Configurable Reference Modeling Languages

Jan Recker[1)], Michael Rosemann[1)], Wil van der Aalst[1,2)], Monique Jansen-Vullers[2)], Alexander Dreiling[3)]

[1)] Faculty of Information Technology
Queensland University of Technology
126 Margaret Street, Brisbane QLD 4000, Australia
Phone: +61 7 3864 9473, Fax: +61 7 3864 9390
{j.recker, m.rosemann, w.vanderaalst}@qut.edu.au

[2)] Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
Phone: +31 40 247 4295, Fax: +31 40 243 2612
{w.m.p.v.d.aalst, m.h.jansen-vullers}@tm.tue.nl

[3)] SAP Research CEC Brisbane, SAP Australia Pty Ltd
133 Mary Street, Brisbane QLD 4000, Australia
Phone: +61 7 3259 9512, Fax: +61 7 3259 9599
alexander.dreiling@sap.com

Configurable Reference Modeling Languages

ABSTRACT

This chapter discusses reference modeling languages for business systems analysis and design. In particular, it reports on reference models in the context of the design-for/by-reuse paradigm, explains how traditional modeling techniques fail to provide adequate conceptual expressiveness to allow for easy model reuse by configuration or adaptation, and elaborates on the need for reference modeling languages to be configurable. We discuss requirements for and the development of reference modeling languages that reflect the need for configurability. Exemplarily we report on the development, definition and configuration of Configurable Event-driven Process Chains. We further outline how configurable reference modeling languages and the corresponding design principles can be used in future scenarios such as process mining and data modeling.

Keywords

IS Models, Systems Development Techniques, Conceptual Design, Process Design

INTRODUCTION

Business systems have evolved as computer-based information systems that present themselves as comprehensive commercial packages for the support of business requirements. Being IT-supported software solutions, they presumptively support and enhance organizations in all their business operations. First attempts towards such corporate-wide integrated information systems were developed in the 1960s (Beer, 1966). The huge success of this idea has led to the proliferation of comprehensive business information systems such as Enterprise Resource Planning (ERP) systems or Enterprise Systems (ES), the current generation of which is known under the label of *process-aware information systems* (Dumas, van der Aalst, & ter Hofstede, 2005). This label has emerged from an act of "silent revolution" that has embraced the IS discipline over the last decades and which has started to shift the focus of attention from a data perspective towards a process perspective. As a result, an increasing number of business processes are now conducted under the governance of process-aware information systems, with the intention of bridging not only business and IT but also people and software through process-based technology.

The successful implementation of process-aware business systems is, however, dependent on a seamless alignment between the system capabilities and the organizational requirements of the enterprise. The process of aligning organizational requirements and system functionality (Rosemann, Vessey, & Weber, 2004) is known as *configuration* and rests on the assumption of similarity between enterprises, in the sense that generic business system functionality, with some customization, is assumed to be applicable to all enterprises in a given industry sector. Following the idea of process-orientation, business system vendors often offer their solutions in the form of pre-defined generic business processes for a set of industry sectors. Oracle, for example, offers system-supported business process solutions that cover 19 industrial sectors (Oracle, 2006) while

SAP offers business process solutions for 24 industrial sectors (SAP, 2006). These industry-specific process “templates” are introduced to organizations to offer a final implementation of the business system in the form of a configured, enterprise-specific set of business processes that are enabled, enacted and supported by the system.

Yet, the act of aligning generic industry-specific to enterprise-specific business processes that reflect organizational requirements has been shown to imply extensive configuration efforts and may lead to significant implementation costs that exceed the price of software licenses by factor five to ten (Davenport, 2000). Some instances even indicate that a misalignment may result in severe business failure if conducted badly. Consider the example of FoxMeyer, once a $5 billion wholesale drug distributor, which filed for bankruptcy in 1996 after Andersen Consulting concluded that the insufficiently aligned SAP installation crippled the firm's distribution (Stein, 1998). Other examples include Mobil Europe and Dow Chemical (Davenport, 1998).

Business systems vendors are aware of these problems and try to increase the manageability of the configuration process of their software solutions. One respective measure is to deliver the products along with extensive documentation and specific implementation and configuration support tools. Conceptual models play a central role within such documentation. They describe functionality and structure of the business systems on a semi-formal level and have become popular under the notion of *reference models*. Though such reference models for business systems exist in the form of function, data, system organization, object and process models, the latter is by far the most popular model type (Rosemann, 2000) and often forms a constituent part of the documentation of software packages.

While the existence of such reference models as part of the system documentation in general is valuable in software implementation projects (Kesari, Chang, & Seddon, 2003), traditional reference models offer little or no support for configuration (Daneva, 2000) This is mainly due to a lack of conceptual support in the form of a configurable modeling language underlying the reference models (Rosemann & van der Aalst, In Press).

Nevertheless, the business system configuration process can significantly benefit from the usage of reference models, for instance in terms of consistency, completeness, adaptability and communicability. Since most business information systems are quite extensively depicted in their reference models, it motivates the idea of utilizing these reference models for the configuration task. However, the language that is used to formulate reference models for the task of system configuration needs to be configurable to support this delicate task. A configurable reference process model should, for instance, provide rules defining how a generic reference process model can be adapted to suit a specific organizational context.

This chapter provides an introduction to configurable reference modeling languages and their role in the configuration process of business information systems. It covers discussions of current shortcomings of reference modeling languages, the need for configurable reference models, and the different stages towards the development and application of configurable reference modeling languages, particularly in the context of business information systems. While we will, during the course of this chapter, address multiple perspectives using the examples of process and data models, our foremost focus lies on the process perspective. We will explicate our argumentations using the example of a configurable reference process modeling language called *Configurable EPCs* (Rosemann & van der Aalst, In Press).

Forthcoming from this introduction we will first discuss traditional reference modeling languages. Then, we will present and discuss design principles for the design of configurable reference modeling languages and then apply the principles in the development of EPCs. Next, we will briefly outline future scenarios for configurable reference modeling languages and their design principles. We close this chapter by discussing some conclusions from our work.

REFERENCE MODELING LANGUAGES

Reference models are generic conceptual models that formalize recommended practices for a certain domain (Fettke & Loos, 2003; Misic & Zhao, 2000). Often labeled with the term 'best practice' reference models claim to capture reusable state-of-the-art practices (Silverston, 2001a, 2001b). The depicted domains can be very different and range from selected functional areas such as financial accounting or Customer Relationship Management to the scope of an entire industry sector, e.g., higher education.

The main objective of reference models is to streamline the design of enterprise-individual (particular) models by providing a generic solution (Rosemann, 2000). The application of reference models is motivated by the ‘design-for/by-reuse’ paradigm, postulating that they should accelerate the modeling process by providing a repository of potentially relevant business processes and structures, ideally in an easy ‘plug & play’ modus. Thus, reference modeling is closely related to the *reuse of information models* (Wisse, 2000) by providing a generic model solution that can be adapted to a specific model reflecting individual requirements.

Reference models are often used for describing the structure and functionality of business systems. In these cases, a reference model can be interpreted as a structured semi-formal description of a particular application. Such *application reference models* correspond to an existing off-the-shelf-solution that supports the functionality and structure described in the model (Rosemann, 2002). They can for example be used for a better understanding and evaluation of the appropriateness of the software.

