Flexible Workflows for an Energy-Oriented Product Development Process[1]

Thomas Reichel, Gudula Rünger, Daniel Steger

Department of Computer Science
Chemnitz University of Technology, Germany
{thomr,ruenger,stda}@cs.tu-chemnitz.de

Abstract: Product development processes are flexible and dynamic in nature. In this domain, workflows are commonly used for representing business processes in change management or for document management. The product development process itself can hardly be represented by a business process that is defined in a design phase and fixed during runtime. However, the workflow technique can be extended by concepts for flexible workflows. Many workflow management systems now provide solutions for flexible process changes. In this article, hierarchical workflows and dynamically selectable sub-workflows are introduced to enable a flexible workflow execution on a standard workflow management system. As proof of concept flexible workflows are applied to an energy-oriented product development process.

1 Introduction

Product development processes are creative and highly-dynamic processes controlled by the design engineers. In contrast, business processes have traditionally been seen as well-structured and fixed. However, the business world changes this view and the optimization of processes as well as quick adaptation to customers' needs become requirements. Workflows that are used to represent such business processes need to allow flexible modeling. The workflow management system (the configuration and executing software system of workflows) has to provide capabilities to evolve workflow definitions as well as to apply ad-hoc changes to workflow instances. In this article, we investigate how these concepts of workflows for business processes can be applied to product development processes.

Research (e.g., [HS08]) has shown that it is impractical to model the whole product development process as a single workflow. Instead, product development utilizes concepts like product data management to organize product related data and documents. Workflows are embedded to organize common activities for document management and

[1] The Cluster of Excellence "Energy-Efficient Product and Process Innovation in Production Engineering" (eniPROD ©) is funded by the European Union (European Regional Development Fund) and the Free State of Saxony.

to track product changes or configurations. These processes are very similar to business processes of other domains. The actual product development can be seen as mental processes of the engineers. Design theories and methodologies try to capture, structure, and document these processes, for example in the VDI-guideline 2221 ([VDI93]), the cognitive science based approach by [Ger90], or its extension in [Col07]. For this article, the property-driven design by [WD03] is exploited. Instead of a predefined business process, product development is described by repeated actions to analyze the product requirements, to synthesize a new product design, and to verify the expected product functionality. Each step requires engineers to apply methods of different engineering domains, mainly based on their expertise and with certain design goals in mind. Usually, these methods involve computer-aided tools to calculate, optimize, or simulate technical and economic product data.

In recent years the design goal "energy-efficiency" for products has become more important. The need to adapt the product development process to energy-efficiency, as proposed by [NFP+09], can be supported by combining the property-driven design approach with explicit, flexible workflows to express the product development process. An obvious advantage of this approach is the possibility of documentation and visualization of the development process. The process is reproducible and overall strategies for product development can be applied. The design goal "energy-efficiency" can be supported by invoking simulation or optimization tools to predict and reduce the energy usage of the final product within the current design context.

This article proposes to model the property-driven design approach by a combination of hierarchical workflows and flexible sub-workflows. This approach utilizes sub-workflows to represent and refine individual analysis, synthesis, and verification activities. To reflect the dynamic flexibility of product development processes, changes can not only be applied during workflow definition, but also while a workflow instance is running. Therefore, recent efforts to systematize flexible workflows, e.g., by change patterns [WRR08] or flexibility patterns [MvdAR08], are considered. Especially, the concept to dynamically select sub-workflows during workflow execution is promising for an implementation of the property-driven design. A new dynamic-selection pattern is inferred from existing flexible workflow concepts that supports parallel execution of sub-workflows in the product development process. As a result, requirements for implementing these concepts in a standard workflow management system are discussed.

The article is organized as follows: Section 2 describes a dynamic-selection pattern based on hierarchical workflows and existing flexible workflow concepts. The workflow technique is applied to a product development process for energy-efficient products in Sect. 3. In Sect. 4, the implementation of the workflow extension with a standard workflow management system is discussed. Section 5 considers related work, and Sect. 6 concludes the article.

2 Hierarchical and flexible workflows

In this section, an overview on business processes as well as workflows is given. Hierarchical workflows and the concept of flexibility for workflows are summarized. Also, a dynamic-selection pattern is introduced that forms the basis for adopting the workflow technology to express the property-driven design approach.

