaborated. But the next step will be to apply this approach to a real world scenario. So we plan to apply this approach to the domain of disaster management where the situation obviously changes. We use the elaborated and here presented concepts as an essential part of the framework for the project INDYCO [20]. This approach should help to provide a software system that supports disaster management by workflow support customized to the actual disaster and therefore improve disaster management.

References