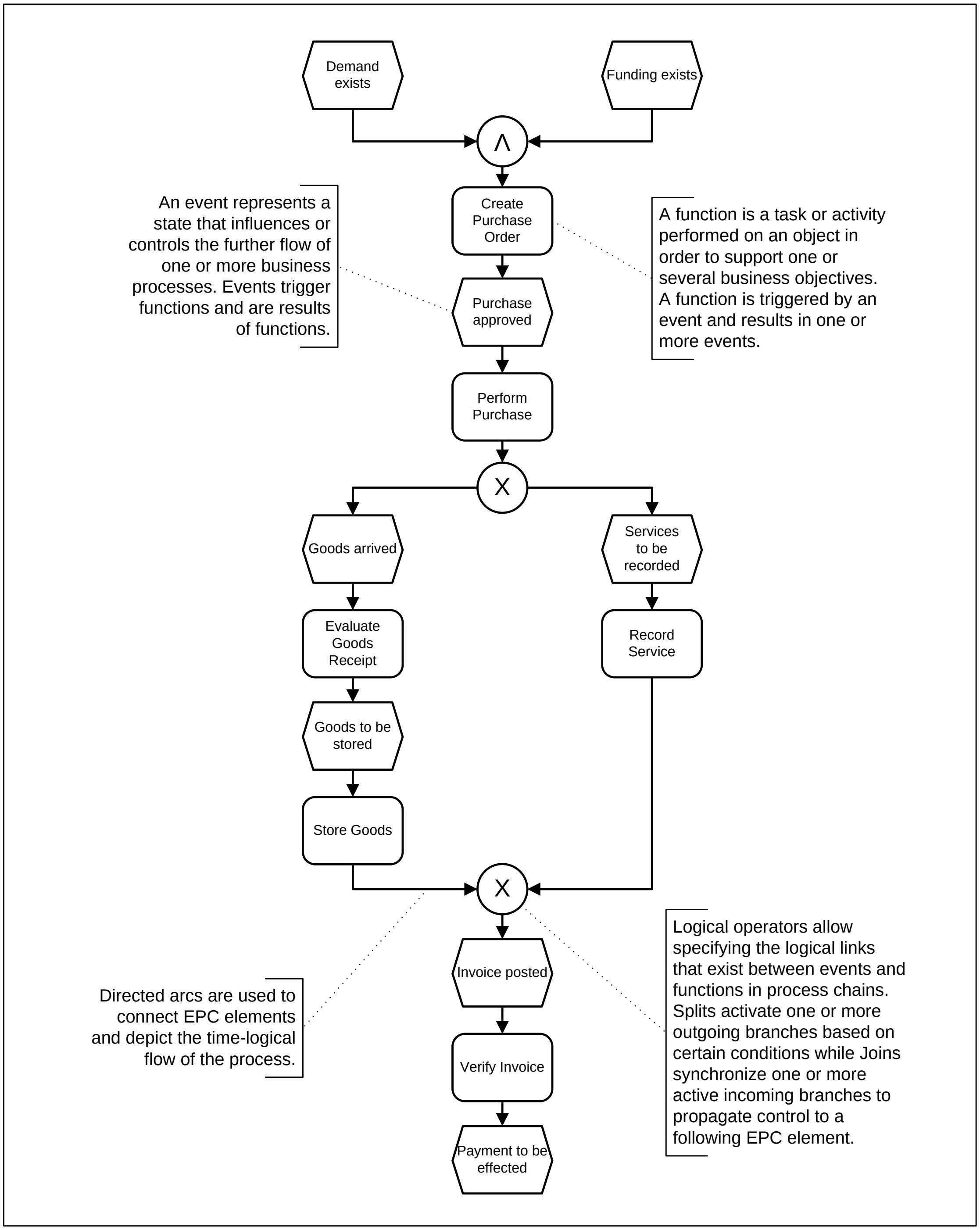
One of the most comprehensive models is the SAP reference model (Curran, Keller, & Ladd, 1997). In its version 4.6 its data model includes more than 4,000 entity types and the reference process models cover more than 1,000 system processes and inter-organizational business scenarios. Most of the other market leading business systems vendors have alternative or similar approaches towards such reference models.

Foundational conceptual work for the SAP reference model had been conducted by SAP AG and the IDS Scheer AG in a collaborative research project in the years 1990-1992 (Keller, Nüttgens, & Scheer, 1992). The outcome of this project was the process modeling language *Event-Driven Process Chains* (EPCs) (Keller et al., 1992; Scheer, 2000), which has been used for the design of the reference process models in SAP. EPCs have become one of the most popular reference modeling languages overall and has for instance been used for the design of many SAP-independent reference models (e.g., Siebel CRM, ITIL, eTOM, or PMBOK).

EPCs basically denote directed graphs, which visualize the control flow and consist of events, functions and connectors. Each EPC starts with at least one event and ends with at least one event. An event triggers a function, which leads to a new event. Three types of connectors (logical AND $\wedge$, logical exclusive OR *XOR*, logical OR $\vee$) can be used to specify the logical links that exist between sequences of events and functions in process chains. They model control flow splits and joins. An AND-split activates all outgoing branches in concurrency while an AND-join waits for all incoming branches to synchronize before propagating control to the following EPC element. An OR-split activates one, two or up to all outgoing branches based on certain conditions while an OR-join synchronizes all incoming branches that are active and then propagates control to the following EPC element. An XOR-split activates one of multiple outgoing branches based on certain conditions while an OR-join propagates control to the following EPC element when the first active incoming branch arrives.

Figure 1 gives an example for an EPC as it potentially can be found as part of a reference model. This model shows an extract of a procurement process. The EPC contains 8 events, 6 functions, and 3 connectors. The events can be seen as pre- and/or post-conditions of functions. For example, the function *Verify Invoice* can be executed if event *Invoice posted* is received and the completion of this function will trigger the event *Payment to be effected*. There are two functions triggering event *Invoice arrived*. The XOR-connector in the lower half of the diagram shows that there is no need to synchronize these two functions, e.g., the completion of *Store Goods* directly triggers event *Invoice posted*. The XOR-connector in the upper half of the diagram splits the control flow in accordance to the condition whether the purchase performed relates to goods (left branch) or services (right branch). The remaining connector denotes an AND-join, meaning that both input events need to be triggered in order to enable function *Create Purchase Order*.

Figure 1. An example for a potential reference model in EPC notation



An event represents a state that influences or controls the further flow of one or more business processes. Events trigger functions and are results of functions.

A function is a task or activity performed on an object in order to support one or several business objectives. A function is triggered by an event and results in one or more events.

Directed arcs are used to connect EPC elements and depict the time-logical flow of the process.

Logical operators allow specifying the logical links that exist between events and functions in process chains. Splits activate one or more outgoing branches based on certain conditions while Joins synchronize one or more active incoming branches to propagate control to a following EPC element.

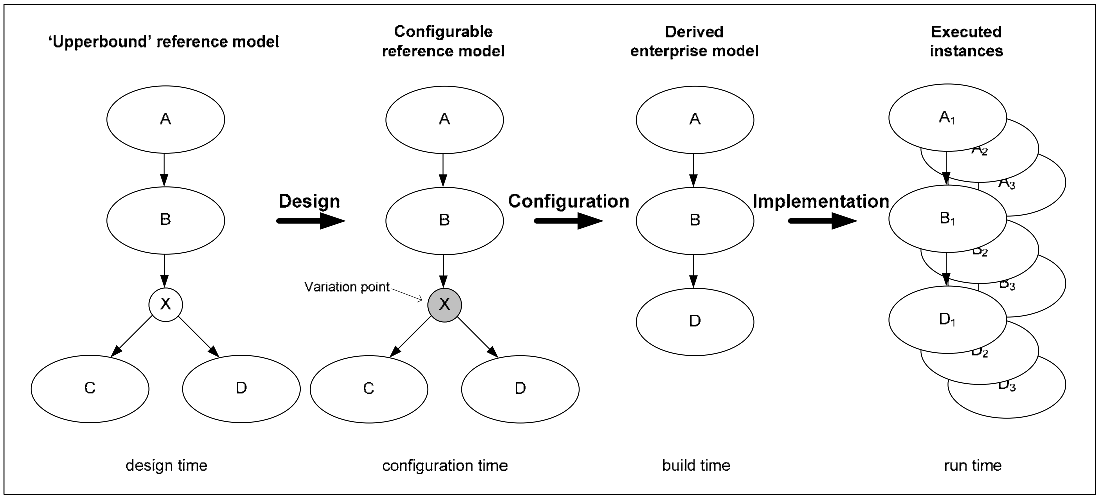
As can be observed from Figure 1, regular EPCs do not contain any configuration information. Therefore, valuable information is lacking. For example, it is not shown that *Record Service,* i.e., the scenario in which procured services need to be audited during execution, is only of interest for a subset of all procurement scenarios, namely those where services are being procured instead of goods. There are cases imaginable where enterprises only enact a procurement process for goods but not services. In these cases the accordant part of the reference model is not applicable to the organization and should be eliminated from the enterprise-specific process model. This implies that the XOR connector may be a choice made for the whole process rather than for an individual process instance. Consider a second example. The EPC shown in Figure 1 neither shows that *Store Goods* is only relevant if *Evaluate Goods Receipt* is conducted. If organizations opt never to procure goods but only services there is no need to implement functionality for goods storage. Also, the model neither gives any insights into the necessity or criticality of potential configurations nor into possible inter-dependencies between configuration decisions. Thus, the model expressive power is limited and cannot guide the configuration of a corresponding business system. Hence, a reference model designed using a traditional reference modeling language is only of limited use for the configuration process due to lacking support on a conceptual level.

