Rule-Based Adaptation of Workflow Patterns for Generic Workflows

Marina Tropmann-Frick, Niklas Sasse
Christian-Albrechts-University of Kiel
Department of Computer Science
Kiel, Germany
Email: {mtr, nsa}@informatik.uni-kiel.de

***Abstract*—This paper focuses on the application of an adaptation rule system for generic workflows based on the basic concepts of rewrite systems in the context of disaster management. 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. We propose to use generic workflows that satisfy these complex requirements and that provide support for organizing processes and information flow in disaster situations thereby providing decision support to crisis managers and "first responders". We illustrate our approach in a case study for disaster management.**

Keywords–adaptation; generic workflow; workflow pattern; rewrite system; term rewriting; disaster management.

I. INTRODUCTION

Workflow management systems are commonly used to plan, execute and control all kinds of business processes. Workflow management helps to optimize the usage of resources and to provide a view on the business processes of a company, which help to make the right management decisions. Profit can be increased by consequently align the business processes to the customer's needs. This makes a workflow management system the basis for controlling system that allows to steer the success of a company [1].

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 [2] [3]. EPC (Event-driven Process Chain) and BPMN (Business Process Model Notation) support such business processes and their modeling [4] [5].

Generic workflows are flexible and adaptable workflows belonging to the area of process-aware information systems (in particular workflow management systems). Process-aware information systems are going to be used in applications which demand higher flexibility [6]–[12]. For example, M. Reichert and B. Weber [13] survey approaches to manage dynamics in process-aware information systems. We can distinguish between design-time and run-time flexibility. Variability, adaptation, evolution and looseness are the four main categories of flexibility.

Disaster management represents an important, versatile and critical domain. The processes of disaster management can be assigned the looseness category of flexibility. They are non-repeatable (every process instance is different), unpredictable (there is no knowledge existing about situation changes during an event) and emergent (the processes emerge during execution when more information becomes available). The situation specific parameters are unknown in the beginning and might change during process execution. Because of the huge number of parameters and possibilities of process development, dynamic approaches can easily become too complex and incomprehensible for dealing with.

Disaster management processes are highly dynamic and there is no possibility to predefine every exception or variation. Therefore, 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.

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 at runtime. We present the structure and basic components of generic workflows and show how our approach satisfies requirements of disaster management in a case study. This work extends our research in [14]–[17], which is going to be used in the EU-project INDYCO [18]. The main goal of this collaborative project is the development of an INtegrated DYnamic decision support system COmponent for disaster management systems.

A. Paper Structure

This paper is structured into four sections. The first section introduces important terms and discusses related work in the context of process aware information systems and disaster management. In Section 2, the structure and components of generic workflows are defined and explained. In Section 3, the essential aspects of term rewriting systems and the approach of their application for the adaptation techniques of generic workflows are described. The following Section 4 involves a case study from the area of disaster management. Afterwards we conclude by summarising and discussing our next steps and future work.

II. GENERIC WORKFLOWS

This section explains the methodical and technical fundamentals of generic workflows. We understand generic workflows as abstract, configurable, mutable and adaptable workflows, which are used to derive the current workflow instance.

 ISBN: 978-1-61208-408-4

The current workflow is developed within the framework of the generic workflow and uses all its services. The current workflow considers the current situation, the current requirements and the current data available.

A. Genericity

The notion of genericity is not new. According to [19], 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 [20]. 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]–[17], 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 [21] is based on government and binding (GB) approach that was introduced by Chomsky [22]. Chomsky proposed a universal theory of languages. Basic concepts of the theory are the atomic units of the syntax [23], [24].
Consider functions $F, F_1, ..., F_n$ of a chosen function algebra. Generic functions are functions

$$\mathcal{F} = (Dom, \phi, F, \psi, Rng), \tag{1}$$

with free configuration parameters, with predicates ϕ for the domain Dom and ψ for the range Rng of $\mathcal{F}$. 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; θ is here an n-ary operator $\theta \in op^{\mathcal{F}}$ (set of function manipulation operators [21]). 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: The next 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]

$$\mathcal{M} = (\mathcal{F}, \mathcal{T}, \mathcal{S}, P), \tag{2}$$

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:

$$P : (\mathcal{F}, \mathcal{S}) \rightarrow \mathbb{R}_{\geq 0}, \tag{3}$$

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.

