Rules Catalogue for Necessary Process Adaptations

Deliverable D5.1

FFG – IKT der Zukunft
SHAPE Project
2014 – 845638

Table 1 Document Information

Project acronym:	SHAPE
Project full title:	Safety-critical Human- & dAta-centric Process management in Engineering projects
Work package:	5
Document number:	5.1
Document title:	Rules Catalogue for Necessary Process Adaptations
Version:	0.2 - ready for review
Delivery date:	E03/2016 (M3)
Actual publication date:	01.04.2016
Dissemination level:	Public
Nature:	Technical Report
Editor(s) / lead beneficiary:	Siemens
Author(s):	S. Sperl, A. Haselböck
Reviewer(s):	S. Bala, S. Steyskal

Table 2 History

Version	Changes	Authors
0.1	Created Structure	S. Sperl
0.2	Requirements chapter added	A. Haselböck

Contents

1 Abstract **1**

2 Introduction and Terminology **1**

3 Flexibility **1**

4 Control Flow Patterns **2**

5 Ad Hoc changes **2**

6 Configurable Process Models **3**

7 Inheritance **4**

8 Case-based Reasoning **5**

9 Declarative Paradigm **5**

9.1 Case Handling Paradigm 6

10 Exception Handling **7**

11 Requirements of Rail Automation **8**

11.1 Requirements from Engineering 8

11.2 Requirements from Verification/Validation 9

11.3 Derived SHAPE Requirements 10

12 Summary **11**

Bibliography **12**

1 Abstract

Business Processes need to correspond with reality, but no plan survives contact with the enemy. The question is then how to adapt business processes when exceptions and a changing environment have to be taken into account. Traditional workflow management systems with their inflexible control policies tend to make reactive control and graceful exception handling difficult. This report attempts to give an overview about possible strategies to cope with changes and exceptions during execution and design time. The list of requirements from Siemens Rail Automation are taken into account when it comes to questions like "Which adaptivity scenarios are investigated with which priority?" and "Which strategies and methods are used to resolve such scenarios?".

2 Introduction and Terminology

Dynamic adaptability was not at topic in early process management, since in the primary use case processes were designed once and then executed repeatedly. But in an interactive environment not all processes can be executed as specified. In order to keep going the process participant then in some way has to act without/outside the management system, ultimately turning the system more into a liability than an asset. Adaptability can then mitigate this problem via inherently more flexible process definitions, techniques that allow the user to change/circumvent the process definition at runtime or reporting systems that allow building a better future version of the process definition.

A business process (BP) is a collection of activities with a starting and an ending point. Workflows are a subset of possible business processes where all all activities can be executed/belong to one organization.

Activities represent a piece of work, are executed by one worker (or resource) and result in either a "commit" or "roll back". Their order of execution is typically ordered by some graphical language such as Petri nets [26], the Business Process Modelling Language [14] or UML Activity Diagrams [7].

3 Flexibility

The question is then how to deal with model changes, redesigns, exceptions or just expected variations in execution, this process adaptability can be accomplished by introducing flexibility in various ways [17]:

- **Flexibility by Variability** requires processes to be handled differently depending on context (i.e. process variants). Typically the parameters relevant for variability are known at design time and the concrete variant is bound at run-time (e.g., country specific regulations).
- **Flexibility by Looseness** is usually required if each process is always slightly different, unpredictable and the exact courses of action emerge during execution (e.g., patient treatment processes)
- **Flexibility by Adaptation** is the case when the BP definition can be changed at runtime. Usually requiring that one or all of the currently running BP instances are migrated to the new BP definition. This category includes exception handling.
- **Flexibility by Evolution** is achieved via planned changes at the process definition level, usually supported by a versioning system.

This is not the only taxonomy of flexibility with regards to process adaptation [6,23], the categories (and labels) are roughly similar but differ in details.