2.1 Hierarchical workflows

Business processes focus on processing business objects in a structured way. The "tasks" (work items) are primarily executed by the users. In the business domain, a *business process* describes an organizational process in an enterprise, not an executable program. Thus, the informal business process description has to be transformed into a *workflow* which can be executed by a workflow engine, e.g., defined by the workflow reference model of the Workflow Management Coalition[2]. Each workflow consists of *activities* (or atomic tasks) that can be processed automatically or with human interaction. Designated start and end tasks, and transitions between activities imply an explicit order of the execution sequence. Special control structures are available to model parallel activities (fork nodes), alternative paths (decision nodes), and join activities (join nodes).

For an additional structuring of workflows, tasks can be organized as sub-workflows. These are fragments of a workflow that can be viewed as separate workflows themselves. In the main workflow, the sub-workflow is replaced by a single activity which transfers the workflow execution to the sub-workflow and returns to the main workflow after the sub-workflow has been finished. This hierarchical structuring of workflows has two advantages. First, repeated fragments of the workflow can be replaced by a single sub-workflow for better maintenance. Second, the workflow can be displayed in several levels of detail. A large, complex workflow can be illustrated by an overall view where no details are visible. Sub-workflows describe specific activities or other sub-workflows and therefore add an additional level of detail to the workflow (see Fig. 2). Human readability of workflows is significantly improved by introducing hierarchical workflows, as for example pointed out by [RM08]. As shown in [TRC+08], business activities like approval, decision making, or notification can be expressed as sub-workflows and can be used as building blocks for the workflow definition.

As a result, a workflow definition by means of hierarchical structures can be introduced:

- The workflow consists of a *main workflow* with an explicit start and end node,
- defines n sub-workflow definitions in a partial order,
- contains additional control structures (loops, forks, joins, etc.), and
- shows the workflow with a detail level of 0.

If all sub-workflows contain exactly one activity, the main workflow is completely defined, otherwise the definition of levels given above is applied recursively to the sub-workflow. Each sub-workflow in the hierarchical structure exposes additional details to

[2] http://www.wfmc.org/

the workflow representation. If a calling workflow shows detail level *l*, each new sub-workflow called from this workflow has level *l+1*.

In standard workflow languages, each activity in the main workflow can represent exactly one sub-workflow definition. The workflow patterns by [vdAtHKB03] provide a way to extend this behavior. One of these workflow patterns, called WCP-15 in [vdAtHKB03], creates multiple instances of a task, not knowing the number of instances prior to runtime. Another workflow pattern, WCP-36, specifies the completion condition of the tasks: Either all running tasks have to finish or an explicit completion criterion is defined to trigger the next task in the workflow.

In this article, pattern WCP-15 and WCP-36 are extended to activate multiple, not necessarily different sub-workflow definitions per activity. This means that an *1:m* relationship between activity and sub-workflows can be established by splitting the main workflow into concurrently executing sub-workflow instances. Additionally, multiple instances of the same sub-workflow definition may be instantiated based on runtime conditions. The main activity is completed only if some (partial join) or all sub-workflow instances (join) are finished. Instances not completed are aborted and rolled back.

A major issue when implementing the described pattern is the synchronization of concurrently running sub-workflows. The sub-workflows are executed in form of parallel threads that are not coordinated with each other. They may be all started at different times after the main activity is triggered. The sub-workflows are not ordered in their execution. None of the workflow pattern of [vdAtHKB03] describes the behavior needed for a parallel synchronized execution of sub-workflows. A possible solution can be to add flexibility to workflows. In the next section, the discussed extension for hierarchical structures is combined with flexible workflow constructs to introduce a new dynamic-selection pattern.

2.2 Adding flexibility to hierarchical workflows

Introducing flexibility in workflows can help to enhance the expressiveness of hierarchical workflows by introducing incomplete descriptions or templates that are specified at runtime. Template definitions embody several cases which otherwise have to be modeled explicitly. In hierarchical workflows, a selection of sub-workflows is enabled at runtime when a certain template activity is reached. This behavior is described by the *Late selection pattern (PP1)* in [WRR08] in three different versions: The selection is based on a user decision, predefined rules filter available selections, but the user decides which one is taken, or the predefined rules decide autonomously. The last version is described by the *worklet* concept of [AtHEvdA06].

