Dealing with Variability in Process-Aware Information Systems:

Language Requirements, Features, and Existing Proposals

Clara Ayora, Victoria Torres, Barbara Weber,
Manfred Reichert, Vicente Pelechano

Ulmer Informatik-Berichte

Nr. 2012-07

Dezember 2012

Dealing with Variability in Process-aware Information Systems: Language Requirements, Features, and Existing Proposals

Clara Ayora[a], Victoria Torres[a], Barbara Weber[b], Manfred Reichert[c], Vicente Pelechano[a]

[a]*Centro de Investigación en Métodos de Producción de Software, Universitat Politècnica de València, Spain*
[b]*Department of Computer Science, University of Innsbruck, Austria*
[c]*Institute of Databases and Information Systems, University of Ulm, Germany*

Abstract

The increasing adoption of Process-aware Information Systems (PAISs), together with the variability of Business Processes (BPs) across different application contexts, has resulted in large process model repositories with collections of related process model variants. To reduce both costs and occurrence of errors, the explicit management of variability throughout the BP lifecycle becomes crucial. In literature, several proposals dealing with BP variability have been proposed. However, the lack of a method for their systematic comparison makes it difficult to select the most appropriate one meeting current needs best. To close this gap, this work presents an evaluation framework that allows analyzing and comparing the variability support provided by existing proposals developed in the context of BP variability. The framework encompasses a set of language requirements as well as a set of variability support features. While language requirements allow assessing the expressiveness required to explicitly represent variability of different process perspectives, variability support features reflect the tool support required to properly cover such expressiveness. Our evaluation framework has

Email addresses: `cayora@pros.upv.es` (Clara Ayora), `vtorres@pros.upv.es` (Victoria Torres), `barbara.weber@uibk.ac.at` (Barbara Weber), `manfred.reichert@uni-ulm.de` (Manfred Reichert), `pele@pros.upv.es` (Vicente Pelechano)

been derived based on an in-depth analysis of several large real-world process scenarios, an extensive literature review, and an analysis of existing PAISs. In this vein, the framework helps to understand BP variability along the BP lifecycle. In addition, it supports PAISs engineers in deciding, which of the existing BP variability proposals meets best their needs.

Key words: Process Model Configuration, Business Process Variability, (Configurable) Process-Aware Information Systems, Process Model Families

1. Introduction

Each product an enterprise develops or produces and each service it provides results from the performance of a set of activities. Business Processes (BPs) are the drivers coordinating these activities [74]. In enterprise computing, *Process-aware Information Systems* (PAISs) provide a guiding framework to understand and deliberate on BPs [72]. A PAIS is defined as an information system that manages and executes operational processes based on BP models involving resources, applications, and data [18].

The increasing adoption of PAISs during recent years has resulted in large process model repositories with numerous collections of BP models [62, 17]. Since these models frequently vary depending on the application context [24, 59], the existing repositories often comprise large collections of related *process model variants* (*process variants* for short). These process variants pursue the same or similar business objective (e.g., the treatment of a patient or maintenance of vehicles in a garage), but at the same time differ in their application context, e.g., regulations found in different countries and regions, type of product or service being delivered, customer categories, or seasonal changes [58, 17, 70].

A collection of *process variants* sharing a number of *commonalities* (e.g., activities found in all process variants), but also showing differences due to their application context is denoted as a *process family*. In large companies such process families comprise dozens or hundreds of process variants [53]. For instance, in the context of the automotive domain, [27] reported on a process family for vehicle repair and maintenance comprising more than 900 variants with country-, garage-, and vehicle-specific differences. In addition, related to the healthcare domain, [45] reported on more than 90 process variants for handling medical examinations. Moreover, also the check-in procedures at airports are characterized by a high level of variability. In the following example, we describe this process—including the different sources for variability—and use it as running example throughout the paper.

Example 1 (Check-in process). This example presents the process every passenger has to go through when checking-in at an airport. Even though this process is similar irrespective of the airport the passenger departs from and the airline she is flying with, many variations are possible depending on different factors. For instance, variability appears due to the type of check-in (e.g., online, at the counter, or at the self-servicing machine) which also determines the type of boarding card (e.g., electronic versus paper-based). Other sources of variability are the flight destination (e.g., extra information is required when traveling to the US) and the ticket class (e.g., economy or business class). These determine both the check-in priority and the option to change seat assignments. Moreover, depending on the type of luggage (e.g., bulk or overweight luggage) the process slightly differs. Temporal variations regarding the availability of the check-in are typical as well (e.g., possibility to check-in several days before departure versus a few hours). In addition, variations are introduced due to different ways to operationalize an activity (e.g., the implementation of activity *Print boarding card* in self-servicing machines is different from the one used by the web system). As a result of all these variations, hundreds of variants can be identified. Figures 1 and 2 show five simplified process variants considering selected variations. Common activities to all variants are colored in grey.

Consider the three process variants from Figure 1. *Variant 1* assumes that the check-in is done online and directly by the passenger who is identified by her passport number. Check-in is available 23 hours before departure. Since the passenger holds a business class ticket, the airline offers her the possibility to change the assigned seat. Moreover, the passenger is flying from an European country to the United States requiring additional information about accommodation. Finally, an electronic boarding card is printed and the passenger drops her luggage at the business class counter. Regarding *Variant 2*, in turn, since the flight destination is within Europe, information regarding accommodation is not required. *Variant 3* is similar to *Variant 2*, but additionally requires the payment of an extra fee at the excess baggage

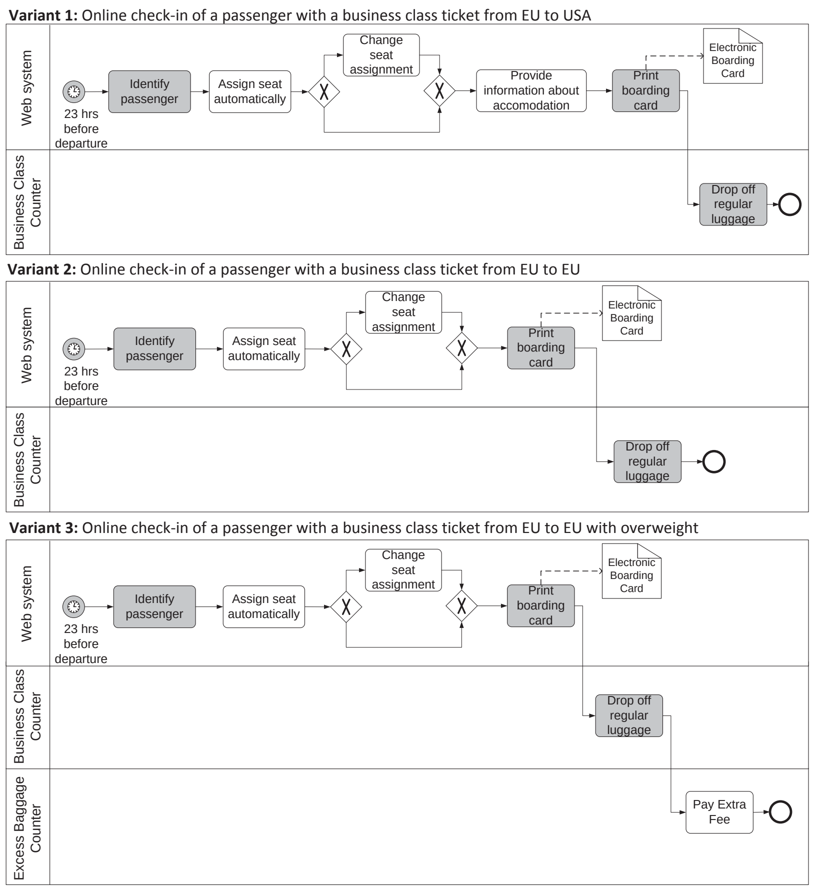


Figure 1: Example of Process Variants Belonging to the Same Process Family

counter due to luggage overweight.

Unlike *Variants 1-3*, *Variants 4-5* (see Figure 2) consider passengers holding an economy class ticket, which means that passengers are not eligible for priority check-in and have to drop-off the luggage at the economy class counter. Moreover, the check-in is not done online, but directly at the economy class counter (*Variant 4*), or at the self-servicing machine (*Variant 5*). In addition, the resulting boarding card is in paper format. Moreover, for *Variant 4* check-in can only be done 3 hours before departure, once the economy class counter has opened.

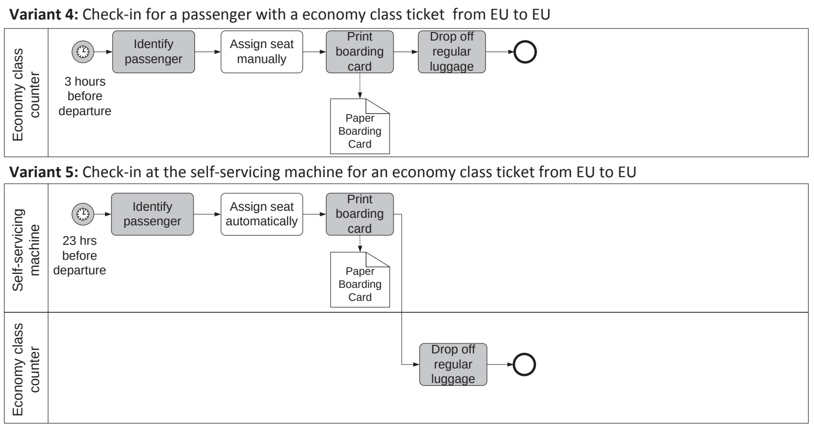


Figure 2: Example of Process Variants Belonging to the Same Process Family

1.1. Problem Statement

Properly dealing with process families (i.e., large process model repositories comprising a large number of related process variants) constitutes a fundamental challenge to reduce process modeling and maintenance efforts in the context of PAISs. Trying to design, implement, and maintain each process variant of a process family from scratch would be too inefficient and costly for enterprises. Thus, there is a great interest in capturing common process knowledge only once and re-using it in terms of *reference process models* (*reference process* for short). Following this trend, a variety of reference processes has been suggested in recent years. Examples include ITIL for IT service management [30], SAP reference models representing BPs as supported in SAP's enterprise resource planning system [52], or medical guidelines for patient treatment [43]. Typically, these reference processes are described in text using a narrative form or by a conventional process modeling language. Even though these examples foster the reuse of common process knowledge, they typically lack comprehensive support for explicitly describing the variations contained in a reference process [60]. In addition, as stated in [27], the lack of variability support in current commercial process modeling tools requires the manual derivation of process variants, which is a cumbersome and error-prone task. Frequently, these individual process variants are specified and maintained either separately in different models (i.e., structural approach) or together in the same process model using conditional branches (i.e., behavioral approach) [28]. Both approaches, however, result in complicated and redundant models which are difficult to comprehend, manage, and maintain [58].

It is well known in Software Product Lines [32], or software aging [54], that software degenerate over time when code is cloned and modified or added by different developers and that large software product families have to continuously be maintained. Similar challenges exist in the context of process families with a large numbers of process variants.

In recent years, several proposals have faced this situation and proposed to deal with BP variability throughout the BP lifecycle [56, 63, 27, 8]. They deal with the modeling, execution, and monitoring of process families and variants. However, there is a lack of profound methods for systematically comparing the different proposals, which makes it difficult for PAISs engineers to select the proposal which best suits to their needs.

To make PAISs better comparable and to facilitate the selection of appropriate PAIS-enabling technologies, process patterns have been introduced [3, 64, 65, 71, 11]. Respective patterns provide means for analyzing the expressiveness of process modeling tools and languages in respect to different process perspectives. In particular, proposed patterns cover activities [68], control flow [3], data flow [64], resources [65], time [41, 42], exceptions [66], and process changes [71]. However, a framework for evaluating a PAIS regarding its ability to deal with BP variability is still missing and will be picked up by this work.

1.2. Contribution

The major contribution of this paper are threefold:

1. It presents results from an analysis of a set of real-world process families that stem from different domains (e.g., automotive industry, healthcare, airport procedures) featuring BP variability.
2. Based on this empirical evidence as well as on a detailed literature review, an evaluation framework for proposals enabling BP variability is developed. This framework comprises a set of *language requirements* needed to accommodate the identified variability needs. In addition, it contains a set of *variability support features.* While the language requirements allow assessing the expressiveness of existing BP variability proposals, variability support features ensure that process families can

be effectively modeled, verified, validated, configured, analyzed, refactored, and evolved. Moreover, they ensure that process variants can be effectively executed, monitored, and dynamically re-configured.

3. Taking this evaluation framework, an in-depth review of existing proposals from academia dealing with BP variability is conducted. This review provides an objective overview of the BP variability support provided by existing proposals as well as the open issues not covered by them yet.

This work can be considered as a reference for implementing PAISs being able to effectively support BP variability along the entire BP lifecycle. In addition, in analogy to the process patterns initiative (see above), we expect the evaluation framework to be applied to different BP variability proposals as well as related tools to evaluate their suitability for BP variability management. In this vein, the framework is expected to support enterprises in deciding which proposal suits their needs best.

The remainder of the paper is organized as follows: Section 2 summarizes background information to contextualize the evaluation framework. Section 3 presents the research methodology applied for developing the evaluation framework. Section 4 introduces the evaluation framework, while Section 5 presents a selection of proposals dealing with BP variability and applies the developed evaluation framework to asses their ability to support BP variability. Finally, Section 6 concludes the paper with a summary and outlook.