4 Control Flow Patterns

Control flow (workflow) patterns are reusable solutions for common modeling problem [22,30]. Originally a set of twenty patterns focused control-flow perspective they were intended as a objective way to compare the relative capabilities of workflow management systems. Now workflow patterns are a de facto standard for evaluating modelling languages like UML 2.0 Activity Diagrams [35], BPEL4WS [34], BPMN [33] and Workflow Management Systems.

With regards to Adaptability, since the intended behaviour is usually declared more explicitly adaptation mechanisms may use specialized handlers (e.g. for Interleaved Routing (execute n activities sequentially in any order)).

5 Ad Hoc changes

When dealing with changes in an already running process by skipping/redoing activities (i.e. changing the execution path) it is easy to miss creating / updating some input data for a later activity or create a live / deadlock. One of the earliest results regarding data flow correctness is published by [16] and shows directly how to correctly handle inserting a new activity that depends on in and output data. A considerably more comprehensive overview about dealing with ad-hoc changes can be seen in [18].

Ad-Hoc changes can be separated into primitive changes (i.e. adding or deleting a single node or edge) or high level adaptation patterns

In the generally supported adaptation mechanisms the user notices a deviation of the specified/expected workflow and initiates changes to his process. If it is not the user that directly adapts the process one speaks of Context-aware systems [24], these react to changes in the context (e.g. availability of people or services) by adapting services, tasks or entire workflows. For example, a workflow arranging transportation contains the tasks for booking a flight, arranging airport pickup and hotel reservation. A predictive context-aware system [25], could note from the weather service that the flight is at risk of being cancelled and book a train ticked and cancel/modify hotel reservations. While a reactive (not predictive) system would initiate the corrective behaviour at the the point of time when the flight is known to be cancelled.

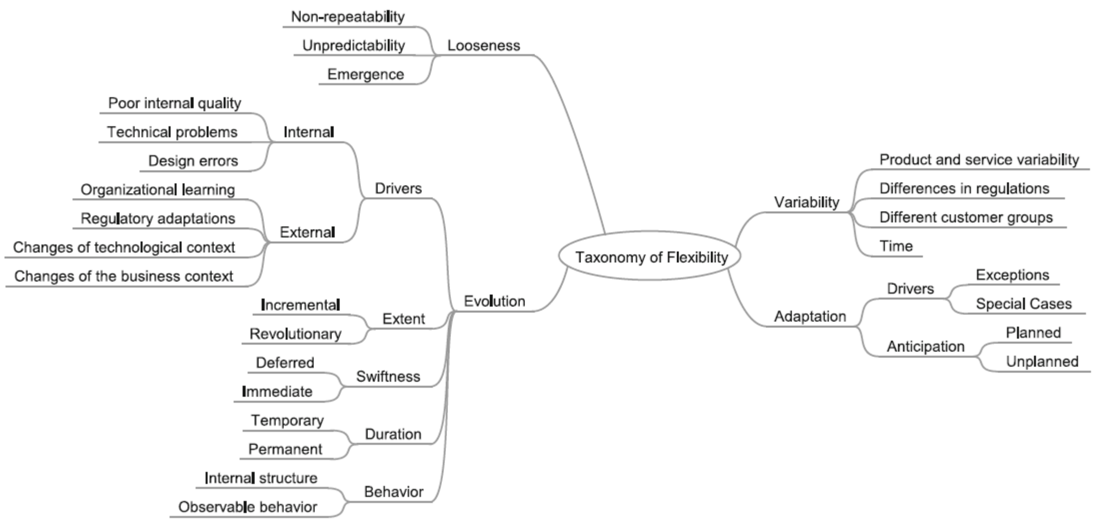


Figure 1 A taxonomy of process flexibility needs [17]:Fig. 3.1

6 Configurable Process Models

Configuration can be seen as *limiting choices by making choices*, in [10] a configurable element can be marked as activated, blocked or hidden. An element configured as blocked then the element and any subsequent activities are removed, if configured as hidden the element is removed but subsequent activities can still be executed. Note, theoretically the notion of configuring a process model can be extended to runtime if one considers a classical "or" decision node as undergoing configuration once a condition on a path becomes true. One way of creating configurable process models is by merging multiple process models into one. [21] created an algorithm that creates a configurable process model which subsumes the input models and further enables tracing back every element to it's source model.

