# VIVACE: A Framework for the Systematic Evaluation of Variability Support in Process-Aware Information Systems

Clara Ayora[a,1], 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, Camino de Vera s/n, 46022 València, Spain, Phone: +34 963 87 70 07, Ext. 83533*
[b] *Department of Computer Science, University of Innsbruck, Technikerstraße 21a, 6020 Innsbruck, Austria, Phone: +43 (0) 512 507 6474*
[c] *Institute of Databases and Information Systems, University of Ulm, James-Franck-Ring, 89069 Ulm, Germany, Phone: +49 731 50 24135*

**Abstract**

*Context:* The increasing adoption of process-aware information systems (PAISs) such as workflow management systems, enterprise resource planning systems, or case management systems, together with the high variability in business processes (e.g., sales processes may vary depending on the respective products and countries), has resulted in large industrial process model repositories. To cope with this business process variability, the proper management of process variants along the entire process lifecycle becomes crucial.
*Objective:* The goal of this paper is to develop a fundamental understanding of business process variability. In particular, the paper will provide a framework for assessing and comparing process variability approaches and the support they provide for the different phases of the business process life-

*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)

[1]Corresponding author.

cycle (i.e., process analysis and design, configuration, enactment, diagnosis, and evolution).
*Method:* We conducted a systematic literature review (SLR) in order to discover how process variability is supported by existing approaches.
*Results:* The SLR resulted in 63 primary studies which were deeply analyzed. Based on this analysis, we derived the *VIVACE* framework. *VIVACE* allows assessing the expressiveness of a process modeling language regarding the explicit specification of process variability. Furthermore, the support provided by a process-aware information system to properly deal with process model variants can be assessed with *VIVACE* as well.
*Conclusions:* *VIVACE* provides an empirically-grounded framework for process engineers that enables them to evaluate existing process variability approaches as well as to select that variability approach meeting their requirements best. Finally, it helps process engineers in implementing PAISs supporting process variability along the entire process lifecycle.

*Keywords:* business process, business process variability, process-aware information systems, process family, systematic literature review

## 1. Introduction

Each product an enterprise develops or produces and each service it delivers to its customers result from the coordinated execution of a set of *activities* (i.e., business functions). In this context, *business processes*[2] act as the drivers enabling this coordination [131]. Consequently, *process-aware information systems* (PAISs) provide a guiding framework in enterprise computing, supporting the entire business process lifecycle [128]. More precisely, a PAIS constitutes an information system that manages, executes, and analyzes the internal business processes of an enterprise (e.g., sales business processes) based on explicitly specified *process models*. In turn, these models may refer to actors (e.g., sellers), application services (e.g., web services), and business data (e.g., products) [35]. Examples of PAISs include workflow management systems (e.g., ADEPT2 [98], YAWL [3]), enterprise resource planning systems (e.g., SAP R/3 [113]), and case management systems (e.g., FLOWer [35], PHILharmonicFlows [59]).

[2]Note that we use the terms *business process* and *process* synonymously throughout the paper.

The increasing adoption of PAISs in enterprises during the last decade has resulted in large process model repositories [105, 32, 37]. Usually, such repositories comprise collections of related *process model variants* (*process variants* for short). On one hand, respective process variants pursue the same or similar *business objective* (e.g., product sales, patient treatment, or car maintenance). On the other, they show differences in several respects due to their varying *application context*, e.g., the regulations to be obeyed in different countries or the type of product to be delivered to customers [100, 32, 125].

A collection of related *process variants* is denoted as *process family.* In practice, a process family may comprise dozens or hundreds of process variants [87]. In the automotive industry, for example, we found a process family dealing with vehicle repair and maintenance in a garage, which comprises more than 900 process variants [48]. The latter share *commonalities* (i.e., process fragments shared by all process variants), but also show country- and vehicle-specific *variations.* In turn, [73] reports on more than 90 process variants for handling medical examinations in a hospital. Finally, consider check-in procedures at an airport, which are characterized by a high degree of variability as well. Example 1 describes the check-in process in detail and discusses its different sources of variability. Note that we will use this process as running example throughout the paper.

**Example 1 (Check-in process)**. We consider 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 flying with, numerous variations exist depending on distinguished factors. For example, variability is caused by the type of check-in (e.g., online, at the counter, or at the self-servicing machine), which, in turn, determines the type of boarding card (e.g., electronic versus paper-based). Other sources of variability include the flight destination (e.g., information about the accommodation is required when traveling to the US) and the type of passenger (e.g., unaccompanied minors and handicapped people might require extra assistance). Depending on the type of luggage (e.g., bulk or overweight luggage), moreover, the process slightly differs since an extra fee might have to be paid. Finally, temporal variations regarding the check-in procedure are typical as well (e.g., possibility to check-in several days before departure versus checking-in a few hours before the flight).

Figures 1 and 2 show six simplified variants of this check-in process represented in terms of the *Business Process Modeling Notation* (BPMN) [18]. These process variants have been modeled and validated in collaboration with subject matter experts. In particular, the process variants share commonalities while also showing differences. Activities common to all process variants are colored in grey. *Variants 1* and *2* (cf. Fig. 1) presume that the check-in is done online by the passenger. First, the passenger is identified and a seat is assigned. *Variant 1* describes the process in case the passenger is flying from Europe to the United States, which requires information about accommodation as well as filling in the electronic system for travel authorization (i.e., ESTA form). Finally, an electronic boarding card is printed and the passenger drops off the luggage at the business class counter. Regarding *Variant 2*, after printing the boarding card, the payment of an extra fee at the airline ticket office is required due to luggage overweight. In turn, for *Variant 3* the check-in is done at the self-servicing machine and the luggage is dropped off at the fast bag drop counter. Finally, for these three process variants, check-in becomes available 23 hours before departure.

In contrast, *Variants 4-6* (cf. Fig. 2) represent the check-in process accomplished at the respective counter at the airport. For example, *Variant 4* describes the check-in for an unaccompanied minor. In this variant, a special seat is assigned and an extra form is filled in. In addition, a copy of the boarding card is required for the relative accompanying the minor to the boarding gate. *Variant 5* refers to a handicapped passenger requiring extra assistance by a person accompanying him, whereas *Variant 6* corresponds to the check-in process of a passenger carrying bulk luggage. In these three process variants, a check-in may only be performed at maximum 3 hours before departure, once the counters will have opened. Finally, the boarding card is printed in paper format.

Figures 1 and 2 exemplify the process family of the check-in process and illustrate the complexity of the latter due to the variability of its application context (e.g., type of passenger, flight destination, and type of luggage). Trying to design, model, implement, and maintain each process variant of such a process family from scratch would be too cumbersome and costly for enterprises [129]. The proper management of process families (i.e., process

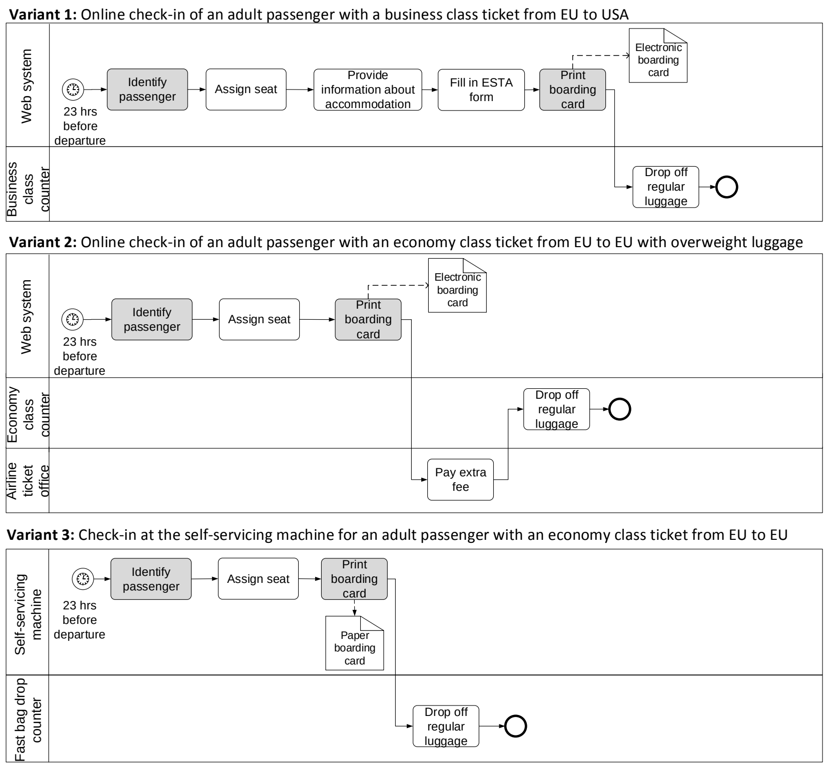


Figure 1: Variants of the check-in process (1)

variability) and their members (i.e., process variants), therefore, constitutes a fundamental challenge for every PAIS.

## *1.1. Problem Statement*

In order to efficiently and effectively manage process families, enterprises are interested in capturing common process knowledge only once and in making it reusable in terms of a *reference process model* (*reference process* for short) [100]. Along this trend, a multitude of reference processes have been suggested in various domains. Examples include ITIL processes for IT service management [50], SAP reference processes for enterprise resource management [83], and medical guidelines for patient treatment [71]. Usually, reference processes are described in a graphical way using a process modeling language like *Business Process Modeling Notation* (BPMN) or *Event-driven Process Chain* (EPC). However, in off-the-shelf process modeling suites like

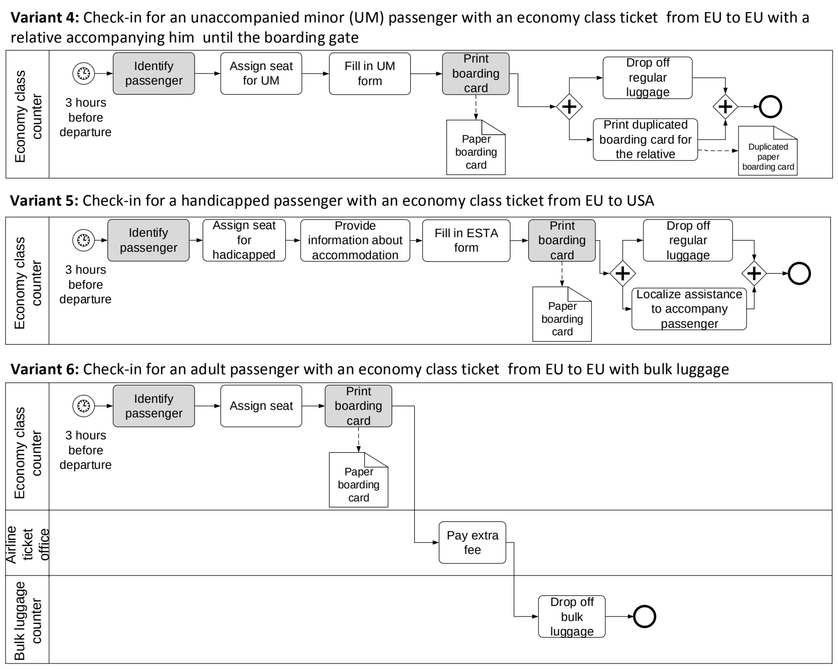

Variant 4: Check-in for an unaccompanied minor (UM) passenger with an economy class ticket from EU to EU with a relative accompanying him until the boarding gate
Economy class counter
3 hours before departure
Identify passenger
Assign seat for UM
Fill in UM form
Print boarding card
Paper boarding card
Drop off regular luggage
Print duplicated boarding card for the relative
Duplicated paper boarding card
Variant 5: Check-in for a handicapped passenger with an economy class ticket from EU to USA
Economy class counter
3 hours before departure
Identify passenger
Assign seat for hadicapped
Provide information about accommodation
Fill in ESTA form
Print boarding card
Paper boarding card
Drop off regular luggage
Localize assistance to accompany passenger
Variant 6: Check-in for an adult passenger with an economy class ticket from EU to EU with bulk luggage
Economy class counter
3 hours before departure
Identify passenger
Assign seat
Print boarding card
Paper boarding card
Airline ticket office
Pay extra fee
Bulk luggage counter
Drop off bulk luggage

Figure 2: Variants of the check-in process (2)

ARIS Architect and Signavio, no proper support for explicitly describing the variations of a reference process exists [103]. In particular, this lack of variability support in existing process modeling suites requires from process engineers to manually derive process variants, which is both a tedious and error-prone task [48, 49].

Motivated by the shortcomings of existing process modeling suites, various approaches enabling process variability along the process lifecycle have been developed; i.e., approaches allowing for the analysis, design, configuration, enactment, diagnosis, and evolution of process families (i.e., collections of related process variants) [96, 106, 48]. By treating variability as a first class citizen, these approaches contribute to avoid model redundancies, foster model reusability, and reduce modeling efforts. However, explicitly capturing and modeling variability introduces additional complexity with respect to the design of the process modeling language used [100].

In order to make these *process variability approaches* amenable for industrial use, the quality of process variant models becomes crucial. In particular, this necessitates proper assistance of process engineers with respect to the modeling and management of process variability. This means that process engineers should be supported in selecting that process variability approach fitting best to their needs. However, there is a lack of profound methods for systematically assessing and comparing existing process variability approaches. Although several attempts to characterize process variability have been made (cf. Sect. 8), a comprehensive and well elaborated framework for assessing process modeling tools and PAISs regarding their ability to deal with process variability is still missing.

### *1.2. Contribution*

The major contribution of this paper is twofold:

1. We present results from a systematic literature review (SLR) of approaches enabling process variability along the process lifecycle. Besides elaborating the state of the art, we want to systematically analyze and assess existing process variability approaches regarding their expressiveness with respect to process variability modeling as well as their support for managing variability along the process lifecycle. In this context, we identify the strengths and shortcomings of these process variability approaches and discuss research opportunities. Overall, our analysis provides a profound understanding of business process variability and approaches supporting it.
2. Based on the empirical evidence provided by the SLR, we derive the *VIVACE framework*, which shall allow for the systematic assessment and comparison of existing process variability approaches. In addition, *VIVACE* enables process engineers to select that variability approach meeting their requirements best as well as helps them in implementing PAISs supporting process variability. In detail, *VIVACE* comprises a core set of *variability-specific language constructs* as well as a core set of features fostering process variability along the process lifecycle (denoted as *variability support features* in the following). In particular, these language constructs (e.g., configurable process region, configuration alternative) allow assessing the expressiveness of existing process variability approaches regarding the modeling of process variability, whereas variability support features shall ensure that process variability can be effectively handled along the process lifecycle. In addition,

it should be possible to efficiently execute as well as to dynamically re-configure the instances of a process variant if required.

This work can be considered as a reference for implementing PAISs being able to effectively cope with process variability along the entire process lifecycle. In addition, we expect the *VIVACE* framework to be applied to various process variability approaches as well as related tools in order to assess their suitability with respect to process variability management. In this vein, the framework is expected to support enterprises and process engineers in deciding which process variability approach suits best to their needs.

The remainder of the paper is organized as follows: Section 2 provides background information required to contextualize process variability. Section 3 then describes the research methodology we applied in the context of the conducted SLR. Section 4 presents the results of the SLR, whereas Section 5 uses the latter to derive the *VIVACE* framework. Furthermore, *VIVACE* is applied to selected approaches obtained from the SLR. Section 6 provides a discussion of corresponding results, whereas Section 7 deals with potential threats regarding the validity of our work. Section 8 discusses related work and Section 9 concludes the paper with a summary and outlook.

## 2. Background

This section summarizes backgrounds related to business process management with a particular emphasis on the support of process variability along the process lifecycle. Referring to different process perspectives, Sect. 2.1 first introduces basic concepts and notions for process modeling and illustrates them along the presented check-in process (cf. Example 1). Section 2.2 then extends these concepts with variability-specific issues including a revised definition of the process lifecycle.

### *2.1. Business Process Perspectives*

According to [131], a business process is defined as "a set of activities performed in coordination in an organizational and technical environment". Analyzing this definition, a business process defines *what* (activities) shall be done, *how* it shall be done (coordination), and by *whom* (organizational and technical environment). In this context, a *business process model* (i.e., *process schema*) constitutes the main artifact for representing the respective process. Basically, a business process model (*process model* for short) is created using the elements (i.e., constructs) of the meta-model depicted in Fig.

3. This meta-model has been adopted from [22]. It allows for the modeling of the functional, behavioral, organizational, informational, temporal, and operational perspectives of a business process [23, 81, 56, 100, 52, 67, 68]. In general, a process model may be subject to variation in all these perspectives. In detail:

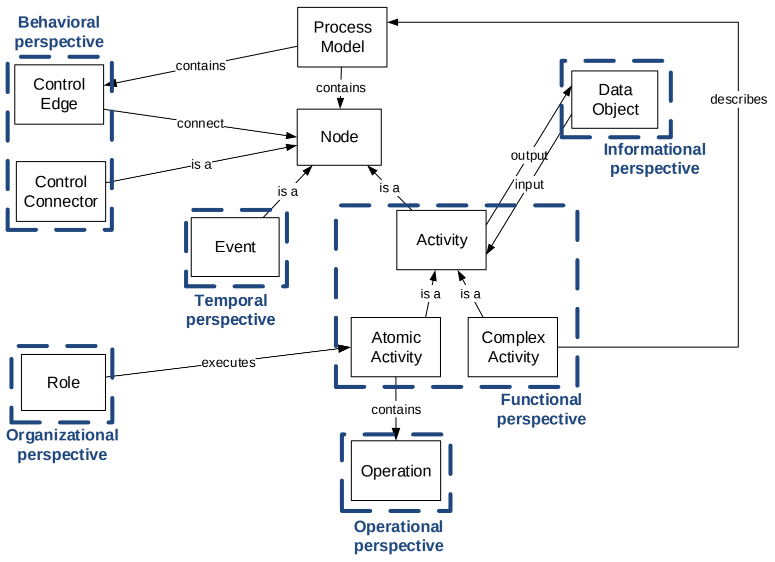


Figure 3: Process meta-model adopted from [22]

- The *functional perspective* specifies the decomposition of a business process into units of work, i.e., it represents the *activities* that may have to be performed to reach a particular *business objective* [23]. An atomic activity is associated with a single action, whereas a complex activity refers to a sub-process or, more precisely, a sub-process model. In Fig. 3, this perspective is represented by entities *activity, atomic activity,* and *complex activity.* Example 2 illustrates variability in respect to the functional perspective.

**Example 2 (Variability in respect to the functional perspective).** Consider *Variants 4-6* of the check-in process. Depending on the type of passenger, the set of activities to be performed may differ; e.g., "Assign seat for UM" in the context of unaccompanied minors, "Assign seat for handicapped" in the context of handicapped passengers, or "Assign seat" for regular passengers. All three activities constitute entities of type *atomic activity* (cf. Fig. 3).

- The *behavioral perspective* captures the (dynamic) behavior of a process model and hence reflects the *control flow* between its activities. The latter defines the order of the activities as well as the constraints for their execution. In Fig. 3, this perspective is represented by entities *control connector* (i.e., gateway) and *control edge* (i.e., arrows). Example 3 illustrates variability in respect to the behavioral perspective.

**Example 3 (Variability in respect to the behavioral perspective).** Consider *Variants 1* and *2* of the check-in process. Their *control flow* differs regarding the model part preceding activity *Print boarding card* (cf. Fig. 1). Activities *Provide information about accommodation* and *Fill in ESTA form* are only performed if the passenger is traveling to the US, but shall be omitted otherwise. Accordingly, there exist two options in the control flow of the process; i.e., either to perform the activities or to skip them.

- The *organizational perspective* represents the different *actors or roles* involved in a process model that are in charge of executing particular process activities (i.e., humans or systems). This perspective is represented by entity *role* in Fig. 3. Example 4 illustrates variability in respect to the organizational perspective.

**Example 4 (Variability in respect to the organizational perspective).** Online check-in is performed by *passengers* using a web system (*Variants 1-2*), whereas check-in at the counter is performed by *airline staff* (*Variants 4-6* in Fig. 2). The passenger and the airline staff constitute entities of type *role* (cf. Fig. 3).

- The *informational perspective* covers *data* and *data flow*, i.e., it represents the informational entities (e.g., data objects) consumed (i.e., used as input for activity execution) or produced (i.e., as output resulting from activity execution) during process execution. The informational perspective is represented by entity *data object* (cf. Fig. 3). Example 5 illustrates variability in respect to the informational perspective.

**Example 5 (Variability in respect to the informational perspective).** Depending on the type of check-in, the resulting boarding card either is an electronic or a paper-based document. The boarding card constitutes an entity of type *data object* (cf. Fig. 3).

- The *temporal perspective* covers temporal constraints restricting the execution and scheduling of activities; e.g., the time an activity may be started or completed, a message arrived, a deadline expired, or an error occurred. This perspective is represented by entity *event* in Fig. 3. Example 6 illustrates variability in respect to the temporal perspective.

**Example 6 (Variability in respect to the temporal perspective).** The availability of the check-in service is delimited from 23 (*Variants 1-3*) to 3 (*Variants 4-6*) hours before departure, depending on the type of check-in. This is represented using different start events (cf. Figs.1 and 2), which constitute entities of type *event* (cf. Fig. 3).