2. Background information

This section provides some basics related to BPs, putting the emphasis on the variability issues that arise when dealing with process families throughout the BP lifecycle. By means of the different BP perspectives, Section 2.1 first introduces the concepts and terms that are used to represent BP models. Afterwards, Section 2.2 extends these concepts by variability-specific issues and a revised definition of the BP lifecycle is provided.

2.1. Business Process Perspectives

According to [74], a BP is defined as "a set of activities performed in coordination in an organizational and technical environment". Analyzing this definition, a BP defines *what* (activities) should be done, *how* (coordination), and by *whom* (organizational and technical environment). In this context, *business process models* (i.e., *process schema*) arise as the main artifacts for representing BPs. BP models are built by instantiating a BP meta-model as the one shown in Figure 3. Depending on the object we focus on, in this meta-model, we can differentiate several BP perspectives [15, 51, 33, 58, 31]. These refer to the functional, behavioral, organizational, informational, temporal, and operational perspectives:

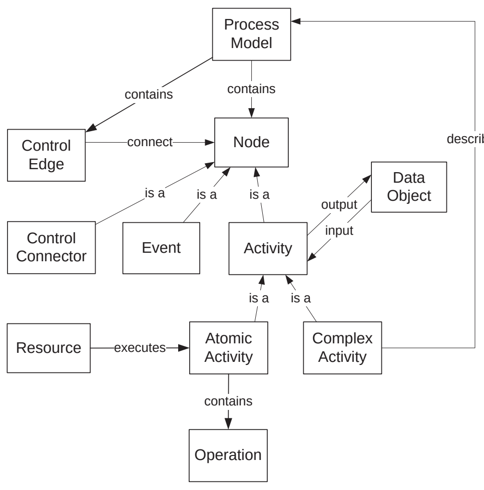


Figure 3: Business Process Meta-model

- *Functional perspective:* specifies the decomposition of BPs, i.e., it represents the *activities* to be performed [15]. While an atomic activity is associated with a single action, a complex activity refers to a sub-process or, more precisely, a sub-process model. This perspective is represented by concepts *activity, atomic activity,* and *complex activity* from Figure 3.

- *Behavioral perspective:* captures the dynamic behavior of a BP model and corresponds to the *control flow* between the activities. A *control flow schema* includes information about the order of the activities or the constraints for their execution. This perspective is represented by concepts *control connector* and *control edge* from Figure 3.

Example 2 (Variability of the behavioral perspective). For example, after executing activity *Assign seat automatically*, two alternative options exist, i.e., performing activity *Change seat assignment* or skipping this activity. These alternatives relate to business class passengers (*Variants 1-3*) and economy class passengers (*Variants 4-5*), respectively.

- *Organizational perspective:* deals with the assignment of *resources* to the activities of a BP model, i.e., it represents the actors, roles (i.e., humans or systems), within an organization being in charge of executing certain BP activities. This perspective is represented by concept *resource* from Figure 3.

Example 3 (Variability of the organizational perspective). While online check-in is performed by the passenger using a web system (*Variants 1-3*), check-in at the counter and at the self-servicing machine is performed by airline personnel (*Variant 4*) and the machine (*Variant 5*), respectively.

- *Informational perspective:* concerns *data* and *data flow*, i.e., it represents the informational entities (e.g., data, artifacts, products, and objects) consumed or produced during the execution of BP activities. It is represented by concept *data object* from Figure 3.

Example 4 (Variability of the informational perspective). Depending on the type of check-in, the resulting boarding card is a digital or a paper-based document.

- *Temporal perspective:* deals with time issues and temporal constraints, i.e., it represents the occurrence of events during the course of a process, which affects the scheduling of activities from this process (e.g., an activity started or ended, a message arrived, a time expired, or an error occurred). This perspective is represented by concept *event* from Figure 3.

Example 5 (Variability of the temporal perspective). The availability of the check-in service is delimited from 23 to 2 hours before departure, depending on the airline and destination.

- *Operational perspective:* refers to the implementation of process activities, i.e., the application services to be executed when an *atomic activity* is performed. It is represented by concept *operation* from Figure 3.

Example 6 (Variability of the operational perspective). The implementation of the *Print boarding card* activity differs depending on the type of check-in selected; i.e., online, at counter, or using the self-servicing machine.

2.2. Business Process Lifecycle

BPs do not only serve for documentation purposes, but are embedded in a lifecycle composed of different phases that are organized in a cyclical structure (see Figure 4) showing their logical dependencies [74, 13, 9]. These phases include *Analysis & Design, Configuration, Enactment, Diagnosis,* and *Evolution.*

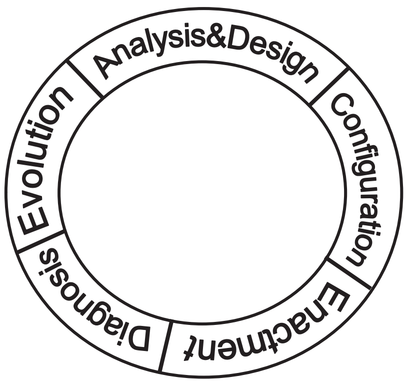


Figure 4: Business Process Lifecycle

Analysis & Design phase. During this phase, and based on domain requirements, relevant BPs are identified, captured in terms of BP models, and subsequently validated and verified. In the context of BP variability, it is important to define (1) what parts of the BP model may vary according to a specific context, (2) what alternatives fit in each of those parts, and (3) which conditions make these alternatives being selected. The first issue refers to the precise identification of the parts being subject to variation, which are commonly known as *variation points*. The second issue refers to the different alternatives that exist for all those variation points which are called *process fragment substitutions*. In addition, to ensure semantically correct combinations, relationships (such as inclusion or exclusion) between these substitutions are important. The third issue refers to the *context* of these variations, which is usually represented by a set of variables gathered in a *context model* (i.e., application environment) in which the BP model is used. After considering all these variability aspects, we obtain a *configurable process model*, which is capable of representing the complete *process family*. Depending on the approach (i.e., behavioral or structural) followed to build a configurable process model, the latter can either consist of one or several artifacts. While a behavioral approach usually results in a unique artifact integrating the behavior of all family members, a structural approach results into a set of artifacts, which separately represent different aspects of the process family (e.g., process family variations and commonalities, and process variant context). Despite these differences, configurable process models—irrespective of the approach used—allow eliminating redundancies by representing variant commonalities only once. Further, they allow fostering model reuse, i.e., variant particularities can be shared among multiple variants. After having created the configurable process model, it must be verified, i.e., it must be ensured that all derivable variants are syntactically correct. Additionally, the configurable process model must be validated, i.e., it must be ensured that the BPs are properly reflected by the model.

Configuration phase. First, based on the current context conditions, an *individualization process* is performed to derive a particular *process variant model* from the configurable process model [38]. Second, according to the chosen enactment system, the derived process variant model is transformed such that it can be deployed on this system [20].

Enactment phase. This phase encompasses the actual enactment time of process variant instances. This implies guaranteeing that process variants are executed according to the specification of the configured process variant model. Moreover, this phase may comprise configuration decisions that can only be made during enactment time. In this phase, monitoring techniques are indispensable to provide accurate information about the current context. In addition, in response to context changes during the execution of process variant instances [67], their dynamic re-configuration may become necessary to allow switching from one variant to another [2]. Unlike adhoc changes (i.e., unplanned changes), re-configuration options are already known and specified in the configurable process model during the *Analysis & Design phase.* However, like build-time configuration, syntactical as well as semantic correctness of process variants must be ensured after dynamic re-configuration [7, 26].

Diagnosis phase. In this phase, information gathered during the *Configuration* and *Enactment* phases (e.g., configuration settings made at configuration time and enactment time respectively) is analyzed to improve the process family and its implementation.

Evolution phase. Finally, during this phase new requirements as well as identified improvement opportunities lead to the evolution of the process family (e.g., by adding new family members). In this case, the configurable process model as well as the associated context model are evolved.

Along the description of these lifecycle phases, we have already presented some important concepts specific to BP variability. These concepts include *configurable process model, variation point, context, process family, process variant,* and *process variant instance.* Figure 5 depicts how these concepts relate to each other as well as the procedures that allow moving from the definition of a process family to an executable process variant instance.

3. Research method

The goal of this paper is to provide a framework for assessing the ability of a PAIS to effectively deal with BP variability. The framework shall not only allow PAISs engineers to assess the expressiveness of different variability proposals, by means of a set of language requirements, but also ensure that large process families can be appropriately modeled, verified, validated, configured, analyzed, refactored, and evolved through a set of variabiltiy support features. Moreover, the framework must ensure that process variants can be effectively executed, monitored, and dynamically re-configured if required.

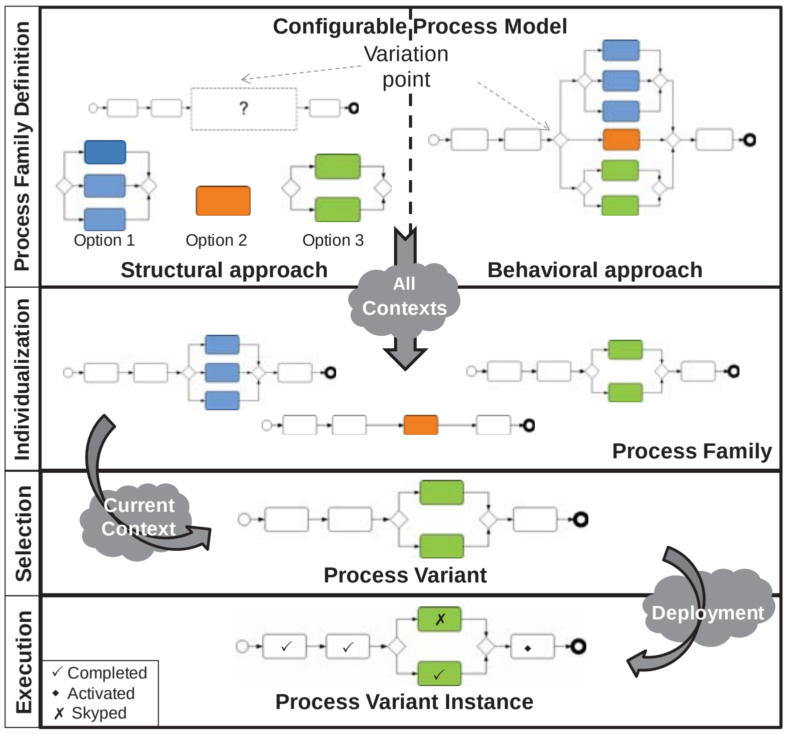


Figure 5: From Process Family Definition to Process Variant Enactment

This section presents the research method we employed for developing the BP variability framework.

3.1. Identification of Language Requirements

We first describe the selection criteria for our language requirements, the procedure we apply for their identification, and the data sources they are based on.

Selection Criteria. To ensure that the language requirements are not specific to a particular domain, case studies from different domains with varying level of complexity were selected. Requirements elicitation was not based on case studies solely, but complemented by a thorough literature review considering both existing research on software product lines and variability in the context of BPs [14].

Language Requirements Identification Procedure. Our goal was to identify a set of language requirements for effectively modeling large process families by explicitly capturing variability in a configurable process model. Since variability is not restricted to one perspective (e.g., behavioral), these requirements should be general enough to cover variability in any perspective [40]. Moreover, they should be independent of any specific proposal or language for modeling BP variability as well as constitute an extension of existing BP modeling languages (e.g., BPMN [12], EPC [1], or YAWL [4]). The derivation of language requirements was then performed as an iterative search process according to the information systems research framework by *Hevner et al.* [29]. Language requirements were iteratively refined after evaluating them based on case studies others than the ones used for gathering the initial requirements. This process finally led to the language requirements presented in Section 4.1.

Sources of Data and Data Collection. As source of the language requirements, we consider several case studies we performed in the context of various research projects. One of our data sources includes the airport check-in process (see Section 1). This process comprises hundreds of variants and shows variability in respect to all process perspectives (see Section 2.1). As another major data source, we use the process family for handling medical examinations described in [45], which encompasses 90 process variants. In addition, we consider the process for vehicle repair and maintenance with around 900 variants exist [27]. Additional case studies include processes related to goods and service provisioning in a public administration in Spain with 8 process variants [10]. As well as case studies described in literature (e.g., music and sound editing process [35]).

3.2. Identification of Variability Support Features

We now describe the selection criteria for variability support features, the procedure used for identifying them, and the data sources they are based on. While the language requirements cover the expressive power needed to explicitly represent BP variability, variability support features constitute typical functionalities offered by PAIS-enabling technologies to handle variability effectively throughout the entire BP lifecycle.

Selection Criteria. In our framework we only consider those features specific to the handling of process variants. As such, they are complementary to features commonly provided by existing BP repositories (e.g., similarity search [16, 50], model merging [39, 19], or reference architecture [75]) and process change frameworks (e.g., support for ad-hoc changes, support for process schema evolution, and process instance migration) [71].

Feature Identification Procedure. To identify fundamental variability support features we have analyzed proposals in respect to their functionality supporting BP variability. As a consequence, the proposed variability support features are of descriptive, rather than prescriptive nature and present an overview of the functionality provided by existing technologies.

Sources of Data and Data Collection. Existing proposals (e.g., [56, 63, 27, 8]), which have been specifically designed to support the management of BP variants, serve as data sources for identifying the variability support features. For describing respective PAISs, we conducted a thorough literature study.

4. Evaluation Framework

This section presents our evaluation framework for BP variability. On the one hand, it provides a set of language requirements needed to explicitly represent variability in a BP model and to assess the expressiveness of different proposals dealing with BP variability. On the other hand, it provides a set of variability support features required to enable variability along the entire BP lifecycle.