A slightly different approach merges configuration with the ability to give different views for different roles of the same process [3,20].

In general configuring process variants is non-trivial when considering the high variability of business processes as well as the many syntactical and semantical constraints the configured process variants may have to obey in different contexts. The Provop approach [11] defines a base process on which a sequence of changes can be applied, like classical insert update delete operations supported by specially marked modification points on the model. Additionally the provop approach allows one to specify constraints between options (e.g. option 1 implies the usage of option 2).

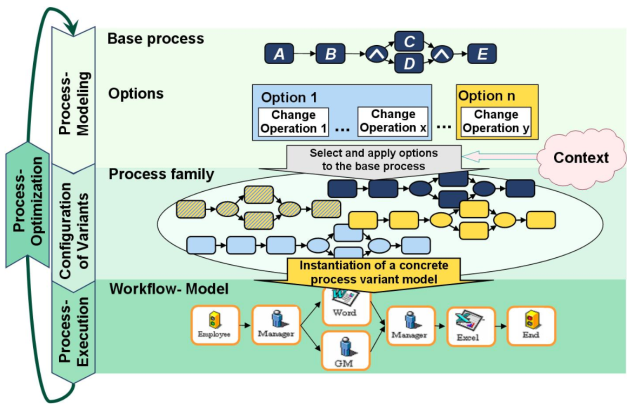


Figure 2 Configurable Process Models. Provop approach [11]:Fig. 3

7 Inheritance

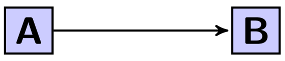
The introduction of inheritance (as known from object-orientation) for workflows intends to enable a more graceful reaction of workflow systems to evolutionary or ad-hoc changes, via inheritance-preserving transformation rules [27,36]. So for example in some cases ad-hoc changes can be seen as specialization under some inheritance relation and therefore not requiring an entire copy (or new version) of a workflow (process definition).

8 Case-based Reasoning

Case-based Reasoning in general is the process of solving new problems based on the solutions of similar past problems [2]. In our context, the idea is to remember past ad-hoc changes to the process model and let those changes inform future redesigns or just help the user in a currently running adaptation [19].

9 Declarative Paradigm

Falling under achieving flexibility by looseness is the declarative paradigm which moves from the classical imperative specification of workflows to a more declarative specification [8,15]. To elaborate, an imperative specification tends to specify what is possible while a declarative approach tends to specify what is impossible. In stronger terms, when using the declarative approach by default all execution paths are allowed which can then be restricted by constraints. This is a answer to the problem of modelling all possible execution paths / requiring execution paths where not necessary.



Imperative: $\{[A, B]\}$
Declarative: $\{[A, B], [A, A], [A, A, B], [A, B, A], \ldots\}$

Figure 3 Example showing possible event log interpretations of "A before B" for different approaches.

Logically the highest amount of flexibility can be achieved by declaring no constraints, at the cost of no support (e.g. enforcing correct execution or recommending effective execution), for a more detailed description of the interplay between flexibility and support see [29].

Similar to imperative problems in the model (e.g. dead/livelock) a declarative model can contain dead activities (activities which can never be executed) or conflicts (there is always a violated constraint in the model). One should note that these checks for complex languages like LTL as used in Declare [29] can quickly become intractable and likely require a restricted featureset for larger problems.

A rather distinctive disadvantage of the declarative approach is understandability, complex formulae (like LTL) tend to become hard to understand much faster than a classical graphical notation.