- The *operational perspective* refers to the implementation of atomic process activities, i.e., the application services (e.g., web services, electronic user forms) to be invoked when these *activities* are started. For a particular atomic activity, different implementations may exist. At enactment time, one of them is then dynamically selected and bound to the execution of this activity. This perspective is represented by entity *operation* in Fig. 3. Example 7 illustrates variability in respect to the operational perspective.

**Example 7 (Variability in respect to the operational perspective).** The implementation of the *Print boarding card* activity differs depending on the type of check-in; i.e., online, counter, or with self-servicing machine.

## *2.2. Business Process Lifecycle*

Business process models not only serve for documentation purposes, but are embedded in a process lifecycle comprising different phases [131, 19, 8]. These phases include *Analysis & Design, Configuration, Enactment, Diagnosis,* and *Evolution.* Fig. 4 depicts the transitions from the analysis and design of a process family to the enactment of a *process variant instance.* The latter represents a concrete case in the operational business of an enterprise [131]. In general, each process variant acts as a blueprint for a set of business process instances. Regarding our running example, for example, the check-in of a particular passenger corresponds to one instance of the process variant representing this check-in.

*Analysis & Design phase.* Based on domain requirements, relevant (emerging or existing) process information is gathered, analyzed, consolidated, and represented in terms of process models. The resulting process models are then validated and verified based on various techniques (e.g., simulation, correctness checks). In the context of process variability, related process variants are *defined* in terms of a *configurable process model*, which represents a complete process family. In particular, a *configurable process model* eliminates model redundancies by representing the commonalities of different process variants only once (cf. Fig. 4). Furthermore, it fosters model reuse since variant particularities can be shared among multiple variants. After creating a *configurable process model*, it must be verified and validated. Verification means that it needs to be ensured that all process variants that may be derived from the *configurable process model* are syntactically correct and sound (e.g., no deadlocks or livelocks). In turn, validation shall ensure that the *configurable process model* properly reflects the semantics of all business processes.

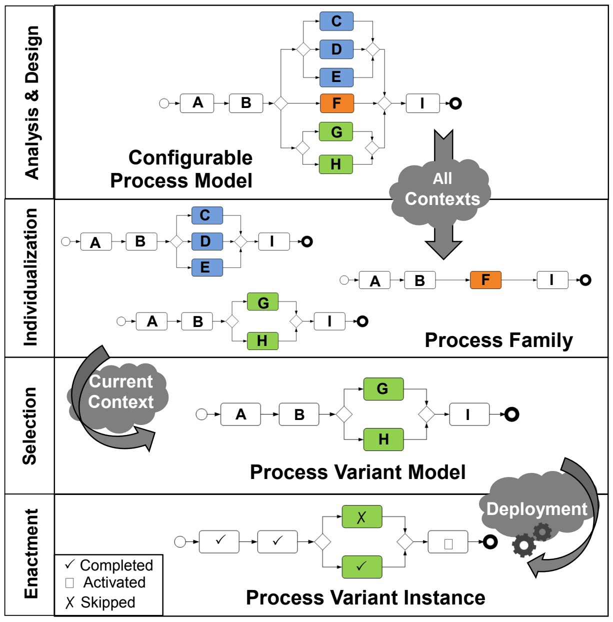


Figure 4: From process family definition to process variant enactment

*Configuration phase.* Based on the respective application environment (i.e., application context) in which a process shall exist [62], an *individualization* as well as a *selection procedure* are performed in order to derive a specific process variant from the *configurable process model* (cf. Fig. 4). Then, the individualized (and selected) process variant is deployed to the target process engine by translating its graphical representation (i.e., process variant model) into an executable representation (i.e., process variant instance). The latter is specified either with a *business process modeling language* or–in case the process is realized in a service-oriented environment–with a service composition language (e.g., *Business Process Execution Language*, WS-BPEL [17]).

*Enactment phase.* This phase deals with the enactment of the instances of a process variant. This implies guaranteeing that process variants are *executed* according to the configured process variant model (cf. Fig. 4).

Moreover, this phase covers configuration decisions that may only be made during enactment time. For example, whether or not a passenger needs to pay an extra fee for an overweight luggage will only become known once she arrives at the counter. Accordingly, monitoring techniques are required to provide accurate information about the current execution state of the process variant instance. In addition, to cope with contextual changes during its execution [119], a dynamic re-configuration might become necessary to switch from the current process variant to another [1]. Unlike ad-hoc changes (i.e., unplanned changes [100]), re-configuration options should be already known at design time and hence be specified in the *configurable process model* during the *Analysis & Design phase.* Again, the syntactical as well as semantical correctness of the process variants must be ensured in the context of dynamic re-configurations [5, 47, 11].

*Diagnosis phase.* In this phase, information gathered during the *Configuration* and *Enactment* phases (e.g., configuration settings applied at configuration or enactment time) is analyzed in order to optimize and evolve the process family and its implementation.

*Evolution phase.* During this phase, emerging requirements as well as identified optimizations lead to the evolution of the process family; e.g., by adding, removing, or changing family members (i.e., process variant models).

## 3. Methodology

To provide a fundamental understanding of business process variability, we conduct a *Systematic Literature Review* (SLR). In general, an SLR is a means of identifying, evaluating, and interpreting relevant data (i.e., research works) in a specific area through a replicable, scientific, and transparent approach, which reduces the probability of any bias [54]. To conduct such an SLR with respect to process variability, we design a protocol following the guidelines, procedures, and policies proposed by *Kitchenham* in [54]. According to the latter, this protocol describes the formulation of the research questions (cf. Sect. 3.1), the search string (cf. Sect. 3.2), the data sources chosen for performing the search (cf. Sect. 3.3), the identification of inclusion and exclusion criteria (cf. Sect. 3.4), the quality assessment questions (cf. Sect. 3.5), the selection of studies[3] (cf. Sect. 3.6), the method for extracting

[3]In the given context, a *study* refers to a paper retrieved in the SLR.

the data from the selected studies (cf. Sect. 3.7), and the way how the obtained data shall be analyzed (cf. Sect. 3.8).

## *3.1. Research Questions Formulation*

The overall goal of our SLR is to identify and analyze studies related to business process variability. Note that a detailed understanding of the way process variability is managed in the context of process families requires an in-depth analysis of various aspects; e.g., modeling languages, language constructs, tools, and features [100]. Our SLR focuses on the analysis of relevant papers regarding their expressiveness for modeling process variability, their support for handling process variability along the process lifecycle, and their empirical evaluations. For this purpose, we consider the following research questions, which will be discussed in the following.

- **RQ1:** What underlying *business process modeling languages* are used for modeling process variability?
- **RQ2:** Which *techniques* are used for representing process variability in a *configurable process model*[4] (cf. Sect. 2.2)?
- **RQ3:** What *language constructs* are provided for representing process variability in a *configurable process model*?
- **RQ4:** Which *process perspectives* (cf. Sect. 4.4) are covered by languages that enable the modeling of process variability?
- **RQ5:** What *tools* exist for enabling process variability?
- **RQ6:** What *variability support features* are provided for fostering process variability in all phases of the process lifecycle (cf. Sect. 2.2)?
- **RQ7:** Have existing process variability approaches been *evaluated*? If yes, how does this evaluation look like?
- **RQ8:** In which domains have existing process variability approaches been applied?

[4]Remember that related process variants are defined in terms of a *configurable process model* (cf. Sect. 2.2), which then represents a complete process family.

Since there exists no standard language for modeling process variability, we are interested in identifying what process modeling languages have been used for this purpose (RQ1). As literature refers to various techniques for creating *configurable process models* [11], in addition, the SLR shall provide an overview of the way these techniques are used (RQ2). In order to allow assessing the expressiveness of existing approaches for modeling process variability, the SLR shall further identify a core set of variability-specific language constructs frequently used by these approaches (RQ3).

As illustrated along the check-in process (cf. Example 1), variability may concern different process perspectives (cf. Sect. 4.4). Accordingly, the SLR shall provide insights into the perspectives covered by existing process variability approaches (RQ4). In order to assess the practical applicability of existing process variability approaches, the SLR shall further identify the available tools supporting these approaches (RQ5). Moreover, the SLR shall create an in-depth understanding of variability support features (e.g., to verify and validate process variants) that foster process variability along the different phases of the process lifecycle (RQ6). To assess the level of maturity of existing process variability approaches, we further investigate whether and–if yes–how these approaches have been empirically evaluated (RQ7). Finally, we analyze the domains in which existing process variability approaches have been applied (RQ8).

## *3.2. Search String*

We subjectively elaborate a search string using keywords we derived based on our in-depth knowledge of the topic and taking the defined research questions into account; i.e., we apply subjective search string definition [135]. Since the keywords may be described with synonymous terms [100], we attempt to use a wide range of terms in order to broadly cover the scope of the SLR. These terms are connected through the logical connector OR.

The search string is iteratively refined with the goal to maximize the number of different candidate studies to be retrieved for the SLR. More precisely, several pilot searches are performed in order to refine the keywords in the search string based on a trial and error approach. We exclude terms whose inclusion does not yield additional studies. These pilot searches are continuously inspected by process variability experts in order to ensure that all relevant studies are found.

The search string of our SLR is as follows:

*'process family' OR 'configurable process model' OR 'process model collection' OR 'reference process model' OR 'configurable workflow' OR 'process variant' OR 'business process variability' OR 'process configuration' OR 'process model configuration'*

## *3.3. Data Source Selection*

The defined search string is applied to relevant data sources to find studies related to the topic (i.e., process variability). More precisely, six electronic libraries are identified by topic experts as a basis for conducting the SLR:

1. SpringerLink
2. IEEE Xplore Digital Library
3. ACM Digital Library
4. Science Direct - Elsevier
5. Wiley Inter Science
6. World Scientific

These libraries include the proceedings of the most relevant conferences, workshops and journals the business process management community publishes its research results in; e.g., *Data & Knowledge Engineering*, *Computers in Industry*, *Information Systems*, *Information and Software Technology*, *Conference on Business Process Management (BPM)*, *Conference on Advanced Information Systems Engineering (CAiSE)*, *Working Conference of Business Process Modeling, Development, and Support (BPMDS)*, *IEEE Enterprise Computer Conference (EDOC)*, *International Conference on Cooperative Information Systems (CoopIS)*, *Symposium on Applied Computing (SAC)*, and *International Conference on Service Computing (SCC).*

With the above selection of libraries, we want to retrieve a maximum number of candidate studies from a minimum number of libraries, while reducing the overlap between them as much as possible. In addition, we check whether papers about the topic, which we have already known, are included in the selected libraries as well. As an additional data source, we consider the literature cited by the retrieved studies themselves; i.e., we apply *backward reference searching* [53]. In turn, this improves SLR results by covering a wide spectrum of directly relevant studies. Finally, Google Scholar Alerts (e.g., "process variability") are continuously analyzed in order to become aware of any publication on the topic emerging during the writing process; i.e., after the search in the specified data sources was performed.

Due to the large amount of data sources chosen, the defined search string is suitably adapted where necessary; e.g., through the use of plural forms (e.g., 'process families' instead of 'process family'). In addition, the search string is applied to full text (i.e., title, abstract, and content of the study) in order to ensure that potentially relevant studies are not excluded.

### *3.4. Inclusion and Exclusion Criteria*

We define the following inclusion and exclusion criteria in order to identify relevant studies for our SLR:

***Inclusion criterion***:

1. The study is related to process variability and describes
   - a process variability approach or
   - process variability support features or
   - an empirical evaluation of a process variability approach.

***Exclusion criteria***:

1. The study is not related to process variability, or it merely mentions process variability terms in a generalized manner.
2. The study is not electronically available or requires the payment of access fees.[5]
3. The study refers to a non-peer reviewed publication (e.g., a preface, editorial, or technical report).
4. The study is not presented entirely in English language.
5. The study presents some type of review (e.g., survey, SLR), but does not deal with outcomes of a particular research work.
6. In case several studies refer to the same process variability approach, all studies except the latest and most complete version is excluded.

A study is eliminated if it meets any of these exclusion criteria. Note that we do not apply any restriction with respect to the publication date.

[5]Note that this only applies to fees that are not covered by the subscriptions to any of the selected data sources.

## *3.5. Quality Assessment*

In addition to the inclusion and exclusion criteria introduced in Sect. 3.4, each selected study is assessed based on a set of quality assessment questions (QA). In particular, this is crucial for interpreting and synthesizing the data extracted from the selected studies [54].

- QA1: Does the study include sufficient data to infer how process variability is explicitly modeled?
- QA2: Does the study include sufficient data about the support of process variability in one or several phases of the process lifecycle?
- QA3: Has the process variability approach described in the study been implemented, formalized or empirically evaluated?

These questions are scored as follows: 1 if the question is satisfied and 0 if it is not satisfied. The intention behind this quality assessment is to ensure a certain level of maturity for the studies included in the SLR. Further, we want to guarantee that studies of pure conceptual nature are not included.

## *3.6. Study Selection*

The SLR is conducted by applying the defined search string to each of the six electronic libraries mentioned in Sect. 3.3 (i.e., inclusion criterion). These queries result in a total of 4947 studies (cf. Fig. 5, Stage 1). Metadata related to them is then imported into an Excel file, which stores the source of each study together with its main information, i.e., title, authors, type of venue (e.g., conference, journal), and complete reference of the study. Following this, each of the studies is reviewed in order to determine its relevance for the SLR. Note that this step is accomplished based on the defined exclusion criteria (cf. Sect. 3.4).

First, the title of each retrieved study is analyzed in order to check whether it actually deals with process variability (i.e., Exclusion Criteria 1-5). In cases the information from the title is not sufficient to decide whether or not to include the study, the corresponding abstract and introduction sections are additionally scanned. After this filtering, we obtain 100 relevant studies (cf. Fig. 5, Stage 2). Following this, we analyze the literature cited in the background or related work sections of these 100 studies (i.e., we apply *backward reference searching*). This results in 25 additional studies, which we include for further consideration (cf. Fig. 5, Stage 3). In the next stage,

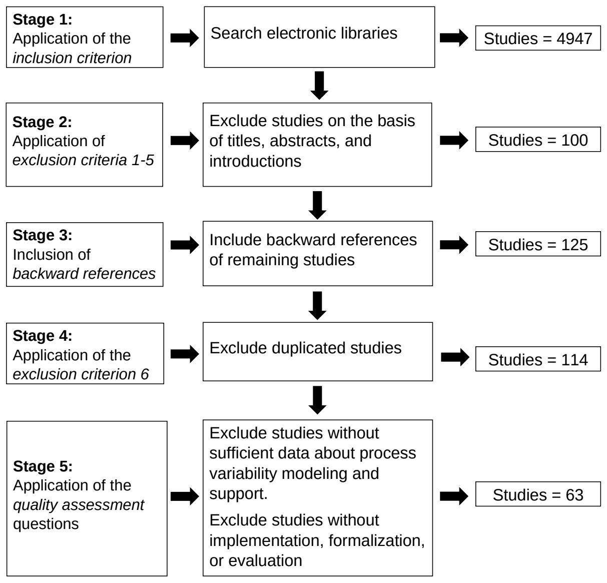

Stage 1:
Application of the inclusion criterion
Search electronic libraries
Studies = 4947
Stage 2:
Application of exclusion criteria 1-5
Exclude studies on the basis of titles, abstracts, and introductions
Studies = 100
Stage 3:
Inclusion of backward references
Include backward references of remaining studies
Studies = 125
Stage 4:
Application of the exclusion criterion 6
Exclude duplicated studies
Studies = 114
Stage 5:
Application of the quality assessment questions
Exclude studies without sufficient data about process variability modeling and support.
Exclude studies without implementation, formalization, or evaluation
Studies = 63

Figure 5: Stages of the study selection process

duplicated studies are removed (i.e., Exclusion Criterion 6) resulting in 114 relevant studies in total (cf. Fig. 5, Stage 4). Finally, studies related to process variability, but without sufficient data about how process variability is modeled or supported or with no tool implementation, formalization or empirical evaluation, are discarded in order to ensure a sufficient level of maturity (i.e., quality assessment questions). Overall, this results in 63 *primary studies* (cf. Fig. 5, Stage 5), which are summarized in Table 1. Each of these studies is associated with a unique identifier (i.e., Study ID), which is used to refer to the respective studies.

During the selection process, we organize these 63 primary studies in three groups:

1. Studies describing process variability approaches: S1 - S34.
2. Studies describing process variability support features: S35 - S50.
3. Studies describing solely empirical evaluations of process variability approaches: S51 - S63.

| **Study ID** | **Study ID** | **Study ID** |
|---|---|---|
| S1-Alférez et al. [10] | S22-Acher et al. [7] | S43-La Rosa et al. [64] |
| S2-Bucchiarone et al. [20] | S23-Reijers et al. [101] | S44-Mahmod et al. [79] |
| S3-Kumar et al. [58] | S24-La Rosa et al. [61] | S45-Gottschalk et al. [41] |
| S4-Frece et al. [36] | S25-La Rosa et al. [65] | S46-Thomas et al. [121] |
| S5-Santos et al. [111] | S26-Montero et al. [85] | S47-Koschmider et al. [57] |
| S6-W. Yao et al. [134] | S27-Moon et al. [86] | S48-Mendling et al. [82] |
| S7-Q. Yao et al. [133] | S28-Gottschalk et al. [40] | S49-Recker et al. [97] |
| S8-Ognjanovic et al. [89] | S29-Lapouchnian et al. [69] | S50-Reinhartz-Berger et al. [102] |
| S9-Gröner et al. [46] | S30-Schnieders et al. [115] | S51-Döhring et al. [34] |
| S10-Boffoli et al. [16] | S31-Lazovik et al. [70] | S52-Derguech et al. [28] |
| S11-Schunselaar et al. [120] | S32-Lu et al. [78] | S53-Lönn et al. [77] |
| S12-Groefsema et al. [44] | S33-Czarnecki et al. [24] | S54-Bulanov et al. [21] |
| S13-Döhring et al. [33] | S34-Becker et al. [15] | S55-Vogelaar et al. [126] |
| S14-Park et al. [90] | S35-van der Aalst et al. [6] | S56-Reinhartz-Berger et al. [104] |
| S15-Nguyen et al. [88] | S36-Li et al. [74] | S57-Scherer et al. [114] |
| S16-Pascalau et al. [92] | S37-Weber et al. [129] | S58-Pascalau et al. [91] |
| S17-Meerkamm et al. [84] | S38-Derguech et al. [27] | S59-Baier et al. [14] |
| S18-Derguech et al. [26] | S39-Yahya et al. [132] | S60-Gottschalk et al. [43] |
| S19-Hallerbach et al. [49] | S40-Koetter et al. [55] | S61-La Rosa et al. [63] |
| S20-de la Vara et al. [25] | S41-Gröner et al. [45] | S62-Schnieders et al. [116] |
| S21-Reinhartz-Berger et al. [103] | S42-van der Aalst et al. [4] | S63-Giese et al. [38] |

Table 1: Final list of primary studies

The specified selection process is carried out by the main author of this paper and continuously checked by her co-authors. More precisely, the co-authors randomly review selected studies to ensure consistency of the process. Further, they ensure the correct application of the inclusion and exclusion criteria as well as the quality assessment questions. All disagreements are resolved through discussion.

## *3.7. Data Extraction Strategy*

To each of the 63 primary studies (cf. Sect. 3.6), a *data extraction process* is applied with the goal to answer the research questions defined in Sect. 3.1. For this purpose, we use Excel sheets to capture and store the relevant information. Appendix A includes an excerpt of these sheets.[6] In detail, we extract the following information:

1. General information about the study; i.e., title, authors, type of venue (e.g., conference, journal), and complete reference of the study.
2. The underlying language used for modeling process variability; e.g., BPMN or EPC (RQ1).

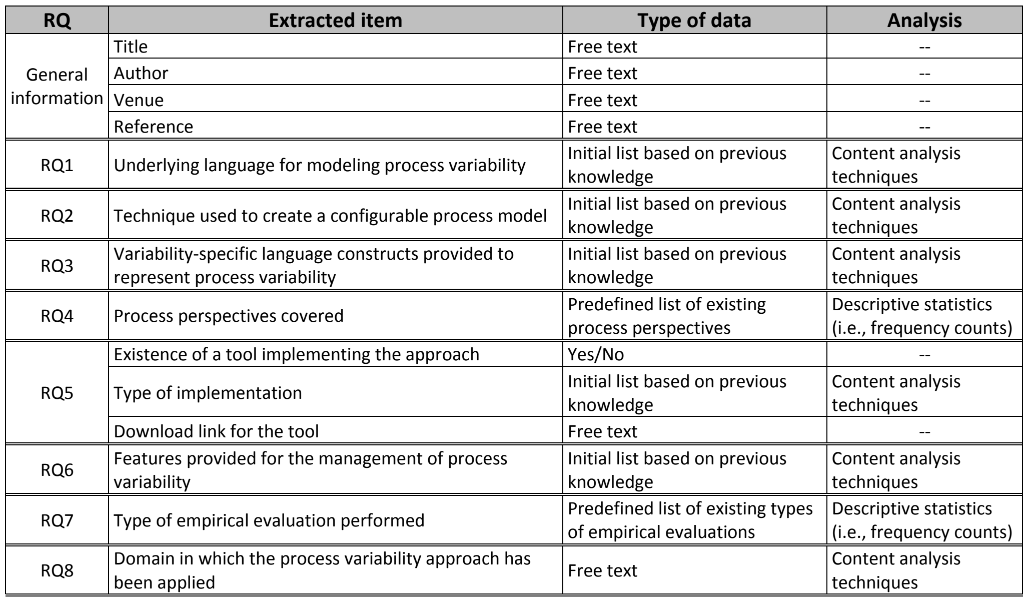
[6]The filled Excel sheets can be downloaded from: `http://www.pros.upv.es/bpvar/SLR/SLRDataExtraction.rar`

3. The technique used to define a configurable process model (RQ2).
4. Variability-specific language constructs that may be used to represent process variability (RQ3).
5. The process perspectives covered; e.g., behavioral, organizational, and informational (RQ4).
6. Information about the implementation of the approach; i.e., availability of a tool implementing the approach, type of implementation, and link for downloading this tool (RQ5).
7. Features provided for the management of process variability (RQ6).
8. Available results from empirical evaluations of a process variability approach and type of evaluation performed (e.g., case study, survey) (RQ7).
9. Domain in which the process variability approach has been applied (RQ8).