4.1. Language Requirements

As described in Section 2.2, a *process family* may be represented following either a *behavioral* or a *structural* approach. Based on the variability requirements identified in our case studies, in addition to the standard constructs, a process modeling language should provide the following language constructs to explicitly represent variability and to allow for the construction of configurable process models:

- **Language Requirement LR1 (Variation Point).** A *variation point* is defined as a precise position within a configurable process model that enables different choices depending on the current context or situation.

Example 7 (Variation point). Regarding the check-in process, depending on the type of ticket (e.g., economy or business class), the airline offers the possibility of changing the seat assignments.

- **Language Requirement LR2 (Alternative Process Element; APE).** An APE is defined as a particular option that may be instantiated at a specific variation point and may refer to any modeling element such as activities and their control flow, resources, data, events, or operations.

Example 8 (Alternative process elements). Several alternatives exist for the check-in process. For example, related to the behavioral perspective, business class passengers may alternatively accept the automatic seat assignment or change it at their convenience, i.e., *Change seat assignment* is an optional activity. Related to the resource perspective, there are different roles that may perform the *Print boarding card* activity: the passenger herself via the web system, the self-servicing machines, or the airline personnel at the economy or business class counters. Alternatives regarding the data perspective are related to different types of boarding cards (e.g., electronic versus paper-based). Alternatives regarding the time perspective can be found in the context of the start event of the check-in process, since the availability of the check-in process may vary from several days to a few hours before departure. Finally, alternatives regarding the operational perspective exist, since there are different implementations of the *Assign seat* activity (depending on whether the activity is performed at the self-servicing machine, online, or at the counter).

- **Language Requirement LR3 (Alternative process element context).** An alternative process element context is defined by a subset of process variables whose values make a particular APE to become instantiated for a variation point.

Example 9 (Alternative process element context). The *Pay excess fee* activity is only executed if the passenger carries an overweight luggage. If so, the overweight variable constitutes this context. Similarly, the *Assign seat* activity can only be performed if the customer holds a business class ticket.

- **Language Requirement LR4 (Alternative process element relationships).** An alternative process element relationship is defined as a constraint of use between two or more APEs. These constraints are defined based on semantic relationships to ensure the proper use of the involved APEs within a specific context.

Example 10 (Alternative process element relationship). An electronic boarding card can only be obtained when using the web system. This implies the definition of an inclusion relationship between the web system role and the electronic boarding card object as well as an exclusion relationships between the self-servicing machine, the economy/business class counters, and the electronic boarding card object.

- **Language Requirement LR5 (Variation point resolution time).** This requirement should allow modelers to distinguish between variation points whose resolution depends on the initial context (*configuration time*) or on the current context of a process instance (*enactment time*).

Example 11 (Variation point resolution time). The online check-in may be configured at configuration time by selecting the variants referring to the web system role. Nevertheless, other activities related to the luggage (i.e., *Pay excess fee*) will be only known when the passenger places the luggage at the desk scales. In this case, decisions regarding the APE to select are postponed until enactment time.

4.2. Variability Support Features

So far, we have introduced a set of fundamental language requirements for coping with BP variability needs. However, assessing expressiveness is not sufficient when comparing different proposals regarding their ability to deal with BP variability. In addition, variability support features need to be taken into account to evaluate the practical applicability of a particular proposal. This section introduces respective variability support features and discusses them along the BP lifecycle (see Section 2).

4.2.1. ***Phase I: Analysis & Design***

In the *Analysis & Design* phase an entire process family is designed, validated and verified in terms of a particular approach (i.e., behavioral or structural); this process family should be described considering the previously presented language requirements such that variants can be clearly identified within the configurable process model.

- **Feature F1.1 (Creating a configurable process model).** Sophisticated tool support is needed for creating a configurable process model. It requires considering all language requirements introduced in Section 4.1 and providing appropriate tool support for them. Since process models are better understood when presented graphically, in addition components like graphical editors, navigation facilities, or visualization features are needed to facilitate the creation of such models.

- **Feature F1.2 (Verifying a configurable process model and its related process family).** Efficient techniques to ensure that a configurable process model is correct should be provided, i.e., it must be guaranteed that only syntactically correct and sound (e.g., absence of deadlocks) process variants may be configured out of the configurable process model.

- **Feature F1.3 (Validating a configurable process model).** Techniques are needed to validate the *semantic correctness* of a configurable process model, i.e., avoiding incorrect domain representations of the process variants.

4.2.2. ***Phase II. Configuration***

The goal of the *Configuration* phase is to derive an executable process model (i.e., a particular member of the process family) from the configurable process model through *individualization* and to deploy it on the enactment system. In this phase, domain experts should be supported in specifying the concrete application context and in deriving process variant fitting to the given context. Finally, configuration settings should be logged to allow for later mining.

- **Feature F2.1 (Configuring a process variant out of a configurable process model).** Tools should provide sophisticated user interfaces and techniques to select the current application context and to derive one correct and valid process variant for it. On the one hand, automated support for checking the correctness and compliance of the configured process variant with the (alternative) process element relationships is required [6]. For example, *Requires/Excludes* constraints may enforce/exclude process fragments at a specific variation point. Users should be prevented (e.g., by informing or forbidding) from deriving invalid configuration settings. On the other hand, techniques enabling a high level of abstraction (e.g., form-based or context-driven configuration) should be provided for setting the current application context and hence for configuring a particular process variant. The resulted process variant should be graphically displayed to users.

- **Feature F2.2 (Logging configuration settings).** Configuration decisions should be logged to enable traceability and learning.

- **Feature F2.3 (Deploying a process variant).** Tool support is needed for transforming the obtained process variant into an executable model ready to be deployed. This support is dependent on the deployment technology and may involve, for instance, implementing the services that operationalize a process activity.

4.2.3. ***Phase III. Enactment***

The *Enactment* phase requires support for run-time configuration, monitoring, and dynamic re-configuration of process variants. Based on the process variants deployed on the enactment system, new process instances can be configured, executed, and monitored. Furthermore, dynamic reconfigurations (i.e., switching from one process variant to another) should be supported in a controlled and robust manner, e.g., enabling automatic reconfigurations when contextual changes occur. Note that this differs from ad-hoc changes known from adaptive PAISs [58], which constitute unplanned changes, instead of pre-specified ones.

- **Feature F3.1 (Configuring a process variant at enactment time).** Since certain configuration decisions can only be made during enactment time, tool support for configuring process variants at enactment time is needed.

Example 12 (Configuring a process variants at enactment time). Whether a passenger carries overweight luggage only becomes known once she arrives at the counter. Hence, the selection of the *Pay extra fee* activity can only be performed while enacting an instance of the process variant.

- **Feature F3.2 (Monitoring the execution of the instances of a process variant).** Tool support is needed for monitoring the context related to the execution of a collection of instances of a process variant (e.g., to discover contextual changes).
- **Feature F3.3 (Re-configuring an instance of a process variant during enactment time).** For a particular instance, it should be possible to dynamically switch its execution from the current process variant model to another one at enactment time. Respective re-configurations may become necessary when contextual changes occur during enactment time.

Example 13 (Re-configuring an instance of a process variant during enactment time). Regarding the check-in process, changes of the passenger status might require variant switches. Consider, for example, a passenger not having entered her frequent flying number when buying the ticket and therefore being initially treated as a regular customer. When providing the frequent flying number a switch to the appropriate process variant should be performed.

4.2.4. ***Phase IV. Diagnosis***

In the *Diagnosis* phase, support for the mining of configured process variants is required to learn from the configuration settings chosen at both design and enactment time.

- **Feature F4 (Optimizing process variant models).** Support for optimizing process variant models is needed [69]. This includes support for identifying process variants that were never configured or deployed

[46]. In turn, this might trigger the evolution of the configurable process model, but also allow discovering frequently applied configuration steps that might be lifted to the configurable process model to reduce future configuration effort [45].

4.2.5. ***Phase V. Evolution***

The *Evolution* phase is concerned with the evolution of a configurable process model to address changing requirements (e.g., need for the adoption of new variants different from the initial ones), increase its quality (e.g., improving model comprehensibility), or optimize its use (i.e., decreasing future configuration efforts).

- **Feature F5.1 (Evolving the schema of a configurable process model).** Configurable process models may have to be evolved to meet emerging needs and contextual changes. Among others, support is required for:
  - correctly evolving the schema of a configurable process model, e.g., by adding/removing variants,
  - maintaining the different schema versions of a configurable process model and to cope with their co-existences, and
  - propagating changes of the configurable process model to all configured process variants and—if desired—to running instances of the process variants as well.

Example 14 (Evolving the schema of a configurable process model). Regarding the check-in process, some airlines may want to provide new support for mobile phone boarding cards, which will require the addition of new variants.

- **Feature F5.2 (Refactoring a configurable process model).** Refactoring support is needed for improving the quality of a configurable process model by altering its schema, but without changing its behavior; i.e., the same process family can be produced on both the old configuration model and the refactored one [73].

5. Applying the Evaluation Framework in Practice: Existing Business Process Variability Proposals

In this section, we evaluate selected proposals regarding their support for BP variability. Section 5.1 describes our evaluation methodology. Evaluation results for language requirements are described in Section 5.2, while the evaluation of variability support features is discussed in Section 5.3. A summary of these results is provided in Section 5.4.

5.1. Evaluation Procedure

Definition of Evaluation Goal. The goal of our evaluation is to measure how well current proposals cope with BP variability regarding all BP perspectives and all phases of the BP lifecycle.

Selection of Evaluation Objects. As evaluation objects we choose PAIS-enabling technologies which have been developed with the aim to explicitly support BP variability. Based on a thorough literature review a list of candidate evaluation objects has been created. To be included in our evaluation we require the existence of a proof-of-concept implementation. As a result, we have included the following five proposals: PESOA [56], C-EPC [63], Provop [27], Rule representation and processing [34], and Worklets [8].

Definition of Evaluation Criteria and Metrics. As evaluation criteria we consider the five language requirements (see Section 4.1) and the twelve variability support features (see Section 4.2) defined by our evaluation framework. We measure the ability of the selected PAISs to deal with variability as the level of support for the described evaluation criteria. For each evaluation criterion we differentiate between *no support* [-], *partial support* [+/-], and *full support* [+]. However, for Language Requirement 2 (i.e., alternative process elements), we use letters to indicate whether the proposal supports the behavioral (B), functional (F), organizational (O), temporal (T), and operational (Op) perspective (e.g., [F, B]).

Analyzing the Evaluation Objects along the Evaluation Criteria. Our evaluation is based on a comprehensive literature study. Moreover, the check-in example is modeled for every proposal.

5.2. Evaluation Results for Language Requirements

For each of the five proposals, we provide evaluation results for the language requirements defined in Section 4.1.

5.2.1. ***PESOA***

PESOA [56] provides a proposal for the development and customization of families of process-oriented software. According to PESOA, a configurable process model is represented by one artifact, which contains variation points (LR1 [+]). They are described by attaching annotations to the activities for which variability may occur. For this, strategies like encapsulation, inheritance, design patterns, and extension points are used. These strategies only refer to the behavioral and functional perspectives (LR2 [F, B]). The context of the control flow alternatives is partially defined since it is specified in terms of features attached to the activities instead of process variables (LR3 [+/-]). Relationships between the alternatives of different variation points are not considered (LR4 [-]) and it is not possible to distinguish whether a variation point is resolved at design time or at enactment time (LR5 [-]).

Example 15 (Check-in process modeled in PESOA). Figure 6 depicts a configurable process model dealing with the check-in process developed with PESOA. It includes the related feature model, which contains the features associated to the alternatives. Commonalities are highlighted in grey, while variation points are shown in white. For example, Figure 6 comprises three optional activities (e.g., *Change seat assignment*, *Provide information about accommodation*, and *Pay extra fee*), which are modeled as extension points. Moreover, an inheritance strategy is defined to represent the alternatives for activity *Assign seat.* Attached to the definition of each alternative, context conditions (i.e., features) can be found that make the alternative be selected. For example, activity *Change seat assignment* is only available for business class passengers. Since variability regarding organizational (i.e., resources) and informational (i.e., data flow) perspectives is not supported, their variations are not shown in Figure 6. Process variants can be derived by selecting features in the associated features model. For example, through the colored features, *Variant 1* from Figure 1 is obtained (see the bottom of Figure 6).

