

Capture and Dissemination of Experience about the Construction of Engineering Processes

Christian Rupprecht, Martin Fünffinger, Holger Knublauch, Thomas Rose

Research Institute for Applied Knowledge Processing (FAW)
Helmholtzstraße 16, D-89081 Ulm, Germany
{rupprech, fuenffin, knublauch, rose}@faw.uni-ulm.de

Abstract. Process know-how is instrumental to govern engineering processes in a network of engineering departments as well as migrate changes of processes due to emerging technological or other opportunities. In this paper we present a process construction kit for the capture and dissemination of process knowledge at the level of process structures as well as process construction experience. Our approach ranges from the formal capture and maintenance of know-how about processes to the implementation of process adaptations in terms of object-oriented concepts. Requirements are drawn from specific reference applications in the automotive engineering and the plant construction domain.

1 Introduction

For the past years, process management has been coined by terms such as business process re-engineering [10,13], workflow support [11], or even process innovation [3]. The prime focus has been the (re-) design of process flows with respect to efficiency, quality or customer satisfaction. That is, optimisation of enterprise resources has been the focal point. Only recently, the attention is shifting from an enterprise resource planning stance towards a business enabling stance to generate expertise. Rather than optimising processes, the prime goal is to generate business opportunities. Process knowledge, be it at the level of process structures or experience about process construction, emerges as critical factor for an enterprise's performance.

As a matter of fact, many aspects of process experience that used to be implicit in the organisational culture, need to be made explicit and even transferable. The prime objective of our research is to elevate the capability to capture and disseminate experience in the construction of engineering processes. The question arises of how to "construct" processes in light of best-practice and technological opportunities, e.g. employ simulation methods instead of crash tests with prototypes.

In our approach, the construction of a process refers to the intellectual task of specifying the activities and their relationships. This task is further compounded by the nature of engineering processes, which are prevalingly characterised by attributes such as innovative, individual, dynamic, interdisciplinary, strongly interrelated, strongly parallel, iterative, communication intensive, anticipatory, planning intensive, uncertain, risky, etc. [18]. Modelling such processes is necessary for reasons of

transparency, documentation, execution, communication, co-ordination, co-operation, planning, what-if analysis, etc.

Original services of a pro-active process construction approach should include:

- initial construction of processes based on best-practice, technological opportunities, and quality guidelines;
- on the fly adaptations along the dimensions of extension, refinement, individualisation, and configuration;
- online alarm amid execution.

Our approach is founded in a process construction kit. The process construction kit consists of a repository for managing basic building blocks and structures of processes, and an array of operators for adapting pieces of processes as well as orchestrating processes from smaller pieces.

Based on experiences in two large European companies in the automotive and utility plant construction sectors, we defined a process engineering environment for the construction, assessment, synchronisation, and individualisation of processes [21,22]. The specific intention is to provide dedicated support to process engineers who themselves are experts in the engineering domain, i.e. there are no third party champions involved that formally represent and adapt processes following a sequence of interviews. The process construction kit aims at the conceptual design of processes in terms of ontologies while at the same time offering automated support for individualising processes.

This paper is organised as follows. In section 2, we elaborate the essentials of processes, process models, and their use with regard to process construction. Section 3 presents our approach for managing process experience at a conceptual and system level. The main objective is to establish a comprehensive framework for coping with developing experience in process construction — ranging from a conceptual framework “down to” an ontology-based formal representation of process construction methods. Related approaches are discussed in section 4, while future extensions of the overall framework are discussed in section 5.

2 Processware

2.1 Processes

In the field of process management, many definitions carrying different foci on the term process can be found. In order to gain a common understanding for this paper on what a process is, we define a *process* as a set of temporally or logically ordered activities intended to reach a goal involving resources. In ordinary usage, a task is often synonymous with a process. In this paper, we understand a *task* as the definition of a goal together with the information about data, objects, resources, rules and other constraints needed for achieving this goal, whereas working on the task is called a process. *Activities* have no externally visible substructure and they are performed by *agents* that can be humans or machines (cf. [12]).

The order of activities can be derived either from their temporal constraints or logical relations among each other. Temporal constraints can be specified, e.g. by assigning starting and ending time to activities. In this case, a temporal order of activities in terms of a time schedule (e.g. gantt chart) can be derived without knowing anything about logical dependencies. Logical relations between activities capture knowledge about the sequence of activities on a more generic level independent of a specific time schedule. In both cases, the result is the ordered sequence of activities, which emphasises the dynamic character of processes.

2.2 Process Models

A process can be regarded as a system where the elements are activities and resources and the relations are the sequential or logical dependencies between those elements. The set of relations describes the *process structure*.