For research questions RQ1, RQ2, RQ3, RQ5, and RQ6, data is extracted by first creating an initial list of categories based on our knowledge and experience about the topic (i.e., process variability) [11, 12, 127, 129]. Once data extraction starts, each study is then thoroughly analyzed and extracted data is assigned to a category based on content analysis techniques [51]; if new categories are identified, they are added to the list. Throughout the analysis, process categories might be merged. In this case, already analyzed studies are re-assigned. However, regarding RQ4 (i.e., process perspectives covered), we start data extraction with a predefined list of the existing process perspectives (cf. Sect. 4.4). Then, we assign each study to the process perspectives it covers; i.e., we use descriptive statistics to analyze the results (i.e., frequency counts). A similar procedure is applied in the context of RQ7. We first create a predefined list of existing types of empirical evaluations. Then, each study is assigned to the type of evaluation it describes. Finally, for RQ8, we include each identified domain in which existing process variability approaches have been applied by analyzing the content of each study. Again, throughout this analysis similar domains might be merged. This may imply the reassignment of already analyzed studies. Fig. 6 summarizes the data extracted and the techniques used for data analysis.

### *3.8. Data Analysis*

Data analysis shall provide suitable information to answer our research questions (cf. Sect. 3.1). This is achieved by synthesizing the data obtained



| RQ | Extracted item | Type of data | Analysis |
|---|---|---|---|
| General information | Title | Free text | -- |
| | Author | Free text | -- |
| | Venue | Free text | -- |
| | Reference | Free text | -- |
| RQ1 | Underlying language for modeling process variability | Initial list based on previous knowledge | Content analysis techniques |
| RQ2 | Technique used to create a configurable process model | Initial list based on previous knowledge | Content analysis techniques |
| RQ3 | Variability-specific language constructs provided to represent process variability | Initial list based on previous knowledge | Content analysis techniques |
| RQ4 | Process perspectives covered | Predefined list of existing process perspectives | Descriptive statistics (i.e., frequency counts) |
| RQ5 | Existence of a tool implementing the approach | Yes/No | -- |
| | Type of implementation | Initial list based on previous knowledge | Content analysis techniques |
| | Download link for the tool | Free text | -- |
| RQ6 | Features provided for the management of process variability | Initial list based on previous knowledge | Content analysis techniques |
| RQ7 | Type of empirical evaluation performed | Predefined list of existing types of empirical evaluations | Descriptive statistics (i.e., frequency counts) |
| RQ8 | Domain in which the process variability approach has been applied | Free text | Content analysis techniques |

Figure 6: Data extraction summary

from the data extraction process. More precisely, the respective research questions are answered by analyzing the identified categories based on the created Excel sheets as well as the results of the quality assessment process. In addition, for answering RQ1-RQ4, only studies of the first type (i.e., S1-S34) are considered since they describe the expressiveness of the respective approach regarding the modeling of process variability. In turn, for answering RQ5 and RQ6, studies of the first and second type are analyzed (i.e., S1-S50) since both types might deal with implementation support for process variability. Finally, for RQ7 and RQ8, studies of the first (i.e., S1-S34) and third (i.e., S51-S63) type are considered since they might provide empirical evaluations of process variability approaches.

In order to simplify the synthesis of the extracted data, we use descriptive techniques to summarize them; e.g., graphics and tabular descriptions.

## 4. Results

This section presents the major results we obtained from the SLR. Fig. 7 shows the temporal distribution of the 63 primary studies by publication year (i.e., from 2004 to 2013). As can be seen, the yearly number of published studies on process variability has increased over time, with a peak in 2012. This indicates a growing interest in the topic (i.e., process variability). The low number of studies published in 2013 might be explained by the fact that the search was conducted during this year, and thus only studies published early in this year are included.

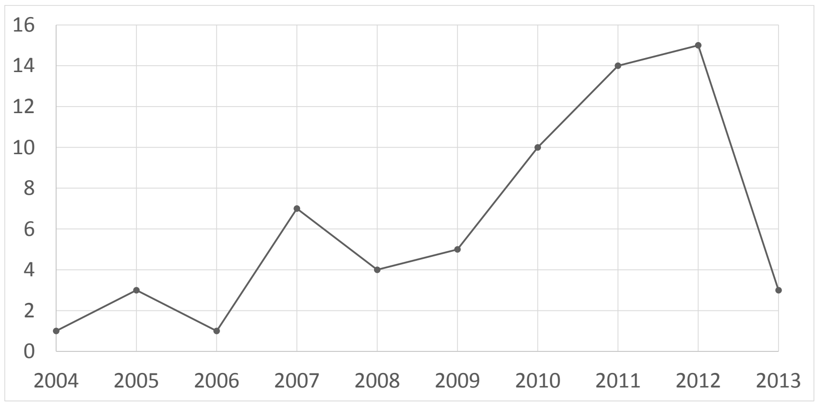


Figure 7: Distribution of primary studies by publication year

An additional analysis is performed regarding the publication venue in which the primary studies are published (primary studies published as book chapters are not taken into account in this analysis; therefore the latter is based on 60 out of the 63 studies). 42 of the primary studies are published in proceedings of conference and workshops (70%), while 18 studies (30%) appeared in journals (cf. Fig. 8). A total of 35 publication venues are identified. Interestingly, there are only two publications venues, namely *BPM* (*Conference on Business Process Management*) and *BPMDS* (*Working Conference of Business Process Modeling, Development, and Support*) with a relatively high number of primary studies (i.e., 4 and 6 studies respectively). All other venues published at most three studies on the topic. Finally, it is noteworthy that process variability is a topic addressed in various fields, i.e., primary studies have been published in a wide range of publication venues from different fields; i.e., publication venues from the business process management field (e.g., *BPM* conference), the web services field (e.g., *ICSOC* conference), the software engineering field (e.g., *JSS* journal), and the information systems field (e.g., *CAiSE* conference). Appendix B includes the list with the full names of the publication venues.

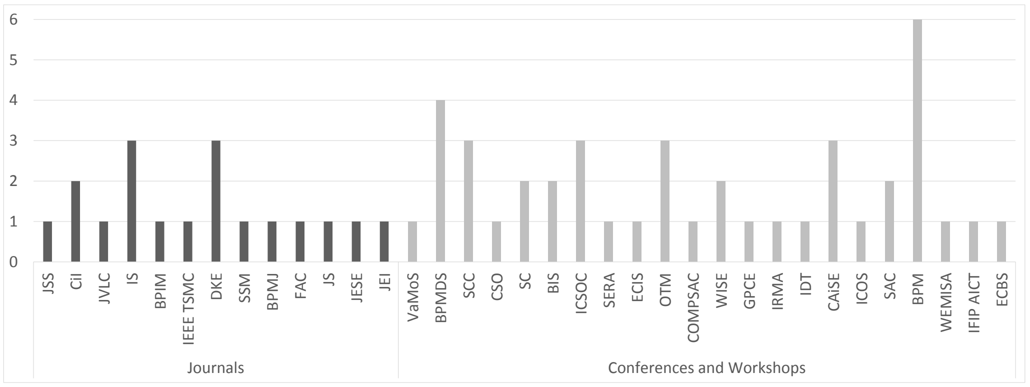


Figure 8: Distribution of primary studies by publication venue

The following sections show the detailed results of our SLR answering each research question separately. Section 4.1 deals with the *modeling languages* used to represent process variability (RQ1). While Sect. 4.2 shows how the different *techniques* for representing process variability in a *configurable process model* are used (RQ2), Sect. 4.3 describes the set of *variability-specific language constructs* identified in the context of the SLR (RQ3). In turn, the *process perspectives* covered by existing variability approaches (RQ4) are summarized in Sect. 4.4. In addition, existing *tools* for managing process variability (RQ5) are presented in Sect. 4.5. Section 4.6 describes the *variability support features* provided in the context of process variability (RQ6). Further, it categorizes them along the different phases of the process lifecycle (cf. Sect. 2.2). While Sect. 4.7 discusses to what extent the presented approaches have been *evaluated* (RQ7), Sect. 4.8 gives insights into the domains in which the process variability approaches have been applied (RQ8). Finally, Sect. 4.9 analyzes aspects cutting across the results of the research questions.

## *4.1. Languages for Modeling Business Process Variability*

We first present the SLR results related to RQ1 (*What underlying business process modeling languages are used for modeling process variability?*). In order to answer this research question, we analyze the group of studies describing process variability approaches (i.e., S1-S34). In particular, these studies refer to the expressiveness of existing approaches with respect to the modeling of process variability. In this section, we focus on the languages they use as basis for modeling process variability.

Figure 9 shows the distribution of the 34 studies according to the modeling languages they use for representing both the *commonalities* (i.e., process fragments shared by all process variants) and *variations* of the members of a process family (i.e., the process variants).

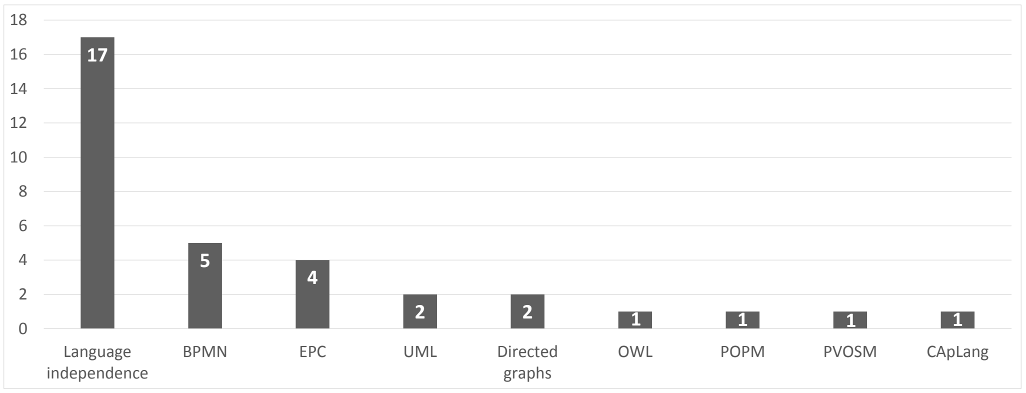


Figure 9: Distribution of studies S1-S34 according to the process modeling language used

As can be seen, 17 studies are conceived to be independent of a particular process modeling language; i.e., S1, S3, S5, S9, S10, S15, S18, S19, S20, S22, S24, and S28-S33. For example, these studies propose the use of feature models, ontologies, rules, or hierarchical indexing structures in order to capture and model process variability. In turn, respective approaches can be used in combination with any process modeling language (e.g., BPMN, EPC, or UML Activity Diagrams) for properly representing process variability.

On the contrary, the other 17 studies propose approaches that extend existing (process) modeling languages with specific constructs for modeling process variability, or that design proprietary languages for this purpose. In particular, 11 studies propose conceptual extensions of existing process modeling languages such as BPMN (i.e., S4, S8, S13, S16, S26), EPC (i.e., S21, S23, S25, and S34), and UML Activity Diagram (S14, S27) in order to enable the explicit modeling of process variability. In turn, 6 studies either make use of languages such as Directed Graphs (S11 and S12) or

OWL[7] (S6), which are common in other fields, or they propose proprietary languages developed for the modeling of process variability; i.e., CApLang (S2), PVOSM (S7), and POPM (S17).

## *4.2. Techniques for Modeling Process Variability in a Configurable Process Model*

We now consider RQ2 (*Which techniques are used for representing process variability in a configurable process model?*). The SLR identifies two techniques that may be used to model process variability (cf. Fig. 10). In particular, these techniques either allow capturing the entire process family (i.e., all process variants) in a single model artifact (i.e., *single artifact technique*) or in a set of related model artifacts (i.e., *multi-artifact technique*). The latter may represent different aspects of the process family, e.g., commonalities of the process variants, variant-specific parts, configuration constraints, and application context. In order to answer RQ2, again we analyze the 34 studies describing process variability approaches (i.e., S1-S34). In particular, these studies refer to the expressiveness of existing approaches regarding the modeling of process variability.

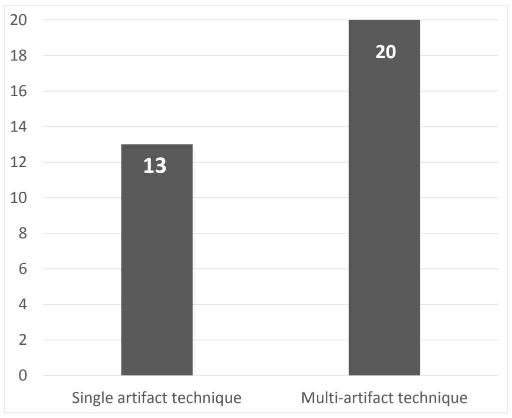


Figure 10: Distribution of studies S1-S34 according to the process variability modeling technique used

The *single artifact technique* has been realized by various studies based on different methods (cf. Fig. 11). The latter include *hiding & blocking* (S11),

[7]Web Ontology Language, `http://www.w3.org/TR/owl-features/`

*configurable nodes* (S7, S25, and S28), and *logic formulae* (S12). Furthermore, *annotations* for BPMN (S4 and S13), *labels* for EPC (S23 and S34), and *meta-model extensions* for UML Activity Diagrams (S27) and BPEL (S31) have been proposed in order to realize configurable process models. Finally, *multiplicity indicators* (S21) and *hierarchical indexing structures* (S18) constitute two specific methods for representing a configurable process model in terms of a single artifact. Note that all these methods enrich the configurable process model with additional information (e.g., *configuration constraints*) in order to guide users when deriving process variants.

In turn, the *multi-artifact technique* has been realized in the following studies: S1, S3, S5, S6, S8, S9, S10, S14-S17, S19, S20, S22, S24, S26, S29, S30, S32, and S33. Basically, approaches using this technique represent a process family in terms of four different modeling artifacts. The latter include a *base model*, a set of *variable process fragments*, *rules* for adapting the base model through adding/deleting the variable process fragments, and an *application context* determining when these rules apply. Thereby, the *base model* is specified using a particular business process modeling language (e.g., BPMN). However, different policies may be applied when defining a base process model, e.g., setting the latter to the most frequently used process variant or to the process model having minimum average edit distance to the process variants of the process family [74].

Concerning the three other artifacts (i.e., variable process fragments, rules to adapt the base model, and application context), different methods for defining them exist (cf. Fig. 11). In turn, these methods are based on specific techniques from various fields (e.g., software product lines, semantic web, and requirements engineering), or they are explicitly designed for the process variability approach at hand. For representing variable process fragments, for example, *features models*, as known from software product lines, can be used (cf. studies S1, S8, S9, S10, S14, S15, S22, S26, S30, and S33). In turn, in the requirements engineering field, S29 refers to *goal models* that may be applied to represent variability at a high level of abstraction. Finally, variable process fragments may be defined based on a set of *process model components* (S16), a *variant list* (S17), or a set of pre-specified *change operations* (S19 and S32).

In turn, the rules for adapting the base model may rely on methods such as *business rules* (S3) and *process model queries* (S16). The approach described by study S29, for example, uses *non-functional constraints* for deriving process variants.

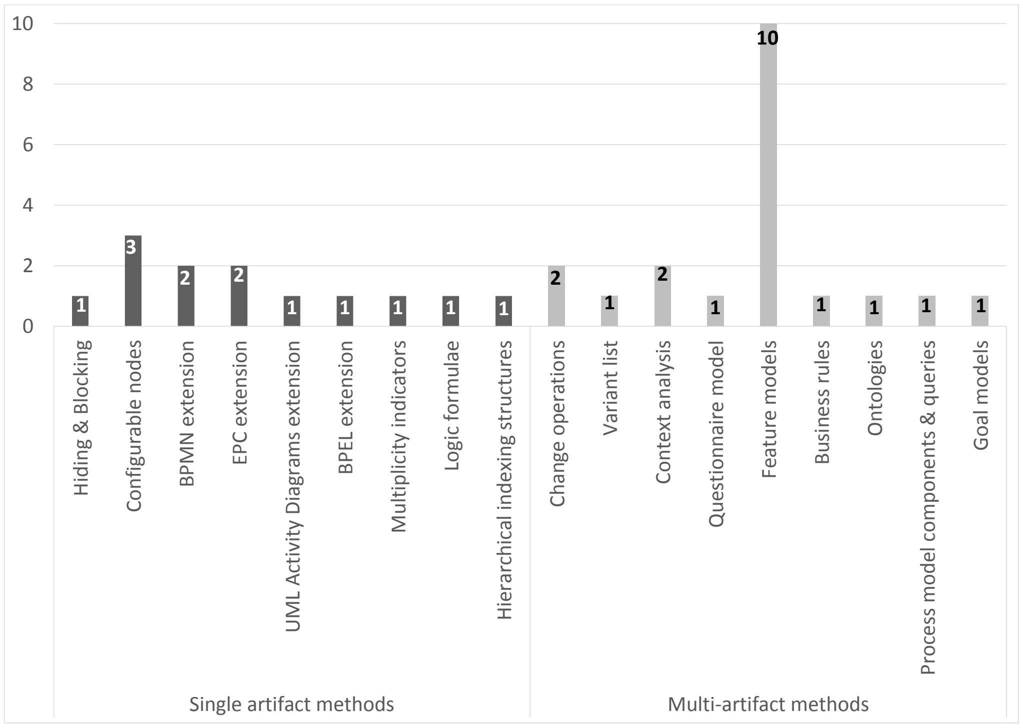


Figure 11: Distribution of studies S1-S34 according to the single and multi-artifact method used

Finally, for defining an application context, studies S1 and S6 use *ontologies* (described in the OWL language) and *semantic rules.* Finally, S5 and S20 redefine a *context analysis method* from a business process perspective in order to analyze context properties.

Note that study S2 has not been considered in the above classification since it describes process variability at the program code level, i.e., S2 does not use process models to represent process variability. Thus, it cannot be classified in any of the described techniques due to the absence of a configurable process model.

## *4.3. Language Constructs for Process Variability*

Regarding RQ3 (*What language constructs are provided for representing process variability in a configurable process model?*), we identified five variability-specific *language constructs* in the SLR: *configurable region, con-*

*figuration alternative, configuration context condition, configuration constraint,* and *configurable region resolution time.* These constructs abstract from concrete process variability approaches since they are defined at a higher level of abstraction. In the following, we describe the identified variability-specific language constructs and illustrate them along the check-in process. Note that for obtaining these constructs, again we only analyze studies describing process variability approaches (i.e., S1-S34).

**Language Construct LC1 (Configurable Region).** A *configurable region* corresponds to a region of a *configurable process model* for which different configuration choices exist, depending on the application context. Studies supporting language construct LC1 include S1-S5, S7-S23, and S25-S34.

**Example 8 (Configurable Region).** Regarding the check-in process (cf. Example 1), activity *Pay extra fee* is only performed if the luggage has overweight. Otherwise, it is skipped. Consequently, at the respective position of the *configurable process model*, there exist two choices depending on the weight of the luggage; i.e., either perform the activity or skip it. Accordingly, the respective position of the *configurable process model* constitutes a *configurable region.*

**Language Construct LC2 (Configuration Alternative).** A *configuration alternative* corresponds to a particular configuration choice that may be selected in the context of a specific *configurable region* (LC1). In general, respective alternatives may refer to any process perspective; i.e., the functional, behavioral, organizational, informational, temporal, and operational perspectives (cf. Sect. 2). The studies that support this construct include S1-S5, S7-S23, and S25-S34. However, note that they do not support *configuration alternatives* with respect to all process perspectives.

**Example 9 (Configuration Alternative).** Several *configuration alternatives* exist for the check-in process. Regarding the behavioral perspective, for example, before performing activity *Print boarding card*, each activity that may be performed or skipped constitutes a *configuration alternative*, e.g., activity *Fill in ESTA form.* Concerning the organizational perspective, there exist different roles that may perform the *Print boarding card* activity, i.e., the passenger himself via the web system, the self-servicing machine, or the airline personnel at the check-in counters (i.e., each role constitutes a *configuration alternative*). The *configuration alternatives* related to the informational perspective refer to the different types of boarding cards resulting from the check-in process (e.g., electronic versus paper-based). Finally, *configuration alternatives* of the temporal perspective refer to the start events "23 hours before departure" or "3 hours before departure".