5.2.2. ***Configurable Event-driven Process Chain (C-EPC)***

Configurable EPC [63, 21] is an extension of EPC (Event-driven Process Chain) to model variability in reference process models. A configurable process model is represented by a single model gathering the whole process

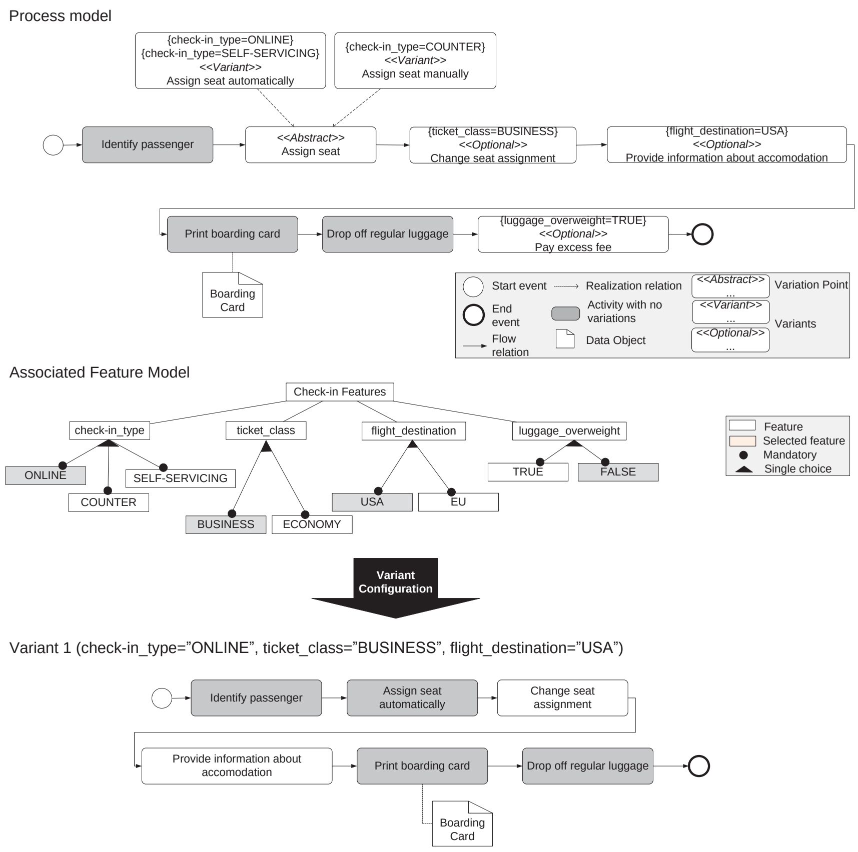


Figure 6: Configurable Process Model for the Check-in Process developed with PESOA

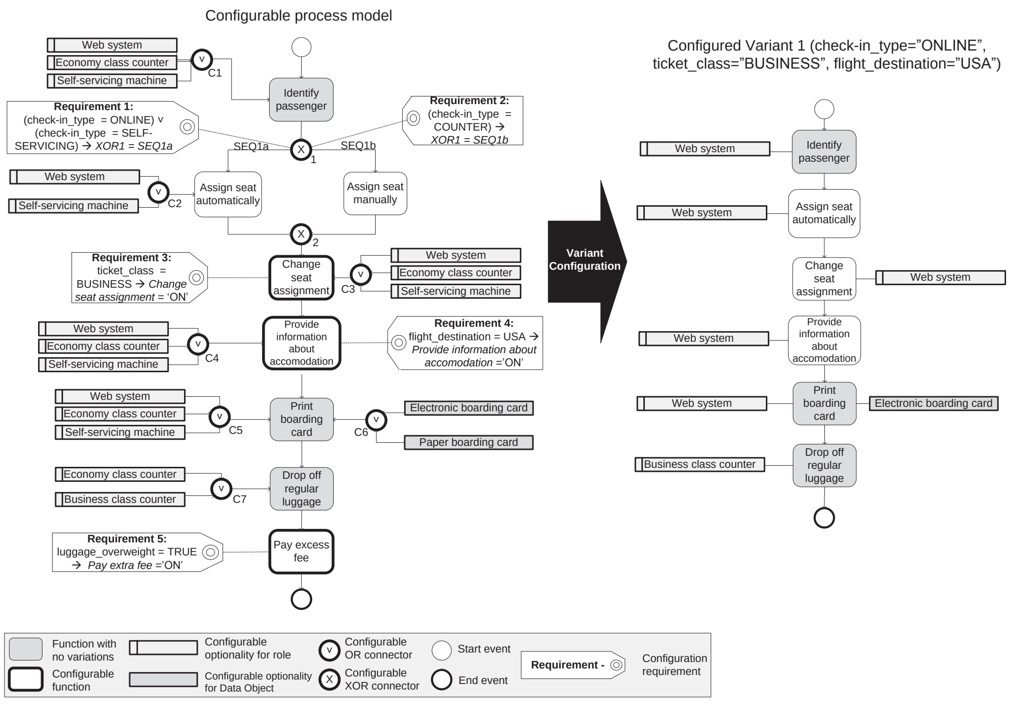
family. The *configurable functions* (i.e., activities) and *connectors* allow identifying the variation points in the model (LR1 [+]). The support originally provided to the functional and behavioral perspectives was extended by [35] to the organizational and informational ones (LR2 [F, B, O, I]). The conditions that instantiate an alternative process element are included in the *configuration requirements* and *guidelines* constructs (LR3 [+]). These conditions are defined as context variables that determine whether or not the requirement shall be applied. Using C-EPCs, it is not possible to explicitly define relationships between alternative process elements. Instead, *configuration requirements* define relationships between functions and connectors along the entire model. The fact that these requirements are scattered throughout the model hinders comprehensibility for modelers (LR4 [+/-]). The C-EPC proposal has been designed to support configuration prior to model deployment, but it does not consider dynamic run-time configuration. It is hence not possible to express when variation points are resolved (LR5 [-]).

Example 16 (Check-in process modeled with C-EPC). The left side of Figure 7 illustrates a configurable process model in C-EPC notation. This figure shows several configurable functions (e.g., *Change seat assignment, Provide information about accommodation*, and *Pay extra fee*), which all constitute optional activities (variability regarding the functional perspective). Moreover, the model comprises one configurable XOR connector, which relates to the behavioral perspective and whose configuration involves selecting just one of the alternative branches. As illustrated in Figure 7, *configuration requirements* are annotated to configurable nodes, defining the context conditions. For example, *Requirement 1* states that, when the check-in type is online or at the self-service machine, activity *Assign seat* is selected automatically; otherwise the activity is perform manually (see *Requirement 2*). Similarly, context conditions for the configurable nodes are defined (see *Requirements 3 to 5*). The model also comprises alternatives regarding the organizational and informational perspective. These alternatives are defined by configurable OR connectors associated to the roles and objects. For example, the *Identify passenger* activity can be performed by the web system, the airline personnel, or the self-servicing machine. In the same line, the boarding card can either be electronic or paper-based. In turn, the right part of the figure shows *Variant 1* from Figure 1 after its configuration. This configuration is performed selecting the correct alternative for each configurable node.

Aligned with C-EPC, other configuring proposals such as C-YAWL (based on hiding and blocking) [5] and Feature-EPC (based on feature models) are worth mentioning [70].

5.2.3. ***Provop***

Provop [27] is a structural proposal for managing large collections of process variants during the BP lifecycle which includes design and enactment



Configurable process model
Configured Variant 1 (check-in_type="ONLINE", ticket_class="BUSINESS", flight_destination="USA")
Web system
Economy class counter
Self-servicing machine
C1
Identify passenger
Requirement 1: (check-in_type = ONLINE) v (check-in_type = SELF-SERVICING) → XOR1 = SEQ1a
Requirement 2: (check-in_type = COUNTER) → XOR1 = SEQ1b
SEQ1a
SEQ1b
Assign seat automatically
Assign seat manually
C2
Requirement 3: ticket_class = BUSINESS → Change seat assignment = 'ON'
Change seat assignment
C3
Provide information about accomodation
Requirement 4: flight_destination = USA → Provide information about accomodation ='ON'
C4
Print boarding card
C5
C6
Electronic boarding card
Paper boarding card
Drop off regular luggage
Business class counter
C7
Requirement 5: luggage_overweight = TRUE → Pay extra fee ='ON'
Pay excess fee
Variant Configuration
Function with no variations
Configurable function
Configurable optionality for role
Configurable optionality for Data Object
Configurable OR connector
Configurable XOR connector
Start event
End event
Requirement -
Configuration requirement

Figure 7: Configurable Process Model for the Check-in Process developed with C-EPC

time activities. A *configurable process model* consists of several artifacts like a *base process* and a set of *variant-specific adjustments*, which can be expressed based on well-defined, high-level change operations (INSERT, DELETE, and MOVE process fragments and MODIFY process attributes [61]). Furthermore, multiple *change operations* may be grouped into so-called *change options* to allow for more complex adjustments. Thus, a specific process variant is determined by applying one or more of these *change options* to the base process. As opposed to C-EPCs, this means that the base process does not need to capture all possible behavior. This is only one alternative for designing this configurable base process. Others are to capture only common behavior or most frequent behavior in the base process, while expressing variability-specific adjustments with *change options*.

Variation points are represented by means of *adjustment points* (LR1 [+]). Alternative process elements can only be defined when talking about control flow options (i.e., behavioral and functional perspective) (LR2 [F, B]). The context conditions that make an alternative process element be instantiated are defined in terms of the context variables comprising a name, description, value range, and mode (static or dynamic depending on whether their value may change during enactment time or not) (LR3 [+]). It is possible to define relationships such as implication, mutual exclusion, application order, hierarchy, and at-most-n-out-of-m-options between the alternative process elements (LR4 [+]) [23]. Since dynamic enactment time configuration is not supported and therefore all configurations are done at design time, it is not possible to explicitly represent the variation point resolution time (LR5 [-]).

Example 17 (Check-in process modeled with Provop). Figure 8 illustrates the Provop proposal for dealing with the check-in process. At the top part of the figure, the base process model from which the process variants can be derived is depicted. Note that this base process model contains several *adjustment points* (delimited by black diamonds) to which *change options* and their change operations may refer. The *change options* applicable to this base process model are shown at the bottom of the figure. For example, *Option 1* specifies that, if the check-in is done at the counter, activity *Assign seat* is automatically replaced (by performing *delete* and *insert change operations*) by a manual seat assignment. *Variant 1* of Figure 1 is obtained after applying *Option 4*.

5.2.4. ***Rule Representation and Processing***

Rule representation and processing [34] is based on the idea of configuring variants through a set of business rules. These rules are applied to a generic template, which constitutes a default configuration that is modified according to context changes. However, variation points cannot be identified in the template since they are not marked in a specific manner to distinguish the exact places being subject to variations (LR1 [-]). Alternatives are described through a set of business rules, whose application leads to the execution of a set of change operations (e.g., insert or delete activity). These change operations, when applied to the basic *process template*, lead to a specific process variant. In addition to control flow, alternatives may also refer to the organizational and informational perspectives (LR2 [F, B, O, I]). Using this proposal, it is easily possible to identify the context related to an alternative process variant (LR3 [+]); the context is represented by context variables.

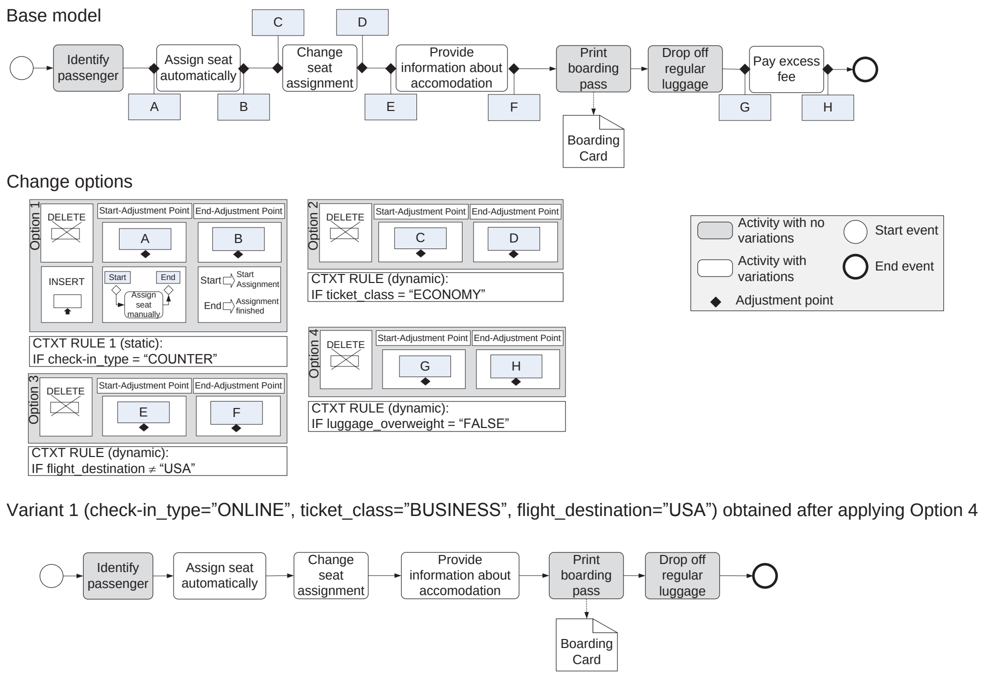
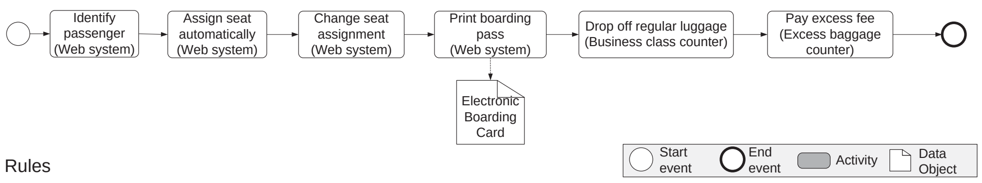

Base model
Identify passenger
Assign seat automatically
Change seat assignment
Provide information about accomodation
Print boarding pass
Drop off regular luggage
Pay excess fee
Boarding Card
A
B
C
D
E
F
G
H
Change options
Option 1
DELETE
INSERT
Start-Adjustment Point
End-Adjustment Point
Start
End
Assign seat manually
Start Assignment
Assignment finished
CTXT RULE 1 (static): IF check-in_type = "COUNTER"
Option 2
CTXT RULE (dynamic): IF ticket_class = "ECONOMY"
Option 3
CTXT RULE (dynamic): IF flight_destination ≠ "USA"
Option 4
CTXT RULE (dynamic): IF luggage_overweight = "FALSE"
Activity with no variations
Activity with variations
Adjustment point
Start event
End event
Variant 1 (check-in_type="ONLINE", ticket_class="BUSINESS", flight_destination="USA") obtained after applying Option 4

Figure 8: Configurable Process Model for the Check-in Process developed with Provop

It is not possible to define relationships between the alternative process elements (LR4 [-]). Moreover, it is not possible to distinguish whether variation points are resolved or alternatives are selected at design time or at enactment time (LR5 [-]).