Worklets describe a dynamic approach to specify hierarchical workflows. A parent process captures an entire workflow. It arranges a number of activities where each activity has a number of worklets attached. Worklets are single sub-workflows kept in a repository. When the parent process is running and a certain activity is enabled, a selection rule automatically chooses an appropriate worklet and integrates it into the parent workflow

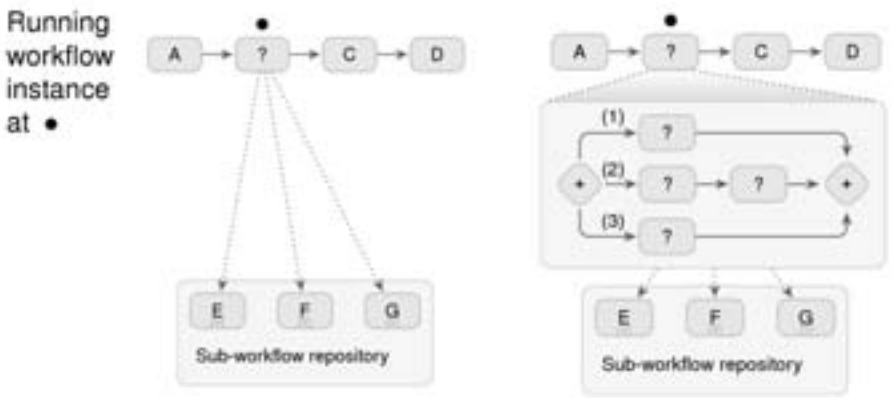

Figure 1: Comparison of late selection of sub-workflows with worklets (left) with an extension where sub-workflows can be arranged at runtime and executed concurrently (right)

enactment. By adding new worklets to the repository changes to workflow definitions can be evolved (see Fig. 1, left).

Worklets allow a dynamic selection of sub-workflows, but neither the coordination of multiple worklets nor the parallel execution is included. These two aspects are addressed in the change patterns of [WRR08]: *Late composition (PP3)* and *Multiple instance activity (PP4)*. With the late composition pattern a given number of sub-workflows can be arranged at runtime. The pattern PP4 supports multiple instances of sub-workflows.

As the existing patterns describe only parts of the needed functionality, a new dynamic selection pattern is introduced. This pattern considers the runtime composition as a new worklet. The user and/or the system chooses a number of sub-workflows that are arranged as a worklet to be called dynamically in the main workflow (see Fig. 1, right (2)). Further flexibility can be achieved by merging late composition with multiple instances (see Fig. 1, right (1) and (3)). This results in a new dynamic-selection pattern that can instantiate multiple worklets and execute them concurrently. As a consequence, parallel execution of dynamically selectable sub-workflows is supported and these sub-workflows can be synchronized with each other.

3 Applying hierarchical and flexible workflows to the product development process

The result of the product development process is a specification to manufacture the product. Domain experts use their domain knowledge of product developing methods and tools to infer product structures from given requirements. A product development process to accomplish this is described in the property-driven design (PDD) approach which uses loops of analysis of the current product behavior, synthesis of new product structures, and verification of the intended product behavior (a special form of analysis). For each design iteration, engineers have to choose the most suitable methods and tools.

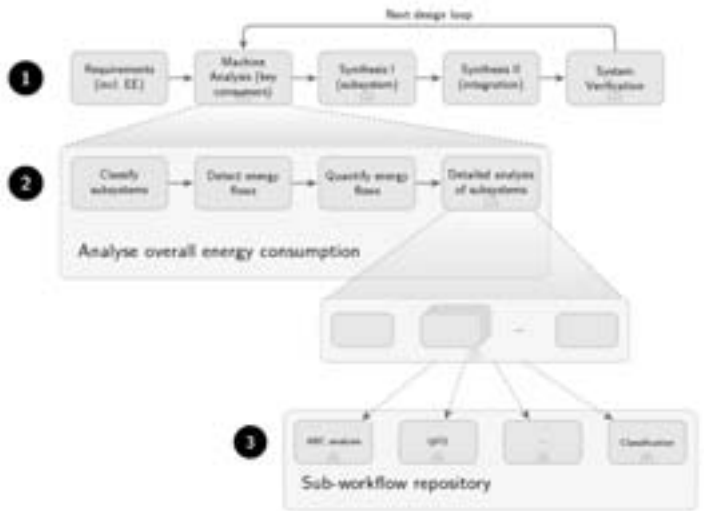

Figure 2: Hierarchical workflow model for an energy-oriented product development process in the PDD approach

Even the number of iterations may vary according to the product's complexity, the type of development, or external conditions. In this section, the principles of hierarchical, flexible workflows are applied to this approach.