Unlike many other definitions, we understand the design of a process model as a *constructive process* rather than a mere mapping of a process structure given in reality. A process structure is not per se existent in reality and cannot be perceived as such; it is created by subjective interpretation of the real world and mental construction out of a world of experience and imaginations [23]. Regarding a process as a system means already creating a mental model of the original process. An original process does not necessarily have to be a “real” process that has occurred in the past or is observed in the present, but it can also be a potential solution of how things could be done in the future. Thus, we define a *process model* as a mental or explicit representation of original processes.

In general, directed graphs are used for the explicit representation of processes. When we speak of process models in this paper, we mean semi-formal, computational representations in symbolic notation, i.e. general process elements like activities and their relations are represented by formal symbols (boxes and vectors) and additional information are attached non-formally, e.g. naming the symbols in natural language.

Practically all process representation techniques use the notion of decomposition [16]: process models can be decomposed into different sub-processes, which again can be made up of other sub-processes. Sub-processes on the lowest level of decomposition are called activities. The decomposition of processes results in an aggregation hierarchy.

Every formal process representation technique (e.g. SADT-diagrams, Petri-nets, event driven process chains etc.) is based on a specific modelling method and a meta-model, which provides guidelines for the construction of process models. Depending on the intended goal of the representation form, other elements of interest besides activities and logical dependencies can be represented in a process model. Those most frequently mentioned are [2]:

- *Agent* – an actor (human or machine) who performs an activity;
- *Role* – a coherent set of activities assigned to an agent as a unit of functional responsibility
- *Artefact* – the output of an activity.

In complex system development processes, most of the output is manifested in documents. As a matter of fact, documents constitute processes. Therefore, in our

approach, we take into account the representation of documents as an artefact created or modified by the enactment of activities.

2.3 Use of Process Models

Process models are created for a specific purpose. Essentially, process modelling is the first step to come up with a formalisation of processes, which serves as the basic platform to build upon [15]. FRICKE ET AL. put forward several aspects why modelling development processes is necessary and worth the effort [7]. Process modelling objectives can be classified into five basic categories [2]:

- *Facilitate human understanding and communication* – process transparency helps people (including the customer) to communicate on the work to be done and to understand what part they play in the game. Process models may serve as a prerequisite for audits and as an excellent learning aid for employees [18].
- *Support process improvement* – process improvements are based on process model assessments, be it in terms of formal reasoning (e.g. static or dynamic analyses) or visual assessment (cf. [19]).
- *Support process management* – a process model provides a sound basis for detailed planning and easier monitoring, measurement and co-ordination of an actual development project.
- *Automate process guidance* – The documentation of process structures enables the capturing and later reuse of process know-how. The objective is to provide “guidance, suggestions, and reference material to facilitate human performance of the intended process” [2].
- *Automate process execution support* – This aims at the derivation of data structures for the development or adaptation of information systems supporting the enactment of processes. Parts of the process can be automated, e.g. by distributing and supplying information and documents electronically with the help of a workflow management system [11].

In the following, we introduce an approach which basically goes along the line of automated process guidance.

3 Approach

Process management constitutes a prime competence to value generation. Process models capture know-how about ways of working in the past or intended in the future. However, a static process model contains no information about *why* work had or has to be done in a certain way. Prevailing process models and supporting tools lack in the adaptation of actual processes. That is, there are sophisticated design, analysis, and management methods and tools, but methods for customising processes in the design and even execution phase are sparse. With our approach, we aim at creating generic, reusable representations of process construction knowledge in terms of an ontology (cf. [6,8]) that can be applied across a variety of future process cases by means of a process construction kit.

3.1 Conceptual Framework

3.1.1 Process Individualisation

Due to the non-repetitive project character of complex system engineering processes, process planning can be performed at the modelling level of process instances. These process instances must be individually tailored according to the current project-specific context (requirements and constraints), in order to serve as a useful basis for process execution and automation at an operational level [22]. We call the process of adapting a process model according to a specific context *process individualisation*. It is necessary when generating new process instances at the beginning of a project as well as during project run-time, when constraints and requirements change.

In general, instantiation and individualisation refer to two different dimensions of process modelling (figure 1). In the sense of object-oriented modelling, process models can be on different levels of information modelling, i.e. a meta model gives guidelines for creating a model on the type level, which can be instantiated for execution (cf. [12]). But this instantiation does not necessarily include the adaptation of the process model to a specific context. That is why the dimension of individualising process models can be regarded as orthogonal to the levels of information modelling. A meta model is always meant to give guidelines for many cases; thus, it does not make sense to develop a project-specific meta model.