DESIGN OF CONFIGURABLE REFERENCE MODELING LANGUAGES

Design principles for a configurable reference modeling language

Following the elaborations in the preceding section and following the idea of reference modeling, i.e., the streamlined development of individual models through "design-for/by-reuse", we postulate that reference modeling languages ought to be configurable. We can reason our argumentation by introducing a simple reference model lifecycle that depicts the different stages of a reference model, ranging from model design to execution (see Figure 2).

Figure 2. Reference Model Lifecycle



The lifecycle is initiated by ES vendors who depict the functionality of their software packages in reference models (*design time*). Such a reference model typically not includes merely one proposed alternative for conducting business in a certain domain but a range of often mutually exclusive alternatives It denotes an 'upper-bound' of business system models that may possibly be implemented in a particular enterprise. An organization might merely favor one of the depicted alternatives and thus only to a subset of system functionality to be implemented. Accordingly they only refer to a subset of the reference model. Figure 2 demonstrates this problem in a simple example. The upper-bound reference model depicts two mutually exclusive alternatives of conducting business, either the sequence A-B-C or A-B-D. A particular enterprise has to select one of these two substitutive alternatives of conducting business under the governance of the respective business system. The XOR split in this case represents a decision point that is of relevance during *configuration time*. Note that a model in this phase cannot necessarily be executed. It rather captures different alternatives for a domain and thus needs to be configured before it can serve as the actual *build time* model, a template for implementing and executing process instances at *run time*.

These types of decision cannot be reflected in traditional reference models due to lacking conceptual support of the underlying reference modeling language. Existing reference modeling techniques do not support the highlighting and selection of different alternatives. The resulting lack of expressiveness denotes a major issue for model users, as (a) it does not become obvious what configuration alternatives exist during system implementation, and (b) the models do not provide any decision support towards the selection of different alternatives.

Contemplating the reference model lifecycle and the shortcomings of traditional reference modeling languages, we have identified the following *design principles* for a configurable reference modeling language:

(a) A configurable modeling language is characterized by its capability to support decisions for the transformation of reference models from configuration time to build time, i.e., the model user can individualize the model by selecting from alternative options before instances will be derived from it. Such configuration decisions on a type level have to be clearly differentiated from decisions on an instance level and can be highlighted as *variation points* in a model (Halmans & Pohl, 2003) that should capture a decision point together with the related possible choices.

(b) A configurable modeling language has to support configurations of business systems regarding processes, functions, control flow and data. In terms of processes, configuration should *address the active parts* of process models, i.e., functionality (functions, tasks, transitions, and the like) and control flow. As events (or states) as more passive parts of processes cannot actively be influenced by an organization, these should not be covered by a configurable reference process modeling language.

(c) It should be possible to differentiate configuration decisions into *mandatory* and *optional decisions*. Mandatory decisions have to be made before the very first instance can be derived from this model. Optional decisions can initially be neglected. It should be possible to maintain defaults for optional configuration decisions. This allows the instantiation of the model even without explicitly making all possible decisions.

(d) Configuration should be differentiated into *global* and *local decisions*. Global decisions are based on the general context, including factors such as industry, country, size etc. The relevant context factors have to be maintained for every variation point. As soon as information regarding the relevant context has been provided, a first (hidden or background) configuration of the reference model can take place, which would lead to “context-aware models”. Local configurations require an explicit study of the relevant reference model as the related decisions may be based on local or individual factors such as available budget, risk profile, time etc.

(e) Configuration decisions should be differentiated into *critical* and *non-critical decisions*. Critical decisions have significant impact on the use of the system and other business processes, can often not be re-done and should be made by the project team. Non-critical decisions are of minor importance, can be made by individual team members and change over time.

(f) Configuration decisions can have *interrelationships*. Such pre-requisites for a configuration decision should be clearly highlighted. This can include other decisions that have to be made before. Moreover, any impact of one decision on other decisions has to be depicted. This means, a logical order between configuration decisions has to be considered. This includes interrelationships within one model, between two process models, or even interrelationships between reference process and related data models (Rosemann & Shanks, 2001).

(g) Variation points should *refer to* further *related information* within the part of the business system it depicts. This may include the system online help and the system configuration module, such as the SAP Implementation Guide (IMG) (Bancroft, Seip, & Sprengel, 1997). Such information can provide valuable support for the decision maker.

(h) The entire configuration process should be guided by recommendations in the form of *guidelines*. Such information could come as benchmarking data from the outside of the system if a critical mass of system users is willing to provide such data. It may include information such as the processing time of a given process path, the number of times a decision has been made in the same industry or the required investments and implementation time for a certain configuration. Such recommendations may as well assist reference model users in assessing the compliance of their configuration to industry best practices.

(i) Reference models can be very comprehensive. Any extension of the underlying modeling languages has to carefully consider the impact on the perceived model complexity. It is advisable to *extend existing reference modeling languages* rather than developing new ones.

In the following we will apply these design principles in the development of a configurable reference modeling language. As process modeling is key to acquiring, communicating and validating business requirements (Daneva, 2004; Welti, 1999) we will focus the process perspective, i.e., the alignment of IT functionality to the actual business processes of an organization. The following section introduces *Configurable EPCs* as the representation language of a reference process modeling approach that considers the configurable nature of a business system and reflects the design principles for configurable modeling techniques.

Configurable Event-Driven Process Chains

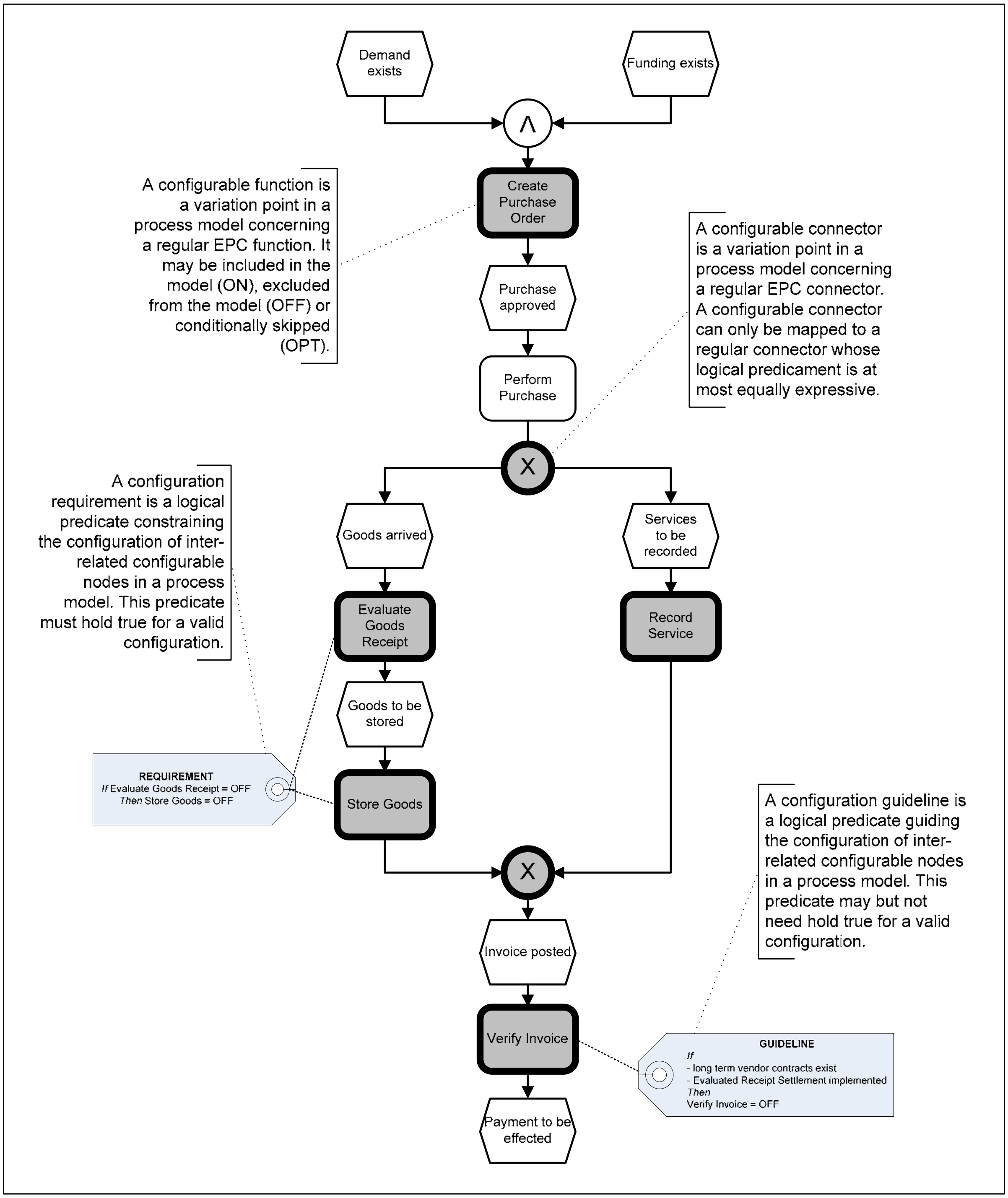
This section introduces the notion of a Configurable EPC (C-EPC).We start our elaborations by referring back to the procurement example given before. Figure 1 shows a potential reference model for the process of procurement in form of a classical EPC. Following this diagram, procurement starts with the creation of a purchase order (function *Create Purchase Order*) when a demand for services or goods exists (event *Demand exists*) and (logical AND-connector $\wedge$) when sufficient funding for the procurement exists (event *Funding exists*). Once the created purchase order has been approved, the procurement can be conducted. The process succeeds with either reception and storage of the arrived goods, or recording of the enactment of the requested service. In either case, an invoice will arrive at some point in time demanding payment for the delivery of goods or services. Then, the invoice needs to be verified, which in turn triggers the effectuation of payment, which ends the process.