Example 18 (Check-in process modeled with Rule representation and processing). Figure 9 shows a configurable process model dealing with the check-in process modeled using the rules representation and processing proposal. The figure illustrates the most common activities (i.e., basic template), the roles (in brackets) that perform them, the control flow relationships, and the data objects for the check-in process. The figure also shows an example rule set that may be applied to the template. For example, *Rule R1* specifies that for economy class passengers, activity *Change seat assignment* is deleted. In turn, *Rule 4* specifies that if the check-in is done at the counter, *Assign seat automatically* is replaced by *Assign seat manually*.

Basic template


Identify passenger (Web system)
Assign seat automatically (Web system)
Change seat assignment (Web system)
Print boarding pass (Web system)
Drop off regular luggage (Business class counter)
Pay excess fee (Excess baggage counter)
Electronic Boarding Card
Start event
End event
Activity
Data Object

Rules

Control flow related
R1: ticket.class= “ECONOMY”, → delete(T3)
R2: flight_destination = USA”, → insert(“Provide information about accomodation”, Safter, T3)
R3: lugagge.overweight = “FALSE”, → delete(T6)

Resource related
R5: check-in.type = “COUNTER”, → role(T1, economy class counter)
R6: check-in.type = “COUNTER”, → role(T2, economy class counter)
R7: check-in.type = “COUNTER”, → role(T4, economy class counter)
R8: check-in.type = “COUNTER”, → role(T5, economy class counter)

Data related
R9: check-in.type = “COUNTER”, → data-out(T4, “Paper boarding card”)
R10: check-in.type = “SELF-SERVICING MACHINE”, → data-out(T4, “Paper boarding card”)

Complex Rules
R4: check-in.type = “COUNTER”, → delete(T2) AND insert(“Assign seat manually”, Safter, T1)
R11: check-in.type = “SELF-SERVICING MACHINE”, → role(T1, “Self-servicing machine”) AND role(T2, ”Self-servicing machine”) AND role(T4, “Self-servicing machine”) AND role(T5, “Self-economy class counter”)

Variant 1 (check-in_type=”ONLINE”, ticket_class=”BUSINESS”, flight_destination=”USA”) obtained after applying R3

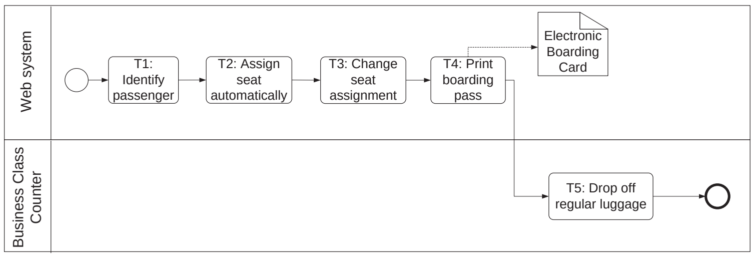


Figure 9: Configurable Process Model for the Check-in Process developed with the Rule Representation Proposal

5.2.5. ***Worklets***

Worklets [8] is a proposal for handling exceptions at enactment time through *dynamic reconfiguration*. A *worklet* is defined as “a small, complete and re-usable workflow specification which handles one specific task in a composite parent process when an exception in it occurs”. Even though this proposal was not conceived to provide BP variability support as such, its re-configuration support justifies its inclusion in our evaluation.

Variation points are not explicitly marked when using the Worklets proposal. However, every activity from the base process model (i.e., default process variant model) that is associated with a repertoire of worklets can be considered as a variation point (LR1 [+/-]). These repertoires comprise the worklets for the associated variation point. Alternative process elements can be single activities, but also entire sub-processes. Therefore, variability therefore relates to both the functional and behavioral perspective (LR2 [F, B]). The conditions that determine the selection of a particular alternative during enactment time are represented by means of Ripple Down Rule (RDR) trees that are associated to activities from the base model. The evaluation of these conditions will determine the selection of the appropriate alternative based on the current context (LR3 [+]). Nevertheless, it is not possible to define relationships between alternative process elements (LR4 [-]). Since the Worklets proposal focuses on handling exceptions at enactment time through dynamic re-configuration, all configuration decisions are made during enactment time. Nevertheless, the model does not contain any particular mark to make this situation explicit (LR5 [-]).

Example 19 (Check-in process modeled with Worklets). Figure 10 illustrates the Worklets proposal when dealing with the check-in process. It depicts the default check-in process (defined in YAWL) as well as the repertoire of alternatives for the activities (i.e., *Assign seat, Provide additional information, Change seat assignment*, and *Pay extra fee*). In addition, the RDR trees are depicted to determine the context for selecting the appropriate alternative. For this, the RDR tree is evaluated starting with the root node, which always evaluates to true. As the next step, context condition *chek-in_type=COUNTER*, for instance, is evaluated. If this condition is satisfied, the true branch is taken and the worklet *Assign seat automatically* is selected. Otherwise, *Assign seat manually* worklet is executed.

5.3. Evaluation Results for Variability Support Features

In the following, for each selected variability proposal, we provide evaluation results for the variability support features defined in Section 4.2.

5.3.1. ***PESOA***

PESOA has been realized as Eclipse plug-in that supports the creation of a *configurable process model* (F1.1 [+]). Concretely, the graphical editor provides support for representing variation points including alternative process elements, but also support for describing the context model. This

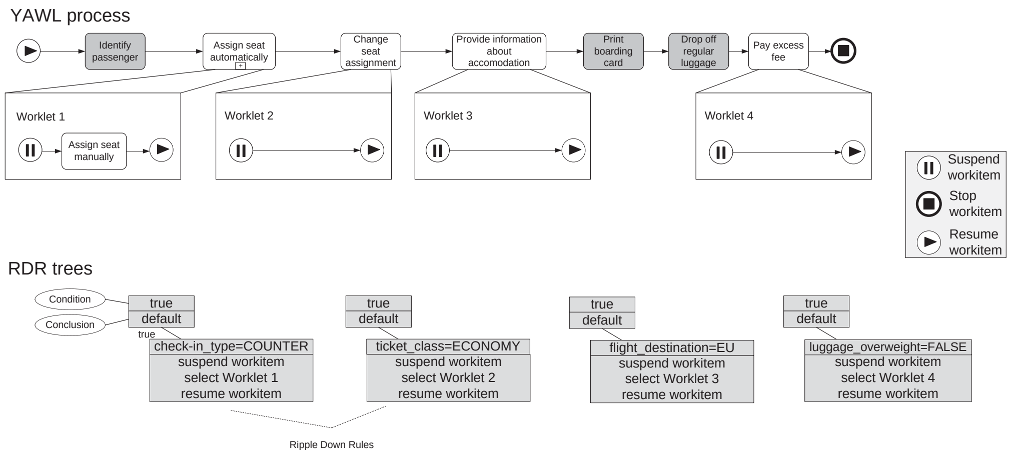


Figure 10: Configurable Process Model for the Check-in Process developed with Worklets

is accomplished by means of feature diagrams, i.e., each process variant is tagged with features, determining the conditions under which this variant is valid. However, neither verification nor validation support is provided (F1.2 [-], F1.3 [-]). The configuration of process models is supported by feature diagrams, i.e., for each disabled feature, corresponding variable parts are removed from the process variant model. The usage of feature diagrams allows for configuring process variants at a high level of abstraction. In addition, resulting process variants can be displayed to users. However, semantic and contextual constraints between variable parts are not considered in the configuration phase. As a consequence, no support is available to inform users about constraint violations (F2.1 [-]). PESOA focuses on the modeling as well as configuration of BP variants. Therefore, features related to the deployment, execution, mining, and evolution of process variants are not taken into account (F2.2 - F5.2 [-]).

5.3.2. ***Configurable Event-driven Process Chain (C-EPC)***

The C-EPC proposal has been implemented as a toolset named *Synergia* [47]. It supports the creation of a configurable process model using a graphical editor. Moreover, it allows defining the context of a configurable process model using *configuration requirements* and *guidelines* (F1.1 [+]). Moreover, the Synergia toolset implements a mapper tool that can be used to verify the construction of the configurable process model [36] (F1.2 [+]). In addition, it is possible to validate configured process models using C-EPC Validator [48] (F1.3 [+]).

The configuration of process variants through domain experts is supported using a questionnaire-based approach, implemented by the *Quaestio* tool [36]. By answering a set of questions, domain experts are assisted and guided when configuring the configurable process model [37]. Questions are asked in such a way that semantic and contextual constraints are always respected and no inconsistent choices can be made. After completing the configuration, *Synergia* allows visualizing the process variant model obtained (F2.1 [+]). Nevertheless, *Synergia* does not consider configuration logs (F2.2 [-]). In combination with YAWL and its configurable extension (C-YAWL), it is possible to transform a configured C-EPC model into a deployable representation, which can then be executed by a YAWL engine (F2.3 [+]).

Run-time configuration, monitoring, and dynamic re-configuration of a process variant are not supported; once the configuration of an C-EPC model is completed, the deployed process is fixed and cannot be changed anymore (F3.1 [-], F3.2 [-], and F3.3 [-]). In addition, support for optimizing process variant models is not provided (F4 [-]).

Evolution of configurable process models is partially supported (F5.1 [+/-]). While configurable process models can be evolved by adding/removing configurable elements, changing the context, or revising the configuration guidelines, no support is provided for handling different versions or propagating model changes to process variants. Refactoring support is not considered by *Synergia* (F5.2 [-]).

5.3.3. ***Provop***

The Provop proposal has been implemented as a *proof-of-concept* prototype based on the *ARIS Business Architect tool* [57]. The creation of a configurable process model is supported by a graphical editor, which allows creating a base model and specifying the options that will configure it, except for distinguishing when a variation point is solved (i.e., design or enactment time). The prototype also supports the definition of a context model through a set of context variables, which represent one specific dimension of the process context. Moreover, relationships between variants can be defined (F1.1 [+]). The created process models can be verified to ensure their correctness (F1.2 [+]), but no validation support is provided (F1.3 [-]).

The configuration of process variants is also supported by the prototype (F2.1 [+]) [57]. For this, depending on the actual context, respective change options are applied to the base process model. The *proof-of-concept* prototype checks whether the defined options violate any constraint defined on the total set of *change options* available. If it detects such constraint violation, it notifies the process engineer accordingly. After selecting the *change options*, they are applied to the configurable process model and the result is shown. The *proof-of-concept* prototype does not maintain configuration logs (F2.2 [-]). Nevertheless, it provides techniques to transform the obtained configuration into an executable model (F2.3 [+]).

Dynamic enactment time configuration is not supported by Provop (F3.1 [-]). On the contrary, dynamic re-configuration of a process variant model based on context changes is supported by including variant branchings in the process model and encapsulating the adjustments of single *change options* within these variant branches (F3.2 [+] and F3.3 [+]). The split condition at the variant branching corresponds to the context rule of the *change option*. Whenever process execution reaches a variant branch, the current context is evaluated: if the split condition evaluates to true the variant branch will be executed, which means that the *change options* will be applied. Otherwise, the variant branch will be skipped and therefore all adjustments of the *change options* will be ignored. In case, constraints between different options exist (i.e., two options being mutually exclusive), Provop ensures that these constraints are enforced when dynamically re-configuring processes during enactment time [27].

No support for optimizing process variant models is provided in the *proof-of-concept* prototype (F4 [-]). Evolution of configurable process models, in turn, is partially supported (F5.1 [+/-]). While configurable process models can be evolved, no support is provided for the handling of different versions or the propagation of model changes to process variants. Finally, refactoring support is provided (F5.2 [+]) [25].

5.3.4. ***Rule Representation and Processing***

The *Rules representation and processing* proposal has been partially implemented as a *proof-of-concept* prototype. It is based on Java and uses the Drools Rule Language (DRL) to represent the rules. The open source rule engine *Drools-expert* [49] is used to process rules and resolve conflicts.

It is possible to create a configurable process model using the *proof-of-concept* prototype (F1.1 [+]). It allows defining a new template as well as the rules that configure the template. The created process models can be verified to ensure their correctness (F1.2 [+]). In particular, the *validity checking* component ensures that the resulting change operations can be applied to the process template, while ensuring syntactical correctness. No validation support is provided (F1.3 [-]).

Configuration of process variants is done by applying the rules specified in the configurable process model (F2.1 [+]). When the *proof-of-concept* prototype detects that a constraint has been violated, it notifies the process engineer about the violation. After configuration, the resulting variant is shown to the modeler. The *proof-of-concept* prototype provides techniques to support the logging of configuration settings (F2.2 [+]). In addition, the prototype allows transforming the configured model obtained into an executable representation (i.e., BPEL), which then can be deployed to the enactment time system (F2.3 [+]).