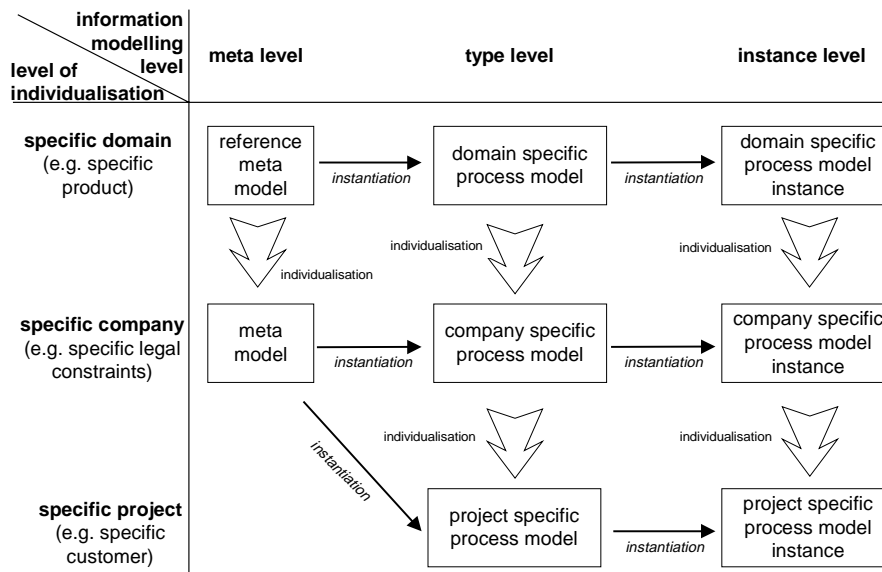


Fig. 1. Levels of modelling and individualisation

3.1.2 Process Construction Kit

In our approach, the specific *context* of a process model is described explicitly by a model of constraints and requirements. Together with the corresponding individual process instance, this model forms a *process case* (see figure 2). These process cases can be reused as best-practice patterns for other cases by copying them.

The likelihood of a process case “fitting” a hundred percent on another project-specific context without any manual adaptations is rather low. Therefore, *reference process models* are needed on a generic level, i. e. they are automatically adaptable to a specific context via defined operators. For example, reference process models (cf. [20]) can refer to a certain product type or a certain procedure (like ISO 9000 certification). New process instances can be generated by automated individualisation.

For complex and innovative engineering processes, it is not sufficient to document a best-practice process once and follow this pattern for ever in all future process cases. Just as little, an overall generic reference process can be modelled that can be adapted to all future process cases. For reasons of higher reusability and flexible connectivity, an extensive process model can be stripped down to temporally and logically isolated units called *process building blocks*. Such building blocks should have only few interfaces to other process models or building blocks, and they should “know” how they may be connected to other building blocks. Similar to reference process models, process building blocks should be generic, in order to allow an automated individualisation and configuration of a new project-specific process instance. Generic reference process models comprise generic building blocks and are augmented with additional, non-generic process structures.

Constraints and requirements have influence on the design of processes. They can be used to describe the context of a process. Values of such constraints and requirements come from a set of discrete values (e.g. a product type or boolean values) or from a continuous interval (e.g. numeric value for the project-specific order value). Constraints and requirements and their relationships are captured at a generic level in terms of an ontology, which provides guidance in modelling an actual context.

Experience in process construction is broken down into single design decisions that are captured in terms of *construction rules*. A construction rule consists of a condition part which refers to constraints and an execution part which triggers construction operators. Construction rules are represented as dependencies among constraints and process building blocks in the ontology. By applying these construction rules to a specific model of constraints and requirements, proposals for adaptations to the process model are generated.

Documents and other resources are connected to the process models in terms of informational dependencies.

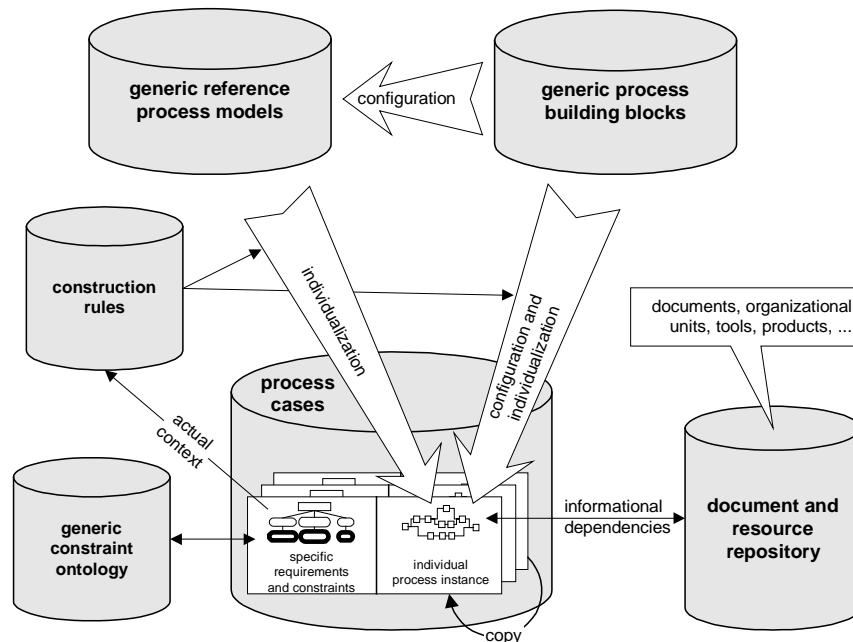


Fig. 2. The Process construction kit

The knowledge base of the process construction kit is extendable, that means, users can define new constraints, requirements, reference process models, process building blocks, and construction rules at any time on a generic level and add them to the ontology. By collecting and maintaining such knowledge from different engineers, process modelling experience (i.e. the reasons for process design decisions in a specific context) can be captured and reused for future process modelling cases.