However, not all organizations implement procurement the same way. For example, not only goods may be purchased but also services, with the former being in a need for appropriate storage while the latter need to be audited during enactment. A particular organization may only want to implement procurement functionality of a business system for either services or goods. Furthermore, for illustration purposes let us assume that a purchase may or may not be related to a purchase order. Similarly, the verification of invoices may or may not be essential for the effectuation of payment, for example in cases where long term contracts to trusted vendors or sophisticated support exists, e.g., in form of Evaluated Receipt Settlement functionality. None of these potential configuration decisions can be visualized using the traditional EPC reference modeling language. In particular, the model does not express possible configuration alternatives and scenarios with respect to the process it represents.

This section introduces *Configurable EPCs* as an approach to depict variation points in a reference process model as well as further configuration information (Rosemann & van der Aalst, In Press).

Adhering to design principle (b) we seek to make the active parts of processes configurable, i.e., functionality and control flow. Accordingly, in a C-EPC, functions and connectors can be configured. As an example, Figure 3 shows the procurement reference process model introduced in the preceding section depicted in C-EPC notation. We will use this example model throughout the remainder of this section to introduce the notion of C-EPCs.

Figure 3. Potential Configurable Reference Model for the Procurement Process, depicted in C-EPC notation



A configurable function is a variation point in a process model concerning a regular EPC function. It may be included in the model (ON), excluded from the model (OFF) or conditionally skipped (OPT).

A configurable connector is a variation point in a process model concerning a regular EPC connector. A configurable connector can only be mapped to a regular connector whose logical predicament is at most equally expressive.

A configuration requirement is a logical predicate constraining the configuration of inter-related configurable nodes in a process model. This predicate must hold true for a valid configuration.

A configuration guideline is a logical predicate guiding the configuration of inter-related configurable nodes in a process model. This predicate may but not need hold true for a valid configuration.

Adhering to design principle (i) C-EPCs extend regular EPCs with the specification of variation points (configurable functions and connectors), configuration requirements and configuration guidelines.

Configurable functions may be included (ON), excluded (OFF), or conditionally skipped (OPT). To be more specific, a decision has to be made whether to perform such a function in every process instance during run time (ON), whether to exclude this function permanently, i.e., it will not be executed in any process instance (OFF), or whether to defer this decision to run time, i.e., for each process instance it has to be decided whether or not to execute the function (OPT).

Referring to the example given in Figure 3, it is for instance possible to configure the procurement process in a way that *Create Purchase Order* and *Verify Invoice* are not to be implemented; therefore they are to be excluded from the enterprise-individual process model. Reflecting this decision in the configurable reference process model, the accordant configurable functions can be switched OFF.

Configurable connectors subsume possible build time connectors that are less or equally expressive. Hence, a configurable connector can only be mapped to a connector type that restricts its behavior. A configurable OR-connector may be mapped to a regular OR-, XOR-, or AND-connector. Or, the OR-connector may be mapped to a single sequence of events and functions (indicated by SEQ_n for some process path starting with node *n*). That is, out of the incoming/outgoing branches of a configurable OR-connector, a single branch is chosen that is to be included in the individual model while the remaining branches are to be excluded from the model. A configurable AND-connector may only be mapped to a regular AND-connector with a decision being made as to how many of *n* available process paths are to be executed in synchronization. A configurable XOR-connector may be mapped to a regular XOR-connector, or the XOR-connector may be mapped to a single process sequence SEQ_n. Table 1 summarizes these mapping constraints.

Table 1. Constraints for the configuration of connectors

Configurable connector	Mapping to OR	Mapping to XOR	Mapping to AND	Mapping to SEQ_n
OR	✓	✓	✓	✓
XOR		✓		✓
AND			✓	

Referring back to the example given in Figure 3, consider the decision that a particular enterprise does not want to implement procurement for both goods and services but instead only for goods. The assessment and recording of services would then be deemed unnecessary. In the reference process model, such a decision can be reflected by mapping the configurable XOR-connector to a single sequence $\text{SEQ}_{\textit{Goods arrived}}$ specifying the process branch containing the handling of received goods.

In order to depict inter-dependencies between configurable EPC nodes, *configuration requirements* can be introduced to limit the configuration possibilities between inter-related configurable nodes. These constraints are best defined via logical expressions in the form of *If-Then*-statements and denote predicates for a set of configurable nodes that *must* hold true for a valid configuration. Consider again the example given in Figure 3. If the goods receipt sub-process is deemed unnecessary, there is no need for the storage of goods, as services cannot be physically stored. A configuration constraint could be that if *Evaluate Goods Receipt* is switched OFF, so must be function *Store Goods*.

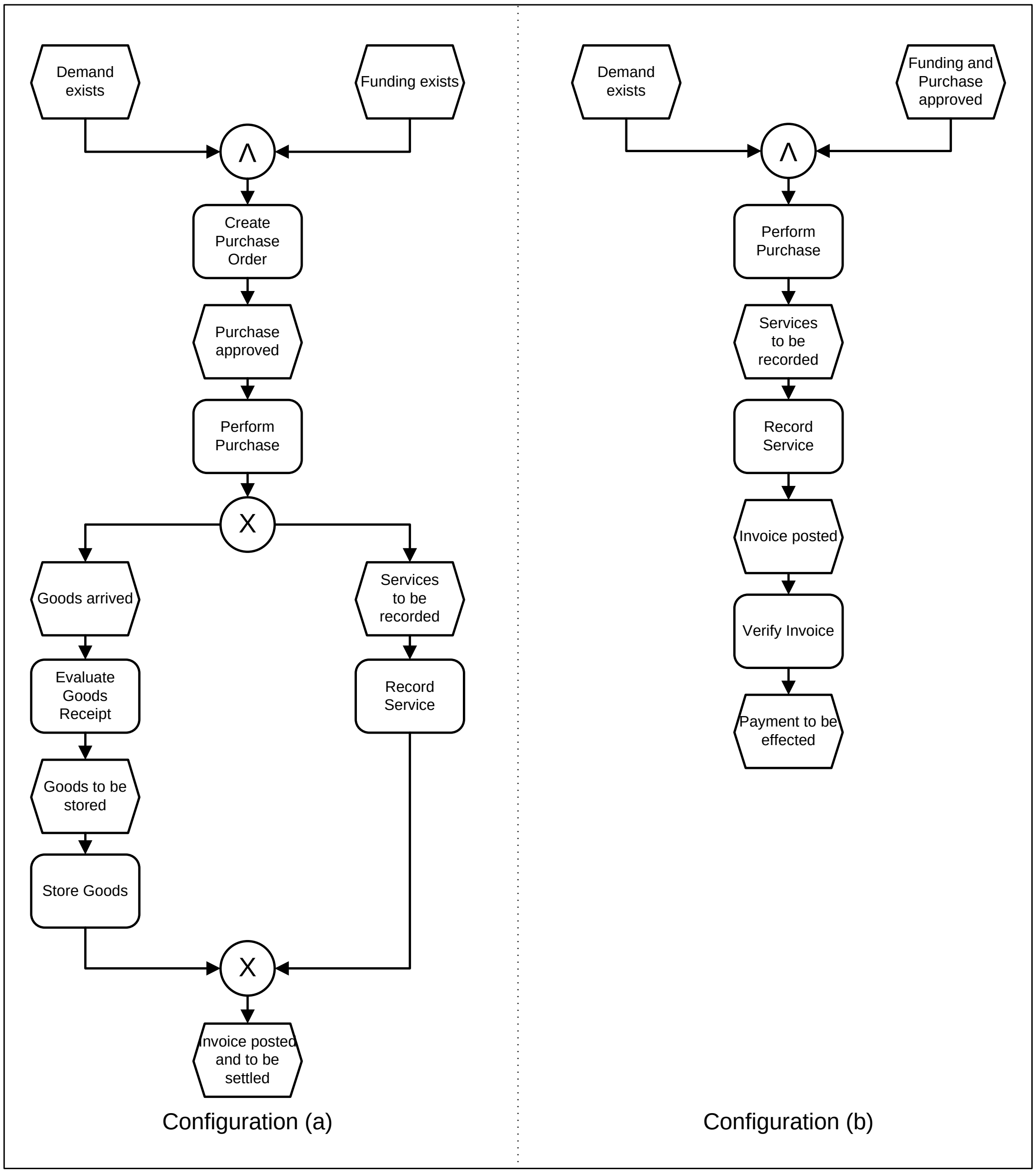
In order to provide input in terms of recommendations and proposed best practices, *configuration guidelines* may be depicted (also in the form of logical expressions) to guide the configuration process semantically. They, too, may be expressed in the form of *If-Then*-statements. They denote logical predicates for a set of configurable nodes that *may but not need* hold true for a given configuration. Again, consider Figure 3. *Verify invoice* may be an unnecessary task if long-term procurement contracts with trusted vendors or advanced Evaluated Receipt Settlement functionality exists that automatically settles invoices based on goods issues. For these scenarios a configuration guideline suggests switching *Verify Invoice* OFF.