A hierarchical workflow with a main workflow and sub-workflows as discussed in Sect. 2 can be exploited to model a product development process with the PDD approach. A main workflow describes the analysis-synthesis-design loop of the PDD approach. Individual methods used by engineers are defined as sub-workflows. The dynamic-selection pattern shown in Fig. 1 (right) allows the engineers to choose the most suitable method.

To further illustrate the approach, a product development process for energy-oriented product designs, described in [NFP+09], is taken as an example. The major design goal for such an energy-oriented development process is to optimize the energy consumption during the product's life cycle, not only for manufacturing, but also for operation and recycling. Though the functionality of the product has to be guaranteed, particular product development methods and tools are incorporated to save energy throughout the product life. Additionally, the costs of the product have to be considered. With hierarchical workflows this process can be modeled as shown in Fig. 2.

The main workflow (1) describes the basic activities according the PDD process: *Analysis*, *Synthesis I+II*, and *Verification*. They are executed in a loop. For energy optimization, individual activities for analysis, synthesis, and verification can be described by sub-workflows. In Fig. 2, the *Machine Analysis* step, which consists of the identification of energy flows in similar products, is shown in detail (2). By finding key energy consumers, the engineers can choose which subsystems of the product should be improved for the new design. For each subsystem, energy flows have to be detected and quantified.

The results are analyzed by engineers and the key consumers are identified. For this analysis, multiple engineering methods need to be made available in form of dynamically selectable sub-workflows (3). The engineers may decide to subdivide energy consumers into classes, e.g., of high energy consumers and low energy consumers, with ABC analysis using the method first applied by [Par71]. Also the engineers can correlate product components with given quality measures by quality function deployment (QFD) [Gra06]. Further classification methods are available. The engineers may use one or more of these analysis methods, either in parallel or one after another. Sub-workflows can be categorized with automatic rules to allow only valid choices of sub-workflows in the current design context (e.g., only analysis sub-workflows).

The following steps in the main workflow (1) in Fig. 2 use the analysis results to synthesize new, energy-optimized subsystems. These steps are organized in similar sub-workflows as shown for the machine analysis. The new design results will be integrated into the product. Finally, the achieved energy-efficiency has to be verified as well as the product's functionality and cost. Results are taken as input for the next design iteration. Although the sequence of actions in the main workflow is fixed, individual sub-workflows differ according to the current development context and the design engineers' needs. Indeed, design engineers may apply various calculation and simulation methods in form of sub-workflows per activity, before they proceed.

In a common product development scenario, multiple engineers work concurrently on different subsystems of the product. Therefore, the corresponding sub-workflows need to be executed concurrently, for example per subsystem or engineer. To coordinate the development activities, PDD provides the *Synthesis II* step, in which modified subsystems are integrated into the entire product. As a consequence, this activity corresponds to a synchronization construct in the workflow.

4 IT support for hierarchical and flexible workflows

This section examines requirements for an IT system to implement the hierarchical and flexible workflows presented in Sect. 2. The solution proposed is based on the JBoss platform[3] that already includes integration components, like the JBoss ESB (enterprise service bus) and the workflow engine jBPM. Utilizing existing components, a service-oriented IT system is constructed that integrates hierarchical workflows for product development according to the PDD approach.

4.1 A workflow-based software architecture

The hierarchical, flexible workflows described can be implemented with a service-oriented approach. The basic idea of services is to modularize functionalities and make them available in a uniform way. When applying this concept to workflows, services with nested workflows can be expressed. If workflows contain activities that realize