Neither run-time configuration (F3.1 [-]) nor re-configuration of a process variant is supported (F3.2 [-] and 3.3 [-]). In the same line, optimization of process variant model is not supported (F4 [-]). Evolution of configurable process models is partially supported (F5.1 [+/-]). While configurable process models can be evolved by adding new rules and changes can be automatically propagated to running process instances, no support for handling different versions is provided. Regarding change propagation, running instances may have to be restarted, continued without change, or migrated to new variants. Finally, no refactoring support is provided (F5.2 [-]).

5.3.5. ***Worklets***

The Worklet proposal has been implemented as a YAWL Custom Service. The creation of configurable process models is possible (F1.1 [+]). The context for selecting a specific alternative can be described by means of the RDR tree (see Section 5.2). Neither verification nor validation support is provided (F1.2 [-] and F1.3 [-]).

Since the Worklets proposal has been developed for enabling run-time flexibility, no support regarding the configuration phase is provided (i.e., all configuration decisions are dynamically made during enactment time) (F2.1 [-] and F2.2 [-]). Process models comprising Worklets (e.g., models specified in terms of YAWL) are executable and can be deployed to a process engine for execution (F2.3 [+]).

Dynamic run-time configuration is not supported by Worklets (F3.1 [-]). On the contrary, using the Worklets proposal, context changes may be considered and process variants be dynamically re-configured (F3.2 [+] and F3.3 [+]). Support for optimizing process variant models is, in turn, not provided (Feature F4 [-]). Whereas evolution of configurable process models is partially supported (F5.1 [+/-]). While configurable process models can be evolved by adding/removing worklets or by adapting the RDR tree, no support is provided for handling different versions or the propagating configurable model changes to process variants. Finally, refactoring support is not provided (F5.2 [-]).

5.4. Summary of Results

Table 1 summarizes the evaluation regarding the language requirements and their support by each of the proposals. As shown, none of the evaluated proposals accomplishes the whole set of requirements. Most of them allow explicitly defining variation points (LR1) as well as the alternatives that may be instantiated at these points (LR2). Nevertheless, these alternatives mainly refer to some perspectives such as the behavioral or functional perspectives, while neglecting other perspectives. For example, none of the proposals supports the definition of alternatives regarding the temporal perspective (i.e., process events). Every proposal supports the definition of the context conditions for which the alternatives shall be used (LR3). Nevertheless, the definition of relationships between alternatives is mostly neglected (LR4). In addition, none of the proposals provides mechanisms to differentiate the variation point resolution time (LR5) that defines the decisions to be taken before model deployment.

Table 2 summarizes the evaluation regarding variability support features. While the modeling of configurable process models (*Phase I*) and the configuration of process variants (*Phase II*) are supported by most of the proposals (F1.1 - F2.3), comprehensive lifecycle support for BP variability has been missing so far. An area not well covered so far is the support of *Phase III (Enactment)* (F3.1 - F3.3), both in terms of *enactment time configuration*, and *dynamic re-configuration* of process variants. In addition, our analysis shows that features related to *Phase IV (Diagnosis)* are not well supported by any of the evaluated proposals (F4). However, there exist techniques (e.g., [44]), such as the one presented by *Günther et al.* [22], which can be applied together with the evaluated proposals. Nevertheless, since this combination was not conceived originally by the tools, they have not been included for this evaluation. Finally, regarding the evolution of configurable process models, existing proposals provide basic support, but lack advanced techniques for the controlled change propagation in the configurable process model to

	PESOA	C-EPC	Provop	Rule	Worklets
LR1: Variation Point	+	+	+	-	+/-
LR2: Alternative process elements	F, B	F, B, O, I	F, B	F, B, O, I	F, B
LR3: Alternative process element context	+/-	+	+	+	+
LR4: Alternative process relationships	-	+/-	+	-	-
LR5: Variation point resolution time	-	-	-	-	-

Table 1: Evaluation of the tools regarding their support for language requirements

existing process variants (F5.1). In addition, refactoring support is mostly missing (F5.2).

6. Summary and outlook

This paper has presented a framework for evaluating BP variability support as it can be found in existing proposals. Based on an in-depth analysis of several large process model repositories from various domains, the framework defines both a set of language requirements and variability support features needed for properly dealing with BP variability. While language requirements allow assessing the expressiveness of a BP modeling languages in terms of variability support, variability support features are needed to ensure the practical applicability of these language features and to support the management of variability throughout the entire BP lifecycle.

The paper has applied the proposed framework to several proposals developed in the BPM field, which aim at explicitly supporting BP variability. Our evaluation shows that currently none of these proposals provides a holistic approach for supporting BP variability. In particular, support for handling perspectives other than control flow is currently limited. Another aspect, which has obtained little attention so far, is the support for dynamic configuration and automatic re-configuration of process variants.

	PESOA	C-EPC	Provop	Rule	Worklets
F1.1 Creating a configurable process model	+	+	+	+	+
F1.2 Verifying a configurable process model	-	+	+	+	-
F1.3 Validating a configurable process model	-	+	-	-	-
F2.1 Configuring a process variant	-	+	+	+	-
F2.2 Logging configuration settings	-	-	-	+	-
F2.3 Deploying a process variant	-	+	+	+	+
F3.1 Configuring a process variant at enactment time	-	-	-	-	-
F3.2 Monitoring the execution of the instances of a process variants	-	-	+	-	+
F3.3 Re-configuring an instance of a process variant during enactment time	-	-	+	-	+
F4 Optimizing process variant models	-	-	-	-	-
F5.1 Evolving a configurable process model	-	+/-	+/-	+/-	+/-
F5.2 Refactoring a configurable process model	-	-	+	-	-

Table 2: Evaluation of the tools regarding variability support features

This paper has provided an overview of the existing support regarding BP variability as well as existing research gaps. It further has provided means to systematically compare different proposals for BP variability. In this vein, the framework not only helps to understand BP variability along the BP lifecycle, but also supports PAISs engineers in deciding, based on the type of variability required, which proposal fits best their needs.

Similar to process patterns [3] or change patterns [71], the proposed framework provides a qualitative perspective on different variability proposals. Our future work will complement this qualitative perspective with a series of empirical evaluations regarding non-functional requirements such as understandability, maintainability, correctness, traceability, and scalability. By considering these advanced of requirements, we can additionally assess the quality of the process family provided by the existing proposals from a quantitative perspective. For conducting the experiments, we plan to use the *Cheetah Experimental Platform* [55] which not only allows testing the outcome of process modeling (i.e., the created process models), but also the *process of process modeling* itself.

Acknowledgements
This work has been developed with the support of MICINN under the project EVERYWARE TIN2010-18011.

References

[1] van der Aalst, W.M.P.: "Formalization and verification of event-driven process chains". Information and Software Technology Journal 41(10), pp. 639–650 (1999).

[2] van der Aalst, W.M.P., Basten, T.: "Inheritance of workflows: An approach to tackling problems related to change". Theoretical Computer Science 270(1-2), pp. 125–203 (2002).

[3] van der Aalst, W.M.P., ter Hofstede, A., Barros, B.: "Workflow Patterns". Distributed and Parallel Databases 14(1), pp. 5–51 (2003).

[4] van der Aalst, W.M.P., Hofstede, A.H.M.: "YAWL: Yet Another Workflow Language". Information Systems 30(4), pp. 245–275 (2005).

[5] van der Aalst, W.M.P., Dreiling, A., Gottschalk, F., Rosemann, M., Jansen-Vullers, M.H.: "Configurable process models as a basis for reference modeling". In: Business Process Management Workshops, vol. 3812, pp. 512–518 (2005).

[6] van der Aalst, W.M.P., Dumas, M., Gottschalk, F., ter Hofstede, A.H.M., La Rosa, M., Mendling, J.: "Preserving correctness during business process model configuration". Formal Aspects of Computing 22(3–4), pp. 459–482 (2010).

[7] van der Aalst, W.M.P., Lohmann, N., La Rosa, M., Xu, J.: "Correctness ensuring process configuration: An approach based on partner synthesis". 8th International Conference on Business Process Management, LNCS vol. 6336, pp. 95–111. Springer, (2010).

[8] Adams, M.J.: "Facilitating dynamic flexibility and exception handling for workflows" PhD thesis, Faculty of Information Technology. Queensland University of Technology, (2007).

[9] Aguilar-Savén, S.: "Business process modelling: review and framework". International Journal Production Economics 90(2), pp. 129–149 (2004).

[10] Ayora, C., Torres, V., Pelechano, V.: "Minor contract process for regional public administration: a business process variability example". Technical report. ProS-TR-2012-01, PROS-UPV (2012).

[11] Becker, J., Delfmann, P., Knackstedt, R.: "Adaptive reference modeling. Integrating configurative and generic adaptation techniques for information models". Reference Modeling. Efficient Information System Design through Reuse of Information Models, Berlin. Physica-Verlag HD Publisher, pp. 23–49 (2007).

[12] Business Process Model and Notation, version 2.0 . Object Management Group (OMG). `http://www.org./spec/BPMN/2.0/` Accessed: April 2012.

[13] Bridgeland, M., Zahavi, R.: "Business modeling: a practical guide to realizing business value". ISBN: 978-0123741516. Morgan Kaufmann/Elsevier Publishers, (2008).

[14] Clemens, P., Northrop, L.: "Software product lines: practices and patterns". SEI Series in Software Engineering, ISBN:0-201-70332-7. Addison-Wesley Publisher, (2001).

[15] Curtis, B., Kellner, M., Over, J.: "Process modeling". Communication of the ACM 35(9), pp. 75–90 (1992).

[16] Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: "Similarity of business process models: metrics and evaluation". Information Systems 36(2), pp. 498–516 (2011).

[17] Dijkman, R., La Rosa, M. and Reijers H.A., "Managing large collections of business process models - Current techniques and challenges", In Computers in Industry 63(2), pp. 91–97 (2012).

[18] Dumas, M., van der Aalst, W.M.P. and Hofstede, A.H.M. ter Eds.: "Process-aware information systems: bridging people and software through process technology". ISBN: 978-0471663065. Hoboken, New Jersey: John Wiley & Sons Publishers, (2005).

[19] Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H.: "Merging event-driven process chains". OTM Conferences, LNCS vol. 5331(1), pp. 418–426 (2008).

[20] Gottschalk, F., van der Aalst,W.M.P., Jansen-Vullers, M.H., La Rosa, M.: "Configurable workflow models". International Journal Cooperative Information System 17(2), pp. 177–221 (2008).

[21] Gottschalk, F.: "Configurable process models". Ph.D. thesis, Eindhoven University of Technology, The Netherlands (2009).

[22] Günther, C.W., Rinderle-Ma, S., Reichert, M., van der Aalst, W.M.P., Recker, J.: "Using process mining to learn form process changes in evolutionary systems". International Journal of Business Process Integration and Management, Special Issue on Business Process Flexibility 3(1), pp. 61–78 (2008).

[23] Hallerbach, A., Bauer, T., Reichert, M.: "Managing process variants in the process lifecycle". 10th International Conference on Enterprise Information Systems 3(2), pp. 154–161 (2008).

[24] Hallerbach, A., Bauer, T., Reichert, M.: "Context-based configuration of process variants". Technologies for Context-Aware Business Process Management, pp. 31–40 (2008).

[25] Hallerbach, A.: "Management von Prozessvarianten". PhD Thesis. University of Ulm, Germany (2009).

[26] Hallerbach, A., Bauer, T., Reichert, M.: "Guaranteeing soundness of configurable process variants in Provop". 11th IEEE Conference on Commerce and Enterprise Computing , pp. 98–105. IEEE Computer Society Press (2009).

[27] Hallerbach, A., Bauer, T., Reichert, M.: "Capturing variability in business process models: the Provop approach". Journal of Software Maintenance 22(6–7), pp. 519–546 (2010).

[28] Hallerbach, A., Bauer, T., Reichert, M.: "Configuration and management of process variants". International Handbook on Business Process Management, Springer Publisher, pp. 237–255 (2010).

[29] Hevner, A.R., March, S.T., Park, J., Ram, S.: "Design science in information systems research". MIS Quartely Journal 28(1), pp. 75–105 (2004).

[30] Hochstein, A., Zarnekow, R., Brenner, W.: "ITIL as common practice reference model for IT service management: Formal assessment and implications for practice". IEEE International Conference on e-Technology, e-Commerce, and e-Services, pp. 704–710 (2005).

[31] Jablonski, S., Bussler, C.: "Workflow management: concepts, architecture and implementation". International Thomson Computer Press Publisher. ISBN 978-3-540-88581-8 (1996).

[32] Jian, M., Zhang, J., Zhao, H., Zhou, Y.: "Enhancing software product line maintenance with source code mining". Wireless Algorithm, Systems, and Applications, LNCS vol. 5258, pp. 538–547 (2008).

[33] Korherr, B.: "Business process modelling - languages, goals and variabilities". PhD Thesis. Vienna University of Technology, (2008).

[34] Kumar, A., Wen, Y.: "Design and management of flexible process variants using templates and rules". International Journal Computers in Industry 63(2), pp. 112–130 (2012).

[35] La Rosa, M., Dumas, M., Hofstede, A., Mendling, J., Gottschalk, F.: "Beyond control flow: extending business process configuration to roles and objects". International Conference on Conceptual Modeling, LNCS vol. 5231, pp. 199–215 (2008).

[36] La Rosa, M.: "Managing variability in process-aware information systems". PhD thesis, Faculty of Science and Technology Queensland University of Technology. Brisbane, Australia, (2009).

[37] La Rosa, M., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: "Questionnaire-based variability modeling for system configuration". Software and System Modeling 8(2), pp. 251–274 (2009).