C. Adaptation

The general understanding of adaptation in computer science means a process in which an interactive system adjusts its behavior to an individual user by processing context information. Context information is defined as any information that can be used to characterize the situation of entities (whether a person, place or object) [25]. A readjustment of a system is often necessary after gaining new or additional context information. The concept of adaptation can be applied for generic workflows as well. The interactive system is the generic workflow itself and the individual adjustments develop

the unfolded workflow based on the given context information available at the time.

At this point it is necessary to define the scope of adaptation. In this case, adaptation means the rule-based transformation of a generic workflow or rather the composition of different mini stories which will eventually build the concrete and unfolded workflow. The processing of context information parameters, the specification and instantiation of generic tasks within the mini stories is to this effect not part of adaptation but rather a part of workflow refinement and goes beyond the scope of this paper.

The adaptation is performed via rule-based transformations of a generic workflow. Therefore, a system of well-structured rules has to be derived. This system should fulfill some properties to avoid runtime errors or complications between different rules. Since adaptation is heavily dependent on context information it is not possible to derive one abstract and universally valid system of rules. A set of rules is merely applicable for similar situations within an application domain.

In disaster scenarios a lot of action patterns are transferable so that similar sets of adaptation rules can be applied to different disaster situations. In combination with the appropriate refinement methods for generic tasks a decision support system can be developed that allows an automated adaptation for effective response in disastrous situations.

III. Rewrite Systems

To create a reliable system of rules for the adaptation of generic workflows the system has to be well-structured. For that purpose the concept of rewrite systems, especially term rewriting systems are considered to derive suitable properties for the rule-based transformation of generic workflows. Rewriting covers a wide range of methods for replacing subterms of formulas or terms with other subterms using rules.

A. Term Rewriting

Term rewriting systems are sets of directed equations which are used to repeatedly replace subterms of a formula until the simplest form of it is reached. The term rewriting introduced by Gorn [26] can be seen as a nondeterministic Markov algorithm which is used as an effective way to analyze and evaluate algorithms.

A term t consists of a set of function symbols F and a set of variables X [27]. Working with terms often requires replacement of parts of the initial terms with new terms. If at the position p of a term t the following subterm $t|_p$ is replaced with a new term s it is represented as $t[s]_p$. Through this operation a new term u is created which equals t except in position p so that $u[s]_p = t$ is true [27]. The new term u can be seen as an abstraction or a specification of t based on the context leading to the transformation. A substitution σ is a special kind of rewrite relation where terms are assigned to several variables of the initial term $\{x_1 \rightarrow s_1, ..., x_m \rightarrow s_m\}$. For example, there is a substitution $\sigma = \{x/g(y)\}$ for a term $f(x)$ then $f(x)\sigma = f(x\sigma) = f(g(y))$ is true after applying the substitution.

In the scope of term rewriting systems we find mainly binary relations $\rightarrow$ which are used on sets of terms T. They are called term rewriting relations. The general idea in term rewriting is the usage of directional equations. A term rewriting rule of a set T is an ordered pair $< l, r >$ of terms that stands in a directional binary relation $l \rightarrow r$. A finite or infinite set of term rewriting rules R over T is called term rewriting system. For a given term rewriting system R and the terms s and t of the set of terms T, s should be replaced by t, then $s \rightarrow_R t$ is true if $s|_p = l\sigma$ and $t = s[r\sigma]_p$ comply for a rule $l \rightarrow r$ in R with position p and substitution σ [27].

Term rewriting systems should be applied to a term as long as it is possible. If no term t exists for a subterm s in T so that $s \rightarrow_R t$ is applicable, the term s is no longer reducible and is in a normal form. A derivation in R is every possible sequence $t_0 \rightarrow_R t_1 \rightarrow_R ... \rightarrow_R t_i \rightarrow_R ...$ which develops from applying term rewriting rules. If every derivation leads to at least one normal form, the term rewriting system is called normalizing. If independent of the derivation the same normal form is always reached from an initial term, the term rewriting system is called canonical [27].