3.1.3 Example for a Process Individualisation

This approach allows one to tailor process models in accordance with project-specific requirements, as it is particularly relevant for processes in the domain of complex system engineering [18]. Figure 3 demonstrates the basic principle of advisor-based guidance with an example for the project-specific adaptation of a plant construction process model. It shows how an advisor derives the suggestion to instantiate, individualise and insert a generic process building block (“*apply for export permission*”) before another process building block (“*decide: bid or no bid*”) due to a specific legal constraint (“*export certification: necessary*”). In figure 3, dashed arrows represent dependencies and solid arrows represent guided system operations.

Amid process modelling, an engineer selects constraints from the constraint ontology. Each selection or later change in a constraint may trigger an advisor, which highlights the parts of the process where potential alterations arise due to the change, generates advice on how to conduct the alterations, and provides a textual explanation.

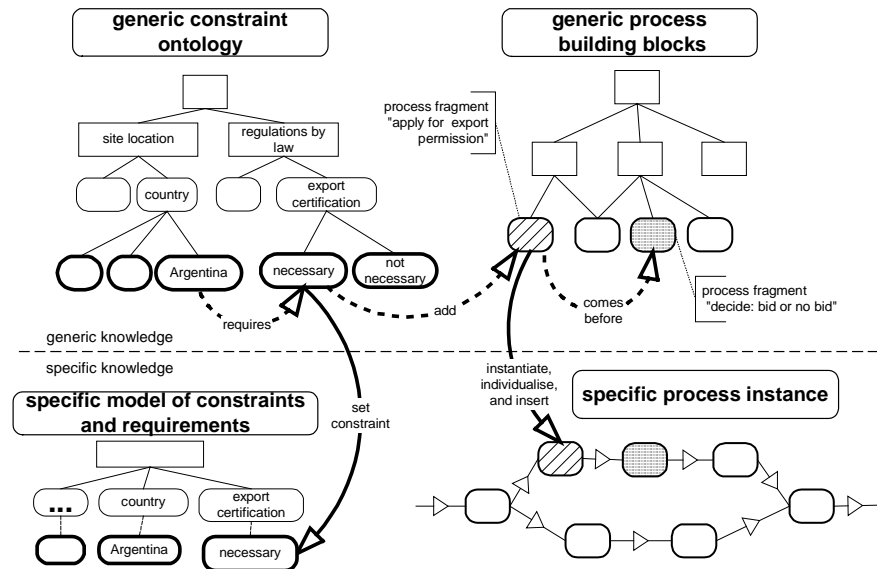


Fig. 3. Basic principle of a process model adaptation

3.2 Technical Approach

3.2.1 Basic Concepts

Technically, we understand the construction of processes as the application of specific operators on a process model to transform that model to an adequate representation of a specific process. From this point of view, processes and activities are objects with certain attributes (e.g. title, status and duration) [1] and associated operators. The set of operators includes operators for instantiation, adaptation and configuration. These operators are context-sensitive, i.e. an operator takes the specific constraints of a process case into account. Objects, operators and constraints together form a network of interrelated concepts.

Operators may either be triggered by user interaction (e.g. a user inserts an activity into a process by applying the `AddActivityOperator`) or by the system itself, responding to a change in the context of a process. In the latter case, the system acts as an agent by proposing an adaptation of the process model, which reflects the context altered. Some of the process adaptation steps may require additional input (e.g. users responsible for an activity), which cannot be solely provided by the system. In this case, the system provides assistance to gather the necessary data in a formalised dialog with the user.

By associating applied operator instances with affected object instance, it is possible to follow the history of changes carried out during the construction of a process case. Moreover, associating operator instances with constraint instances that lead to their application, it is also possible to record *why* a change has occurred. In this way, the process of constructing an explicit process model is recorded in the form

of a history of changes made to a process instance and the reasons that lead to the change.

Thoroughly analysing the history of process cases – be it by human effort or computer support - may lead to the identification of new or more efficient construction rules. Dependencies between constraints, operators, and process objects are stored on a generic level in an ontology. The structure of generic reference process models and process building blocks is recorded in a construction plan, which is made up of a collection of operators. Operators in turn may also trigger other operators. By recursively traversing and applying the network of operators that describes generic process models one ends up at basic operators like `createActivity` or `addActivityToProcess`.