[38] La Rosa, M., Dumas, M., ter Hofstede, A.H.M.: "Modelling Business Process Variability for Design-Time Configuration". Handbook of Research on Business Process Modeling, ISBN: 978-1605662886. Information Science Reference - Imprint of: IGI Publisher (2009).

[39] La Rosa, M., Dumas, M., Uba, R. Dijkman, R.: "Merging business process models". In Proceedings of the 18th International Conference on Cooperative Information Systems, LNCS vol. 6426, pp. 96–113, Springer (2010).

[40] La Rosa, M., Dumas, M. Hofstede, H.M., Mendling, J.: "Configurable multi-perspective business process models". Business Process Management Journal 12(2), pp. 1–23 (2011).

[41] Lanz, A., Weber, B. Reichert, M.: "Workflow time patterns for process-aware information systems". Proceedings Enterprise, Business-Process, and Information Systems Modelling, 11th International Workshop BPMDS and 15th International Conference EMMSAD, LNCS pp. 94–107 (2010).

[42] Lanz, A., Weber, B., Reichert, M.: "Time patterns for process-aware information systems". Requirements Engineering Journal (2012) (accepted for publication).

[43] Lenz, R., Reichert, M.: "IT Support for healthcare processes - premises, challenges, perspectives". Data and Knowledge Engineering 61(1), pp. 39–58 (2007).

[44] Li, C., Reichert, M. and Wombacher, A.: "Mining process variants: goals and issues". In IEEE International Conference on Service Computing 2(1), pp. 573–576 (2008).

[45] Li, C.: "Mining process variants: challenges, techniques, examples". PhD Thesis. University of Twente. Netherlands (2010).

[46] Li, C, Reichert, M., Wombacher, A.: "Mining business process variants: challenges, scenarios, algorithms". Data & Knowledge Engineering 70 (5), pp. 409–434 (2011).

[47] `http://www.processconfiguration.com/download.html` Accessed: April 2012.

[48] `http://www.mendling.com/EPML/C-EPC-Validator.xsl` Accessed: April 2012.

[49] JBoss Community, Drools Expert, 2011. `http://www.jboss.org/drools/drools-expert` Accessed: April 2012.

[50] Lu, R., Sadiq, S.W., Governatori, G.: "On managing business processes variants". Data Knowledge Engineering 68(7), pp. 642–664 (2009).

[51] Melao, N., Pidd, M.: "A conceptual framework for understanding business processes and business process modeling". Information Systems Jounal 10(2), pp. 105–129 (2000).

[52] Mendling, J., Verbeek, H., van Dongen, B., van der Aalst, W., Neumann, G.: "Detection and prediction of errors in EPCs of the SAP reference model". Data & Knowledge Engineering 64(1), pp. 312–329 (2008).

[53] Müller, D., Herbst, J., Hammori, M., Reichert, M.: "IT support for release management processes in the automotive industry". 4th International Conference on Business Process Management, LNCS vol. 4102, pp. 368–377 (2006).

[54] Parnas, D.L. "Software aging". 16th International Conference on Software Engineering, ICSE '94, pp. 279–287 (1994).

[55] Pinggera, J., Zugal, S., Weber, B.: "Investigating the process of process modeling with Cheetah Experimental Platform". Empirical Research in Process-Oriented Information Systems 30(2), pp. 13–18 (2012).

[56] Puhlmann, F., Schnieders, A., Weiland, J., Weske, M.: "Variability mechanisms for process models". Technical report, BMBF-Project (2006).

[57] Reichert, M., Rechtenbach, S., Hallerbach, A., Bauer, T.: "Extending a business process modeling tool with process configuration facilities: The

Provop demonstrator". BPM Demos, CEUR Workshop Proceedings, vol. 489. CEUR-WS.org (2009).

[58] Reichert. M, Weber, B.: "Enabling flexibility in process-aware information systems: challenges, methods, technologies". Springer (2012) (to appear).

[59] Reinhartz-Berger, I., Soffer, P., Sturm, A.: "Organisational reference models: supporting an adequate design of local business processes". International Journal Business Process Integration and Management 4(2), pp. 134–149 (2009).

[60] Reinhartz-Berger, I., Soffer, P., Sturm, A.: "Extending the adaptability of reference models". IEEE Transactions on Systems, Man, and Cybernetics, Part A 40(5), pp. 1045–1056 (2010).

[61] Rinderle-Ma, S., Reichert, M., Weber, B.: "On the formal semantics of change patterns in process-aware information systems". 27th International Conference on Conceptual Modeling, LNCS vol. 5231, pp. 279–293 (2008).

[62] Rosemann, M. "Potential pitfalls of process modeling: Part A". Business Process Management Journal 12(2), pp. 249–254 (2006).

[63] Rosemann, M., van der Aalst, W.M.P.: "A configurable reference modeling language". Information Systems 32(1), pp. 1–23 (2007).

[64] Russell, N., ter Hofstede, A., Edmond, D., van der Aalst, W.M.P: "Workow data patterns". Technical Report FIT-TR-2004-01, Queensland University of Technology (2004).

[65] Russell, N., ter Hofstede, A., Edmond, D., van der Aalst, W.M.P.: "Workow resource patterns". Technical Report WP 127, Eindhoven Univ. of Technology (2004).

[66] Russell, N., van der Aalst, W.M.P., ter Hofstede, A.: "Exception handling patterns in process-aware information systems". 18th International Conference on Advanced Information Systems Engineering, pp. 288–302 (2006).

[67] Soffer, P.: "Scope analysis: identifying the impact of changes in business process models." Software Process: Improvement and Practice 10(4), pp. 393–402 (2005).

[68] Thom, L., Reichert, M., Iochpe, C.: "Activity patterns in process-aware information systems: basic concepts and empirical evidence". International Journal of Business Process Integration and Management 4(2), pp. 93–110 (2009).

[69] Vergidis, K., Tiwari, A., Majeed, B.: "Business process analysis and optimization: beyond reengineering". IEEE Transactions on Systems, Man, and cybernetics, 38(1), pp. 69–82 (2008).

[70] Vervuurt, M.: "Modeling business process variability: a search for innovative solutions to business process variability modeling problems". Student Theses of University of Twente. October 2007.

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

[72] Weber, B. Sadiq, S. Reichert, M. "Beyond rigidity - dynamic process lifecycle support". Computer Science 23, pp. 47–65 (2009).

[73] Weber, B., Reichert, M., Reijers, H.A., Mendling, J.: "Refactoring large process model repositories". Computers in Industry 62(5), pp. 467–486 (2011).

[74] Weske, M.: "Business process management: concepts, languages, architectures". Springer-Verlag Berlin Heidelberg Publisher. ISBN: 978-3-540-73521-2 (2007).

[75] Yan, Z, Dijkman, R.M., Grefen, P.W.P.J.: "Business process model repositories - Framework and survey". Information and Software Technology 54(4), pp. 380–395 (2012).

92-08*	*Uwe Schöning* On Random Reductions from Sparse Sets to Tally Sets
92-09*	*Hermann von Hasseln, Laura Martignon* Consistency in Stochastic Network
92-10	*Michael Schmitt* A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function
92-11	*Johannes Köbler, Seinosuke Toda* On the Power of Generalized MOD-Classes
92-12	*V. Arvind, J. Köbler, M. Mundhenk* Reliable Reductions, High Sets and Low Sets
92-13	*Alfons Geser* On a monotonic semantic path ordering
92-14*	*Joost Engelfriet, Heiko Vogler* The Translation Power of Top-Down Tree-To-Graph Transducers
93-01	*Alfred Lupper, Konrad Froitzheim* AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager
93-02	*M.H. Scholl, C. Laasch, C. Rich, H.-J. Schek, M. Tresch* The COCOON Object Model
93-03	*Thomas Thierauf, Seinosuke Toda, Osamu Watanabe* On Sets Bounded Truth-Table Reducible to P-selective Sets
93-04	*Jin-Yi Cai, Frederic Green, Thomas Thierauf* On the Correlation of Symmetric Functions
93-05	*K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam* A Conceptual Approach to an Open Hospital Information System
93-06	*Klaus Gaßner* Rechnerunterstützung für die konzeptuelle Modellierung
93-07	*Ullrich Keßler, Peter Dadam* Towards Customizable, Flexible Storage Structures for Complex Objects
94-01	*Michael Schmitt* On the Complexity of Consistency Problems for Neurons with Binary Weights
94-02	*Armin Kühnemann, Heiko Vogler* A Pumping Lemma for Output Languages of Attributed Tree Transducers
94-03	*Harry Buhrman, Jim Kadin, Thomas Thierauf* On Functions Computable with Nonadaptive Queries to NP
94-04	*Heinz Faßbender, Heiko Vogler, Andrea Wedel* Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers

94-05 *V. Arvind, J. Köbler, R. Schuler*
On Helping and Interactive Proof Systems

94-06 *Christian Kalus, Peter Dadam*
Incorporating record subtyping into a relational data model

94-07 *Markus Tresch, Marc H. Scholl*
A Classification of Multi-Database Languages

94-08 *Friedrich von Henke, Harald Rueß*
Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

94-09 *F.W. von Henke, A. Dold, H. Rueß, D. Schwier, M. Strecker*
Construction and Deduction Methods for the Formal Development of Software

94-10 *Axel Dold*
Formalisierung schematischer Algorithmen

94-11 *Johannes Köbler, Osamu Watanabe*
New Collapse Consequences of NP Having Small Circuits

94-12 *Rainer Schuler*
On Average Polynomial Time

94-13 *Rainer Schuler, Osamu Watanabe*
Towards Average-Case Complexity Analysis of NP Optimization Problems

94-14 *Wolfram Schulte, Ton Vullinghs*
Linking Reactive Software to the X-Window System

94-15 *Alfred Lupper*
Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

94-16 *Robert Regn*
Verteilte Unix-Betriebssysteme

94-17 *Helmuth Partsch*
Again on Recognition and Parsing of Context-Free Grammars:
Two Exercises in Transformational Programming

94-18 *Helmuth Partsch*
Transformational Development of Data-Parallel Algorithms: an Example

95-01 *Oleg Verbitsky*
On the Largest Common Subgraph Problem

95-02 *Uwe Schöning*
Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

95-03 *Harry Buhrman,Thomas Thierauf*
The Complexity of Generating and Checking Proofs of Membership

95-04 *Rainer Schuler, Tomoyuki Yamakami*
Structural Average Case Complexity

95-05 *Klaus Achatz, Wolfram Schulte*
Architecture Indepentent Massive Parallelization of Divide-And-Conquer Algorithms

95-06 *Christoph Karg, Rainer Schuler*
Structure in Average Case Complexity

95-07 *P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe*
ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen

95-08 *Jürgen Kehrer, Peter Schulthess*
Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik

95-09 *Hans-Jörg Burtschick, Wolfgang Lindner*
On Sets Turing Reducible to P-Selective Sets

95-10 *Boris Hartmann*
Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen Netzen am Beispiel Truck Backer-Upper

95-11 *Thomas Beuter, Peter Dadam*:
Prinzipien der Replikationskontrolle in verteilten Systemen

95-12 *Klaus Achatz, Wolfram Schulte*
Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists

95-13 *Andrea Mößle, Heiko Vogler*
Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes

95-14 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
A Generic Specification for Verifying Peephole Optimizations

96-01 *Ercüment Canver, Jan-Tecker Gayen, Adam Moik*
Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche mit VSE

96-02 *Bernhard Nebel*
Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class

96-03 *Ton Vullinghs, Wolfram Schulte, Thilo Schwinn*
An Introduction to TkGofer

96-04 *Thomas Beuter, Peter Dadam*
Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am Beispiel der Domäne Concurrent-Engineering

96-05 *Gerhard Schellhorn, Wolfgang Ahrendt*
Verification of a Prolog Compiler - First Steps with KIV

96-06 *Manindra Agrawal, Thomas Thierauf*
Satisfiability Problems

96-07 *Vikraman Arvind, Jacobo Torán*
A nonadaptive NC Checker for Permutation Group Intersection

96-08 *David Cyrluk, Oliver Möller, Harald Rueß*
An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with Composition and Extraction

96-09 *Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte*
Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT–Ansätzen

96-10 *Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Formalizing Fixed-Point Theory in PVS

96-11 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Mechanized Semantics of Simple Imperative Programming Constructs

96-12 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Generic Compilation Schemes for Simple Programming Constructs

96-13 *Klaus Achatz, Helmuth Partsch*
From Descriptive Specifications to Operational ones: A Powerful Transformation Rule, its Applications and Variants

97-01 *Jochen Messner*
Pattern Matching in Trace Monoids

97-02 *Wolfgang Lindner, Rainer Schuler*
A Small Span Theorem within P

97-03 *Thomas Bauer, Peter Dadam*
A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration

97-04 *Christian Heinlein, Peter Dadam*
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow Dependencies

97-05 *Vikraman Arvind, Johannes Köbler*
On Pseudorandomness and Resource-Bounded Measure

97-06 *Gerhard Partsch*
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien für den digitalen Mobilfunkstandard DECT

97-07 *Manfred Reichert, Peter Dadam*
$ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08 *Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler*
The Project NoName - A functional programming language with its development environment

97-09 *Christian Heinlein*
Grundlagen von Interaktionsausdrücken

97-10 *Christian Heinlein*
Graphische Repräsentation von Interaktionsausdrücken

97-11 *Christian Heinlein*
Sprachtheoretische Semantik von Interaktionsausdrücken