[3] http://www.jboss.org

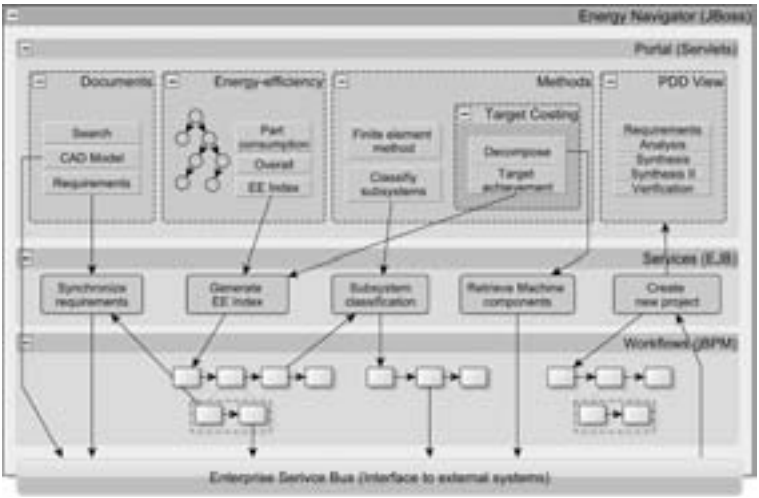

Figure 3: IT System for implementing an energy-oriented product development process

service calls, a hierarchical workflow is formed. This concept is introduced by [vAtH+09] and named by the term *flexibility as a service*. In their implementation, flexible workflow technology, like worklets, is embedded in services. The YAWL workflow engine [vt03] acts as an intermediate layer to orchestrate the services, and is extended with an interface to the flexible workflow services.

The implementation concept described in this article uses the *flexibility as a service*-approach, but instead of extending the workflow engine, a workflow service integration interface is provided by integrating an enterprise service bus (ESB). This software component provides necessary means for a service-oriented IT system and a three-layered software architecture is created as shown in Fig. 3.

The architecture consists of a base layer implementing the workflow technology with the jBPM workflow engine (*Workflows*). The workflows describe human activities and system activities to guide the execution of the product development process. The next layer is a service layer that can nest the workflows and provides a uniform interface to external systems (*Services*). An *Enterprise Service Bus* (ESB) is incorporated as an integration component. In the energy-oriented product development, relevant data needs to be collected from various IT systems and should be made available to the engineers. In the machine analysis, for example, this data may consist of experimental measurements or estimations about energy consumptions during product manufacturing or operation. An ESB provides a wide range of standardized interfaces to connect the workflow management system to external systems storing this data (e.g., an enterprise resource planning system). The message-oriented communication in combination with the jBPM engine (see [JBo10]) can be used to call a service that starts the sub-workflow and continues the main workflow until a synchronization point is reached. Abilities to dynamically route and transform messages allow a simple integration of additional IT systems, for example an enterprise resource planning system. In summary, external calls are made available as services to create a uniform service layer for workflow calls and system calls.

A user layer is the third layer in the software architecture (*Portal*). It enables the engineer to interact with the system. The user can view the current development process and the tasks assigned. The user interface allows the engineer to select and start the sub-workflows representing valid methods in the current design context. Results of analysis and synthesis activities can be accessed in form of documents or in an energy-efficiency view. This view shows collected and aggregated energy data to provide the engineers with input for their development efforts and energy comparisons between different designs.

4.2 Flexible sub-workflow selection

The implementation concept proposed in Sect. 4.1 is realized in a standard workflow management system utilizing the workflow constructs provided by JPDL[4] and custom Java extensions supplied by the workflow engine jBPM.

The dynamic-selection pattern can be subdivided into several components. First, sub-workflows in form of services need to be queried from the service repository of the ESB. Then, a rule-based action will automatically preselect sub-workflows according to the current development step, like analysis or synthesis, or other criteria. Next, the filtered list of services is presented to the user, who has now the task to identify which one of the sub-workflows should be called. For the assignment, the user role needs to be taken into account, and only sub-workflows corresponding to this role should be presented. After the user has selected a sub-workflow, the sub-workflow is called asynchronously by splitting up the workflow execution into two paths. One path activates and registers the sub-workflow in a list of running sub-workflows. On the other path, a loop-back returns to the selection of available sub-workflows and a next sub-workflow can be started. Multiple activities can run in parallel until the user decides to continue to the next step in the main workflow, instead of selecting another sub-workflow. All running sub-workflows have to be finished or cancelled before continuing. To achieve this, the workflow execution is blocked until an event listener receives the signal of an empty list of running sub-workflows.

Exception handling for the dynamic-selection pattern needs to be addressed. In all cases, the user should be able to continue the main workflow. Hence, exception handling in the running sub-workflows should include a loop-back to the selection of available sub-workflows.