The idea of freeing the strict control flow via the usage of preconditions is not new, like the Vortex Paradigm [12] or Document-Driven Workflow [32]. Case handling is probably one of more used implementation of approaches that fall under the umbrella term artifact(data)-centric business processes.

9.1 Case Handling Paradigm

Case handling integrates processes and associated data in a tighter manner, using produced data not only for routing decisions, but also for determining which parts of the process have already been accomplished [1,31] (Note, often a case is defined as a workflow instance e.g. YAWL). In other words a case follows the idea of specifying "given this situation what is to be/can be done". A data object is a piece of information which if present has an associated value (and is usually edited via forms).
Case Handling prioritizes the following design goals:

- Provide all pieces of available information.
- Activities are enabled on the basis of available information.
- Information can be registered/modified at any time.

Each activity is associated with three sets of data objects:

- Authorized data objects that are visible while performing the activity.
- Mandatory data objects which once set cause the activity to be finished.
- Restricted data objects that can be changed during the activity.

Additionally data objects can be free, meaning they can be edited as long as the case is being handled. Undo, skip and redo functionality in CH is automatically supported by clearing the associated mandatory data objects. Since undo,skip and redo are ubiquitous many process paths or ad-hoc changes required for classical PMs become unnecessary.

A standard application of case handling is in the healthcare domain, where patients must be registered without possibly knowing their name, sudden complications can require new/interrupt activities or new relevant information (like allergies) can become available at any moment.

A simple example showing multiple parts (packages) that require development and verification by a separate role. CH naturally supports creating forms where the associated data objects (if the role has the necessary rights) can be edited. The dotted line represents that this activity is allowed to change the connected data objects. The not shown roles in Case Handling are handled by authorization rules.

Case handling in general does not adequately support interaction between different cases or multiple cases of instances. Object aware Process Management [4] intends to fill the gap by providing an entire object structure for interaction (in

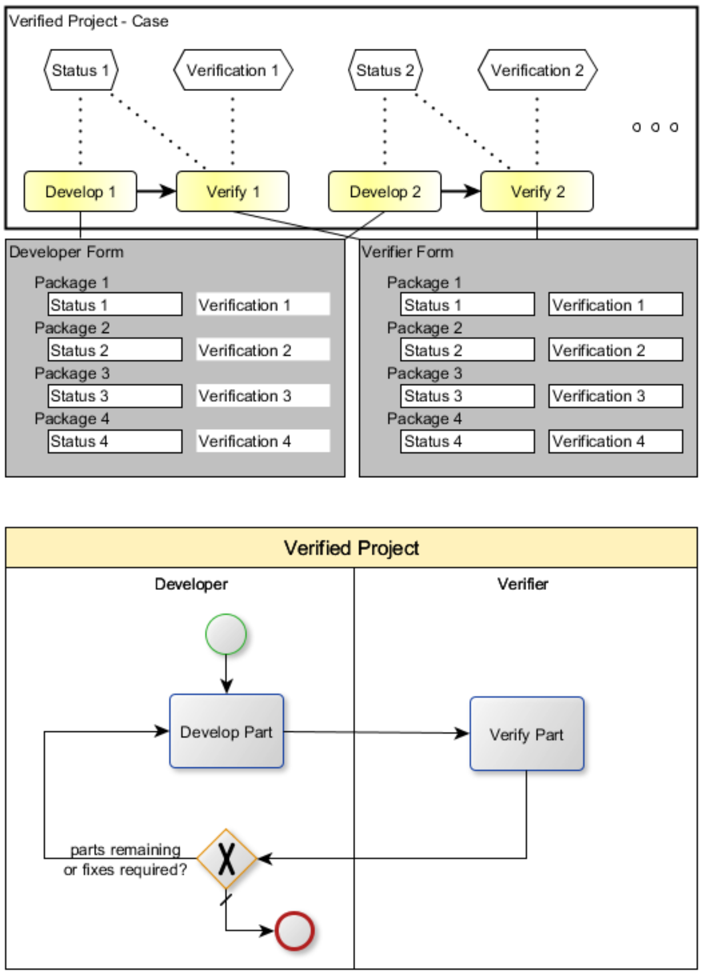


Figure 4 The same problem as Case and Process.

contrast to classical CH where there are only atomic data objects and at best a case can be considered an object). Currently the only implementation for object aware processes is the PHILharmonicFlows framework [5].