97-12	*Gerhard Schellhorn, Wolfgang Reif* Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers
97-13	*Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn* Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme
97-14	*Wolfgang Reif, Gerhard Schellhorn* Theorem Proving in Large Theories
97-15	*Thomas Wennekers* Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen
97-16	*Peter Dadam, Klaus Kuhn, Manfred Reichert* Clinical Workflows - The Killer Application for Process-oriented Information Systems?
97-17	*Mohammad Ali Livani, Jörg Kaiser* EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications
97-18	*Johannes Köbler,Rainer Schuler* Using Efficient Average-Case Algorithms to Collapse Worst-Case Complexity Classes
98-01	*Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf* Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung
98-02	*Thomas Bauer, Peter Dadam* Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse
98-03	*Marko Luther, Martin Strecker* A guided tour through *Typelab*
98-04	*Heiko Neumann, Luiz Pessoa* Visual Filling-in and Surface Property Reconstruction
98-05	*Ercüment Canver* Formal Verification of a Coordinated Atomic Action Based Design
98-06	*Andreas Küchler* On the Correspondence between Neural Folding Architectures and Tree Automata
98-07	*Heiko Neumann, Thorsten Hansen, Luiz Pessoa* Interaction of ON and OFF Pathways for Visual Contrast Measurement
98-08	*Thomas Wennekers* Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons
98-09	*Thomas Bauer, Peter Dadam* Variable Migration von Workflows in *ADEPT*
98-10	*Heiko Neumann, Wolfgang Sepp* Recurrent V1 – V2 Interaction in Early Visual Boundary Processing

98-11	*Frank Houdek, Dietmar Ernst, Thilo Schwinn* Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment
98-12	*Gerhard Schellhorn* Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers
98-13	*Gerhard Schellhorn, Wolfgang Reif* Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers
98-14	*Mohammad Ali Livani* SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN
98-15	*Mohammad Ali Livani, Jörg Kaiser* Predictable Atomic Multicast in the Controller Area Network (CAN)
99-01	*Susanne Boll, Wolfgang Klas, Utz Westermann* A Comparison of Multimedia Document Models Concerning Advanced Requirements
99-02	*Thomas Bauer, Peter Dadam* Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation
99-03	*Uwe Schöning* On the Complexity of Constraint Satisfaction
99-04	*Ercument Canver* Model-Checking zur Analyse von Message Sequence Charts über Statecharts
99-05	*Johannes Köbler, Wolfgang Lindner, Rainer Schuler* Derandomizing RP if Boolean Circuits are not Learnable
99-06	*Utz Westermann, Wolfgang Klas* Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets
99-07	*Peter Dadam, Manfred Reichert* Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications. Paderborn, Germany, October 6, 1999, GI–Workshop Proceedings, Informatik ’99
99-08	*Vikraman Arvind, Johannes Köbler* Graph Isomorphism is Low for ZPP^{NP} and other Lowness results
99-09	*Thomas Bauer, Peter Dadam* Efficient Distributed Workflow Management Based on Variable Server Assignments
2000-02	*Thomas Bauer, Peter Dadam* Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT
2000-03	*Gregory Baratoff, Christian Toepfer, Heiko Neumann* Combined space-variant maps for optical flow based navigation

2000-04 *Wolfgang Gehring*
Ein Rahmenwerk zur Einführung von Leistungspunktsystemen

2000-05 *Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel*
Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

2000-06 *Wolfgang Reif, Gerhard Schellhorn, Andreas Thums*
Fehlersuche in Formalen Spezifikationen

2000-07 *Gerhard Schellhorn, Wolfgang Reif (eds.)*
FM-Tools 2000: The 4th Workshop on Tools for System Design and Verification

2000-08 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Effiziente Durchführung von Prozessmigrationen in verteilten Workflow-Management-Systemen

2000-09 *Thomas Bauer, Peter Dadam*
Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

2000-10 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Adaptives und verteiltes Workflow-Management

2000-11 *Christian Heinlein*
Workflow and Process Synchronization with Interaction Expressions and Graphs

2001-01 *Hubert Hug, Rainer Schuler*
DNA-based parallel computation of simple arithmetic

2001-02 *Friedhelm Schwenker, Hans A. Kestler, Günther Palm*
3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

2001-03 *Hans A. Kestler, Friedhelm Schwenker, Günther Palm*
RBF network classification of ECGs as a potential marker for sudden cardiac death

2001-04 *Christian Dietrich, Friedhelm Schwenker, Klaus Riede, Günther Palm*
Classification of Bioacoustic Time Series Utilizing Pulse Detection, Time and Frequency Features and Data Fusion

2002-01 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata

2002-02 *Walter Guttmann*
Deriving an Applicative Heapsort Algorithm

2002-03 *Axel Dold, Friedrich W. von Henke, Vincent Vialard, Wolfgang Goerigk*
A Mechanically Verified Compiling Specification for a Realistic Compiler

2003-01 *Manfred Reichert, Stefanie Rinderle, Peter Dadam*
A Formal Framework for Workflow Type and Instance Changes Under Correctness Checks

2003-02 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Supporting Workflow Schema Evolution By Efficient Compliance Checks

2003-03	*Christian Heinlein* Safely Extending Procedure Types to Allow Nested Procedures as Values
2003-04	*Stefanie Rinderle, Manfred Reichert, Peter Dadam* On Dealing With Semantically Conflicting Business Process Changes.
2003-05	*Christian Heinlein* Dynamic Class Methods in Java
2003-06	*Christian Heinlein* Vertical, Horizontal, and Behavioural Extensibility of Software Systems
2003-07	*Christian Heinlein* Safely Extending Procedure Types to Allow Nested Procedures as Values (Corrected Version)
2003-08	*Changling Liu, Jörg Kaiser* Survey of Mobile Ad Hoc Network Routing Protocols)
2004-01	*Thom Frühwirth, Marc Meister (eds.)* First Workshop on Constraint Handling Rules
2004-02	*Christian Heinlein* Concept and Implementation of C+++, an Extension of C++ to Support User-Defined Operator Symbols and Control Structures
2004-03	*Susanne Biundo, Thom Frühwirth, Günther Palm(eds.)* Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence
2005-01	*Armin Wolf, Thom Frühwirth, Marc Meister (eds.)* 19th Workshop on (Constraint) Logic Programming
2005-02	*Wolfgang Lindner (Hg.), Universität Ulm , Christopher Wolf (Hg.) KU Leuven* 2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm
2005-03	*Walter Guttmann, Markus Maucher* Constrained Ordering
2006-01	*Stefan Sarstedt* Model-Driven Development with ACTIVECHARTS, Tutorial
2006-02	*Alexander Raschke, Ramin Tavakoli Kolagari* Ein experimenteller Vergleich zwischen einer plan-getriebenen und einer leichtgewichtigen Entwicklungsmethode zur Spezifikation von eingebetteten Systemen
2006-03	*Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari* Eine qualitative Untersuchung zur Produktlinien-Integration über Organisationsgrenzen hinweg
2006-04	*Thorsten Liebig* Reasoning with OWL - System Support and Insights –
2008-01	*H.A. Kestler, J. Messner, A. Müller, R. Schuler* On the complexity of intersecting multiple circles for graphical display

2008-02	*Manfred Reichert, Peter Dadam, Martin Jurisch,l Ulrich Kreher, Kevin Göser, Markus Lauer* Architectural Design of Flexible Process Management Technology
2008-03	*Frank Raiser* Semi-Automatic Generation of CHR Solvers from Global Constraint Automata
2008-04	*Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander* Entscheidungsdokumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse
2008-05	*Markus Kalb, Claudia Dittrich, Peter Dadam* Support of Relationships Among Moving Objects on Networks
2008-06	*Matthias Frank, Frank Kargl, Burkhard Stiller (Hg.)* WMAN 2008 – KuVS Fachgespräch über Mobile Ad-hoc Netzwerke
2008-07	*M. Maucher, U. Schöning, H.A. Kestler* An empirical assessment of local and population based search methods with different degrees of pseudorandomness
2008-08	*Henning Wunderlich* Covers have structure
2008-09	*Karl-Heinz Niggl, Henning Wunderlich* Implicit characterization of FPTIME and NC revisited
2008-10	*Henning Wunderlich* On span-P^{cc} and related classes in structural communication complexity
2008-11	*M. Maucher, U. Schöning, H.A. Kestler* On the different notions of pseudorandomness
2008-12	*Henning Wunderlich* On Toda’s Theorem in structural communication complexity
2008-13	*Manfred Reichert, Peter Dadam* Realizing Adaptive Process-aware Information Systems with ADEPT2
2009-01	*Peter Dadam, Manfred Reichert* The ADEPT Project: A Decade of Research and Development for Robust and Fexible Process Support Challenges and Achievements
2009-02	*Peter Dadam, Manfred Reichert, Stefanie Rinderle-Ma, Kevin Göser, Ulrich Kreher, Martin Jurisch* Von ADEPT zur AristaFlow® BPM Suite – Eine Vision wird Realität “Correctness by Construction” und flexible, robuste Ausführung von Unternehmensprozessen

2009-03	*Alena Hallerbach, Thomas Bauer, Manfred Reichert* Correct Configuration of Process Variants in Provop
2009-04	*Martin Bader* On Reversal and Transposition Medians
2009-05	*Barbara Weber, Andreas Lanz, Manfred Reichert* Time Patterns for Process-aware Information Systems: A Pattern-based Analysis
2009-06	*Stefanie Rinderle-Ma, Manfred Reichert* Adjustment Strategies for Non-Compliant Process Instances
2009-07	*H.A. Kestler, B. Lausen, H. Binder H.-P. Klenk. F. Leisch, M. Schmid* Statistical Computing 2009 – Abstracts der 41. Arbeitstagung
2009-08	*Ulrich Kreher, Manfred Reichert, Stefanie Rinderle-Ma, Peter Dadam* Effiziente Repräsentation von Vorlagen- und Instanzdaten in Prozess-Management-Systemen
2009-09	*Dammertz, Holger, Alexander Keller, Hendrik P.A. Lensch* Progressive Point-Light-Based Global Illumination
2009-10	*Dao Zhou, Christoph Müssel, Ludwig Lausser, Martin Hopfensitz, Michael Kühl, Hans A. Kestler* Boolean networks for modeling and analysis of gene regulation
2009-11	*J. Hanika, H.P.A. Lensch, A. Keller* Two-Level Ray Tracing with Recordering for Highly Complex Scenes
2009-12	*Stephan Buchwald, Thomas Bauer, Manfred Reichert* Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen
2010-01	*Hariolf Betz, Frank Raiser, Thom Frühwirth* A Complete and Terminating Execution Model for Constraint Handling Rules
2010-02	*Ulrich Kreher, Manfred Reichert* Speichereffiziente Repräsentation instanzspezifischer Änderungen in Prozess-Management-Systemen
2010-03	*Patrick Frey* Case Study: Engine Control Application
2010-04	*Matthias Lohrmann und Manfred Reichert* Basic Considerations on Business Process Quality
2010-05	*HA Kestler, H Binder, B Lausen, H-P Klenk, M Schmid, F Leisch (eds):* Statistical Computing 2010 - Abstracts der 42. Arbeitstagung
2010-06	*Vera Künzle, Barbara Weber, Manfred Reichert* Object-aware Business Processes: Properties, Requirements, Existing Approaches

2011-01	*Stephan Buchwald, Thomas Bauer, Manfred Reichert* Flexibilisierung Service-orientierter Architekturen
2011-02	*Johannes Hanika, Holger Dammertz, Hendrik Lensch* Edge-Optimized À-Trous Wavelets for Local Contrast Enhancement with Robust Denoising
2011-03	*Stefanie Kaiser, Manfred Reichert* Datenflussvarianten in Prozessmodellen: Szenarien, Herausforderungen, Ansätze
2011-04	*Hans A. Kestler, Harald Binder, Matthias Schmid, Friedrich Leisch, Johann M. Kraus (eds):* Statistical Computing 2011 - Abstracts der 43. Arbeitstagung
2011-05	*Vera Künzle, Manfred Reichert* PHILharmonicFlows: Research and Design Methodology
2011-06	*David Knuplesch, Manfred Reichert* Ensuring Business Process Compliance Along the Process Life Cycle
2011-07	*Marcel Dausend* Towards a UML Profile on Formal Semantics for Modeling Multimodal Interactive Systems
2011-08	*Dominik Gessenharter* Model-Driven Software Development with ACTIVECHARTS - A Case Study
2012-01	*Andreas Steigmiller, Thorsten Liebig, Birte Glimm* Extended Caching, Backjumping and Merging for Expressive Description Logics
2012-02	*Hans A. Kestler, Harald Binder, Matthias Schmid, Johann M. Kraus (eds):* Statistical Computing 2012 - Abstracts der 44. Arbeitstagung
2012-03	*Felix Schüssel, Frank Honold, Michael Weber* Influencing Factors on Multimodal Interaction at Selection Tasks
2012-04	*Jens Kolb, Paul Hübner, Manfred Reichert* Model-Driven User Interface Generation and Adaption in Process-Aware Information Systems
2012-05	*Matthias Lohrmann, Manfred Reichert* Formalizing Concepts for Efficacy-aware Business Process Modeling
2012-06	*David Knuplesch, Rüdiger Pryss, Manfred Reichert* A Formal Framework for Data-Aware Process Interaction Models
2012-07	*Clara Ayora, Victoria Torres, Barbara Weber, Manfred Reichert, Vicente Pelechano* Dealing with Variability in Process-Aware Information Systems: Language Requirements, Features, and Existing Proposals

Ulmer Informatik-Berichte
ISSN 0939-5091

Herausgeber:
Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik
89069 Ulm