5 Related Work

There are two lines of related work. On the one hand, research concentrates on IT support for product development processes. The main purposes of this work are the integration of CAx systems and the management of product data. On the other hand, workflow management systems for general business processes have been developed with the claim

[4] http://www.jboss.org/jbpm

to be applicable to different domains. Such systems exploit quite different approaches to include flexible workflows.

In SIMNET, a workflow management system for engineering changes in a collaborative environment was developed [YGH04]. Workflows are used to coordinate communication between distributed teams or companies. The project concentrates on a distributed, evolving product model; concepts for flexible workflows are not discussed.

FORFLOW is a research program [JFJ+08] that focuses on cooperative processes in product development. It utilizes process management concepts and product development methodologies to model product development processes in general. The contribution is a workflow management system which provides flexibility through late composition of activities as described in the change pattern PP3 of [WRR08]. Given a number of activities with a partial order and an optimization strategy, a recommended execution order of the activities is calculated. Flexibility is addressed by allowing changes in the execution order, but not by introducing constructs to dynamically change the current workflow execution or adding additional activities at runtime.

Enterprise models are IT representations of processes, information, and resources in a company. An example for an enterprise modeling language is CIMOSA [BV99]. It provides behavioral rules (AND, OR and XOR) to compose sets of tasks. The OR construct allows the execution of one, several, or all tasks of a given set of tasks. In contrast to the approach shown, the composed tasks in CIMOSA have to be available at design time and are not provided via a repository that allows changes at runtime. Additionally, each task in the OR construct is executed at least once; a repeated task execution is not part of the construct.

The DECLARE framework [vPS09] provides a different approach to workflow definition than standard workflow management. Instead of defining a strict order of activities by control structures, constraints between activities are assigned. Hence, any order of activities that will not violate the constraints is allowed during workflow enactment. Product based workflows [VRv08] combine the declarative workflow concept with a tight integration of processes and data. It is an example for a data-driven process concept described also in other approaches, e.g. [MHHR06]. An executable workflow is automatically inferred from the product data model, related operations, and a design strategy, e.g., for cost or quality. Though the approach is based on a specific product data model, it cannot be used for a product development process starting with an empty product model.

YAWL is an expressive workflow language [vt03]. Worklets extend the YAWL workflow management system to support flexible sub-workflow selection by automatic rules [AtHEvdA06]. Though the principle of worklets is rather similar to the approach described in this article, the rule-based approach is not sufficient to express the dynamic choices in product development and to assist the engineers when making design decisions.

Another workflow management system for dynamic processes is ADEPT2 [RDR+09]. The system allows ad-hoc changes to running workflow instances and process schema

evolution that can be propagated to workflow instances. Approaches are taken towards detection and prevention of deadlocks or errors during process changes. In ADEPT2 changes are applied through a separate workflow editor. The user is not actively sculpting the process herself or himself, as it is the case with dynamically selectable sub-workflows.

The described approaches contribute to model different aspects of the product development process with workflows, but none of them can be fully applied to the property-driven design methodology. In particular, workflow constructs that express the analysis-synthesis loops of PDD are necessary. Consequently, a new approach based on a combination of hierarchical structures with flexible workflows is chosen in the presented approach.

6 Conclusion

In this article, an approach for applying flexible workflow technologies to complex product development processes is presented. Using the property-driven design methodology for product development enables the modeling of the process with hierarchical workflow structures and dynamically selectable sub-workflows. When designing energy-efficient products such a systematic approach is required. Most of the product's energy consumption, e.g., in operation, is a consequence of design decisions made during product development. Hence, an explicit model of the product development process is necessary to which energy optimization strategies can be applied.

Special attention is put on flexible workflow techniques which are provided by the concept of a new dynamic-selection pattern. This pattern is meant to be implemented on a standard workflow system. The selection pattern supports a semi-automated sub-workflow selection, which is pre-filtered by the system, and allows the user to select sub-workflows according the current development activity. A key concept for this approach is to provide workflows as services.

The contribution of this article is an extension of existing flexible workflow concepts by a combination of hierarchical structures with flexible selection. The discussed pattern is applied to the product development process to express the dynamic and parallel activities when developing new products. In conclusion, the approach is exploited to implement an energy-oriented development process in an IT system that utilizes a service-oriented architecture.