In summation, the notion of a C-EPC potentially facilitates a selection and modification of process flows and process activities within a reference process model. As can be seen from Figure 3, configurable nodes are denoted as usual EPC nodes shaped by thick circles, while both configuration requirements and guidelines are depicted as notes-like boxes attached to a number of configurable nodes.

Configuration using Configurable EPCs

According to the reference model lifecycle (see Figure 2), at configuration time a configurable reference process model can be configured in the sense that configuration alternatives within the model are selected in a way that a configuration scenario is created which is deemed desirable for the particular organization. Such a configuration maps all configurable nodes to concrete values, i.e., regular EPC nodes, while adhering to configuration requirements (and possibly also configuration guidelines). Figure 4 shows two possible regular EPCs resulting from a configuration of the C-EPC shown in Figure 3. Consider the EPC depicted in the left part of Figure 4: In this case, the particular enterprise decided to relate purchase requests to purchase orders, hence, the function *Create Purchase Order* is included. Similarly, as the organization only purchases from long-known, trusted vendors, an extra invoice verification activity was deemed unnecessary. Hence, the accordant function *Verify Invoice* was excluded from the model. Furthermore, procurement in this case has to cater for either physical goods or services. Hence, the configurable XOR-connector has been mapped to a regular XOR-connector, allowing at run time for the procurement of either services or goods, for both of which accordant activities have been included as well. In the left part of Figure 4, Configuration (a) shows the process model resulting from the configuration *{(Create Purchase Order,ON),(XOR,XOR),(Evaluate Goods Receipt,ON),(Store Goods,ON),(Record Service,ON),(XOR,XOR),(Verify Invoice,OFF)}*.

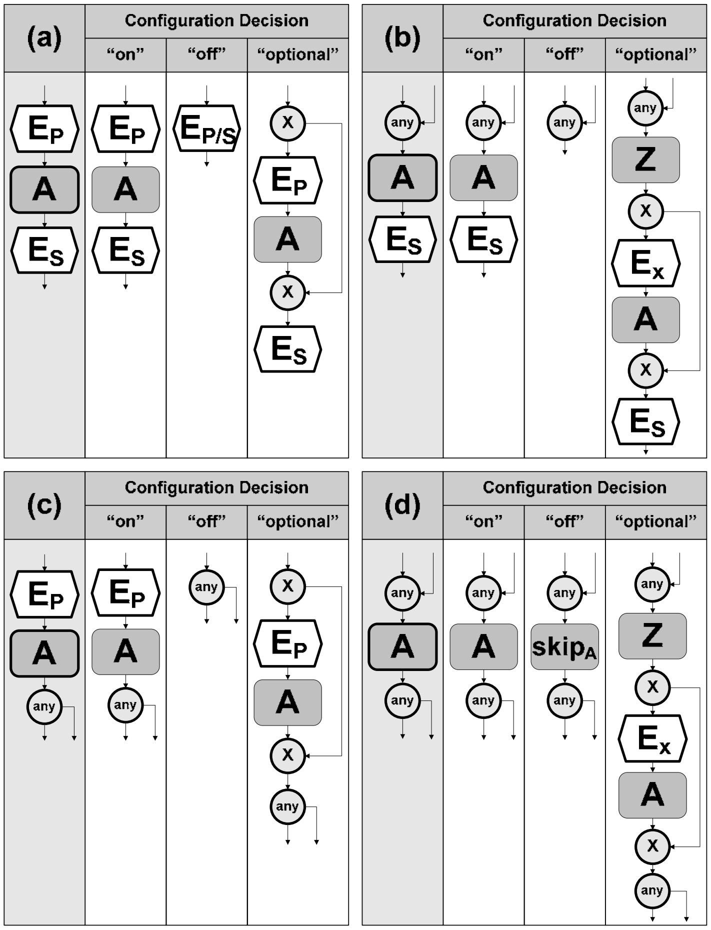
Figure 4. Two possible configurations of the C-EPC shown in Figure 3



Configuration (b) shows an EPC resulting from the configuration *{(Create Purchase Order,OFF),(XOR,SEQ*$_{\textit{Services to be recorded}}$*),(Goods Receipt,OFF),(Storage,OFF),(Service recording,ON),(XOR,SEQ*$_{\textit{Services to be recorded}}$*),(Verify Invoice,ON)}*. As both EPC models do not conflict against the configuration requirements depicted in Figure 3, both configurations are *valid*. Note here that a valid configuration is also *suitable* if it further satisfies all configuration guidelines.

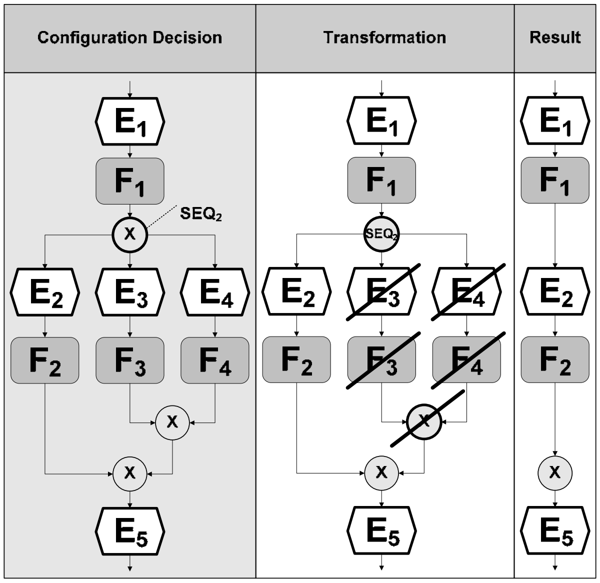
Strictly speaking, deriving a correct build time EPC from a configured C-EPC involves three kinds of tasks: (a) derivation of a partial EPC model for each configured function, (b) derivation of a partial EPC model for each configured connector, and (c) recalculation of the complete EPC process graph by excluding unnecessary paths. The calculation of the build time EPC should be governed by the minimality criterion: if elements have to be added by configuration, add as few elements as possible; if elements have to be removed by configuration, remove as many as possible, and optimize the graph so as to include no unnecessary paths (Mendling, Recker, Rosemann, & van der Aalst, in press; Recker, Rosemann, van der Aalst, & Mendling, 2006). Theoretically, there are four constellations in which a configured function may appear in a C-EPC (Dreiling, Chiang, Rosemann, & van der Aalst, 2005; Recker, Rosemann et al., 2006): (a) between two events, (b) between a connector and an event, (c) between an event and a connector, and (d) between two connectors. Figure 5 illustrates the derivation rules for these four cases (connectors labeled with *any* indicate that any connector type is allowed to make the rule applicable). In case (a) a configurable function mapped to OPT generates two additional XOR-connectors. This mapping is proposed in accordance to the minimality criterion as it introduces a minimal set of additional elements. In case (b) the configurable function mapped to OPT generates an additional function and two XOR-connectors. This additional function allows for the XOR-split decision, otherwise there would have been a split connector subsequent to a join connector, which is not lawful. Case (c) is similar to case (a) – instead of the succeeding event a successor split connector (any) is given. In Case (d) the configurable function mapped to OFF may not simply be excluded. As the *any* join may be the last connector in a chain of several connectors, the exclusion of the configurable function may not be possible in every case (if the connector chain is composed of join connectors only, events preceding the connector chain can be eliminated together with the function. If the connector chain also includes split connectors, there are further functions at the end of the chain that require the events in order to comply with the EPC alternation rule). The optional function follows a similar idea as applied in case (b). All of these derivation rules preserve the correctness of the model.