Note, Case Handling is in use under several terms; "case management", "case file handling", "case file management" and "case folder management" (under these labels it is usually aligned with law and government institutions).

10 Exception Handling

In general adaptivity regarding exceptions is mainly interested in unexpected exceptions and as such can be seen as having the following goals [13]:

- **Detecting Exceptions.** Under the assumption that not all anomalies are known a-priori, it is of interest to be able to detect new exceptional situations and have the means to detect the origin point of an exception.

- **Avoiding Exceptions** The system itself can be the source of errors, as such approaches such as open systems, incremental adoption, flexible execution, reusable processes can often help prevent the rise of exceptions.
- **Handling Exceptions** Depending on the severity/frequency of the exception handling strategies are; tolerating minor deviations, local exception handling that only changes the affected process, and global exception handling that is used for evolving process definitions.

In general the handling of exceptions can be separated in three groups; trying alternatives, adding behaviour and cancelling behaviour. Adding behaviour then includes fixing, retrying or reworking process parts while cancelling behaviour includes actions that cancel or undo a process.

11 Requirements of Rail Automation

During several meetings, experts from Siemens Rail Automation defined and refined requirements on the integration of a business process management system (BPMS) into their engineering and verification process. The main input came from the head of the engineering group head of the verification/validation group. In the following, we summarize those requirements that are related to adaptivity and flexibility aspects of their engineering and verification processes.

11.1 Requirements from Engineering

Distributed engineering teams. Rail Automation projects usually are large and complex technical projects, meaning that most often more than one engineer/team of engineers is involved. Engineers working on a project are often located at different sites (e.g., Austria and Slovakia).

Parallel projects. Usually, an engineer or a team of engineers works on several projects in parallel.

Different system types. The portfolio of Siemens Rail Automation in Austria contains various different types of systems. The main one in Austria is the so-called "Anlagenbauprozess" for configuring electronic interlocking systems in Austria. For interlocking systems in countries other than Austria, a different technology and tool set is used. Other types of systems are: ETCS (European Train Control System) level 1 and 2, level crossings, etc.

Different project types. The two main variants of a project for a system are: (1) Build a new system; (2) Modify/extend an existing system.

Work parallelization. In principle, the different processes are sequential processes. Due to distributed work and parallel projects, many process steps are performed in parallel.

Conformance checks. There are many constraints specified on the different process steps and assets (e.g., the right versions of input files). Currently, many of those checks are done at the end of a project. Doing checks as early as possible could save rollback and rework time.

Skills. Engineers have different experiences and skills. The goal is to assign only those engineers to a process step who are (a) available, (b) have the necessary skills, and (c) preferably already know the facility in case of required modifications/extensions of the system.

Documentation of time and costs. Documentation of time and costs of engineering/verification tasks in a project is usually not done on basis of single steps, but on process phases: "How many hours are used for engineering or verifying a project?" Documentation on a finer level (e.g. on the level of a single process step) could facilitate a better resource allocation and planning of future projects.

Data access rights. Currently, there are no hard restrictions on who is reading or changing which data.

Change interdependencies. Currently, it is very difficult to know or track interdependencies of changes at different parts of the system. This leads to challenges like: (a) change tracking, (b) change effect interdependency control, (c) communication of changes, and (d) to make all the involved people work according to the interdependency model.

Non-human resources. The main, critical non-human resources are hardware and workstations in the test laboratory. Similar to human resources, optimal allocations of test laboratory equipment can essentially reduce system engineering and test effort.