The basic concepts of term rewriting systems can be applied for generic workflows. In this case, the set of terms is comparable with a generic workflow that can be represented as a graph with a tree structure. The different paths are derivations of the generic workflow and possible compositions of mini stories. When a leaf is reached the term is in normal form. That means the generic workflow is concrete and unfolded. The navigation that leads the flow of the generic workflow is based on rules similar to term rewriting systems. The substitutions or rather abstractions or specifications are based on context information instead of matching variables. Therefore, the left side of an adaptation rule represents context information and the right side the mini story that has to be executed if the current context matches the rules context. By following this procedure and repeatedly applying the adaptation rule system the generic workflow will more and more unfold until a normal form is reached and the concrete workflow is complete.

In order to create a system that matches a broad variety of situations given through context information a lot of adaptation rules need to be derived. Furthermore, additional properties have to be fulfilled or at least need to be considered to prevent unexpected problems by application of an adaptation rule system to a generic workflow.

B. Termination

One of the most important characteristics of the term rewriting systems is the termination property. A binary relation $\rightarrow$ on a set T terminates if there are no endless chains $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow ...$ of elements generated from T. For a term rewriting system R this implies that it terminates if there are no endless derivations $t_1 \rightarrow_R t_2 \rightarrow_R ...$ developed by the application of R on the set of terms T. That means, every term in T has at least one normal form. In case of term rewriting systems termination can be proved by using term orders, e.g., *Knuth-Bendix-Order*, *Recursive-Path-Order* or *Polynomial-Order* [28] [29].

For generic workflows termination implies that no endless sequence of mini stories is generated by applying the adaptation rule system. In practice it is obvious that a generic workflow cannot be endless because at some point every business process comes to an end, especially if dealing with disaster scenarios. Some problems arise based on the flexibility of generic workflows. E.g., it is possible and required to repeat the same mini story more than once. This can lead to cycles

 ISBN: 978-1-61208-408-4

within the workflow and for this reason to endless activity chains. Therefore, it is necessary to define assumptions to maintain the termination property.

For this problem we can use the mini stories as labels on the nodes of our workflow graph, so that there can be many repetitions of the same mini story on different nodes between which we can distinguish. In order to monitor the whole process the adaptation rule system must be controlled by a higher instance. For that purpose a controller must be implemented which monitors the workflow during runtime. Its main priority is then to prevent runtime errors and exceptions by using appropriate abort criteria.

C. Church-Rosser Property

Presumably the set of term rewriting rules is finite in a term rewriting system. The *Church-Rosser* property is used to describe, whether for the terms s and t of the set T the statement $s =_R t$ is true. Therefore, the term rewriting system R is applied to both terms to check whether the results are identical. If derivations differ it is possible that different normal forms are reached during the usage of the term rewriting system. This problem does not exist if in every situation (for all derivations of the initial term) a term exists where they can be reunited. This means that every derivation of a term leads exactly to one single normal form. This characteristic is described as Church-Rosser property [30].

A binary relation $\rightarrow$ on a set of terms T is called Church-Rosser relation if its reflexive-transitive-symmetric closure $\leftrightarrow^*$ is maintained in the junction relation $\rightarrow^* \circ \leftarrow^*$. That means for every term t_1 and t_2 in a set of terms T, if $t_1 \leftrightarrow^* t_2$ is true, there exists another term s in T, for which $t_1 \rightarrow^* s$ and $t_2 \rightarrow^* s$ is true as well [27].

The Church-Rosser property is equivalent to the slightly simpler property of confluence [31]. A binary relation $\rightarrow$ on a set of terms T is confluent if the reflexive-transitive-symmetric closure of the relation $\leftarrow^* \circ \rightarrow^*$ is maintained in the junction relation $\rightarrow^* \circ \leftarrow^*$. That means for every term u, t_1 and t_2 in a set of terms T with $u \rightarrow^* t_1$ and $u \rightarrow^* t_2$ there exists another term s in T, so that $t_1 \rightarrow^* s$ and $t_2 \rightarrow^* s$ are true as depicted in Figure 1 [27]. So the confluence of a term rewriting system R on a set of terms T implies the impossibility of more than one normal form for every particular term $t_1, ..., t_i$ of T. But at the same time it does not guarantee the existence of a normal form for every term $t_1, ..., t_i$ of T because of the termination property that must be fulfilled as well in this case.