By these means, the system is capable of applying the explicit knowledge about process construction in terms of operators and to generate proposals for the adaptation of the process model, which are suitable for a current situation. Process cases, the generic process models, constraints and operators together form a knowledge base, which is used for both the construction of new cases and the derivation of new generic process models, constraints and operators. In the following, we focus on applying constraints and operators for the construction of process models.

3.2.2 Design and Implementation of an Ontology for Process Construction

Knowledge based systems, such as the process construction kit, should contain an explicit model of the underlying knowledge, so that the system's behaviour can be visualised, analysed and edited on a high level of abstraction. The notion of ontologies is fundamental in this context. An *ontology* [24] is an explicit specification of concepts and the relationships among them. Ontologies can be either represented by a network of interrelated terms, or - as demonstrated in [14] - as a hierarchy of object oriented classes and instances. In the latter case, the classes define the types of entities in the knowledge base, whereas the instances represent specific entities.

We have designed a class-based ontology for process construction. The basic principles of this ontology can be compared with an event-driven, learning neural network. The ontology consists of nodes which are connected by directed edges, forming a network of interacting concepts. Nodes either represent process model objects (such as activities, documents and their relations), operators or evaluations. Evaluable nodes have a state of a given type, such as boolean, discrete, string or numeric. Constraints are represented in the states of these nodes.

The state of the network is defined by the initial state of its nodes and the sequence of incoming events, such as user actions or state changes. Whenever a node changes, it notifies its predecessors which then in turn may change their state. Thus, events may propagate through the net, finally reaching condition and operator nodes. If an operator node is reached, the associated action is being applied in the given context. Operators might instantiate new processes or building blocks, update attributes of a given activity, or insert a building block into a process model instance.

An excerpt of the ontology's class hierarchy is presented in figure 4. All classes are derived from the abstract base class `Node`, which provides services for the event flow between adjacent nodes. The abstract class `ProcessObject` describes entities from the process model. `Activities` are basic `ProcessObjects`, representing executable real-world process steps. `Processes` can have a number of child `ProcessObjects`. Instances of the classes `BooleanNode`, `DiscreteNode`,

StringNode and NumericNode can be evaluated to a value of their respective type. They either deliver constant values (as in the DiscreteValue class), variable values (as in the DiscreteVariable class) or derived values (as in the Equality class, which is linked to two EvaluableNodes and evaluates to true if the values of these adjacent nodes are equal). Boolean nodes, such as instances of the Equality class, can trigger Operator nodes, such as the AddActivityOperator, which constructs a new Activity with given properties. Operators can also be nested to complex operator sequences. Thus, complex process models can be generated and modified. The applied Operators are stored together with the ProcessObjects, serving as a backward reference, which can be used to analyse the development process leading to a process model.

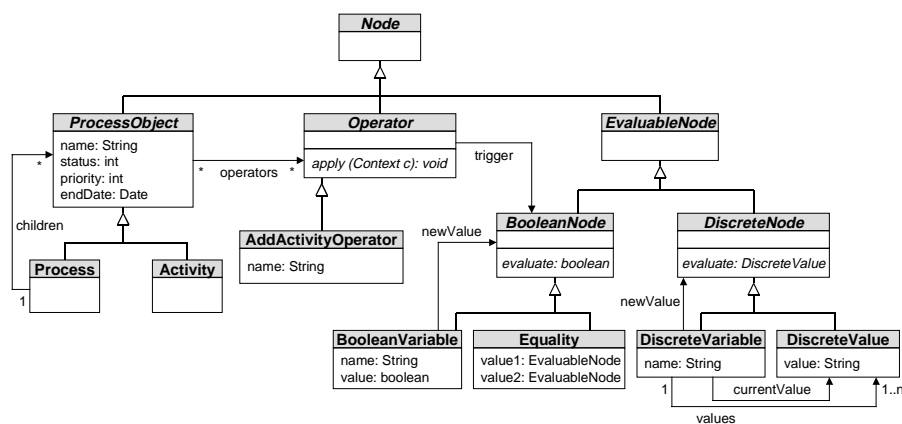


Fig. 4. A simplified excerpt of the classes implementing the ontology for process construction (in UML notation)

An ontology built from these simple types of nodes represents the generic process models, the knowledge about constructing a process, and the current process case. At the same time, the ontology is executable and can be used to create or modify an existing process model. To do so, the concepts (or classes) from the ontology are linked to object-oriented methods which can be applied to them. Using the framework for implementing ontologies presented in [14], it is possible to attach methods to ontologies without sacrificing the abstract high-level view of the ontology. This implementation technique is based on object-oriented *reflection*, i.e. the ability to analyse the structure of a class hierarchy and the state of an object network at run time. This allows one to display and modify the ontology with generic tools, e.g. to visualise the ontology's reasoning processes at run time.