11.2 Requirements from Verification/Validation

The main goal of the verification/validation process is the final approval of a national safety authority. In the following, we outline the main requirements of verification/validation processes.

Verification and validation process. Part of the overall project management plan are both the verification and validation process, which are usually started after the engineering process has ended. Doing some of the verification/validation steps in parallel with the engineering process could save time by detecting errors earlier.

Process approvals. Engineering, verification and validation processes must have several approvals, like "Verfahrenssicherheitsnachweis" and "Gutachten". An extensive and up-to-date process documentation is an important prerequisite of these approvals.

SIL 4. Most of the systems built by Rail Automation must be engineered under the condition of SIL 4 (Safety Integration Level). SIL 4 is a very high level and usually cannot be achieved by making each single process step SIL 4 compliant, but the whole process is designed to result in SIL 4 systems by integrating diverse redundant engineering steps. Example: The output of an engineering tool is re-transformed to the input data and checked against them by a separate step/tool.

Standards and norms. The main standards are EN50126 [1] (life-cycle model), EN50128 (important for the engineering process), and EN50129 (risk analysis, safety targets, hazard rates). Some of the rules in these standards and norms can be checked automatically. The BPMS should support such automatic checks.

Traceability. Tracing of steps between input and output should be improved. Documentation of process steps is an important input for validation (usually at the end of a project).

Process documentation. Engineering must prove that the system has been engineered in compliance with the defined, safe engineering process. Currently, there are no automated monitoring/tracking capabilities, so the challenge is to know whether all process steps are executed correctly or not. Most of the checks are currently performed at the end of the process.

Delta verification. In modification/extension projects, much verification work could be saved, if only those parts of the system data are checked that have actually changed.

11.3 Derived SHAPE Requirements

Derived from Siemens' Rail Automation requirements we identify the following main requirements concerning adaptivity and flexibility as input for our SHAPE reference architecture and tool selection:

[1] In the Bachelor's Thesis of Julia Fuchsbauer [9] - written in the course of the project SHAPE - the rules of EN50126 have been systematically extracted and categorized.

Automated Resource (Re-)Allocation of human resources. The BPMS should support automated resource allocation and re-allocation of human resources. Data like user roles, skill matrix, experience and performance values of users (about skills and actual projects / systems) should be used for an optimal allocation. Experience and performance values could be learned or refined from process logs. The project manager should be able to overrule computed resource allocation. Resource allocation should satisfy hard and soft constraints. Example of a hard constraint: Verifier must be different to engineer. Example of a hard constraint: If a user is assigned the first time to a task of a certain type, an other, experienced user must be assigned to that task, too. Example of a soft constraint: Assign task to engineer with highest experience points.

Automated Resource (Re-)Allocation of non-human resources. Similar to human resources, the BPMS should support automated allocation and re-allocation of non-human resources. Hardware, workstations and laboratory time are the main non-human resources.

Exception handling. The BPMS should be able to handle exceptional situations. In addition to the standard process paths, the process definition should contain the most important exception paths. In exceptional cases, the process must not prohibit deviations of the defined process paths. Deviations must be documented.

Guide back to standard path. In cases of process deviations, the BPMS should guide users back to a standard path. Consistency of external data must be restored; e.g.: entries in the project logbook must be updated whenever tasks are rolled-back and re-entered.

12 Summary

Rail Automation requirements show that engineering and verification processes of safety-critical systems strongly rely on various different kinds of data. Two examples are (i) resource allocation, which is based on, e.g., roles, skills, experiences, and availability of team members, and (ii) document generation, which collects all the necessary data produced in different tasks for filling out templates. From the view of adaptivity/flexibility, we argued in Section 9 that a declarative paradigm better supports data-oriented processes than classical imperative process workflow specifications. Nevertheless, an architectural decision for Camunda[2] and the

[2] https://camunda.com/

notation language BPMN has been made in SHAPE. In future work, we will integrate data-centered adaptivity patterns into the SHAPE reference architecture.