**Language Construct LC3 (Configuration Context Condition).** A *configuration context condition* defines the conditions under which a particular *configuration alternative* (LC2) of a *configurable region* (LC1) shall be selected. Studies that consider this construct include S1, S2, S5, S6, S8, S13-S15, S19, S20, S24, S28, S30, S33, and S34.

**Example 10 (Configuration Context Condition).** Before activity *Print boarding card* will be performed in a check-in process (cf. Figs. 1 and 2), different alternatives exist. For example, activity *Fill in ESTA form* is only performed if the passenger is traveling from EU to US. In this case, the *configuration context condition* "flight destination" determines whether or not this activity will be performed.

**Language Construct LC4 (Configuration Constraint).** A *configuration constraint* is defined as a restriction regarding the selection of *configuration alternatives* (LC2). Respective constraints are based on semantic restrictions to ensure the proper use of the defined *configuration alternatives* (e.g., exclusion or inclusion relationships). The studies supporting this language construct include S2, S5-S10, S12, S14-S17, S19, S21, S22, S24-S29, and S31-S33.

**Example 11 (Configuration Constraint).** Regarding the check-in process, activity *Localize assistance to accompany passenger* shall be performed when handicapped passengers check-in. Accordingly, a *configuration constraint* is or needs to be introduced in the *configurable process model* to express that this activity shall be only performed if the passenger is a handicapped person. Otherwise, the activity shall be skipped.

**Language Construct LC5 (Configurable Region Resolution Time).** The *configurable region resolution time* allows modelers to distinguish between *configurable regions* (LC1) whose configuration either depends on the initial or the current context of a process instance (i.e., configuration or enactment time). Studies supporting this construct include S15, S28 and S32.

**Example 12 (Configurable Region Resolution Time).** Regarding the check-in process, the process variant specifying the online check-in may be configured at configuration time by selecting the activities referring to the *web system* role. However, the activity related to the overweight luggage (i.e., *Pay excess fee*) is only performed if the passenger places the luggage at the desk scales. In this case, the decision whether or not this activity will be performed is postponed to enactment time.

Fig. 12 summarizes which studies support which variability-specific constructs. As shown, *configurable regions* (LC1) and *configuration alternatives* (LC2) are supported by 32 (out of 34) studies. In turn, *configuration context conditions* (LC3) are covered by 15 studies, while 24 studies consider the definition of *configuration constraints* (LC4). Finally, only 3 studies allow specifying the *configurable region resolution time* (LC5).

Interestingly, only 2 studies cover the entire set of the language constructs we identified (i.e., LC1-LC5): S15 and S28. In turn, 7 studies cover four language constructs (e.g., LC1, LC2, LC3, LC4); i.e., S2, S5, S8, S14, S19, S32, and S33. Altogether, Fig. 12 confirms the high relevance of the five language constructs in respect to the explicit modeling of process families and the variability inherent to them.

| | | **Studies supporting the language construct** |
|---|---|---|
| **Variability-specific language constructs** | LC1 Configurable Region | S1-S5, S7-S23, S25-S34 |
| | LC2 Configuration Alternative | S1-S5, S7-S23, S25-S34 |
| | LC3 Configuration Context Condition | S1, S2, S5, S6, S8, S13-S15, S19, S20, S24, S28, S30, S33, S34 |
| | LC4 Configuration Constraint | S2, S5-S10, S12, S14-S17, S19, S21, S22, S24-S29, S31-S33 |
| | LC5 Configurable Region Resolution Time | S15, S28, S32 |

Figure 12: Variability-specific language constructs and studies supporting them

## *4.4. Covered Process Perspectives*

This section describes the SLR results in respect to research question RQ4, which refers to the process perspectives covered (*Which process perspectives are covered by languages that allow for the modeling of process variability?*). For answering RQ4, we re-analyze the studies describing process variability approaches (i.e., S1-S34).

As illustrated by Fig. 13, the most frequent process perspectives covered by existing process variability approaches are the *functional* and *behavioral* ones. As shown, 33 studies consider both process perspectives; i.e., the respective approaches define process variability in respect to the control flow perspective (i.e., S1-S24 and S26-S34). In turn, the *informational* perspective is only considered by 7 studies (i.e., S3, S4, S15, S17, S25, S32, and S34) and the *organizational* one by 5 studies (i.e., S3, S17, S22, S25, and S34). Finally, none of the identified studies considers variability of the *temporal* or *operational* perspective.

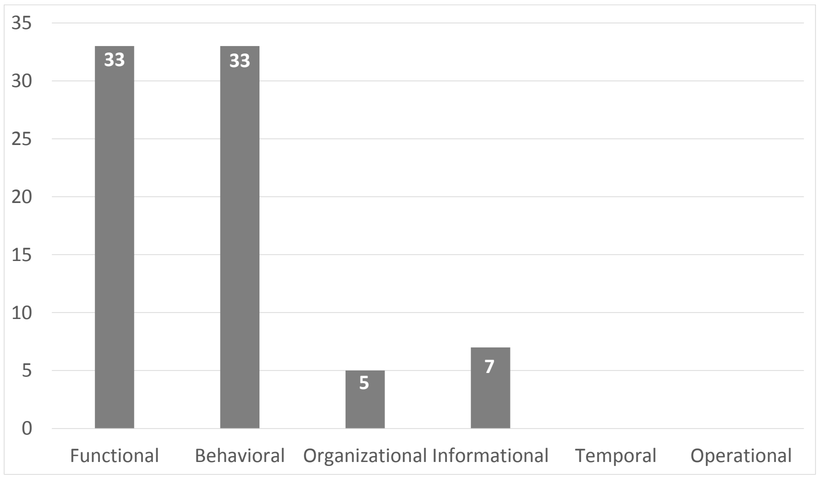


Figure 13: Distribution of studies S1-S34 according to the process perspectives covered

As can be seen in Fig. 13, most studies solely cover variability with respect to control flow (i.e., the *functional* and *behavioral* perspectives). However, studies S3, S17 and S34 are more complete covering four perspectives (i.e., *functional, behavioral, organizational,* and *informational*). Finally, studies S4, S15, S22, and S32 at least cover 3 different perspectives (e.g., *functional, behavioral,* and *informational*).

## *4.5. Existing Tools for Managing Process Variability*

This section deals with available tools providing support for process variability (i.e., RQ5: *What tools exist for enabling process variability?*). For answering research question RQ5, we analyze the studies describing process variability approaches (i.e., S1-S34) and variability support features (i.e., S35-S50). In particular, these studies might refer to available tool implementations.

The SLR reports on 41 tools for managing process variability. Out of them, however, only 10 are available online; i.e., these tools can be downloaded from websites (including manuals and tutorials); i.e., S2, S11, S13, S22, S24-S26, S28, S29, and S33. Figure. 14 lists these tools.

Furthermore, the SLR reveals that most tool implementations constitute proof-of-concept prototypes not yet ready for industrial adoption; i.e., they were developed with the goal to validate the feasibility of the proposed approaches. Besides this, the kind of tool differs, depending on the objective of the respective study. In detail, existing implementations provide graphical editors for modeling process variability (S29), model transformations that allow generating the entire family of process variants from a feature model (S26), and extensions of existing process modeling languages for explicitly representing variability (S5, S13). In some studies, these implementations are realized as an Eclipse plug-in (S1, S8, S15, S25, S27, S28) or a proprietary Java tool (S3 and S24).

Other implementations integrate existing tools for realizing a particular process variability approach. Examples include S6, S9 and S33, which implement a set of plug-ins to integrate a feature model editor with the Protègè tool, a transformation from BPMN to Description Logic, and Rational Software Modeler, respectively. Finally, other process variability approaches are implemented by extending commercial BPM suites such as *ARIS Architect* (S19, S23), *IBM Rational Software Architect* (S30), and *WebSphere BPEL4WS* (S31).

## *4.6. Variability Support Features*

This section summarizes the variability support features extracted in the context of RQ6 (*What variability support features are provided for fostering process variability in all phases of the process lifecycle?*). For answering this research question, we analyze studies describing process variability approaches (i.e., S1-S34) and variability support features (i.e., S35-S50). We organize the features along the phases of the process lifecycle (cf. Sect. 2.2).

| Study ID | Reference |
|---|---|
| S2 | soa.fbk.eu/node/218 |
| S11 | www.promtools.org/prom6/ |
| S13 | www.markus-doehring.de/phd/index.php?option=com_content&view=article&id=59&Itemid=63 |
| S22 | modalis.polytech.unice.fr/softwares/manvarwor |
| S24 | www.processconfiguration.com/download.html |
| S25 | www.processconfiguration.com/download.html |
| S26 | www.eclipse.org/m2m/atl/atlTransformations/#FM2BPMN |
| S28 | www.yawlfoundation.org/ |
| S29 | https://se.cs.toronto.edu/trac/ome/wiki |
| S33 | gp.uwaterloo.ca/fmp2rsm |

Last accessed: April, 2014

Figure 14: Downloading links of available tools

**Phase I: Analysis & Design**

In the *analysis & design* phase, a process family is designed, modeled, validated, and verified using a particular process variability approach (cf. Sect. 2.2). In this context, language constructs such as the ones introduced in Sect. 4.3 are provided in order to specify and represent the common as well as the variable parts of the process variants of a process family in a *configurable process model*. Relevant features identified for this phase are as follows:

**Feature F1.1 (Modeling a configurable process model).** Tool support is needed for designing a *configurable process model* that represents an entire process family (i.e., the collection of all process variants). In this context, we must consider all language constructs introduced (cf. Sect. 4.3) as well as appropriate tool support for them. Since graphical process models are usually more comprehensible than non-graphical ones [131], in addition, graphical editors, navigation features, and visualization support are required to facilitate the creation of such models. Studies S1, S3, S4, S6, S7, S9, S10, S12-S23, S25-S28, and S30-S34 provide various techniques supporting this feature.

**Feature F1.2 (Verifying a configurable process model and its related process family).** Efficient techniques are needed in order to ensure that configurable process models are *syntactically* correct and *behaviorally* sound.[8] This means, it must be guaranteed that solely syntactically correct

[8]Regarding a sound process, all activities may be executed in the context of at least

and behaviorally sound process variants can be derived from a *configurable process model*. The verification may be accomplished during the creation of the *configurable process model* or latter. Feature F1.2 is considered by studies S23, S32, S35, S40, S41, S48, and S49.

**Feature F1.3 (Validating a configurable process model).** Techniques are needed for validating the *semantic* correctness of *configurable process models*. In particular, it must be ensured that a *configurable process model* properly covers all relevant variants of a business process. Again, such a validation may be accomplished during the creation of a *configurable process model* or afterwards. Studies S42 and S46 use logic formulas to address this issue.

**Feature F1.4 (Evaluating the similarity of different process variants).** In order to reduce modeling efforts, techniques for determining the similarity between related process variants are needed. Before adding a process variant to a process family, for example, it needs to be checked whether a similar process variant already exists. This is crucial in order to avoid redundancies and duplications. Studies S32, S44 and S47 provide methods and algorithms for this purpose.

**Feature F1.5 (Merging process variants).** In order to avoid mode redundancy and foster model reusability, techniques for integrating (i.e., merging) a collection of related process variants in a *configurable process model* are needed. Corresponding techniques are provided by studies S11, S23, S38, S39, and S43. Usually, a *configurable process model* resulting from their application covers the behavior of all process variants merged. In addition, reversibility techniques that allow deriving any of the input process variants from the *configurable process model* through individualization are useful as well. Studies S11, S38 and S43 describe methods for realizing such reversibility; i.e., they ensure the traceability of each variant after having performed the merging process.

**Phase II: Configuration**

The goal of the *configuration* phase is to derive an executable process variant (i.e., a member of the process family) through a configuration of the *configurable process model*. This is denoted as *individualization* process. Furthermore, the resulting process variant then needs to be deployed on the enact-

one process instance and no deadlocks or livelocks may occur.

ment system (e.g., workflow management system). The SLR reveals Feature F2 for this phase:

**Feature F2 (Configuring a process variant out of a configurable process model).** In general, tools should provide sophisticated user interfaces. Furthermore, proper techniques for retrieving the current application context and deriving an appropriate process variant for it are required. On one hand, algorithms for checking the syntactical correctness and soundness of configured process variants as well as their conformance with the specified *configuration constraints* (cf. Sect. 4.3) are required. For example, *inclusion (exclusion)* constraints may enforce (exclude) *configuration alternatives* with respect to a specific *configurable region*. In particular, users should be prohibited from deriving invalid process variants, e.g., by informing them about constraint violations. On the other hand, techniques enabling a high level of abstraction are required when specifying a particular application context; i.e., the configuration of a particular process variant should be accomplished at a high level of abstraction. Furthermore, the process variant resulting from a configuration (i.e., individualization) procedure should be graphically displayed to users. This feature is supported by studies S1, S8-S11, S14, S15, S17, S21-S26, S28-S30, S33, S34, S41, and S50 based on different techniques for configuring a process variant. For example, studies S24 and S25 provide a *questionnaire model* (i.e., form-based questionnaire) that allows individualizing a *configurable process model* by answering questions about the respective application context. Another well-known configuration technique is provided by *feature models* (e.g., S41 and S50). The latter map features to *configuration alternatives*; i.e., when a feature is selected, the *configurable process model* becomes configured automatically. Other techniques we discovered with respect to the support of the configuration phase include *configuration algorithms* (e.g., S8 and S23), *goal models* (S29), and *decision tables* (S10). Fig. 15 illustrates abstract examples of these techniques. To be more precise, it depicts a *questionnaire model* (part A), a *feature model* (part B), a *goal model* (part C), and a *decision table* (part D).

**Phase III: Enactment**

During the *enactment* phase, instances of a (configured) process variant may be created, started and executed. In this context, it should be possible to dynamically configure or re-configure a process variant if required [11]; i.e., to switch from one process variant to another during enactment time. Note that such dynamic (re-)configuration might become necessary due to con-

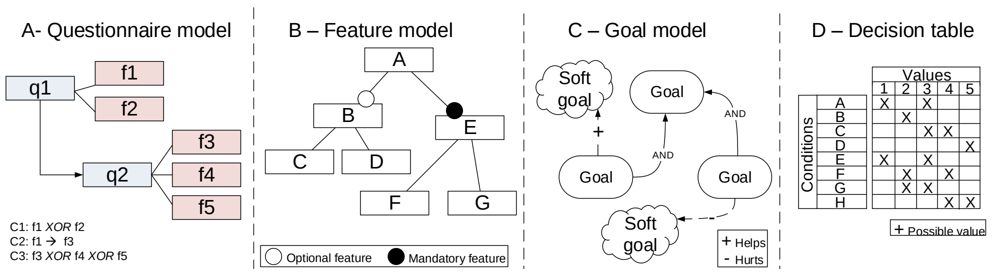


Figure 15: Example of configuration techniques

textual changes occurring during enactment time [10]. However, a dynamic (re-)configuration must be accomplished in a controlled and robust manner. For example, automated re-configurations of a sound process variant instance should always result in a sound process variant instance again. Note that this differs from ad-hoc changes as supported in adaptive PAISs [100]. Usually, ad-hoc changes correspond to unplanned dynamic changes, whereas a dynamic re-configuration switches the execution of a process instance from its current variant model to another pre-specified one. In detail, the SLR reveals the following features for the enactment phase:

**Feature F3.1 (Configuring specific regions of a process variant at enactment time).** Certain configuration decisions can solely be made at enactment time when required data becomes available. In order to address this issue, late modeling techniques [100] for configuring process variants at enactment time are provided by S4, S5, S13, and S32. This feature is illustrated by Fig. 16: A part of the process variant (indicated as activity X) is deemed to be of dynamic nature. In particular, X is defined based on a set of activities (i.e., C, D, E, F, G, and H) and a corresponding set of constraints restricting their use (presented as logic formulas in Fig. 16). During enactment time, for a given process variant instance, X may be concretized based on available context data or user decisions. For the given scenario, Fig. 16 shows a particular design of the process for which X is substituted by a process fragment composed out of activities G and H. Note that this substitution is valid in terms of the prescribed constraints (e.g., G and H *exclude* each other).

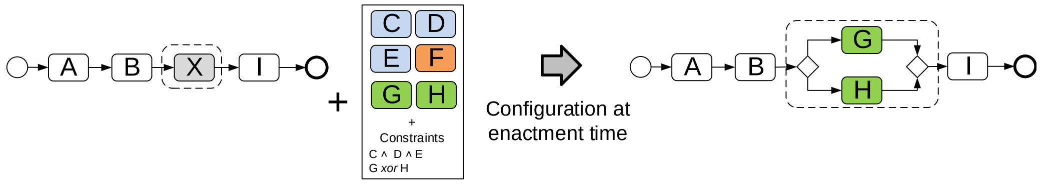


Figure 16: Configuring a specific region of a process variant at enactment time

**Example 13 (Configuring specific regions of a process variant at enactment time).** Whether or not a passenger carries overweight luggage is not known until she arrives at the counter. Hence, the *Pay extra fee* activity may only be selected when enacting instances of the respective process variant.

**Feature F3.2 (Dynamically re-configuring an instance of a process variant at enactment time).** For a particular instance of a process variant, it might become necessary to dynamically switch its execution from the current process variant model to another pre-specified one. Such dynamic re-configurations might be required, for example, to react to contextual changes [100]. Studies S1, S2, S12, and S19 provide techniques that support this advanced feature. Fig. 17 illustrates it for a process variant instance that is dynamically re-configured to another variant model (i.e., the execution of activity F substitutes the one of activities C, D and E).

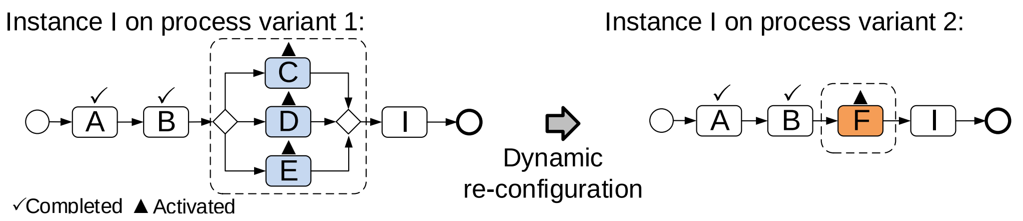

Instance I on process variant 1:
A
B
C
D
E
I
✓Completed ▲Activated
Dynamic
re-configuration
Instance I on process variant 2:
A
B
F
I

Figure 17: Dynamically re-configuring an instance of a process variant

**Example 14 (Dynamically re-configuring an instance of a process variant at enactment time).** Regarding the check-in process, changes of the passenger status might require dynamic variant switches. For example, consider 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 later, a switch to another process variant needs to be performed.

**Phase IV: Diagnosis**

In the *diagnosis* phase, a collection of configured process variants is analyzed to learn from the configuration settings made at design and enactment time.

**Feature F4 (Analyzing a collection of process variants).** Techniques for learning from the configuration settings chosen when configuring the process variants at design or enactment time are needed; i.e., by analyzing the structure as well as the behavior of a given collection of process variants, an improved *configurable process model* might be obtained. Studies providing support for this advanced feature include S36 and S45.

**Phase V: Evolution**

This phase deals with the *evolution* of a process family and the *configurable process model* representing its members in order to cope with changing and evolving requirements. Examples of such evolutionary changes include the addition of new process variants (i.e., variants that cannot be configured out of the *configurable process model* so far), the removal of existing ones (i.e., process variants that must no longer be configured), and the modification of existing process variants to increase their quality (e.g., to improve model comprehensibility). In order to enable such an evolution, a *configurable process model* must be changed accordingly. In this context, the SLR reveals the following features:

**Feature F5.1 (Versioning of a configurable process model).** Techniques allowing for the co-existence of different versions of the same *configurable process model* are needed, particularly in the context of long-running processes. For example, study S46 presents a method using *version-graph models* to support this feature. Fig. 18 shows four versions of a *configurable process model* and the associated *version graph* to manage them.

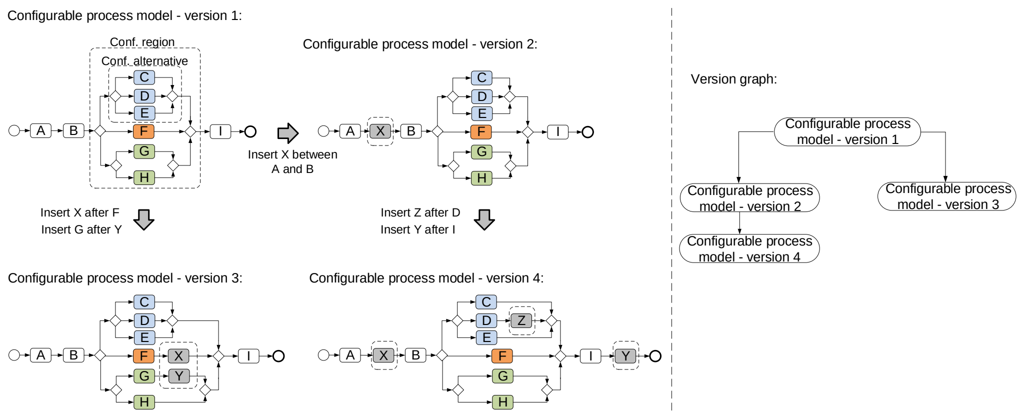

Configurable process model - version 1:
Conf. region
Conf. alternative
Configurable process model - version 2:
Insert X between A and B
Insert X after F
Insert G after Y
Insert Z after D
Insert Y after I
Configurable process model - version 3:
Configurable process model - version 4:
Version graph:
Configurable process model - version 1
Configurable process model - version 2
Configurable process model - version 3
Configurable process model - version 4

Figure 18: Versioning of a configurable process model

**Feature F5.2 (Propagating changes of a configurable process model to already configured process variants).** When changing a process fragment in a *configurable process model*, which is common to several process variants, the changes must be propagated across all (already configured) process variants in order to maintain overall consistency of the process family and to reduce maintenance efforts. Techniques for propagating changes of a *configurable process model* to already configured process variants are described in studies S3 and S37. Fig. 19 illustrates this feature.