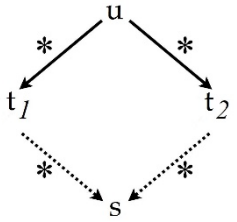

FIGURE 1. CONFLUENCE

Regarding the adaptation rule system for generic workflows the fulfillment of confluence is not crucial because for the concept of generic workflows it isn't important that the resulting unfolded workflow is in every case the same. In reality, it is rather the opposite. This results from the fact that after every application of an adaptation rule new context information is processed. Therefore, it is possible that the priority for the execution of different mini stories may change over time and affect the finally concrete workflow. However the important effect that can be shown through confluence is the prevention of execution for some mini stories by application of certain adaptation rules.

D. Critical Pairs

The consideration of critical pairs is an extension of the Church-Rosser property and the confluence. In general confluence is a nondeterministic criteria [32] but with special methods it is possible to force confluence for finite and terminating term rewriting systems.

Two different term rewriting rules $l \rightarrow r$ and $s \rightarrow t$ overlap if they are both applicable on the same term. In this case, a decision is necessary to determine which rule shall be executed. The proof of *Knuth & Bendix* [33] considers all possible positions of terms and subterms on which term rewriting rules can be applied and it shows that every critical pair can be solved by using the superposition test. Thereby confluence can be forced for a finite and terminating term rewriting system.

The handling of critical pairs is a very important part by creating an adaptation rule system for generic workflows because there occur a lot of them, especially when dealing with disaster scenarios where many activities are performed simultaneously by various actors and the amount of available resources is limited.

To determine which adaptation rule should be executed if more than one rule applies at the same time for the given context of a generic workflow, it is necessary to prioritize the tasks of the mini stories. The processing of the prioritization is also part of the controller which monitors the generic workflow during runtime. The controller has to consider different actors and the amount of available resources before it decides which action alternatives should be performed. The goal of prioritization is the effective usage of all resources and appropriate selection of mini stories possible for a given context. Thus the concrete, unfolded workflow is the most efficient way to carry out the treated business process.

IV. CASE STUDY

By the case study we want to illustrate how our ideas can be used in a practical way. In order to demonstrate our approach and create a first adaptation rule system for generic workflows we selected two scenarios from the disaster management area. The scenarios were reviewed to deviate rules and to gather information that the controller can use for evaluation of different situations. To have the ability to compare both regarded scenarios are flooding events, so that rules and information deviated from the first scenario can directly be evaluated with the second. In both cases activities on the micro level and context information is supposed to be given so that the description is more on the abstraction level of mini stories.

The first scenario is an emergency plan that describes the actions which have to be performed mainly by a task force during a flooding disaster. It is an abstract scenario

 ISBN: 978-1-61208-408-4

and refers to a region in Austria near a huge European river. The plan distinguishes different levels of escalation. At first all measures are described which will help dealing with the situation without taking any severe damage or encounter exceptional circumstances. These measures contain among other things description of how the information flow works and of allocations of actors to their respective roles. Furthermore, the plan shows how to handle some critical situations, e.g., the burst of a dam or a complete system failure. For these exceptional escalations suitable countermeasures are specified.

After analyzing the emergency plan adaptation rules for the generic workflow were derived. For the deriving process it was important to consider the level of abstraction. On the one hand the rules cannot be too specific because they should also be applicable for similar disaster scenarios. On the other hand, if they are too abstract they are of little help in guiding the sequence of the generic workflow at all. Based on the given information first adaptation rule system was formed containing several rules which describe and process the activities during a flooding event. the rules were prioritized and divided into three different categories so that the controller should be able to decide between rules if necessary.

The derived adaptation rule system was then evaluated with the second disaster scenario in order to prove whether the rules and the prioritization are applicable to control a generic workflow during similar situations. The second scenario deals with a real flooding disaster which took place in Germany in 2010. It was triggered by heavy and ongoing rainfall and led to a quick escalation of the situation. Similar to the emergency plan the response actions were performed mainly by a task force, the information flow and the involved actors are accurately documented as well.