The knowledge encoded in the ontology can also be used by various problem solving engines, such as the advisor that generates proposals for possible process adaptations. Such engines can traverse the network of adjacent concepts in order to identify related entities. Furthermore, the ontology implicitly contains the generic construction rules, which can be extracted and presented in a user-readable way. These construction rules can be generated automatically by learning from the operators applied in the given contexts.

3.2.3 Example Scenario

Figure 5 presents an excerpt of the ontology instances for the example presented in section 3.1.3. In this ontology, a given process has an attribute “DestinationCountry”, which is modelled in a namesake `DiscreteVariable`. This variable can take one of its possible values, as shown below. From an abstract point of view, the knowledge encoded in this ontology specifies that the `AddActivityOperator` for the “Apply for export permission” activity shall be executed if the destination country variable is “Argentina”.

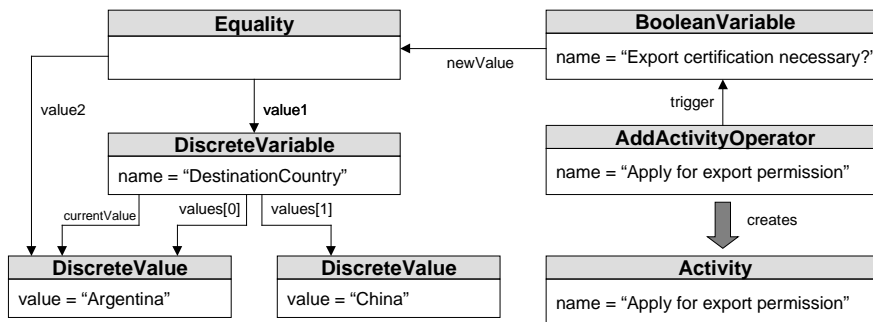


Fig. 5. A small ontology representing knowledge about when an application for export permission is necessary and what to do in this case

A sample use case of this ontology is shown in figure 6. The scenario starts with the user creating a new process on the GUI. The GUI creates a new `Process` instance, displays it on the screen and selects it. This will set the new `Process` to be the GUI’s current editing context. Then the user specifies the destination country of the process to be “Argentina”. The GUI assigns this new value to the `DestinationCountry` variable. This variable notifies its predecessors in the ontology network about its state change. Thus, the `Equality` is being evaluated and notifies in turn its predecessor node, i.e. the `ExportPermission` variable. This finally triggers the `AddActivityOperator`, which creates a new `Activity` and uses the GUI context to add it to the selected process.

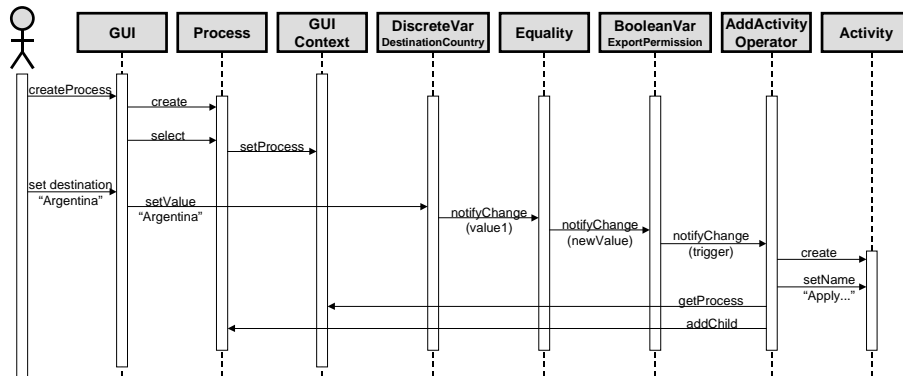


Fig. 6. A simple scenario based on the ontology from figure 5

4 Related Work

Most of the commercially available process modelling tools use non-generic methods for the process description. Non-generic reference process models can only serve as static templates that need to be adapted to specific requirements manually.

However, some approaches in research are related to our work:

With the *ARIS Process Generator*, an approach for automated customising of process models to a specific company has been developed and implemented in a prototype [9]. The context is described by answering pre-defined, semi-structured interviews. Such a tool is meant to be used by third party champions like organisation consultants for the configuration of standard software. However, there are no concepts for expanding and maintaining knowledge about process modelling, as it is necessary for capturing experience from the process participants.

ProTail is a tool-prototype for automated adaptation of the V-model, a standard for software development processes [17]. Here, a process model is individualised according to specific project constraints. However, the description of the process context is limited to the pre-defined constraints of the V-model. Mutual dependencies between the constraints are not considered.

The aim of *CoMo-Kit* is to capture and reuse knowledge about the execution of processes by recording the selection of process variants during run-time [4]. But the approach does not support the description of a project-specific context by constraints and the automated derivation of process model adaptations.