Fig. 20 summarizes the variability support features extracted from the SLR and presents the primary studies supporting them. Note that, in addition to these variability support features, well known features for managing single (i.e., individual) process models are applicable in the context of process families as well. As example consider algorithms measuring process model similarity [29, 30] or techniques enabling process model refactorings [31]. Both might improve the management of process families as well [100]. This work excludes such standard features since it focuses on variability-specific language constructs and support features.

## *4.7. Empirical Evaluation of Process Variability Approaches*

This section refers to empirical evaluations existing in the context of process variability (i.e., RQ7: *Have existing process variability approaches been evaluated? If yes, how does this evaluation look like?*). For answering this research question, we analyze studies describing process variability approaches

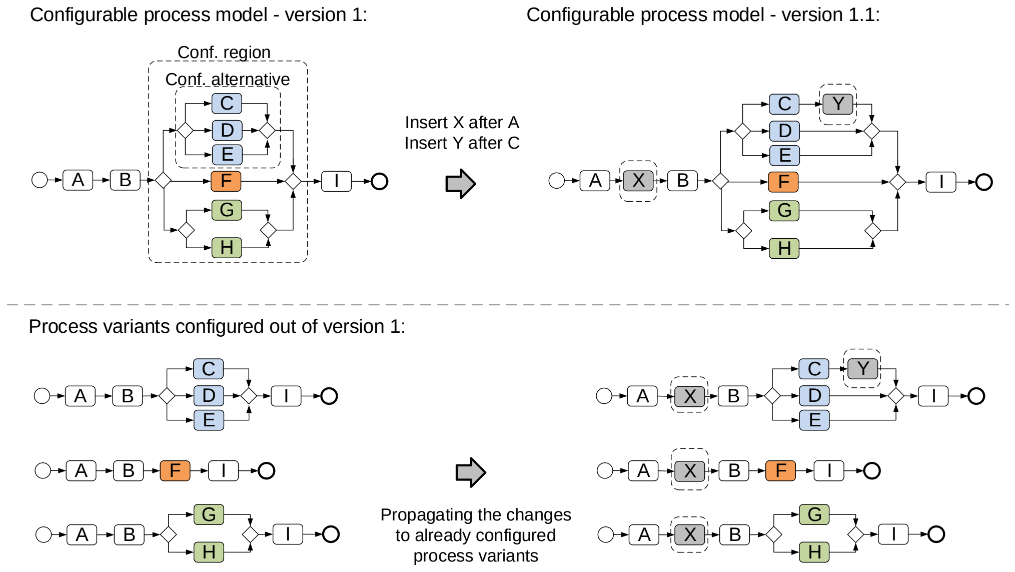
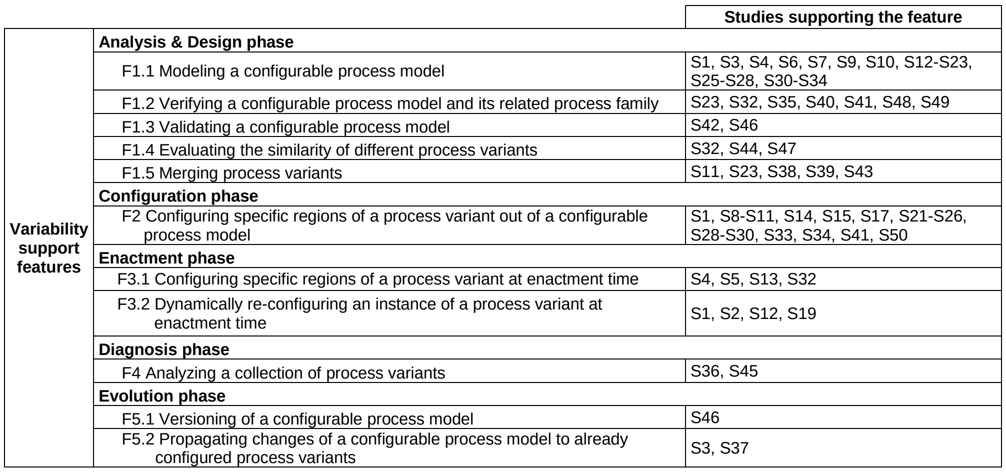

Configurable process model - version 1:
Conf. region
Conf. alternative
A
B
C
D
E
F
G
H
I
Insert X after A
Insert Y after C
Configurable process model - version 1.1:
X
Y
Process variants configured out of version 1:
Propagating the changes to already configured process variants

Figure 19: Propagating changes between configured process variants

(i.e., S1-S34) and empirical evaluations of these approaches (i.e., S51-S63).

The SLR reveals that two different methods have been applied to empirically evaluate process variability approaches: case studies and experiments (cf. Fig. 21). Case studies constitute the most popular method applied in the context of 12 studies (i.e., S28, S51-54, S56, S57, S58, and S60-S63). Furthermore, 4 studies deal with experimental validations (i.e., S1, S23, S55, and S59). Study S1 uses the *Goal-Question-Metric method* to evaluate the design of a *configurable process model*. In turn, S23 reports on interviews with practitioners after they interacted with a *configurable process model*. In turn, study S55 uses similarity metrics to cope with the complexity (e.g., size) of configurable process models. Finally, S59 provides mapping patterns to compare two process variability approaches (i.e., S23 and S28) in terms of complexity (e.g., size of the resulting models).

It is noteworthy that the SLR confirms that there only exist few concrete evaluations of process variability approaches. In turn, this indicates a lack of empirical evaluations of existing process variability approaches, which have not matured to the level of general industrial adoption yet (e.g., regarding scalability and usability).



| | | Studies supporting the feature |
|---|---|---|
| Variability support features | **Analysis & Design phase** | |
| | F1.1 Modeling a configurable process model | S1, S3, S4, S6, S7, S9, S10, S12-S23, S25-S28, S30-S34 |
| | F1.2 Verifying a configurable process model and its related process family | S23, S32, S35, S40, S41, S48, S49 |
| | F1.3 Validating a configurable process model | S42, S46 |
| | F1.4 Evaluating the similarity of different process variants | S32, S44, S47 |
| | F1.5 Merging process variants | S11, S23, S38, S39, S43 |
| | **Configuration phase** | |
| | F2 Configuring specific regions of a process variant out of a configurable process model | S1, S8-S11, S14, S15, S17, S21-S26, S28-S30, S33, S34, S41, S50 |
| | **Enactment phase** | |
| | F3.1 Configuring specific regions of a process variant at enactment time | S4, S5, S13, S32 |
| | F3.2 Dynamically re-configuring an instance of a process variant at enactment time | S1, S2, S12, S19 |
| | **Diagnosis phase** | |
| | F4 Analyzing a collection of process variants | S36, S45 |
| | **Evolution phase** | |
| | F5.1 Versioning of a configurable process model | S46 |
| | F5.2 Propagating changes of a configurable process model to already configured process variants | S3, S37 |

Figure 20: Variability support features and studies supporting them

## *4.8. Application Domains*

This section gives insights into the domains in which existing process variability approaches have been applied (i.e., RQ8: *In which domains have existing process variability approaches been applied?*). For this purpose, we analyze studies describing process variability approaches (i.e., S1-S34) and empirical evaluations of these approaches (i.e., S51-S63). We observe that process variability approaches have been applied to different domains. The latter include e-government (S12, S28, S53, S54, S55, and S60), retail (S10, S16, and S58), finance (S23), automotive industry (S2), healthcare (S17 and S61), and film production (S24 and S25). Note that this list only includes those domains for which there exists a clear evidence that the process variability approach is applied to real business processes; i.e., we do not consider domains for which fictitious processes are described.

## *4.9. Aspects Cutting Across Research Questions*

The SLR results described in the previous sections are not completely independent from each other. This section analyzes the relations among the results.

First, the analysis made in the context of research questions RQ1 and RQ2 reveals that many of the language-independent process variability approaches rely on a *multi-artifact technique* for building configurable process

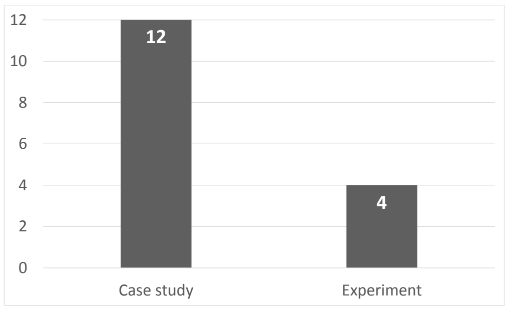


Figure 21: Methods applied to empirically evaluate process variability approaches

models. This applies to 14 out of 17 language-independent process variability approaches: S1, S3, S5, S9, S10, S15, S19, S20, S22, S24, S29, S30, S32, and S33.

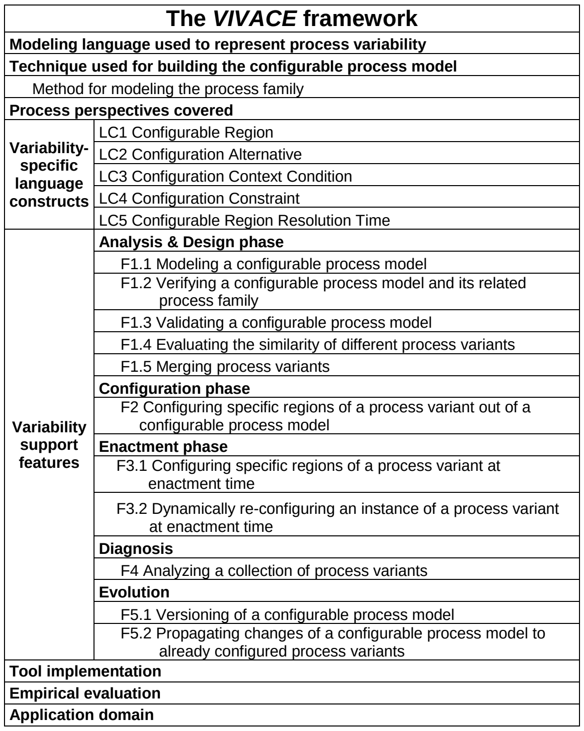
Second, the *multi-artifact technique* (RQ2) enables a broader support of the variability-specific language constructs identified (RQ3). Most process variability approaches relying on a multi-artifact technique (e.g., S1, S14, and S15) support three or more of these language constructs. This may be explained by the fact that the use of additional artifacts allows defining broader aspects of a configurable process model; e.g., configuration context conditions or configuration constraints.

Third, it is noteworthy that the identified language constructs (RQ3) are mainly supported with respect to the *functional* and *behavioral* process perspectives (i.e., control flow) (RQ4). This is plausible since the most supported variability-specific language constructs (i.e., *configurable region* and *configuration alternative*) are related to the control flow of the process.

Fourth, tool support available for a particular process variability approach (RQ5) does not entail an empirical evaluation of the respective approach (RQ7). Although tools may facilitate the evaluation of an approach, only 3 process variability approaches (i.e., S1, S23 and S28) have been both implemented and empirically evaluated.

## 5. The VIVACE Framework: Synthesizing the SLR Results

This section synthesizes the data obtained in the SLR and presents the *VIVACE* framework. The latter aggregates the results we gathered in the context of the defined research questions (cf. Sect. 3.1). Hence *VIVACE* draws a complete picture on process variability support, i.e., the framework refers to the process modeling language, the techniques provided for building a *configurable process model*, the process perspectives covered, the language constructs provided, the features provided for supporting process variability in the different phases of the lifecycle, tool implementation, and empirical evaluation (cf. Fig. 22).



| **The *VIVACE* framework** | |
|---|---|
| **Modeling language used to represent process variability** | |
| **Technique used for building the configurable process model** | |
| Method for modeling the process family | |
| **Process perspectives covered** | |
| **Variability-specific language constructs** | LC1 Configurable Region |
| | LC2 Configuration Alternative |
| | LC3 Configuration Context Condition |
| | LC4 Configuration Constraint |
| | LC5 Configurable Region Resolution Time |
| **Variability support features** | **Analysis & Design phase** |
| | F1.1 Modeling a configurable process model |
| | F1.2 Verifying a configurable process model and its related process family |
| | F1.3 Validating a configurable process model |
| | F1.4 Evaluating the similarity of different process variants |
| | F1.5 Merging process variants |
| | **Configuration phase** |
| | F2 Configuring specific regions of a process variant out of a configurable process model |
| | **Enactment phase** |
| | F3.1 Configuring specific regions of a process variant at enactment time |
| | F3.2 Dynamically re-configuring an instance of a process variant at enactment time |
| | **Diagnosis** |
| | F4 Analyzing a collection of process variants |
| | **Evolution** |
| | F5.1 Versioning of a configurable process model |
| | F5.2 Propagating changes of a configurable process model to already configured process variants |
| **Tool implementation** | |
| **Empirical evaluation** | |
| **Application domain** | |

Figure 22: The *VIVACE* framework

Like other frameworks [117, 9], *VIVACE* is intended to systematically assess and compare process variability approaches with respect to their expressiveness and the features provided for the support of process variability in the different phases of the process lifecycle. In order to illustrate the way *VIVACE* can be applied in practice, we exemplarily assess selected process variability approaches found in the context of the SLR. In detail, the latter include *C-EPC* (S28) (cf. Sect. 5.1), *Provop* (S19) (cf. Sect. 5.2), and *PE-SOA* (S30) (cf. Sect. 5.3). We select these approaches since they are (1) well established and highly cited, (2) there exists a mature tool support for them, and (3) they represent variability using different techniques (cf. Sect. 4.2). For each selected approach, based on *VIVACE* we provide a general description, discuss its expressiveness with respect to process variability modeling, and assess its lifecycle support for process variability. As evaluation criteria, we consider the results gathered in the context of the presented research questions (cf. Sect. 3.1).

For both, variability-specific language constructs (cf. Sect. 4.3) and variability support features (cf. Sect. 4.6), we differentiate between *no support* [-], *partial support* [+/-], and *full support* [+]. In addition, regarding the process perspectives supported (RQ4), we use codes to indicate the perspectives covered by the approach: behavioral (B), functional (F), organizational (O), informational (I), temporal (T), and operational (Op). In this vein, we use these codes for Language Construct LC2 (i.e., *configuration alternative*) in order to indicate the process perspectives it covers. Finally, we summarize evaluation results in Sect. 5.4.

## *5.1. Applying VIVACE to Configurable EPC*

**General description.** A possible way of realizing a *configurable process model* is to enrich a process model with *configurable nodes*. A modeling language supporting this approach is C-EPC (i.e, Configurable EPC). C-EPC extends an existing process modeling language (i.e., EPC) by introducing configurable elements. In particular, this allows merging the behavior of all valid process variants in one and the same artifact, i.e., the *configurable process model* corresponds to one artifact (*single artifact technique*). *Configurable nodes* have been introduced for other process modeling languages as well (e.g., YAWL [3]). Fig. 23 illustrates the use of C-EPC in the context of the check-in process (cf. Example 1). Configurable nodes are depicted with a thicker line. The *configurable nodes* correspond to process fragments with single entry and single exit (i.e., SESE fragment). They may have two different forms. On one hand, the SESE fragment may consist of a splitting *configurable connector*, immediately followed by a set of branches representing configuration alternatives, and a joining *configurable connector*; i.e., the *configurable connectors* delimit a configurable region (e.g., Configurable region 1 in Fig. 23). Alternatively, the SESE fragment may consist of a *configurable function* (e.g., Configurable region 2 in Fig. 23), which may be configured as ON (i.e., the function shall be kept in the configured process model), OFF (i.e., the function shall not be included in the configured process model), or OPT (i.e., the function shall be conditionally included in the configured process model deferring the decision about its execution to enactment time). In turn, SESE fragments representing the different *configuration options* are included as branches between two *configurable connectors* (e.g, *Localize assistance to accompany passenger* in Configurable Region 1 in Fig. 23). Further, note that the *application context* is represented separately from the *configurable process model* in a *questionnaire model* [61]. Note that the latter is not depicted in Fig. 23 due to lack of space. Finally, semantic constraints with respect to the configuration of *configurable functions* and *connectors* (e.g., mutual exclusion, inclusion) may be specified in terms of *configuration requirements* linked to the configurable nodes. For example, *Configuration Requirement 1* in Fig. 23 states that the *configurable function Fill in unaccompanied form* is only included if SEQ1b is selected in XOR1; i.e., activity *Assign seat for UM* is selected.

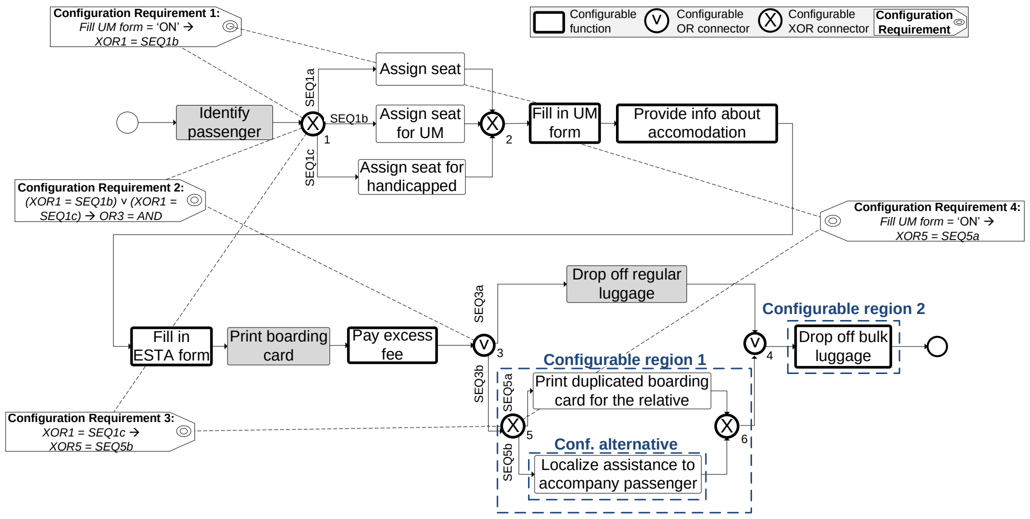

Configuration Requirement 1: Fill UM form = 'ON' → XOR1 = SEQ1b
Configurable function
Configurable OR connector
Configurable XOR connector
Configuration Requirement
Identify passenger
SEQ1a
SEQ1b
SEQ1c
Assign seat
Assign seat for UM
Assign seat for handicapped
Fill in UM form
Provide info about accomodation
Configuration Requirement 2: (XOR1 = SEQ1b) v (XOR1 = SEQ1c) → OR3 = AND
Configuration Requirement 4: Fill UM form = 'ON' → XOR5 = SEQ5a
Drop off regular luggage
SEQ3a
SEQ3b
Configurable region 2
Fill in ESTA form
Print boarding card
Pay excess fee
Drop off bulk luggage
Configurable region 1
Print duplicated boarding card for the relative
SEQ5a
SEQ5b
Conf. alternative
Localize assistance to accompany passenger
Configuration Requirement 3: XOR1 = SEQ1c → XOR5 = SEQ5b

Figure 23: Configurable process model of the check-in process (in C-EPC notation)

**Process variability expressiveness.** Regarding the identified variability-specific language constructs presented in Sect. 4.3, in C-EPC, a *configurable region* (LC1) is specified in terms of a *configurable connector* or *function* (LC1 [+]). In turn, a *configuration alternative* (LC2) corresponds to a SESE fragment that may either be included as a branch between two *configurable connectors* (e.g., *Localize assistance to accompany passenger* in Configurable Region 1 in Fig. 23) or excluded. Basically, *configuration alternatives* consider the *functional* and *behavioral* perspectives. In addition, extended support with respect to the *organizational* and *informational* perspectives is provided [61] (LC2 [F, B, O, I]). In turn, *configuration context conditions* (LC3 [+]) are represented separately in a *questionnaire model* (see [61]). A *configuration constraint* is specified in terms of a *configuration requirement.* The latter may be linked to the configurable nodes that delimit the configurable region to which the respective configuration alternatives belong (LC4 [+]); e.g., *Configuration Requirement 1* in Fig. 23. Finally, *configurable region resolution time* is supported since *configurable functions* can be configured to OPT, deferring their configuration to enactment time when the context information becomes available (LC5 [+]); e.g., *configurable function Pay excess fee* in Fig. 23.