During evaluation it was tested if the derived adaptation rule system is also applicable to a real scenario. In general most of the adaptation rules were suitable for the second scenario and the prioritization of the performed actions and selected mini stories was mostly correct. Minor changes and additional rules which could be derived from this scenario were then added to the adaptation rule system. Therefore, it could be shown that the system is applicable to similar scenarios.

However the current system isn't able to control the generic workflow during a disaster scenario by itself. Much more information has to be gathered through processing a lot of disaster situations, so that the adaptation rule system can handle any given situation which is described through context information. Especially the functionality of the controller is important to give sufficient support for making intelligent decisions and to learn from new situations.

V. Conclusion and Future Work

In this contribution we discuss an adaptation rule system approach for generic workflows and its application for disaster management. 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 intend to use generic workflows for coordination of disaster management processes. They allow accurate, fast and dynamic activity guidance and information coordination in complex situations.

The theoretically elaborated concepts were tested positively while they were applied to a real world scenario. Therefore, it is possible to integrate the adaptation rule system into the framework of generic workflows. This concept should help to provide a software system to support decision making and workflow control in disastrous situations.

Our next step will be to specify a refinement mechanism for generic workflows that could fit to our overall concept. We intend also to develop a toolkit which allows information gathering about past disaster management processes and preparation of the information for generic workflows.

References

[1] L. Fischer, Ed., Workflow Handbook 2003. Future Strategies Inc., Published in association with the Workflow Management Coalition, 2003.

[2] D. Georgakopoulos, M. Hornick, and A. Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," in Distributed and Parallel Databases, 1995, pp. 119–153.

[3] N. Russell, A. H. Hofstede, D. Edmond, and W. van der Aalst, "Workflow Data Patterns: Identification, Representation and Tool Support," in 24th International Conference on Conceptual Modeling, L. M. L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos, and O. Pastor, Eds. Klagenfurt, Austria: Springer, 2005, pp. 353–368.

[4] L.Fischer, Ed., BPMN 2.0 Handbook Second Edition. Future Strategies Inc., Published in collaboration with the Workflow Management Coalition (WfMC), 2012.

[5] D. M. Stephen A. White, BPMN Modeling and Reference Guide. Future Strategies Inc., 2008.

[6] C. Ellis, K. Keddara, and G. Rozenberg, "Dynamic change within workflow systems," in Proceedings of conference on Organizational computing systems, ser. COCS '95. New York, NY, USA: ACM, 1995, pp. 10–21.

[7] Y. Han and A. Sheth, "On Adaptive Workflow Modeling," in Proceedings of the 4th International Conference on Information Systems Analysis and Synthesis, Orlando, Florida, July 1998, pp. 108–116.

[8] P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, and M. Teschke, "A Comprehensive Approach to Flexibility in Workflow Management Systems," in WACC99, Work Activities Coordination and Collaboration, ACM Press, San Francisco, USA, February 1999, pp. 79–88.

[9] M. Momotko and K. Subieta, "Dynamic change of Workflow Participant Assignment," in ADBIS 2002. LNCS. Springer, 2002.

[10] J. Klingemann, "Controlled Flexibility in Workflow Management," in Proceedings of the 12th International Conference on Advanced Information Systems Engineering, ser. CAiSE '00. London, UK, UK: Springer-Verlag, 2000, pp. 126–141.

[11] W. M. P. van der Aalst, "How To Handle Dynamic Change and Capture Management Information? An Approach Based on Generic Workflow Models," Comput. Syst. Sci. Eng., vol. 16, no. 5, 2001, pp. 295–318.

[12] R. Müller, U. Greiner, and E. Rahm, "AgentWork: a Workflow System Supporting Rule-Based Workflow Adaptation," Data and Knowledge Engineering, vol. 51, no. 2, 2004, pp. 223 – 256.

[13] M. Reichert and B. Weber, Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Berlin-Heidelberg: Springer, 2012.