Figure 5. Derivation rules for configured functions



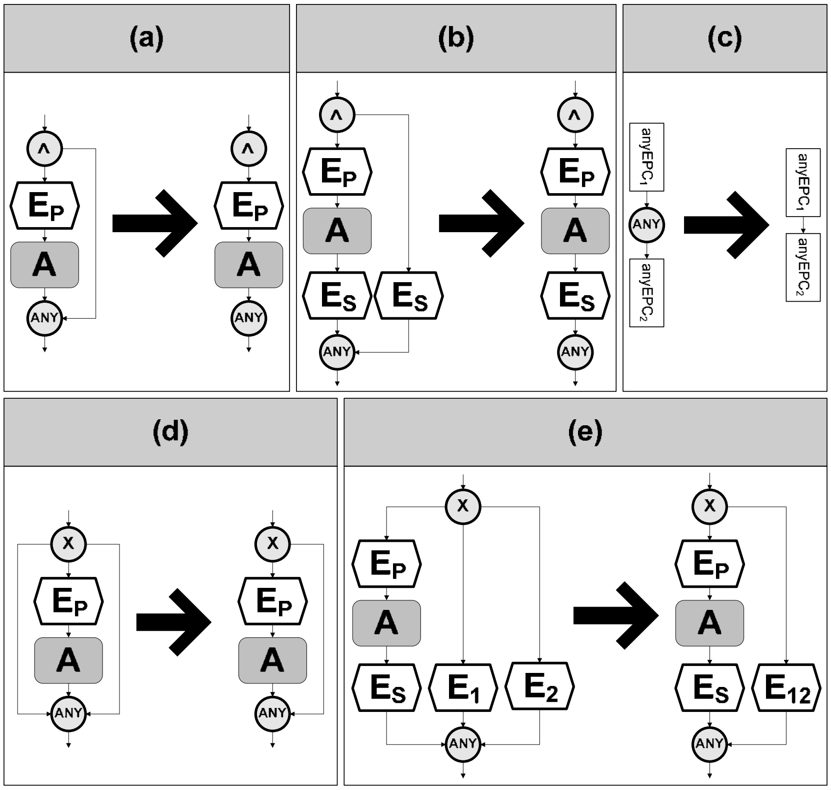
Configured connectors can mostly be derived in a straight-forward manner. If a configurable connector is not configured to a sequence, only its label has to be adopted. If a connector is configured to a sequence SEQ$_n$, those succeeding paths that are not to be included in the build time model have to be eliminated. This means that all subsequent elements are to be excluded from the model until a join connector is reached. If there are no more paths to be eliminated, it must further be checked whether there are join connectors in the model that do not link to any incoming arc. Paths starting with these joins have to be eliminated, too, and the check must be repeated. This procedure is iterated until there are no more connectors without incoming arcs. Figure 6 illustrates this procedure by presenting the case of a split connector whose outgoing paths are eliminated. Following our argumentation this connector and its successor path must be eliminated until a join connector is reached. Again, these derivation rules preserve the correctness of the model.

Figure 6. Example: Connector configured to SEQ$_2$



After deriving configured functions and configured connectors, the resulting EPC may still include unnecessary process graph structures. Functions that are switched OFF and connectors that are configured to SEQ*n* may lead to empty paths or connectors with only one incoming and one outgoing arc (for instance the XOR connector in the resulting model shown in Figure 6). In order to comply with the minimality criterion, certain graph reduction rules have to be applied. Figure 7 gives five reduction rules that are sufficient to derive EPCs that comply with the minimality criterion. Rule (a) eliminates arcs *a* from an AND-split to an AND-join if there is a path from the split to the join that does not pass *a*. Rule (b) deletes a path of concurrency if that path only includes an event and no function. Rule (c) eliminates connectors that only have one incoming and one outgoing arc. Rule (d) deletes an arc between an OR split or an XOR split and a join connector if there is another arc between them. Rule (e) merges two events if they both are successors of an OR split or an XOR split and predecessor of the same join connector. These reduction rules preserve a minimal process graph structure that represents the control flow of the configured process flow variant.

Figure 7. Reduction rules to derive minimal EPCs



The previous derivation rules can be summarized in the definition of a respective derivation algorithm. The algorithm includes the steps 1-4 for connector configuration, 5-6 for graph reduction, 7 for function configuration, and 8-9 for graph reduction. We start with the configuration of connectors as sequence configurations might already reduce the model; in particular, it may lead to the exclusion of configurable functions. Furthermore, connector configuration may result in unnecessary connectors. The graph is reduced in steps 5-6, as the removal of unnecessary connectors before handling configurable functions allows applying the derivation rules (a) and (c) of Figure 7, which in turn result in a smaller graph than rules (b) and (d). Still, function configuration may also result in unnecessary connectors that have to be removed in steps 8-9.

1. Map configured connectors to regular connectors in adherence to the configuration value.
2. If the configuration value is SEQ$_n$ eliminate paths (including all nodes) $i \neq n$, until a join connector or an end node is reached.
3. Check whether there is a connector c without any incoming arcs. If yes, goto 4. If no, goto 5.
4. Eliminate all paths starting with connector c until a join connector or an end node is reached. Goto 3.
5. Check whether one of the reduction rules shown in Figure 7 is applicable. If yes, goto 6. If no, goto 7.
6. Apply one reduction rule and goto 5.
7. Configure functions according to the rules shown in Figure 5.
8. Check whether one of the reduction rules shown in Figure 7 is applicable. If yes, goto 9. If no, end.
9. Apply one reduction rule and goto 7.

Steps 1 to 9 ensure that all configurable nodes in a C-EPC are either deleted from the model or mapped to regular EPC counterparts. At this stage, we can ensure that the resulting process graph does neither contain semantically ambiguous process paths nor unnecessary ones. What we cannot ensure is a formal semantics of the resulting EPC (Kindler, 2005; van der Aalst, 1999). Yet, our extension (and the respective reduction) approach allows for the application of existing formalization approaches, e.g., (Kindler, 2005; van der Aalst, 1999) as a semantic foundation for (derived) EPCs.

The algorithm as shown here rests on the specification of C-EPCs in XML (Mendling, Recker, Rosemann, & van der Aalst, 2005; Recker, Rosemann et al., 2006) using the interchange format EPML (Mendling & Nüttgens, in press) and can be implemented using the object-oriented scripting language XOTcl (Neumann & Zdun, 2000) (the prototype program and the EPML specifications can be downloaded from http://wi.wu-wien.ac.at/~mendling/EPML).