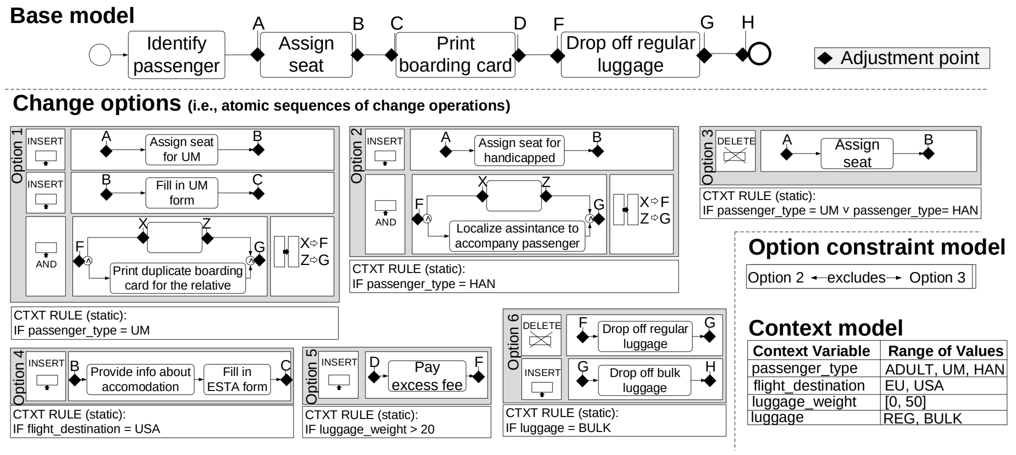
**Process variability support.** The C-EPC approach has been implemented in a toolset called *Synergia*, which provides a number of variability support features as well [75]. In particular, Synergia supports the creation of a *configurable process model* using a graphical editor. Moreover, it allows defining the context of a *configurable process model* using *configuration requirements* (F1.1 [+]). Further, the Synergia toolset provides a mapper tool that can be used to verify a *configurable process model* and its related process family [60] (F1.2 [+]). In addition, it is possible to validate configured process models using *C-EPC Validator* [76] (F1.3 [+]). No support is provided for measuring the similarity between process variants (F1.4 [-]), whereas sophisticated merging techniques are presented in [64] (F1.5 [+]). The configuration of process variants is supported by a questionnaire-based approach, which has been implemented in the *Quaestio* tool [60]. By answering a set of questions, domain experts are assisted and guided in configuring *configurable process models* and hence in deriving a specific process variant (F2.1 [+]); i.e., domain facts (answers) are mapped to configuration choices. Configuration at enactment time and dynamic re-configuration of a process variant are not considered; once the configuration of a C-EPC model is finished, the resulting process model variant is deployed and cannot be changed anymore (F3.1 [-], F3.2 [-]). In turn, support for optimizing process variant models is provided in [41] (F4.1 [+]). No explicit support exists for handling different versions of a *configurable process model* or propagating model changes to process variants (F5.1 [-], F5.2 [-]). In addition, the C-EPC approach has been empirically evaluated in a case study [77]. Finally, C-EPC has been applied to business processes from different domains (e.g., e-government and film production).

*5.2. Applying VIVACE to Provop*

**General description.** In Provop, a pre-specified *base process model* (*base process* for short) is adjusted to the given application context through a sequence of model changes; i.e., context-specific structural adaptations are applied in order to derive a particular process variant from a *configurable process model* in Provop [48, 47]. Furthermore, a base process may be specified with any process modeling language; i.e., Provop provides a language-independent approach.

Fig. 24 illustrates how the check-in process family can be represented in Provop. In particular, the *configurable process model* can be represented through several artifacts (i.e., *multi-artifact technique*). The top of Fig. 24 depicts the base process based model out of which the process variants can be configured. Figure 24 further shows the structural adaptations (i.e., *change options*) that may be applied in isolation or combination to this base process when configuring a particular process variant. In Provop, configurable regions of the base process are specified by a SESE fragment, delimited by two *adjustment points* (i.e., black diamonds). For example, in Fig. 24 a *configurable region* comprises the process fragment delimited by *adjustment points* A and B. In turn, a configuration alternative is specified in terms of a *change option*, which includes (1) a list of atomic *change operations* modifying a configurable region of the *base process* and (2) a *context rule* that defines the context conditions under which the *change operations* shall be applied (e.g., Option 1 in Fig. 24). The application context is specified in terms of *context rules*, which include a set of context variables and their values specifying the conditions under which a configuration alternative (i.e., a *change option*) shall be chosen (e.g., Option 2 shall be applied if the passenger is a handicapped person). All *context variables* and their allowed values are gathered in the *context model* (cf. Fig. 24). Finally, semantic constraints (e.g., mutual exclusion or inclusion) may be specified between two *change options* in the *option constraint model*; e.g., if Option 2 is applied, Option 3 shall be excluded (cf. Fig. 24).

**Process variability expressiveness.** In Provop, a *configurable region* is specified in terms of a SESE fragment of the *base process*, delimited by two *adjustment points* (LC1 [+]). In turn, a *configuration alternative* (LC2) is specified in terms of a *change option*. These alternatives may be defined with respect to the control flow perspective, i.e., behavioral and functional



| Context Variable | Range of Values |
|---|---|
| passenger_type | ADULT, UM, HAN |
| flight_destination | EU, USA |
| luggage_weight | [0, 50] |
| luggage | REG, BULK |

Figure 24: Provop model of the check-in process

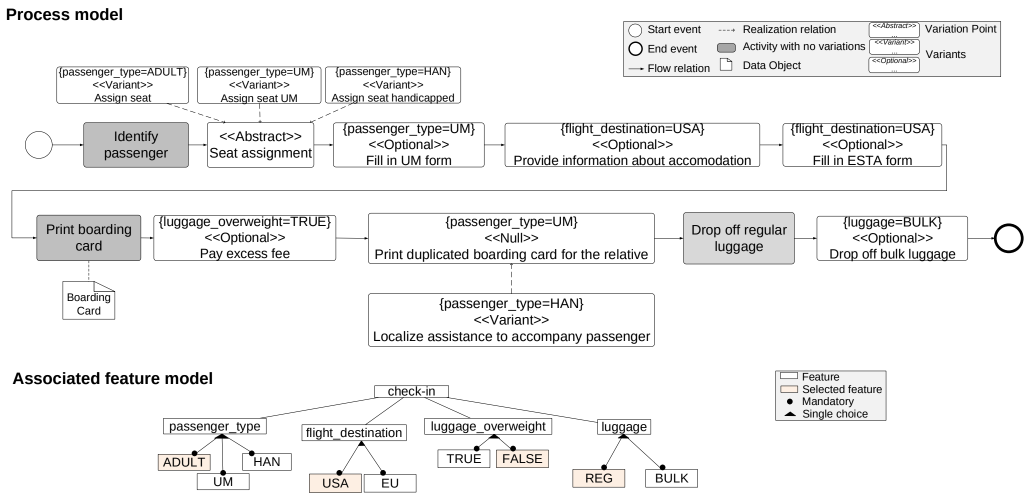
perspective (LC2 [F, B]). In turn, *configuration context conditions* can be specified in terms of *context rules* (LC3 [+]), while *configuration constraints* are specified as constraints between *change options* in the *option constraint model* (LC4 [+]). Finally, Provop does not allow for the specification of the *configurable region resolution time* (LC5 [-]).

**Process variability support.** The Provop approach has been implemented in a *proof-of-concept* prototype based on the *ARIS Business Architect tool* [99]. The creation of a *configurable process model* is supported by a graphical editor, which allows creating a base model and specifying the options that may be used to configure it (F1.1 [+]). The prototype further supports the definition of a context model through a set of context variables. Moreover, relationships between variants can be defined in the *option constraint model*. The configured process models can be verified to ensure their correctness (F1.2 [+]). However, no validation support is provided (F1.3 [-]). In addition, techniques for measuring the similarity of process variants [72] as well as for merging process variants [74] are provided (F1.4 [+], F1.5 [+]). The prototype further provides configuration support (F2.1 [+]). Depending on the actual context, a (sub-)set of the available change options is applied to the base process model resulting in a context-specific process variant. The Provop prototype checks whether the defined options violate any constraint defined on the total set of change options. If such a constraint violation is detected, the prototype notifies the process engineer accordingly. After selecting the change options relevant in the given application context, these are applied to the *base process* and the resulting process variant is displayed to the user. Dynamic configuration at enactment time is not supported by the Provop prototype (F3.1 [-]). Opposed to this, dynamic re-configuration of a process variant instance due to contextual changes at enactment time is supported by including variant branches in the process model and encapsulating the adjustments of single change options within these variant branches (F3.2 [+]). Support for analyzing a collection of process variants is provided through a separate tool [73], which can import the process variants created by the Provop prototype (F4.1 [+]). In turn, the evolution of *configurable process models* is not considered; i.e., neither support for handling different versions of a *configurable process model* nor support for propagating the changes of the *base process* to already configured process variants has not been provided yet (F5.1 [-], F5.2 [-]). Finally, Provop has been illustrated using processes from the automotive and healthcare domains.

## *5.3. Applying VIVACE to PESOA*

**General description.** The PESOA approach represents a process family through a *configurable process model* including a set of annotations. More precisely, annotations are attached to the process activities that may be subject to variation [96]. For this purpose, language-independent techniques like encapsulation, inheritance, design patterns, and extension points are used. In principle, therefore, PESOA may be applied to any process modeling language. Further, the application context of the variable activities is specified in terms of features attached to them. Accordingly, process variants are configured by selecting features in the respective feature model. However, the relationships that may exist between the alternatives of different variation points are not considered.

Fig. 25 illustrates the *configurable process model* corresponding to the check-in process as defined in PESOA. It composes the related feature model, which, in turn, contains the features associated with the configuration alternatives. For example, the *configurable process model* comprises five optional activities (e.g., *Fill in UM form*, *Provide information about accommodation*, and *Drop off bulk luggage*) specified in terms of extension points. Moreover, an inheritance technique is provided in order to model the alternatives for activity *Seat assignment*. Attached to the definition of each alternative, there are context conditions (i.e., features) defining when the alternative shall be selected (i.e., *multi-artifact technique*). For example, activity *Pay excess fee* will only become available if variable *luggage_overweight* has value TRUE.



Process model
Start event
End event
Flow relation
Realization relation
Activity with no variations
Data Object
Variation Point
Variants
{passenger_type=ADULT} <<Variant>> Assign seat
{passenger_type=UM} <<Variant>> Assign seat UM
{passenger_type=HAN} <<Variant>> Assign seat handicapped
Identify passenger
<<Abstract>> Seat assignment
{passenger_type=UM} <<Optional>> Fill in UM form
{flight_destination=USA} <<Optional>> Provide information about accomodation
{flight_destination=USA} <<Optional>> Fill in ESTA form
Print boarding card
{luggage_overweight=TRUE} <<Optional>> Pay excess fee
{passenger_type=UM} <<Null>> Print duplicated boarding card for the relative
Drop off regular luggage
{luggage=BULK} <<Optional>> Drop off bulk luggage
Boarding Card
{passenger_type=HAN} <<Variant>> Localize assistance to accompany passenger
Associated feature model
Feature
Selected feature
Mandatory
Single choice
check-in
passenger_type
flight_destination
luggage_overweight
luggage
ADULT
UM
HAN
USA
EU
TRUE
FALSE
REG
BULK

Figure 25: PESOA model of the check-in process

**Process variability expressiveness.** *Configurable regions* are defined by attaching annotations to those activities that are subject to variation (LC1 [+]). In addition, *configuration alternatives* can be modeled using techniques like encapsulation, inheritance, design patterns, and extension points; e.g., *Provide information about accommodation* in Fig. 25. However, these alternatives only refer to the behavioral and functional perspectives (LC2 [F, B]). Furthermore, the *configuration context conditions* of the alternatives are defined in terms of features attached to the activities instead of process variables (LC3 [+]). Finally, neither *configuration constraints* between *configuration alternatives* nor *configurable region resolution time* can be specified (LC4 [-], LC5 [-]).

**Process variability support.** PESOA has been realized as Eclipse plug-in that supports the creation of a *configurable process model* (F1.1 [+]). Its graphical editor allows representing *configurable regions* including *configuration alternatives.* In addition, it supports the definition of *configuration context conditions.* This is accomplished based on feature diagrams [24], 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 [-]). In addition, neither similarity nor merging techniques exist (F1.4 [-], F1.5 [-]). The configuration of process models is supported by feature diagrams, i.e., for each disabled feature, the corresponding variable parts are removed from the process variant model (F2.1 [+]). Note that the usage of feature diagrams allows configuring process variants at a high level of abstraction. In addition, the resulting process variants are displayed to users. Since PESOA focuses on the modeling as well as configuration of process variants, features related to the deployment, execution, diagnosis, and evolution of process variants are not taken into account (F3.1 - F5.2 [-]). Finally, PESOA has been illustrated using processes from the retail domain.

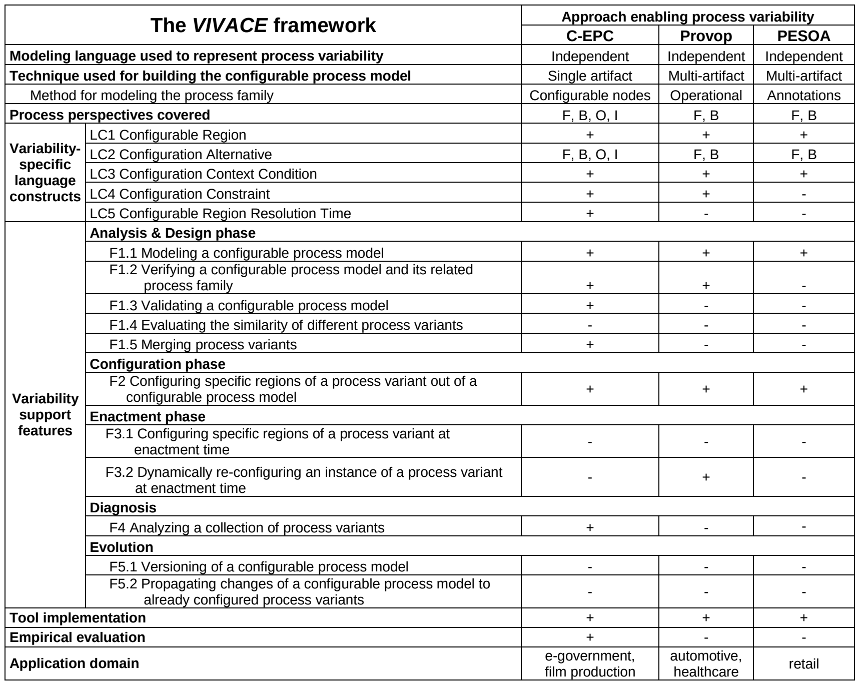
### *5.4. Summary of the Evaluation*

Fig. 26 illustrates the *VIVACE* framework as well as its use for evaluating the selected process variability approaches. As shown, none of the selected process variability approaches supports the framework completely. To be more precise, the language constructs are fully supported only by one approach (i.e., C-EPC). In addition, the features introduced in Sect. 4.6 are only partially supported. The latter can be explained by the fact that all approaches lack a support of the late phases of the process lifecycle (i.e., enactment, diagnosis, and evolution).[9]

In absence of an established reference framework for process variability, *VIVACE* covers different aspects related to process variability. In particular, it covers modeling aspects of process variability (i.e., modeling languages and techniques, variability-specific language constructs, and process perspectives covered), existing support for process variability along the lifecycle (i.e., variability support features), existing tools, empirical evaluations of process variability, and application domains in which process variability approaches have been applied. In this context, and based on the descriptions provided in [39], in the following we discuss the *completeness, expandability,* and *consistency* of *VIVACE.*

- **Completeness**: The process variability aspects covered by *VIVACE*

[9]Note that we will apply *VIVACE* to other approaches as well. Respective results will be made available on a website.



| The *VIVACE* framework | | Approach enabling process variability | | |
|---|---|---|---|---|
| | | C-EPC | Provop | PESOA |
| **Modeling language used to represent process variability** | | Independent | Independent | Independent |
| **Technique used for building the configurable process model** | | Single artifact | Multi-artifact | Multi-artifact |
| Method for modeling the process family | | Configurable nodes | Operational | Annotations |
| **Process perspectives covered** | | F, B, O, I | F, B | F, B |
| **Variability-specific language constructs** | LC1 Configurable Region | + | + | + |
| | LC2 Configuration Alternative | F, B, O, I | F, B | F, B |
| | LC3 Configuration Context Condition | + | + | + |
| | LC4 Configuration Constraint | + | + | - |
| | LC5 Configurable Region Resolution Time | + | - | - |
| **Variability support features** | **Analysis & Design phase** | | | |
| | F1.1 Modeling a configurable process model | + | + | + |
| | F1.2 Verifying a configurable process model and its related process family | + | + | - |
| | F1.3 Validating a configurable process model | + | - | - |
| | F1.4 Evaluating the similarity of different process variants | - | - | - |
| | F1.5 Merging process variants | + | - | - |
| | **Configuration phase** | | | |
| | F2 Configuring specific regions of a process variant out of a configurable process model | + | + | + |
| | **Enactment phase** | | | |
| | F3.1 Configuring specific regions of a process variant at enactment time | - | - | - |
| | F3.2 Dynamically re-configuring an instance of a process variant at enactment time | - | + | - |
| | **Diagnosis** | | | |
| | F4 Analyzing a collection of process variants | + | - | - |
| | **Evolution** | | | |
| | F5.1 Versioning of a configurable process model | - | - | - |
| | F5.2 Propagating changes of a configurable process model to already configured process variants | - | - | - |
| **Tool implementation** | | + | + | + |
| **Empirical evaluation** | | + | - | - |
| **Application domain** | | e-government, film production | automotive, healthcare | retail |

Figure 26: *VIVACE* framework applied to three selected approaches

have been identified from an SLR, which provides a fundamental and profound understanding of business process variability. As a consequence of this methodological choice, the *VIVACE* framework only describes how process variability is currently supported (i.e., description and classification of existing literature), but not how support for process variability should look like (e.g., important variability support features that have not been addressed by existing research yet). In addition, the results of the SLR, and thus the resulting *VIVACE* framework, are influenced by the defined research questions (cf. Sect. 3.1). Although we are confident that these cover the most important process variability aspects (e.g., variability-specific language constructs and variability support features) and for characterizing existing literature, *completeness* cannot be guaranteed.

- **Expandability**: If other research proposes more aspects that cannot be fully mapped to the existing framework (e.g., need for adding other variability-specific language constructs), *VIVACE* may be *expanded* with the newly identified aspects. However, with every expansion of the framework it might potentially become necessary that aspects need to be merged and subsequently renamed (e.g., a newly added features is similar, but not identical, to an existing one and hence the two features might be merged and a new label covering both features might be assigned).

- **Consistency**: The orthogonality of the process variability aspects covered in *VIVACE* minimize the occurrence of *inconsistencies* in the results obtained after its application. For example, the assessment of a specific variability-specific language construct (e.g. configurable region) does not have an impact over the evaluation of variability support features.

## 6. Discussion

The data presented in the previous sections allowed us to answer the research questions that had guided the SLR. This section interprets the results of the SLR. Further, it provides a general discussion.

*First*, we learned that we can distinguish between language-independent and language-specific process variability approaches (RQ1). Basically, language-independent approaches extend an existing process modeling language, but are intended to be applicable to other process modeling languages as well. However, we observed that the latter is far from being trivial due to existing language peculiarities. Despite the fact that language independence is useful for generalizing a process variability approach, it might be more suitable to focus on a well established process modeling language (e.g., standardized languages) as well as to develop variability-specific techniques optimized for this language. In particular, this would facilitate its industrial adoption and evaluation.

*Second*, the SLR revealed that 13 studies implement a *single artifact* technique to represent a process family. By contrast, 20 studies provide a *multi-artifact technique* making use of various modeling artifacts in order to represent the different aspects of a process family; e.g., common parts, variant-specific parts, constraints, and application context. Note that these figures do not provide evidence about which technique is more exposed. Hence, additional research is needed; e.g., experiments evaluating the techniques in different settings (e.g., varying model size or covered process perspectives).

*Third*, despite the high number of process variability approaches identified through the SLR, a common set of variability-specific language constructs is essentially supported by most approaches (RQ3); i.e., *configurable regions* and *alternatives* (supported by 32 studies), *configuration context conditions* (15 studies), *configuration constraints* (24 studies), and *configurable region resolution time* (3 studies). Although existing approaches use different terminology (e.g., *configurable region* vs. *variation point*) and realize the language constructs in different ways (cf. Sect. 4.3), the five languages constructs we identified are the most prevalent ones in existing literature on process variability. In turn, this can be considered as a valuable insight. However, only two studies report on process variability approaches supporting all five variability-specific language constructs; i.e., most existing approaches do not cover the entire set of constructs. Hence, additional efforts are required to support all variability-specific language constructs in an integrated way in order to cover process variability in a more complete sense. In addition, the language constructs we identified as well as their descriptions can be used as a standard vocabulary when dealing with process variability. Thus, they will contribute to improve the communication among PAIS engineers.