References

1 Wil M.P. van der Aalst and P.J.S. Berens. Beyond Workflow Management: Product-Driven Case Handling. In S. Ellis, T. Rodden, and I. Zigurs, editors, *International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001)*, pages 42–51. ACM Press, New York, 2001.

2 Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, 1994.

3 Jörg Becker, Patrick Delfmann, Alexander Dreiling, Ralf Knackstedt, and Dominik Kuropka. Configurative process modeling–outlining an approach to increased business process model usability. In *Proceedings of the 15th IRMA International Conference*, pages 1–12. Hershey Idea Group, 2004.

4 Carolina Ming Chiao, Vera Künzle, and Manfred Reichert. Enhancing the case handling paradigm to support object-aware processes. In Rafael Accorsi, Paolo Ceravolo, and Philippe Cudré-Mauroux, editors, *Proceedings of the 3rd International Symposium on Data-driven Process Discovery and Analysis, Riva del Garda, Italy, August 30, 2013*, volume 1027 of *CEUR Workshop Proceedings*, pages 89–103. CEUR-WS.org, 2013.

5 Carolina Ming Chiao, Vera Künzle, and Manfred Reichert. A tool for supporting object-aware processes. In Georg Grossmann, Sylvain Hallé, Dimka Karastoyanova, Manfred Reichert, and Stefanie Rinderle-Ma, editors, *18th IEEE International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, EDOC Workshops 2014, Ulm, Germany, September 1-2, 2014*, pages 410–413. IEEE, 2014.

6 Riccardo Cognini, Flavio Corradini, Stefania Gnesi, Andrea Polini, and Barbara Re. Research challenges in business process adaptability. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 1049–1054, New York, NY, USA, 2014. ACM.

7 Marlon Dumas and Arthur H. M. ter Hofstede. Uml activity diagrams as a workflow specification language. In *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, &#171;UML&#187; '01, pages 76–90, London, UK, UK, 2001. Springer-Verlag.

8 Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo A. Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative versus imperative process modeling languages: The issue of understandability. In Terry A. Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, Pnina Soffer, and Roland Ukor, editors, *BMMDS/EMMSAD*, volume 29 of *Lecture Notes in Business Information Processing*, pages 353–366. Springer, 2009.

9 Julia Fuchsbauer. How to manage processes according to en50126. Bachelor thesis, Vienna University of Economics and Business, July 2015.

10 Florian Gottschalk, Wil M. P. van der Aalst, Monique H. Jansen-Vullers, and Marcello La Rosa. Configurable workflow models. *Int. J. Cooperative Inf. Syst.*, 17(2):177–221, 2008.

11 Alena Hallerbach, Thomas Bauer, and Manfred Reichert. Capturing variability in business process models: the provop approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(6-7):519–546, 2010.

12 Richard Hull, François Llirbat, Eric Simon, Jianwen Su, Guozhu Dong, Bharat Kumar, and Gang Zhou. Declarative workflows that support easy modification and dynamic browsing. In *Proceedings of the international joint conference on Work activities coordination and collaboration 1999, San Francisco, California, USA, February 22-25, 1999*, pages 69–78. ACM, 1999.

13 Peter J. Kammer, Gregory Alan Bolcer, Richard N. Taylor, Arthur S. Hitomi, and Mark Bergman. Techniques for supporting dynamic and adaptive workflow. *Computer Supported Cooperative Work*, 9(3/4):269–292, 2000.

14 OMG. Business Process Model and Notation (BPMN), Version 2.0. Technical report, Object Management Group, January 2011.

15 M. Pesic and W. M. P. van der Aalst. A declarative approach for flexible business processes management. In *Proceedings of the 2006 International Conference on Business Process Management Workshops*, BPM'06, pages 169–180, Berlin, Heidelberg, 2006. Springer-Verlag.