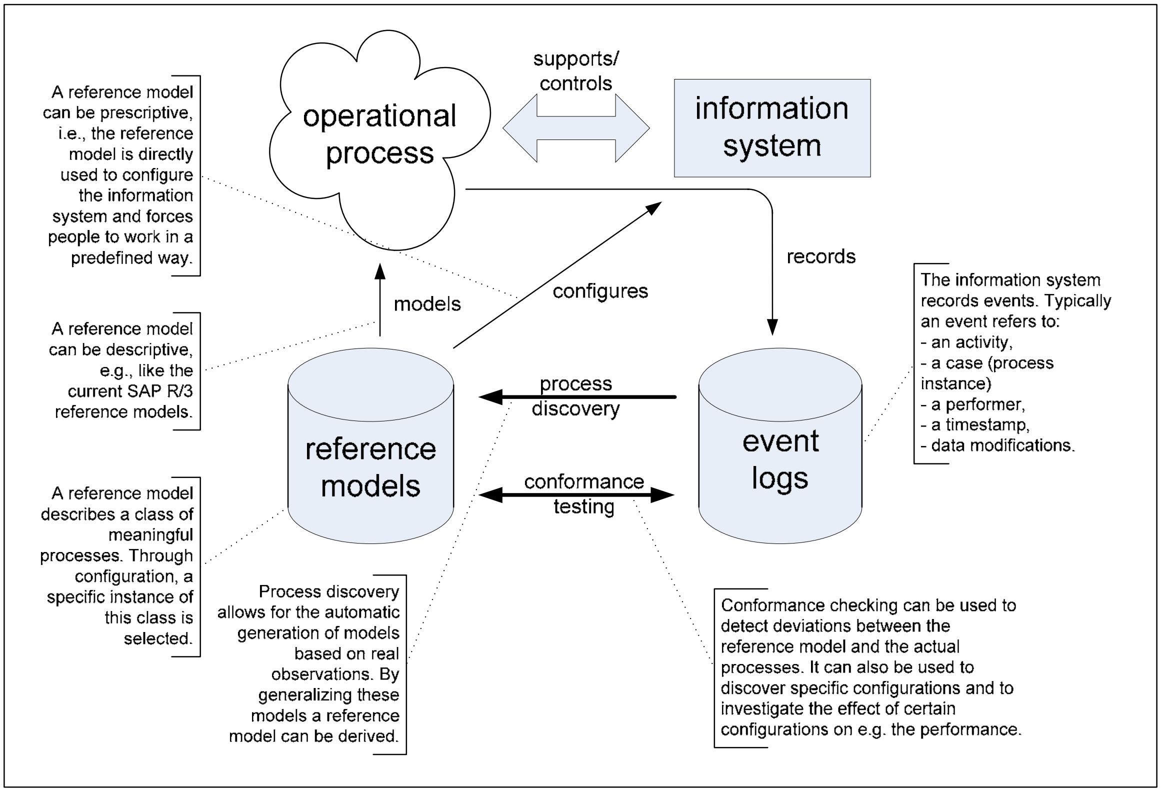
FUTURE TRENDS

Mining Configurable Reference Models

Most of the work reported in this chapter discusses the use of configurable process models as a way to actually configure an ES, i.e., the model is used to realize the system. However, configurable process models (e.g., C-EPCs) can also be used as a way to analyze the processes supported by the system and to "discover" the actual system configuration. As a starting point for such types of analysis one can use audit trails (also known as event or transaction logs) and apply *process mining techniques*.

The goal of process mining is to extract information about processes from event logs (van der Aalst et al., 2003). Process mining techniques such as the alpha algorithm (van der Aalst, Weijters, & Maruster, 2004) typically assume that it is possible to sequentially record events such that (a) each event refers to an activity (i.e., a well-defined step in the process) and (b) each event refers to a case (i.e., a process instance). Moreover, there are other techniques explicitly using additional information such as (c) the performer also referred to as originator of the event (i.e., the person/resource executing or initiating the activity), (d) the timestamp of the event, or (e) data elements recorded with the event (e.g., the size of an order). This information can be used to automatically construct process models. For example, the Multi-Phase Mining approach (van Dongen & van der Aalst, 2004) can be used to construct an EPC describing the behavior observed in the log. There are mature tools such as the ProM framework (van Dongen, Alves de Medeiros, Verbeek, Weijters, & van der Aalst, 2005) available to construct different types of models based on process executions.

Figure 8: The relation between reference models and process mining



A reference model can be prescriptive, i.e., the reference model is directly used to configure the information system and forces people to work in a predefined way.

A reference model can be descriptive, e.g., like the current SAP R/3 reference models.

A reference model describes a class of meaningful processes. Through configuration, a specific instance of this class is selected.

Process discovery allows for the automatic generation of models based on real observations. By generalizing these models a reference model can be derived.

The information system records events. Typically an event refers to:
- an activity,
- a case (process instance)
- a performer,
- a timestamp,
- data modifications.

Conformance checking can be used to detect deviations between the reference model and the actual processes. It can also be used to discover specific configurations and to investigate the effect of certain configurations on e.g. the performance.

There are several ways to use event logs in the context of configurable reference models (see Figure 8). Reference models can be *descriptive* or *prescriptive*, i.e., they are used to describe a process or control respectively guide the system. The SAP reference models are expressed in terms of EPCs describing how people should/could use the SAP system. In reality, however, the real process may deviate from the modeled process, e.g., the implementation is not consistent with the specification, or people use a SAP solution in a way not modeled in any of the EPCs. Even if reference models are more of a prescriptive nature, it is still interesting to investigate how people really use the system.

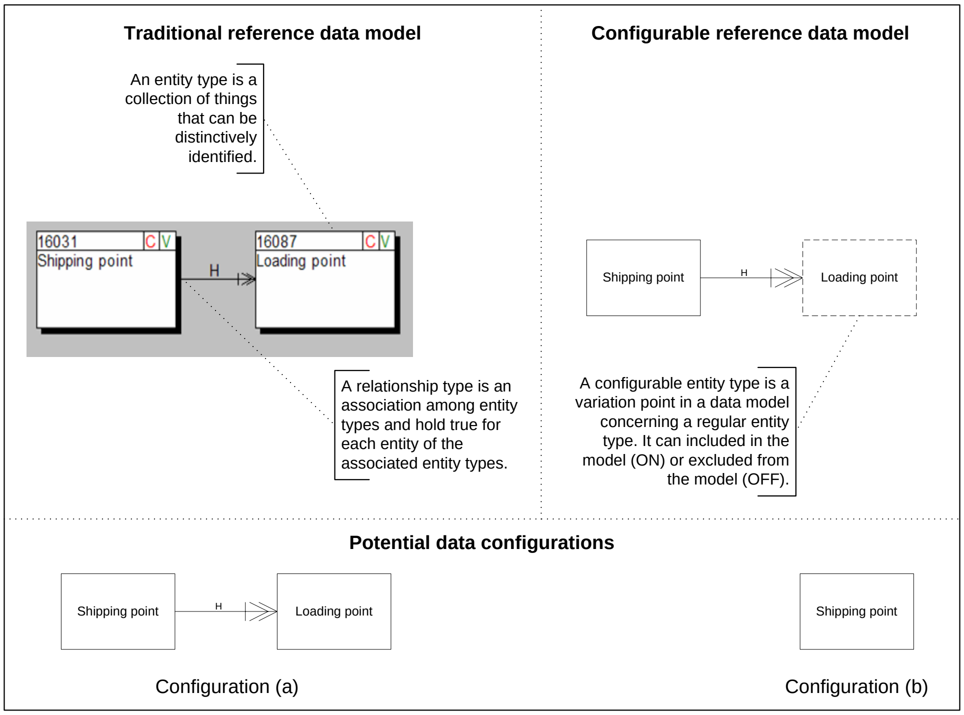
Figure 8 shows that reference models can be used to configure an information system (prescriptive) or to merely model the desired process (descriptive). Independent of the way the reference model is used; most information systems log events in the form of audit trails or transaction logs. The information can be used for *process discovery* and *conformance testing*. Process discovery aims at the construction of models based on the logs without explicitly using some a-priori reference model. This approach is used to construct models that can be used for comparison with existing reference models, or to generate input for the construction of new reference models. Conformance testing can be used to compare real processes with some a-priory knowledge represented in the form of a reference model. It may be used to see if some descriptive reference model is actually followed in reality. Note that system users may deviate from the procedure prescribed in the reference models. Such information can be used for auditing or process improvement. Moreover, the configuration itself can be investigated, e.g., which configuration is used, what is the effect of using a specific configuration, etc.

Process mining is far from trivial. Knowledge of the many ways in which a system may be used, can assist process mining techniques, as illustrated by Jansen-Vullers, van der Aalst, & Rosemann (2006). Based on inspecting the event logs it is relatively easy to discover the particular configuration being used. Moreover, event logs can be used to "diagnose" a configuration. For example, using process mining it is possible to automatically locate the bottlenecks and present them in the context of the configurable process model (e.g., a function in the C-EPC). This may assist the reconfiguration of the system. Furthermore, process mining techniques can be used to compare different configurations and their effects on the performance of the resulting process, which supports an “evidence-based” approach towards business process management.