*Fourth*, regarding the process perspectives covered by existing process variability approaches (RQ4), most efforts (i.e., 33 studies) have been spent on modeling the variability of the *functional* and *behavioral* perspectives (i.e., activities and control flow). In turn, the *informational* and *organizational* perspectives are only covered by rather few studies (7 and 5 studies, respectively). On the contrary, none of the identified studies considers variability of the *temporal* or *operational* perspective even though variability support for these two perspectives is crucial in practice. As illustrated in the context of the check-in process, variability occurs with respect to all process perspectives. Accordingly, it must be properly handled for each of them in order to be able to represent processes families from the real world. Thus, an integrated approach for modeling process variability, which covers all language constructs as well as process perspectives, is needed. As a consequence, additional research is needed in order to extend existing process variability approaches such that they cover other process perspectives (i.e., the *temporal* and the *operational* ones) as well. Finally, it is noteworthy that the identified language constructs (RQ3) are mainly supported for the *functional* and *behavioral* perspectives (i.e., control flow). However, since variability is prevalent for all process perspectives, these language constructs need to be extended to cover all perspectives.

*Fifth*, process variability approaches should not only allow for the modeling of variability, but also provide tool support for managing process variants along the process lifecycle. In this line, the SLR has shown that 41 studies provide a tool implementation (RQ5). However, most of these implementations constitute proof-of-concept prototypes. In addition, only 10 of the 41 implementations are made available on respective websites. When making tools accessible to others, researchers promote a wider adoption of their solutions by practitioners. Thus, tools can be applied to real scenarios and other users might improve them.

*Sixth*, the SLR findings reveal a core set of 11 variability support features to deal with process variability along the process lifecycle (RQ6). Five of these features are related to the modeling of a *configurable process model*: *modeling, verifying, validating, evaluating the similarity of variants,* and *merging process variants.* Regarding the *configuration* phase, there exist studies dealing with the *configuration* of process variants. In tunr, for the *enactment* phase, techniques for *dynamically configuring* parts of a process variant instance as well as for *dynamically re-configuring* process variant instances exist. Besides, several studies report on techniques for *analyzing* process variants. Finally, there are techniques for *maintaining different versions* of *configurable process models* as well as for *propagating changes* across process variants; i.e., techniques supporting the evolution of process families over time. Overall, this set of features will allow process engineers to evaluate the practical applicability of existing process variability approaches.

Basically, existing work on process variability has focused on modeling issues. However, more efforts are required with respect to the implementation, enactment, diagnosis, and evolution of process families and corresponding variants. In addition, similar to single (i.e., individual) business processes [124, 74, 73], more advanced techniques for optimizing process families need to be provided. For example, this includes support for identifying process variants that may be derived from a *configurable process model*, but never have been configured or deployed. In turn, respective optimizations might trigger the evolution of the *configurable process model.* In addition, optimization techniques might discover frequently applied configuration steps, which are then lifted up to the *configurable process model* to reduce future configuration efforts. Furthermore, refactoring support should be provided in order to improve the quality of a *configurable process model*; i.e., altering its schema, but without changing its behavior. These examples suggest emerging research areas related to process families (i.e., process variability).

Finally, there is no integrated process variability approach supporting the entire set of features along the process lifecycle. Basically, existing process variability approaches support the modeling, verification and configuration of *configurable process models*. However, none of the identified approaches supports a wider range of features; i.e., only very few process variability approaches support more than one feature; e.g., study S1 supports 3 features (i.e., F1.1., F2.1, and F3.2), while study S3 refers to 2 features (i.e., F1.1 and F5.1). As a consequence, more efforts are required in order to cover the entire set of features in an integrated way.

Used in combination with the identified language constructs, variability-specific features will help process engineers to analyze, compare, and assess the support provided by existing process variability approaches. While language constructs allow assessing the expressiveness required to explicitly model process variability in process families, the variability-specific features reflect the support required to adequately cope with such expressiveness along the different phases of the process lifecycle. Thus, another purpose of our work is to use these language constructs and variability-specific features as an *evaluation framework*, which we denote as *VIVACE* (cf. Fig. 26). As opposed to other variability frameworks [117, 9], *VIVACE* has been the result of a systematic analysis of existing literature identifying which aspects are actually supported when dealing with process variability. Accordingly, we expect our framework to be applied to different process variability approaches as well as related tools in order to evaluate their suitability for process variability management. In the same vein, we expect *VIVACE* to support process engineers in implementing a PAIS supporting process variability along the process lifecycle. Finally, *VIVACE* may assist process engineers in selecting the variability approach meeting their requirements best.

*Seventh*, only few process variability approaches have been extensively evaluated in practice so far (RQ7); i.e., 15 studies. However, the absence of a broader empirical evidence limits the acceptance of respective approaches and aggravates their practical use.

*Eighth*, the SLR revealed that process variability approaches have been applied in various domains, e.g., healthcare, retail, and e-government. This emphasizes the presence of variability in business processes from different domains.

## 7. Threats of Validity

The main threats to the validity of our work are **selection bias**, **inaccuracy in data extraction & analysis**, and **reliability** [54, 94, 107, 118].

To ensure that the SLR is complete as far as possible and no important literature is missing, we used six well-known literature sources. These include the most important conference and journals on the topic (i.e., process variability). In addition, by scanning the references of the retrieved studies (i.e., backward reference searching), we ensured the completeness of the SLR. Further, we ensured that all relevant literature previously known to us was found by the SLR as well.

*First*, our systematic search was conducted in 2013. Accordingly, studies published later were not included in our work.[10] In order to minimize the selection bias, the selection process performed by the main author was continuously reviewed by her co-authors. To be more precise, the co-authors randomly reviewed selected studies to ensure the consistency of the selection process, along with the correct application of inclusion and exclusion criteria. Disagreements emerging in this context were resolved through comprehensive discussions. Paper duplication constitutes another potential threat. Hence, the selected studies were checked twice in order to detect and remove duplicate papers, including only the most recent and complete version.

*Second*, data extraction and analysis were carried out by the main author. This might comprise subjective decisions since several primary studies did not provide a clear description of objectives and results. To mitigate this risk, a rigor extraction process was applied based on the guidelines of Kitchenham [54]. In addition, the co-authors continuously checked the work of the first author resolving disagreements through discussion.

*Finally*, we ensure reliability as the search process can be replicated by other researchers. Since the data extraction process also considers subjective factors (e.g., in cases where studies did not provide clear descriptions), however, there is no guarantee that other researchers will obtain exactly the same results as presented in this work.

[10]We continuously scan newly emerging papers and approaches to evolve our framework as well as to apply it to the emerging approaches.

## 8. Related Work

The goal of this paper is to provide a fundamental and profound understanding of process variability as well as to comprehensively assess the support provided by existing process variability approaches along the entire lifecycle of process families. For this purpose, we conducted an SLR (i.e., systematic literature review) on existing process variability approaches. To the best of our knowledge, there exist three *refereed* works that have to some degree analyzed process variability in a systematic way as well.

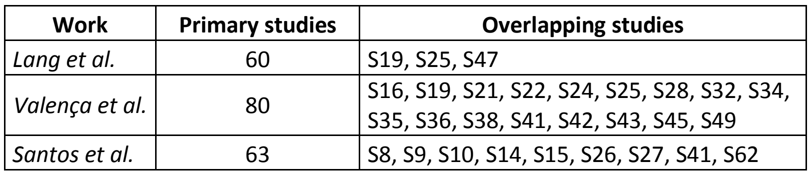
*Lang* [66] reports on the results of a systematic mapping on flexible process modeling. A set of research questions is defined for identifying the types of approaches enabling flexible processes, the research method used (e.g., analysis, implementation), the contributions of the study (i.e., tool, method), the context in which these approaches were developed (i.e., industrial vs. academic), and their quality assessment. This work is broader in scope compared to our SLR since it focuses on process flexibility in general (see [100] for a comprehensive description of process flexibility issues). On the contrary, we focus on a concrete aspect of process flexibility (i.e., the support of process variability through *configurable process models*) with the goal of providing profound insights into how existing approaches enable process variability support (i.e., modeling languages, techniques, language constructs, and features). Other aspects of process flexibility (i.e., looseness, adaptation, and evolution) were out of the scope of our work and hence were not included in the SLR. In addition, we identified a set of language constructs and variability support features tailored towards the support of process variability; i.e., we establish the *VIVACE* framework for evaluating and comparing existing process variability approaches.

In [123], *Valença et al.* present a systematic mapping on process variability, which summarizes the theoretical background of this topic. Although the goal of this paper is related to the one of our work, the authors solely deal with design time issues. On the contrary, our SLR considers all phases of the process lifecycle. This is relevant since execution and maintenance aspects of process variability will enrich its understanding. In addition, *Valença et al.* consider complementary keywords in the search string; e.g., "change", "agility", "reuse", and "similarity". As a result, they also retrieve studies related to features for managing single (i.e., individual) business processes (e.g., [29]). However, we restricted our search to variability-specific issues to set a clear focus on process variability support. Furthermore, we explore how process variability is actually supported and implemented by existing approaches, i.e., we provide a complete evaluation framework.

Finally, *Santos et al.* [112] conduct an SLR on how Software Product Lines (SPL) techniques have been applied to business process management. For this purpose, the authors analyze coarse-grained aspects of SPL techniques for business processes such as domain and application engineering, SPL architectures, variability management, and feature modeling. In our work, we are specifically investigating how variability can be managed in process families (e.g., languages, constructs, features). We attempt to provide detailed insights into the modeling of process variability as well as into the way process variants can be configured, enacted, evolved, and maintained, no matter what the used techniques are. On the contrary, [112] only considers process variability approaches related to SPL techniques (e.g., use of feature models to represent process variability), neglecting other methods for dealing with process variability (e.g., annotated *configurable process models* [115]).

Fig. 27 presents a comparative summary of these works based on the retrieved studies. While column "Primary studies" presents the identified primary studies of each work, column "Overlapping studies" shows the studies identified in our SLR as well. In terms of primary studies, there is no significant difference between the SLRs. To be more precise, *Lang et al.* [66] retrieved 60, *Santos et al.* [112] 63, and our work 63 primary studies. Concerning *Valença et al.* [123], there is a higher number of primary studies (i.e., 80 studies). This can be explained with the complementary keywords the authors use in the search string (e.g., "change", "agility"). The overlap of the studies is relatively low between our SLR and *Lang et al.* and *Santos et al.*, respectively, since the goals of these works are different. On the contrary, the overlap increases in the case of *Valença et al.* since the focus is more related to our SLR (i.e., process variability). However, note that we expand the analysis of process variability to other phases of the lifecycle and identify the set of features specifically tailored to process families as well.

In the context of software processes, there exist SLRs analyzing variability in software process tailoring. For example, [80] identifies the requirements and mechanisms that consistently support process tailoring. Finally, [93] describes the tools, techniques, approaches, and experiences of variability in software engineering processes. However, these works were not deeply analyzed since their focus is not on business process variability.



| Work | Primary studies | Overlapping studies |
|---|---|---|
| *Lang et al.* | 60 | S19, S25, S47 |
| *Valença et al.* | 80 | S16, S19, S21, S22, S24, S25, S28, S32, S34, S35, S36, S38, S41, S42, S43, S45, S49 |
| *Santos et al.* | 63 | S8, S9, S10, S14, S15, S26, S27, S41, S62 |

Figure 27: Comparison of related works

## 9. Summary and Outlook

Our work aims to provide a fundamental understanding of process variability and comprehensively assesses the support provided by existing approaches enabling process variability along the process lifecycle. For this purpose, a *systematic literature review* was conducted. In this context, we retrieved a total of 4947 studies of which 63 were identified as primary studies for providing such an understanding. After analyzing these studies, a set of variability-specific language constructs were identified. These constructs allow assessing the ability of an approach to properly model process variability. Regarding the business process modeling languages used, our work reveals that existing process variability approaches tend to be language-independent. However, focusing on a specific language might foster the evaluation of respective approaches as well as their adoption in practice. Another crucial aspect concerns the process perspectives covered. While the *behavioral, functional, organizational,* and *informational* perspectives are rather well supported, there is a lack of support regarding variability in the *temporal* and *operational* perspectives.

We further identified a set of variability support features for dealing with process families. Respective features are intended to foster the evaluation of existing process variability approaches with respect to their practical applicability. Interestingly, not many tools exist in this context. Furthermore, there is no integrated process variability approach supporting the entire set of features along the process lifecycle. Note that this would significantly facilitate the management of process families from different domains. Finally, our SLR revealed that more effort should be spent for empirically evaluating process variability approaches.

The aggregated results obtained from our systematic literature review were used to define the *VIVACE framework*. *VIVACE* supports process engineers in (1) evaluating existing process management technologies enabling process variability, (2) selecting which of them meets their requirements best, and (3) implementing a PAIS that will effectively support variability along the process lifecycle,

Similar to workflow patterns [2, 67, 68, 108, 109, 110] and process change patterns [127, 13], our framework will provide a qualitative perspective on different approaches enabling process variability. 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 [130]. By also considering these advanced requirements, we will be able to additionally assess the quality of an existing process variability approach from a quantitative perspective [122]. In the context of these experiments, we will use the *Cheetah Experimental Platform* [95]. *Cheetah* 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., Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. Theoretical Computer Science 270(1-2), pp. 125–203, (2002).

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

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

[4] 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).

[5] van der Aalst, W.M.P., Lohmann, N., La Rosa, M., Xu, J.: Correctness ensuring process configuration: An approach based on partner synthesis. In Proc. BPM'10, pp. 95–111, (2010).

[6] van der Aalst, W. M. P., Lohmann, N., La Rosa, M.: Ensuring correctness during process configuration via partner synthesis. Information Systems 37, pp. 574–592, (2012).

[7] Acher, M., Collet, P., Lahire, P., France, R. B.: Managing variability in workflow with feature model composition operators. In Proc. SC'10, pp. 17–33, (2010).

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

[9] Aiello, M., Bulanov, P., Groefsema, H.: Requirements and tools for variability management. In Proc. COMPSACW'10, pp. 245–250, (2010).

[10] Alférez, G. H., Pelechano, V., Mazo, R., Salinesi, C., Diaz, D.: Dynamic adaptation of service compositions with variability models. Journal of Systems and Software 91, pp. 24–47, (2013).

[11] Ayora, C., Torres, V., Reichert, M., Weber, B., Pelechano, V.: Towards run-time flexibility for process families: open issues and research challenges. In Proc. BPM Workshops'12, pp. 477–488, (2012).

[12] Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: Dealing with variability in process-aware information systems: language requirements, features, and existing proposals. Technical Report UIB-2012-07, Faculty of Engineering and Computer Science, University of Ulm, (2012).

[13] Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: Enhancing modeling and change support for process families through change patterns. In Proc. BPMDS'13, pp. 246-260, (2013).

[14] Baier, T., Pascalau, E., Mendling, J.: On the suitability of aggregated and configurable business process models. In Proc. BPMDS'10, pp. 108–119, (2010).

[15] Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., Kuropka, D.: Configurative process modeling - Outlining an approach to increased business process model usability. In Proc. IRMA'04, (2004).

[16] Boffoli, N., Caivano, D., Castelluccia, D., Visaggio, G.: Business process lines and decision tables driving flexibility by selection. In Proc. SC'12, pp. 178–193, (2012).

[17] OASIS Web Services Business Process Execution Language. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel Accessed: April 2014.

[18] Business Process Model and Notation, version 2.0. Object Management Group (OMG). http://www.bpmn.org/ Accessed: April 2014.

[19] Bridgeland, M., Zahavi, R.: Business modeling: a practical guide to realizing business value. Morgan Kaufmann Publishers, (2008).

[20] Bucchiarone, A., Antares Mezzina, C., Pistore, M.: CAptLang: A language for context-aware and adaptable business processes. In Proc. VaMoS'13, pp. 1–5, (2013).

[21] Bulanov, P., Groefsema, H., Aiello, M.: Business process variability: A tool for declarative template design. In Proc. ICSOC'11, pp. 241–242, (2012).

[22] Business Process Definition MetaModel Volume II: Process Definitions. http://www.omg.org/spec/BPDM/1.0/volume2/PDF Accessed: April 2014.

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

[24] Czarnecki, K., Antkiewicz, M. 2005. Mapping features to models: A template approach based on superimposed variants. In Proc. GPCE'05, pp. 422–437, (2005).

[25] de la Vara, J.L., Ali. R., Dalpiaz, F., Sánchez, J., Giorgini, P.: COMPRO; A methodological approach for business process contextualisation. In Proc. OTM'10, pp. 132–149, (2010).

[26] Derguech, W., Bhiri, S.: An indexing structure for maintaining configurable process models. In Proc. BPMDS'10, pp. 157–168, (2010).

[27] Derguech, W., Bhiri, S.: An automation support for creating configurable process models. In Proc. WISE'11, pp. 199–212, (2011).

[28] Derguech W., Gao, F., Bhiri, S.: Configurable process model for logistics case study for customs clearance processes. In Proc. BPM'12 Workshops, pp. 119–130, (2012).

[29] Dijkman, R.: Diagnosing differences between business process models. In Proc. BPM'08, pp. 261–277, (2008).

[30] 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).

[31] Dijkman, R., Gfeller, B., Küster, J., Völzer, H.: Identifying refactoring opportunities in process model repositories. Information and Software Technology 53, pp. 937–948, (2011).

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

[33] Döhring, M., Zimmermann, B.: vBPMN: Event-aware workflow variants by weaving BPMN2 and business rules. In Proc. BPMDS'11, pp. 332–341, (2011).

[34] Döhring, M., Reijers, H. A., Smirnov, S.: Configuration vs. adaptation for business process variant maintenance: an empirical study. Information and Systems (to appear), (2013).

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

[36] Frece, A., Juric, M. B.: Modeling functional requirements for configurable content- and context-aware dynamic service selection in business process models. Journal of Visual Languages & Computing 23, pp. 223–247, (2012).

[37] Galster, M., Lapre, L., Avgeriou, P.: SOA in Variability-Intensive Environments: Pitfalls and Best Practices. IEEE Software 31(1), pp. 77–84, (2014).

[38] Giese, C., Schnieders, A., Weiland, J.: A practical approach for process family engineering of embedded control software. In Proc. ECBS'07, pp. 229–240, (2007).

[39] Gómez-Perez, A.: Evaluation of ontologies. International Journal of Intelligent Systems 16(3), pp. 391–409, (2001).

[40] Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M. H.: Configurable process models - A foundational approach. Reference modeling, Physica-Verlag HD, pp. 59–77, (2007).

[41] Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M. H.: Mining reference process models and their configurations. In Proc. OTM'08 Workshops, pp. 263-272, (2008).

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

[43] Gottschalk, F., Wagemakers, T.A.C., Janse-Vullers, M.H., van der Aalst, W.M.P., La Rosa, M.: Configurable process models: Experiences from a municipality case study. In Proc. CAiSE'09, pp. 486–500, (2009).

[44] Groefsema, H., Bulanov, P., Aiello, M.: Declarative enhancement framework for business processes. In Proc. ICSOC'11, pp. 496–504, (2011).

[45] Gröner, G., Wende, C., Boskovic, M., Silva Parreiras, F., Walter, T., Heidenreich, F., Gasevic, D., Staab, S.: Validation of families of business processes. In Proc. CAiSE'11, pp. 551–565, (2011).

[46] Gröner, G., Boskovic, M., Silva Parreiras, F., Gasevic, D.: Modeling and validation of business process families. Information Systems 38(5), pp. 709–726, (2012).

[47] Hallerbach, A., Bauer, T., Reichert, M.: Guaranteeing soundness of configurable process variants in Provop. In Proc. CEC'09, pp. 98–105, (2009).

[48] 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).

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

[50] 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).

[51] Hsieh, H.F.,Shannon, S.E.: Three approaches to qualitative content analysis. Qualitative Health Research 12, pp. 1277–1288, (2005).

[52] Jablonski, S., Bussler, C.: Workflow management: concepts, architecture and implementation. International Thomson Computer Press Publisher, (1996).

[53] Jalali, S., Wohlin, C.: Systematic literature studies: database searches vs. backward snowballing. In Proc. ESEM'12, pp. 29–38, (2012).

[54] Kitchenham, B. A.: Guidelines for performing systematic literature reviews in software engineering, Version 2.3. Keele University and University of Durham, EBSE Technical Report, (2007).

[55] Koetter, F., Weidmann, M., Schleicher, D.: Guaranteeing soundness of adaptive business processes using ABIS. In Proc. BIS'11, pp. 74–85, (2011).

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

[57] Koschmider, A., Oberweis, A.: How to detect semantic business process model variants?. In Proc. SAC'07, pp. 1263–1264, (2007).

[58] 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).

[59] Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. Journal of Software Maintenance and Evolution: Research and Practice, 23(4), pp. 205–244, (2011).

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

[61] 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).

[62] 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. Information Science Reference - Imprint of: IGI Publisher, (2009).

[63] La Rosa, M., Mendling, J.: Domain-driven process adaptation in emergency scenarios. In Proc. BPM'08 Workshops, pp. 290–297, (2009).

[64] La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Merging business process models. In Proc. OTM'10, pp. 96-113, (2010).

[65] 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).

[66] Lang, A.: Flexible business process modeling – A Systematic mapping study. Master Thesis. Athabasca University. April 2012.

[67] Lanz, A., Weber, B. Reichert, M.: Workflow time patterns for process-aware information systems. In Proc. BPMDS'10, pp. 94–107, (2010).

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

[69] Lapouchnian, A, Yu, Y., Mylopoulos, J.: Requirements-driven design and configuration management of business processes. In Proc. BPM'07, pp. 246–261, (2007).

[70] Lazovik, A., Ludwig, H.: Managing process customizability and customization: model, language and process. In Proc. WISE'07, pp. 373–384, (2007).