References

[AtHEvdA06] M. Adams, A. ter Hofstede, D. Edmond, and W. van der Aalst. Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In *On the Move to Meaningful Internet Systems 2006*, pp.291–308, Springer, 2006.

[BV99] G. Berio and F. B Vernadat. New developments in enterprise modelling using CIMOSA. *Comput. Ind.*, 40(2-3):99–114, 1999.

[Col07] G. Colombo. Towards the design of intelligent CAD systems: An ontological approach. *Advanced Engineering Informatics*, 21(2):153–168, 2007.

[Ger90] J. Gero. Design Prototypes: A Knowledge Representation Schema for Design. *AI Magazine*, 4(11):26–36, 1990.

[Gra06] J. Grady. *System requirements analysis.* Elsevier Academic Press, 2006.

[HS08] S. Ha and H. Suh. A timed colored Petri nets modeling for dynamic workflow in product development process. *Computers in Industry*, 59(2-3):193–209, 2008.

[JBo10] JBoss® ESB Documentation. 2010. http://www.jboss.org/jbossesb/docs.html.

[JFJ+08] S. Jablonski, M. Faerber, F. Jochaud, M. Götz, and M. Igler. Enabling Flexible Execution of Business Processes. In *OTM '08: Proc. of the OTM Confederated Int. Workshops and Posters on On the Move to Meaningful Internet Systems*, pp.10–11, Mexico, 2008.

[MHHR06] D. Müller, J. Herbst, M. Hammori, and M. Reichert. IT support for release management processes in the automotive industry. *Business Process Management*, 4102/2006:368–377, 2006.

[MvdAR08] N. Mulyar, W. van der Aalst, and N. Russell. Process flexibility patterns. BETA Working Paper Series WP 251, Eindhoven University of Technology, Netherlands, 2008.

[NFP+09] R. Neugebauer, U. Frieß, J. Paetzold, M. Wabner, and M. Richter. Approach for the Development of Energy-efficient Machine Tools. *Journal of Machine Engineering*, 9(2):51–62, 2009.

[Par71] V. Pareto. *Manual of political economy.* Augustus M. Kelley Pubs, 1971.

[RDR+09] M. Reichert, P. Dadam, S. Rinderle-Ma, M. Jurisch, U. Kreher, and K. Goeser. Architecural principles and components of adaptive process management technology. *PRIMIUM – Process Innovation for Enterprise Software*, LNI P-151:81–97, 2009.

[RM08] H. Reijers and J. Mendling. Modularity in Process Models: Review and Effects. In *Business Process Management*, pp.20–35, Springer, 2008.

[TRC+08] L. Thom, M. Reichert, C. Chiao, C. Iochpe, and G. Hess. Inventing Less, Reusing More, and Adding Intelligence to Business Process Modeling. In *Database and Expert Systems Applications*, vol. 5181/2008, pp.837–850, Springer, 2008.

[vAtH+09] W. van der Aalst, M. Adams, A. ter Hofstede, M. Pesic, and H. Schonenberg. Flexibility as a Service. In *Database Systems for Advanced Applications*, pp.319–333, Springer, 2009.

[vdAtHKB03] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[VDI93] VDI-Guideline 2221. Systematic approach to the development and design of technical systems and products. Association of German Engineers, 1993.

[vPS09] W. van der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - Research and Development*, 23(2):99–113, 2009.

[VRv08] I. Vanderfeesten, H. Reijers, and W. van der Aalst. Product based workflow support: A recommendation service for dynamic workflow execution. *BPM Center Report BPM-08-03, BPMcenter.org*, 2008.

[vt03] W. van der Aalst and A. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30:245–275, 2003.

[WD03] C. Weber and T. Deubel. New theory-based concepts for PDM and PLM. In *Proc. of the 14th Int. Conf. on Engineering Design - ICED 03*, 2003.

[WRR08] B. Weber, M. Reichert, and S. Rinderle-Ma. Change patterns and change support features - Enhancing flexibility in process-aware information systems. *Data & Knowledge Engineering*, 66(3):438–466, 2008.

[YGH04] J. Yang, M. Goltz, and S. Han. Parameter-based Engineering Changes for a Distributed Engineering Environment. *Concurrent Engineering*, 12(4):275–286, 2004.