[14] B. Thalheim and M. Tropmann-Frick, "Mini Story Composition for Generic Workflows in Support of Disaster Management," in Proceedings of the 24th international workshop on Database and Expert Systems Applications, ser. DEXA 2013. IEEE Computer Society, 2013, pp. 36–40.

[15] B. Thalheim, M. Tropmann-Frick, and T. Ziebermayr, "Application of Generic Workflows for Disaster Management," in Proceedings of the 23rd European-Japanese Conference on Information Modelling and Knowledge Bases, ser. Information Modeling and Knowledge Bases XXIII, Y. Kiyoki, T. Tokuda, and N. Yoshida, Eds. IOS Press, 2013, pp. 68–85.

[16] M. Tropmann-Frick, B. Thalheim, D. Leber, C. Liehr, and G. Czech, "Generic Workflows - A Utility to Govern Disastrous Situations," in Proceedings of the 24th European-Japanese Conference on Information Modelling and Knowledge Bases, ser. Information Modeling and

 ISBN: 978-1-61208-408-4

Knowledge Bases XXIV, Y. Kiyoki, T. Tokuda, and N. Yoshida, Eds. IOS Press, 2014, pp. 473–485.

[17] M. Tropmann-Frick and T. Ziebermayr, “Generic approach for dynamic disaster management system component,” in Proceedings of the 25th international workshop on Database and Expert Systems Applications, ser. DEXA 2014. IEEE Computer Society, Sept 2014, pp. 160–164.

[18] INDYCO, “http://www.is.informatik.uni-kiel.de/ project/indyco/en/,” accessed 06.04.2015.

[19] Webster’s Third New International Dictionary, 1993.

[20] D. R. Musser and A. A. Stepanov, “Generic Programming,” in Lecture Notes in Computer Science 358. Springer Verlag, 1989, pp. 13–25.

[21] A. Bienemann, Context-Driven Generation of Specifications for Interactive Information Systems, ser. Dissertationen zu Datenbanken und Informationssystemen. AKA, 2008.

[22] N. Chomsky, Some Concepts and Consequences of the Theory of Government and Binding, ser. Linguistic Inquiry Monographs. MIT Press, 1982.

[23] N. Chomsky, The minimalist program. Cambridge: MIT Press, 1995.

[24] E. Stabler, “Derivational Minimalism,” in Logical aspects of computational linguistics, C. Retore, Ed., vol. LNCS 1328. Springer, 1998, pp. 68–95.

[25] D. Crestani, A. Jean-Marie, and C. Coves, “Petri Nets Analysis: Complexity and Finite Coverability Graph in Modular Design,” Studies in Informatics and Control, vol. 14, no. 1, 2005, pp. 55–64.

[26] S. Gorn, J. Hart, and S. Takasu, “Explicit Definitions and Linguistic Dominoes,” in Systems and Computer Science. University of Toronto Press, 1967, pp. 77–105.

[27] N. Dershowitz and J.-P. Jouannaud, “Handbook of Theoretical Computer Science (Vol. B),” J. van Leeuwen, Ed. MIT Press, 1990, pp. 243–320.

[28] N. Dershowitz, “Termination,” in Proceedings of the First International Conference on Rewriting Techniques and Applications (Dijon, France), ser. Lecture Notes in Computer Science, vol. 202. Springer-Verlag, Berlin, 1985, pp. 180–224.

[29] ——, “Orderings for Term-Rewriting Systems,” in Theoretical Computer Science, vol. 17, no. 3, 1982, pp. 279–301.

[30] A. Church and J. B. Rosser, “Some Properties of Conversion,” in Transactions of the American Mathematical Society, vol. 39, no. 3, 1936, pp. 472–482.

[31] M. H. A. Newman, “On Theories with a Combinatorial Definition of ”Equivalence”,” in Annals of Mathematics, vol. 43, no. 2, 1942, pp. 223–243.

[32] G. Huet, “Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems: Abstract Properties and Applications to Term Rewriting Systems,” J. ACM, vol. 27, no. 4, Oct. 1980, pp. 797–821.

[33] D. E. Knuth and P. B. Bendix, “Simple Word Problems in Universal Algebras,” in Automation of Reasoning. Springer-Verlag, Berlin, 1983, pp. 342–376.

 ISBN: 978-1-61208-408-4