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

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

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

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

[75] `http://www.processconfiguration.com/download.html` Accessed: April 2014.

[76] `http://www.mendling.com/EPML/C-EPC-Validator.xsl` Accessed: April 2014.

[77] Lönn, C.M., Uppström, E., Wohed, P., Juell-Skielse, G.: Configurable process models for the Swedish public sector. In Proc. CAiSE'12, pp. 190–205, (2012).

[78] Lu, R., Sadiq, S., Governatori, G.: On managing business process variants. Data & Knowledge Engineering 68(7), pp. 642–664, (2009).

[79] Mahmod, N. M., Chiew, W. Y.: Structural similarity of business process variants. In Proc. ICOS'10, pp. 17–22, (2010).

[80] Martínez-Ruiz, T., Münch, J., García, F., Piattini, M.: Requirements and constructors for tailoring software processes: a systematic literature review. Software Quality Journal 20, pp. 229–260, (2011).

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

[82] Mendling, J., Recker, J., Rosemann, M., van der Aalst, W. M. P.: Generating correct EPCs from configured C-EPCs. In Proc. SAC'06, pp. 1505–1510, (2006).

[83] 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).

[84] Meerkamm, S., Jablonski, S.: Configurable process models: experiences from a medical and an administrative case study. In Proc. ECIS'11, (2011).

[85] Montero, I., Sea, J., Ruiz-Cortés, A.: From feature models to business processes. In Proc. IEEE SCC'08, pp. 605–608, (2008).

[86] Moon, M., Hong, M., Yeom, K.: Two-level variability analysis for business process with reusability and extensibility. In Proc. COMPSAC'08, pp. 263–270, (2008).

[87] Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT support for release management processes in the automotive industry. In Proc. BPM'06, pp. 368–377, (2006).

[88] Nguyen, T., Colman, A. W., Han, J.: Modeling and managing variability in process-based service compositions. In Proc. ICSOC'11, pp., 404–420, (2011).

[89] Ognajanovic, I., Mohabbati, B., Gasevic, D., Bagheri, E., Boskovic, M.: A metaheuristic approach for the configuration of business process families. In Proc. SCC'12, pp. 25–32, (2012).

[90] Park, J., Yeom, K.: A modeling approach for business processes based on variability. In Proc. SERA'11, pp. 211–218, (2011).

[91] Pascalau, E., Rath, C.: Managing business process variants at eBay. In Proc. BPM'10, pp. 91–105, (2010).

[92] Pascalau, E., Awad, A., Sakr, S., Weske, M.: Partial process models to manage business process variants. Business Process Integration and Management 5(3), pp. 240–256, (2011).

[93] Pedreira, Ó., Piattini, M., Luaces, M. R., Brisaboa, N.: A systematic literature review of software process tailoring. ACM SIGSOFT Software Engineering Notes 32(3), pp. 1–6, (2007).

[94] Perry, D. E., Porter, A. A., Votta, L. G.: Empirical studies of software engineering: a roadmap. In Proc. ICSE'2000, pp. 345–355, (2000).

[95] 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).

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

[97] Recker, J., Rosemann, M., van der Aalst, W.M.P., Mendling, J.: On the syntax of reference model configuration - Transforming the C-EPC into Lawful EPC models. In Proc. BPM'05 Workshops, pp. 497–511, (2005).

[98] Reichert M., Rinderle S., Kreher U., Dadam P.: Adaptive process management with ADEPT2. In Proc. ICDE'05, pp. 1113–1114, (2005).

[99] 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).

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

[101] Reijers, H.A., Mans, R.S., van der Toorn, R.A.: Improved model management with aggregated business process models. Data & Knowledge Engineering 68, pp. 221–243, (2009).

[102] Reinhartz-Berger, I., Soffer, P., Sturm, A.: A domain engineering approach to specifying and applying reference models. In Proc. Workshop Enterprise Modelling and Information Systems Architectures 75, pp. 50–63, (2005).

[103] 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).

[104] Reinhartz-Berger, I., Sturm, A.: Comprehensibility of UML-based software product line specifications: A controlled experiment. Journal Empirical Software Engineering, pp. 1–36, (2012).

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

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

[107] Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14(2), pp. 131–164, (2009).

[108] 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).

[109] 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).

[110] Russell, N., van der Aalst, W.M.P., ter Hofstede, A.: Exception handling patterns in process-aware information systems. In Proc. CAiSE'06, pp. 288–302, (2006).

[111] Santos, E., Pimentel, J., Castro, J., Finkelstein, A.: On the dynamic configuration of business process models. In Proc. BMMDS'12, pp. 331–346, (2012).

[112] Santos Rocha, R., Fantinato, M.: The use of software product lines for business process management: A systematic literature review. Information and Software Technology 55(8), pp. 1355–1373, (2013).

[113] SAP Business Suite `http://www.sap.com/index.html` Accessed: April 2014.

[114] Scherer, R., Sharmak, W.: Process risk management using configurable process models. In Proc. IFIP AICT'11, pp. 341–348, (2011).

[115] Schnieders, A., Puhlmann F.: Variability modeling and product derivation in e-business process families. Technologies for Business Information Systems, pp. 63–74, (2007).

[116] Schnieders, A., Weske, M.: Activity diagram based process family architectures for enterprise application families. Journal Enterprise Interoperability, pp. 67–76, (2007).

[117] Sinnema, M., Deelstra, S., Hoekstra, P.: The COVAMOF derivation process. In Proc. ICSR'06, pp. 101–114, (2006).

[118] Sjoeberg, D. I. K., Hannay, J. E., Hanse, O., Kampenes, V. B., Karahasanovic, A., et al.: A survey of controlled experiments in software engineering. IEEE Transactions on Software Engineering 31(9), pp. 733–753, (2005).

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

[120] Schunselaar, D. M. M., Verbeek, E., van der Aalst, W. M. P., Reijers, H. A.: Creating sound and reversible configurable process models using CoSeNets. In Proc. BIS'12, pp. 24–35, (2012).

[121] Thomas, O.: Design and implementation of a version management system for reference modeling. Journal of Software 3(1), pp. 49–62, (2008).

[122] Torres, V., Zugal, S., Weber, B., Reichert, M., Ayora, C., Pelechano, V.: A qualitative comparison of approaches supporting business process variability. In Proc. BPM Workshops'12, pp. 560–572, (2012).

[123] Valena, G., Alves, C., Niu, N.: A systematic mapping study on business process variability. International Journal of Computer & Science Information Technology 5(1), (2013).

[124] 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).

[125] 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.

[126] Vogelaar, J.J.C.L., Verbeek, H.M.W., Luka, B., Aalst, W.M.P.: Comparing business processes to determine the feasibility of configurable models: A case study. In Proc. BPM'12 Workshops, pp. 50–61, (2012).

[127] 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).

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

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

[130] Weber, B., Pinggera, J., Torres, V., Reichert, M.: Change Patterns in Use: A Critical Evaluation. In Proc. BPMDS'13, pp. 261–276, (2013).

[131] Weske, M.: Business process management: concepts, languages, architectures. Springer-Verlag Berlin Heidelberg Publisher, (2007).

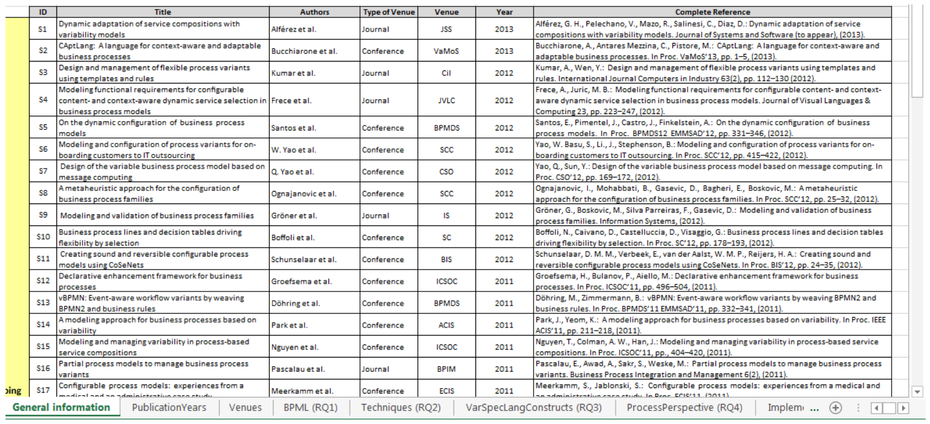
[132] Yahya, B. N., Bae, H.: Generating reference business process model using heuristic approach based on activity proximity. In Proc. IDT'11, pp. 469–478, (2011).

[133] Yao, Q., Sun, Y.: Design of the variable business process model based on message computing. In Proc. CSO'12, pp. 169–172, (2012).

[134] Yao, W. Basu, S., Li., J., Stephenson, B.: Modeling and configuration of process variants for on-boarding customers to IT outsourcing. In Proc. SCC'12, pp. 415–422, (2012).

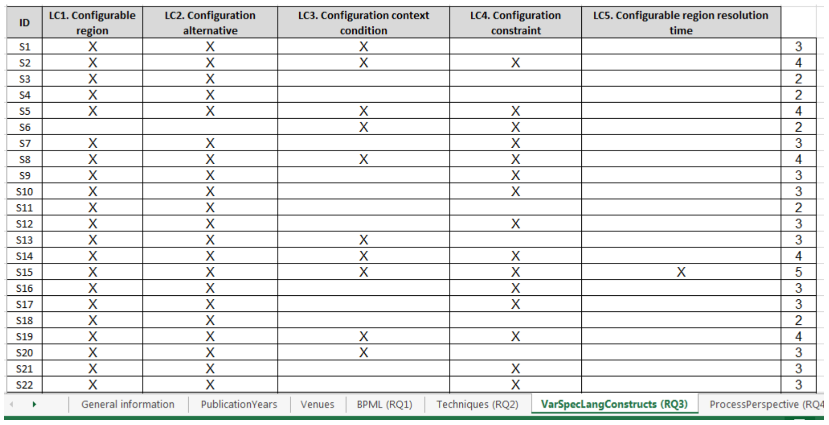
[135] Zhang, H., Babar, M. A., Tell, P.: Identifying relevant studies in software engineering. Information & Software Technology 53(6), pp.625–637, (2011).

# Appendix A: Excerpts of Excel Sheets for Data Extraction



| ID | Title | Authors | Type of Venue | Venue | Year | Complete Reference |
|---|---|---|---|---|---|---|
| S1 | Dynamic adaptation of service compositions with variability models | Alférez et al. | Journal | JSS | 2013 | Alférez, G. H., Pelechano, V., Mazo, R., Salinesi, C., Diaz, D.: Dynamic adaptation of service compositions with variability models. Journal of Systems and Software (to appear), (2013). |
| S2 | CAptLang: A language for context-aware and adaptable business processes | Bucchiarone et al. | Conference | VaMoS | 2013 | Bucchiarone, A., Antares Mezzina, C., Pistore, M.: CAptLang: A language for context-aware and adaptable business processes. In Proc. VaMoS'13, pp. 1–5, (2013). |
| S3 | Design and management of flexible process variants using templates and rules | Kumar et al. | Journal | CiI | 2012 | 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). |
| S4 | Modeling functional requirements for configurable content- and context-aware dynamic service selection in business process models | Frece et al. | Journal | JVLC | 2012 | Frece, A., Juric, M. B.: Modeling functional requirements for configurable content- and context-aware dynamic service selection in business process models. Journal of Visual Languages & Computing 23, pp. 223–247, (2012). |
| S5 | On the dynamic configuration of business process models | Santos et al. | Conference | BPMDS | 2012 | Santos, E., Pimentel, J., Castro, J., Finkelstein, A.: On the dynamic configuration of business process models. In Proc. BPMDS12 EMMSAD'12, pp. 331–346, (2012). |
| S6 | Modeling and configuration of process variants for on-boarding customers to IT outsourcing | W. Yao et al. | Conference | SCC | 2012 | Yao, W. Basu, S., Li., J., Stephenson, B.: Modeling and configuration of process variants for on-boarding customers to IT outsourcing. In Proc. SCC'12, pp. 415–422, (2012). |
| S7 | Design of the variable business process model based on message computing | Q. Yao et al. | Conference | CSO | 2012 | Yao, Q., Sun, Y.: Design of the variable business process model based on message computing. In Proc. CSO'12, pp. 169–172, (2012). |
| S8 | A metaheuristic approach for the configuration of business process families | Ognjanovic et al. | Conference | SCC | 2012 | Ognjanovic, I., Mohabbati, B., Gasevic, D., Bagheri, E., Boskovic, M.: A metaheuristic approach for the configuration of business process families. In Proc. SCC'12, pp. 25–32, (2012). |
| S9 | Modeling and validation of business process families | Gröner et al. | Journal | IS | 2012 | Gröner, G., Boskovic, M., Silva Parreiras, F., Gasevic, D.: Modeling and validation of business process families. Information Systems, (2012). |
| S10 | Business process lines and decision tables driving flexibility by selection | Boffoli et al. | Conference | SC | 2012 | Boffoli, N., Caivano, D., Castelluccia, D., Visaggio, G.: Business process lines and decision tables driving flexibility by selection. In Proc. SC'12, pp. 178–193, (2012). |
| S11 | Creating sound and reversible configurable process models using CoSeNets | Schunselaar et al. | Conference | BIS | 2012 | Schunselaar, D. M. M., Verbeek, E., van der Aalst, W. M. P., Reijers, H. A.: Creating sound and reversible configurable process models using CoSeNets. In Proc. BIS'12, pp. 24–35, (2012). |
| S12 | Declarative enhancement framework for business processes | Groefsema et al. | Conference | ICSOC | 2011 | Groefsema, H., Bulanov, P., Aiello, M.: Declarative enhancement framework for business processes. In Proc. ICSOC'11, pp. 496–504, (2011). |
| S13 | vBPMN: Event-aware workflow variants by weaving BPMN2 and business rules | Döhring et al. | Conference | BPMDS | 2011 | Döhring, M., Zimmermann, B.: vBPMN: Event-aware workflow variants by weaving BPMN2 and business rules. In Proc. BPMDS'11 EMMSAD'11, pp. 332–341, (2011). |
| S14 | A modeling approach for business processes based on variability | Park et al. | Conference | ACIS | 2011 | Park, J., Yeom, K.: A modeling approach for business processes based on variability. In Proc. IEEE ACIS'11, pp. 211–218, (2011). |
| S15 | Modeling and managing variability in process-based service compositions | Nguyen et al. | Conference | ICSOC | 2011 | Nguyen, T., Colman, A. W., Han, J.: Modeling and managing variability in process-based service compositions. In Proc. ICSOC'11, pp., 404–420, (2011). |
| S16 | Partial process models to manage business process variants | Pascalau et al. | Journal | BPIM | 2011 | Pascalau, E., Awad, A., Sakr, S., Weske, M.: Partial process models to manage business process variants. Business Process Integration and Management 6(2), (2011). |
| S17 | Configurable process models: experiences from a | Meerkamm et al. | Conference | ECIS | 2011 | Meerkamm, S., Jablonski, S.: Configurable process models: experiences from a medical and |

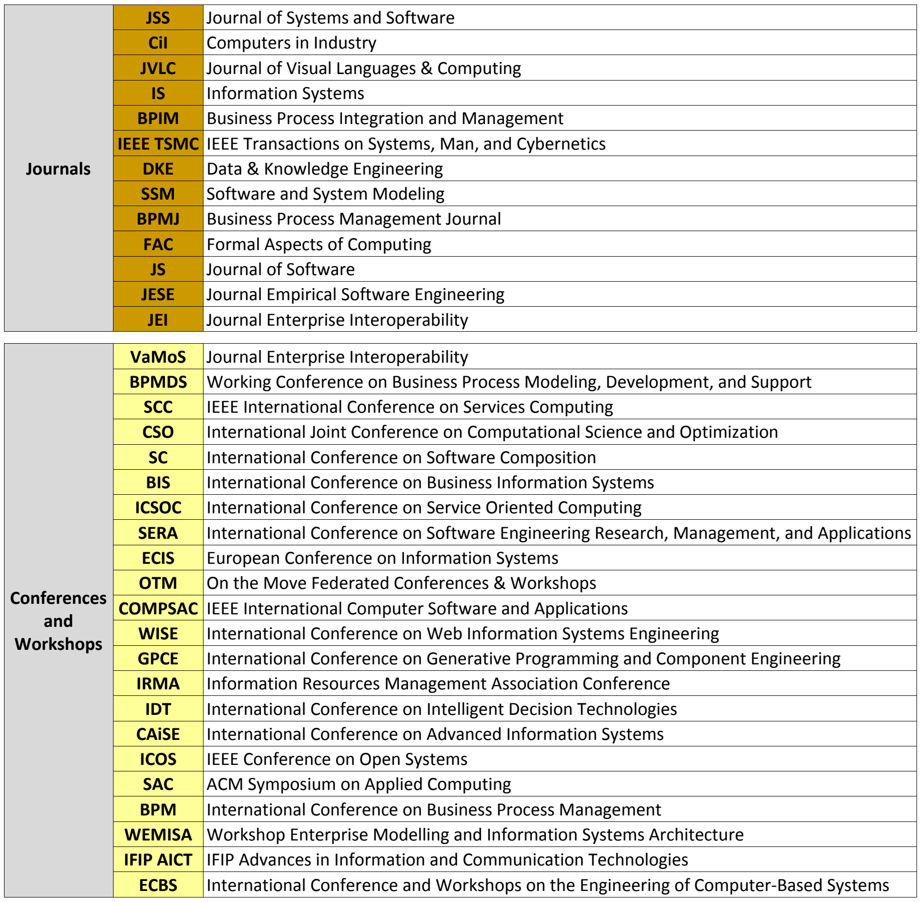
Figure 28: Overview of the Excel sheet for the general information table



| ID | LC1. Configurable region | LC2. Configuration alternative | LC3. Configuration context condition | LC4. Configuration constraint | LC5. Configurable region resolution time | |
|---|---|---|---|---|---|---|
| S1 | X | X | X | | | 3 |
| S2 | X | X | X | X | | 4 |
| S3 | X | X | | | | 2 |
| S4 | X | X | | | | 2 |
| S5 | X | X | X | X | | 4 |
| S6 | | | X | X | | 2 |
| S7 | X | X | | X | | 3 |
| S8 | X | X | X | X | | 4 |
| S9 | X | X | | X | | 3 |
| S10 | X | X | | X | | 3 |
| S11 | X | X | | | | 2 |
| S12 | X | X | | X | | 3 |
| S13 | X | X | X | | | 3 |
| S14 | X | X | X | X | | 4 |
| S15 | X | X | X | X | X | 5 |
| S16 | X | X | | X | | 3 |
| S17 | X | X | | X | | 3 |
| S18 | X | X | | | | 2 |
| S19 | X | X | X | X | | 4 |
| S20 | X | X | X | | | 3 |
| S21 | X | X | | X | | 3 |
| S22 | X | X | | X | | 3 |

Figure 29: Overview of the Excel sheet for RQ3

# Appendix B: List of Publication Venues



| | | |
|---|---|---|
| Journals | **JSS** | Journal of Systems and Software |
| | **CiI** | Computers in Industry |
| | **JVLC** | Journal of Visual Languages & Computing |
| | **IS** | Information Systems |
| | **BPIM** | Business Process Integration and Management |
| | **IEEE TSMC** | IEEE Transactions on Systems, Man, and Cybernetics |
| | **DKE** | Data & Knowledge Engineering |
| | **SSM** | Software and System Modeling |
| | **BPMJ** | Business Process Management Journal |
| | **FAC** | Formal Aspects of Computing |
| | **JS** | Journal of Software |
| | **JESE** | Journal Empirical Software Engineering |
| | **JEI** | Journal Enterprise Interoperability |
| Conferences and Workshops | **VaMoS** | Journal Enterprise Interoperability |
| | **BPMDS** | Working Conference on Business Process Modeling, Development, and Support |
| | **SCC** | IEEE International Conference on Services Computing |
| | **CSO** | International Joint Conference on Computational Science and Optimization |
| | **SC** | International Conference on Software Composition |
| | **BIS** | International Conference on Business Information Systems |
| | **ICSOC** | International Conference on Service Oriented Computing |
| | **SERA** | International Conference on Software Engineering Research, Management, and Applications |
| | **ECIS** | European Conference on Information Systems |
| | **OTM** | On the Move Federated Conferences & Workshops |
| | **COMPSAC** | IEEE International Computer Software and Applications |
| | **WISE** | International Conference on Web Information Systems Engineering |
| | **GPCE** | International Conference on Generative Programming and Component Engineering |
| | **IRMA** | Information Resources Management Association Conference |
| | **IDT** | International Conference on Intelligent Decision Technologies |
| | **CAiSE** | International Conference on Advanced Information Systems |
| | **ICOS** | IEEE Conference on Open Systems |
| | **SAC** | ACM Symposium on Applied Computing |
| | **BPM** | International Conference on Business Process Management |
| | **WEMISA** | Workshop Enterprise Modelling and Information Systems Architecture |
| | **IFIP AICT** | IFIP Advances in Information and Communication Technologies |
| | **ECBS** | International Conference and Workshops on the Engineering of Computer-Based Systems |

Figure 30: List of publication venues