The *Handbook of Organisational Processes* is meant to serve as a handbook, which users can browse for alternative process models at various levels of specialisation and decomposition. The approach “allows people to explicitly represent the similarities (and differences) among related processes and to easily find or generate sensible alternatives for how a given process could be performed” [16]. Alternatives can be compared only within a bundle of related alternatives by trade-off matrices on multiple dimensions. The specialisation of a generic process model is

performed according to defined goals and the type of organisation. This approach is not suitable for the automated adaptation of a process model to an individual, project-specific context.

With the approach of *WEGA*, process modelling know-how is captured in generic design patterns and reused for the construction of process models [5]. The approach is based on the notion of software reuse by design patterns. A pattern manager supports the retrieval and application of appropriate design patterns. The design patterns can be adapted to a specific situation, but the adaptation is limited to a manual selection of pre-defined parts.

5 Conclusion

We have presented an approach for project-specific individualisation of engineering processes. Know-how about engineering processes is a crucial ingredient for a successful operation of any virtual organisation. Large companies in fact already “live” the concept of virtual organisations in their internal operations. Our approach is founded in the assumption that engineering experts will capture and maintain their know-how about engineering processes. Initial pilot applications have proven this approach, i.e. to move process modelling tasks from consulting experts towards engineering experts and process engineers.

The process construction kit is operational in that it allows the formal representation of processes and operators at an abstract level while compiling modifications to the implementational level for execution. Knowledge about process design is represented in terms of operator sequences related to constraints. Each constraint is represented as a Boolean node.

We have developed a dedicated ontology editor that allows one to specify and modify process models at a generic as well as instance level [14]. Currently, we are designing an ontology for classifying engineering processes and refining operators for individualisation in project-specific contexts. Particular emphasis will be put on a domain-specific ontology for the support of simultaneous engineering teams in the automotive sector. Main intention is to support the co-ordination of process tasks in a network of engineers, the identification of critical process structures and states, and the advice about options for process amendments. In order to do so, the ontology will be founded upon basic patterns of construction processes derived from empirical studies of automotive development processes.

In general, the question arises of how to learn process knowledge, be it success stories or lessons learned in worst cases. Co-operation with engineering experts is one side of the coin for capturing experience about the construction of engineering processes. In an enterprise-wide perspective, communities of practice might even serve as organisational means to additionally extract and disseminate best practice and lessons learned from worst cases. Yet, automatic mining or case based reasoning might be another solution. We will explore metrics to assess processes based on various reference numbers for quality, goal compliance, etc.

So far, the process modelling component of our system has been tested at user sites for the capture of processes, in particular emphasising the notion of graphical process presentation for visual assessment [21]. The assessment of processes draws upon the

duality of formal reasoning and mental perception [19]. Advisors care about specific features of processes that are based upon formal theories. Visualisation techniques are employed to support process engineers in cases where formal theory about process characteristics are missing. That is, automated layout algorithms are used to generate different perspectives on processes.

Moreover, we call for intuitive interaction means that organises processes in spatial regions and allows engineers to walk along process chains and navigate to neighbouring regions, be it a neighbourhood with regard to time, process or information dependencies.

Acknowledgements

Part of this work is being supported by the German Federal Ministry of Education and Research (BMB+F) as part of INVITE (Research Grant No. 01 IL 901 B 4). For further information on INVITE: <http://www.invite.de>.

References

1. Bider, I., Khomyakov, M.: Business Process Modeling: Motivation, Requirements, Implementation. In: Demeyer, S., Bosch, J. (eds.): Object-Oriented Technology: ECOOP'98 Workshop Reader, LNCS 1543, Springer (1997) 217-218
2. Curtis, B., Kellner, M.I., Over, J.: Process Modeling. In: Communications of the ACM 35 (1992) 9, 75-90
3. Davenport, T.H.: Process Innovation: Reengineering Work through Information Technology, Harvard Business School Press, Boston (1993)
4. Dellen, B., Maurer, F., Pews, G.: Knowledge Based Techniques to Increase the Flexibility of Workflow Management. In: Data & Knowledge Engineering Journal, North Holland (1997)
5. Ferstl, O.K., Hammel, C., Pfister, A., Popp, K., Schlitt, M., Sinz, E.J., Wolf, S.: Verbundprojekt WEGA: Wiederverwendbare und Erweiterbare Geschäftsprozess- und Anwendungssystem-Architekturen. In: Statusband des BMBF Softwaretechnologie, Berlin (1998) 3-21
6. Fox, M.S., Gruninger, M.: Ontologies for Enterprise Integration. In: Proc. 2nd Conf. on Cooperative Information Systems, Toronto, Ontario (1994)
7. Fricke, E., Negele, H., Schrepfer, L., Dick, A., Gebhard, B., Härtle, N.: Modeling of Concurrent Engineering Processes for Integrated Systems Development. In: Proc. 17th Digital Avionics Systems Conf.: Electronic in Motion, Bellevue, WA (1998)
8. Gruninger, M., Fox, M.S.: The Role of Competency Questions in Enterprise Engineering. In: Proc. IFIP WG5.7 Workshop on Benchmarking - Theory and Practice, Trondheim, Norway (1994)
9. Hagemeyer, J., Rolles, R., Scheer, A.-W.: Der Schnelle Weg zum Sollkonzept: Modellgestützte Standardsoftwareeinführung mit dem ARIS Process Generator. In: Scheer, A.-W. (ed.): Publications of the Institut für Wirtschaftsinformatik at the University of Saarbrücken, Germany, Report No. 152, Saarbrücken (1999)
10. Hammer, M., Champy, J.: Reengineering the Corporation – a Manifesto for Business Revolution, Harper Collins Publishers, New York (1993)