16 Manfred Reichert and Peter Dadam. A framework for dynamic changes in workflow management systems. In *DEXA Workshop*, pages 42–48, 1997.

17 Manfred Reichert and Barbara Weber. *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Springer, Berlin, 2012.

18 Stefanie Rinderle, Manfred Reichert, and Peter Dadam. Correctness criteria for dynamic changes in workflow systems - a survey. *Data Knowl. Eng.*, 50(1):9–34, 2004.

19 Stefanie Rinderle, Barbara Weber, Manfred Reichert, and Werner Wild. Integrating process learning and process evolution - A semantics based approach. In van der Aalst et al. [28], pages 252–267.

20 Marcello La Rosa, Marlon Dumas, Arthur H. M. ter Hofstede, and Jan Mendling. Configurable multi-perspective business process models. *Inf. Syst.*, 36(2):313–340, 2011.

21 Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco M. Dijkman. Merging business process models. In Robert Meersman, Tharam S. Dillon, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2010 - Confederated International Conferences: CoopIS, IS, DOA and ODBASE, Hersonissos, Crete, Greece, October 25-29, 2010, Proceedings, Part I*, volume 6426 of *Lecture Notes in Computer Science*, pages 96–113. Springer, 2010.

22 N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow control-flow patterns: A revised view. Technical Report BPM-06-22, BPM Center, 2006.

23 Helen Schonenberg, Ronny Mans, Nick Russell, Nataliya Mulyar, and Wil M. P. van der Aalst. Towards a taxonomy of process flexibility. In Zohra Bellahsène, Carson Woo, Ela Hunt, Xavier Franch, and Remi Coletta, editors, *CAiSE Forum*, volume 344 of *CEUR Workshop Proceedings*, pages 81–84. CEUR-WS.org, 2008.

24 Sucha Smanchat, Sea Ling, and Maria Indrawan. A survey on context-aware workflow adaptations. In Gabriele Kotsis, David Taniar, Eric Pardede, and Ismail Khalil Ibrahim, editors, *MoMM*, pages 414–417. ACM, 2008.

25 Abhishek Tiwari and Arvind K. T. Sekhar. Review article: Workflow based framework for life science informatics. *Comput. Biol. Chem.*, 31(5-6):305–319, October 2007.

26 Wil M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

27 Wil M. P. van der Aalst and Twan Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theor. Comput. Sci.*, 270(1-2):125–203, 2002.

28 Wil M. P. van der Aalst, Boualem Benatallah, Fabio Casati, and Francisco Curbera, editors. *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649. Springer, 2005.

29 Wil M. P. van der Aalst, Maja Pesic, and Helen Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - R&D*, 23(2):99–113, 2009.

30 Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

31 Wil M. P. van der Aalst and Mathias Weske. Case handling: A new paradigm for business process support. *Data Knowl. Eng.*, 53(2):129–162, May 2005.

32 Jianrui Wang and Akhil Kumar. A framework for document-driven workflow systems. In van der Aalst et al. [28], pages 285–301.

33 Stephen A. White. Process Modeling Notations and Workflow Patterns. On BPMN website, 2004.

34 Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, and Arthur H. M. ter Hofstede. Analysis of web services composition languages: The case of BPEL4WS. In Il-Yeol Song, Stephen W. Liddle, Tok Wang Ling, and Peter Scheuermann, editors, *Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling, Chicago, IL, USA, October 13-16, 2003, Proceedings*, volume 2813 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2003.

35 Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, Arthur H. M. ter Hofstede, and Nick Russell. Pattern-based analysis of the control-flow perspective of UML activity diagrams. In Lois M. L. Delcambre, Christian Kop, Heinrich C. Mayr, John Mylopoulos, and Oscar Pastor, editors, *Conceptual Modeling - ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24-28, 2005, Proceedings*, volume 3716 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2005.

36 George M. Wyner and Jintae Lee. Defining specialization for process models. pages 131–174. MIT Press, 2003.