11. Jablonski, S.: Process Modelling and Execution in Workflow Management Systems. In: Proc. Int. Working Conf. on Dynamic Modelling and Information Systems, Nordwijkerhout, Netherlands (1994)
12. Jennings, N.R., Norman, T.J., Faratin, P., O'Brian, P., Odgers, B.: Autonomous Agents for Business Process Management, Int. Journal of Applied Artificial Intelligence (to appear) (2000). <http://www.elec.qmw.ac.uk/dai/pubs/>
13. Johansson, H.J., McHugh, P., Pendlebury, A.J., Wheeler III, W.A.: Business Process Reengineering: Breakpoint Strategies for Market Dominance, Chichester (1993)
14. Knublauch, H., Rose, T.: Reflection-enabled Rapid Prototyping of Knowledge-based Systems. In: Proc. OOPSLA'99 Workshop on Object-oriented Reflection and Software Engineering, Denver, CO (1999)
15. Kratz, N., Rose, T.: Modelling and Analyzing Processes in Production and Administration. In: Tzafestas, S.G. (ed.): Management and Control in Manufacturing Systems, Springer, Berlin (1997) 118-142
16. Malone, T.W., Crowston, K., Lee, J., Pentland, B., Dellarocas, C., Wyner, G., Quimby, J., Osborne, C., Bernstein, A.: Tools for Inventing Organisations: Towards a Handbook of Organizational Processes. Technical Report of the Center for Coordination Science, Massachusetts Institute of Technology, Cambridge (1997)
17. Münch, J., Schmitz, M., Verlage, M.: Tailoring Großer Prozessmodelle auf der Basis von MVP-L. In: Montenegro, S., Kneuper, R., Müller-Luschnat, G. (eds.): Vorgehensmodelle – Einführung, Betrieblicher Einsatz, Werkzeug-Unterstützung und Migration: Papers of the 4th Workshop, GMD-Report No. 311, Berlin-Adlershof, Germany (1997) 63-72
18. Negele, H., Fricke, E., Schrepfer, L., Härtle, N.: Modeling of Integrated Product Development Processes. In: Proc. 9th Annual Int. Symposium of INCOSE Systems Engineering: Sharing the Future, Brighton, UK (1999)
19. Peter, G., Rose, T., Rupprecht, C.: Towards Reducing the Complexity of Process Modeling by Advisors, Explicit Context Modeling, and Visualisation Techniques. In: Proc. 10th Mini EURO Conf. on Human Centered Processes (HCP '99), Brest, France (1999) 315-320
20. Remme, M., Allweyer T., Scheer A.-W.: Implementing Organizational Structures in Process Industry Supported by Tool-Based Reference Models. In: Proc. Conf. on Computer Integrated Manufacturing in Process Industries CIMPRO, East Brunswick, New Jersey, USA (1994) 233-247
21. Rose, T.: Visual Assessment of Engineering Processes in Virtual Enterprises. In: Communications of the ACM 41 (1999) 12 45-52
22. Rupprecht, C., Peter, G., Rose, T.: Ein Modellgestützter Ansatz zur kontext-spezifischen Individualisierung von Prozessmodellen. In: Wirtschaftsinformatik 41 (1999) 3 226-236. An earlier version of this article appeared in: Scheer, A.-W., Nüttgens, M. (eds.): Electronic Business Engineering – Proc. 4th Int. Conf. on Business Informatics in Saarbrücken, Germany, Physica, Heidelberg (1999) 353-373
23. Schütte, R.: Grundsätze Ordnungsmäßiger Referenzmodellierung: Konstruktion Konfigurations- und Anpassungsorientierter Modelle. Gabler, Wiesbaden (1998)
24. Studer, R., Fensel, D., Decker, S., Benjamins, V.R.: Knowledge Engineering: Survey and Future Directions. In: Proc. 5th German Conf. on Knowledge-based Systems, Würzburg, Germany (1999)