Configurable Data Modeling Languages

So far we have covered the configurability of reference process models. Yet, given that reference models are often used in the context of business systems, there are more perspectives to consider. Business systems are not only popular since they provide process-oriented support for typical functional areas such as Procurement or Materials Management but also since they provide integrated data repositories across the whole enterprise. Accordingly, available reference models not only depict business processes but also the data structure of business systems. As an example, the SAP reference data model covers in the version 4.6 more than 230 business objects clustering more than 4,000 entity types. A configuration approach needs to place emphasis on the configuration of reference data models as well. Consider an organizational perspective:
Reference data models are of particular importance to the configuration of system organizational units as they precisely depict the given opportunities of a business system. A subset of the SAP reference data model (approx. 30-40 entity types) allows for a complete description of the interrelations between system organizational units such as company, factory, or distribution channel, which facilitates configuration decisions as to the system organizational structure.
Similar to the process perspective, current reference data models are typically based on traditional modeling techniques such as the Entity-Relationship Modeling (ERM) notation (Chen, 1976). *Entity types* are used to group and depict distinct subjects of interest, e.g., customers, organizations, sales order items etc. These entities may possess various attributes for further specification. Relationships between such elements of interest are depicted using *relationship types* that specify the type of association between distinct entities. *Cardinalities* can further be used to specify the extent of dependency between associated entity types.
Classical data modeling techniques do not allow for the depiction of configuration information such as variation points or configuration requirements (Rosemann & Shanks, 2001). In the following we discuss some configuration decisions that can be made and how they could be depicted in reference data models. Extracts of the SAP reference data model are used as an example. The structure of this analysis follows the main constructs of Entity-Relationship-Models, i.e., entity types and relationship types (Chen, 1976). Note that the variant used here is called SAP-Structured ERM, refer, for instance, to (Seubert, Schäfer, Schorr, & Wagner, 1994).
Transparent examples for model configurations related to *optional entity types* can be found in Enterprise Systems in the definition of system organizational structures. The Sales & Distribution solution in SAP, for example, requires for a decision whether shipping points of an enterprise are to be subdivided into loading points. The IMG (Bancroft et al., 1997) marks this decision as optional. This variation point, however, cannot be reflected in the available reference data model (see Figure 9) as the data structure is statically fixed.

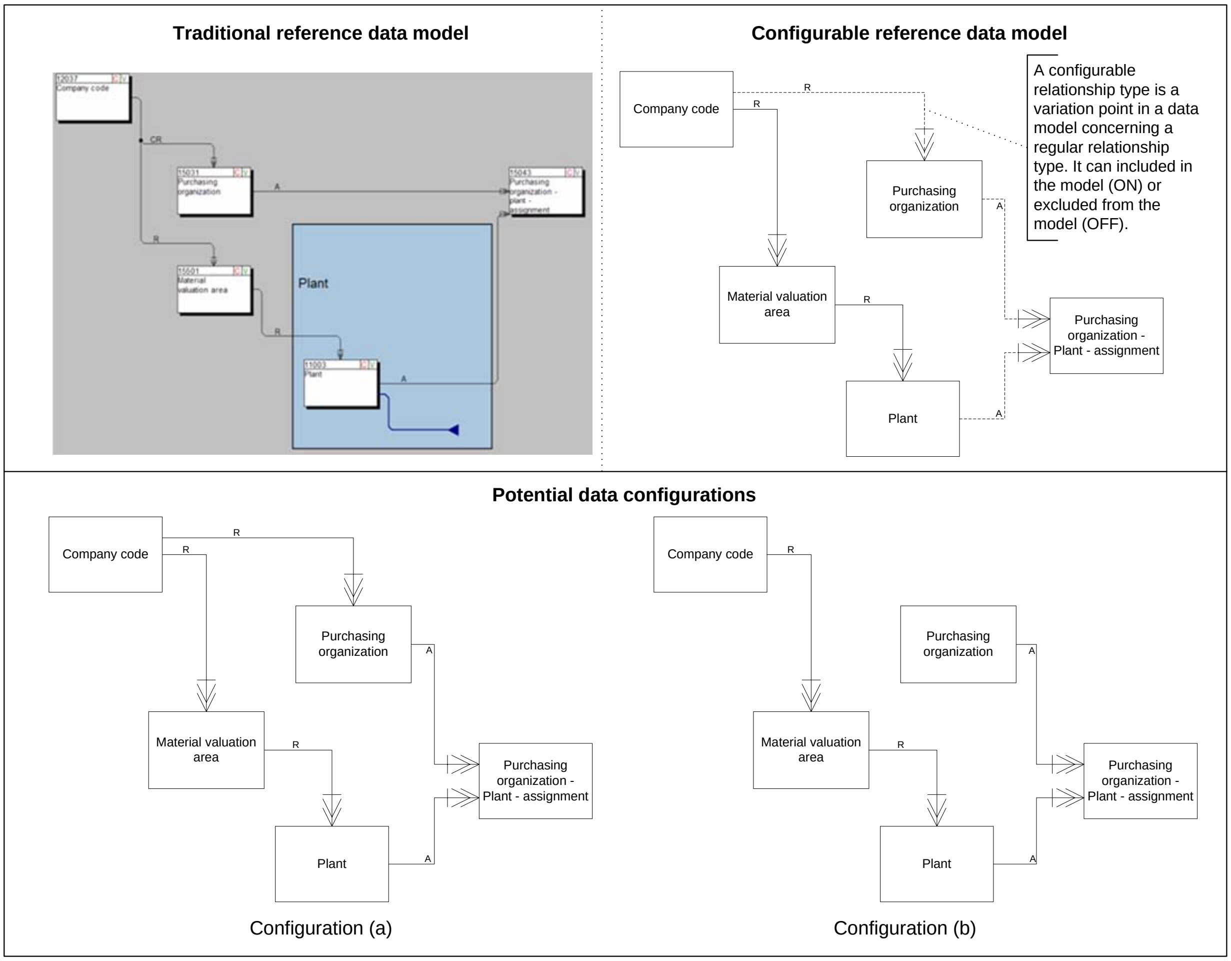
Figure 9. Configuration of reference data models – Entity types


Traditional reference data model
An entity type is a collection of things that can be distinctively identified.
16031 C V
Shipping point
H
16087 C V
Loading point
A relationship type is an association among entity types and hold true for each entity of the associated entity types.
Configurable reference data model
Shipping point
H
Loading point
A configurable entity type is a variation point in a data model concerning a regular entity type. It can included in the model (ON) or excluded from the model (OFF).
Potential data configurations
Shipping point
H
Loading point
Shipping point
Configuration (a)
Configuration (b)

In a configurable reference data model, optional entity types such as Loading Point could be highlighted with a dotted line, thereby indicating that such organizational structure may (a) or may not be implemented (b).

The configuration of *optional relationship types* includes two decisions. First, if the relationship type is required at all. If the relationship is required, a second decision is related to what cardinalities the relationship should have. Again, consider an organizational perspective: The IMG allows for the decision whether or not to assign a purchasing organization to a company code, i.e., whether procurement may be effectuated *company-specific* for all plants assigned to that company (Figure 10, configuration (a)), or whether procurement may be effectuated *plant-specific* for all the plants assigned to the purchasing organization (Figure 10, configuration (b)), irrespective of the super-ordinate company code. Again, the available reference data model cannot reflect this decision as the relationship between the entity types Company Code and Purchasing Organization is fixed.

Figure 10. Configuration of reference data models – Relationship types


Traditional reference data model

Configurable reference data model

A configurable relationship type is a variation point in a data model concerning a regular relationship type. It can included in the model (ON) or excluded from the model (OFF).

Potential data configurations

Configuration (a)

Configuration (b)

A configurable reference data model could highlight this variation point by using a dotted line for the connection between these entity types.

There is a need to further explore configurability of reference data models. We only presented a brief outline of a proposed conceptual extension to existing reference data modeling techniques. Our short discussion revealed that, following the idea of configurable reference process modeling, the design principles that led to the development of C-EPCs may also be used to extend or refine other reference modeling techniques towards configurability (leading for example to C-ERMs). Exemplarily we elaborated on the conceptual development of a configurable data modeling technique that allows for the modeling of optional entity types and optional relationship types. Clearly, this has to be considered work-in-progress but nevertheless denotes an important and interesting research facet in the future of (configurable) reference modeling.

CONCLUSIONS

This chapter discussed and introduced extensions to conceptual modeling languages in order to facilitate the configuration of reference models. These modeling languages have been developed in light of a number of critical design principles which are of relevance following the paradigm of information model reuse. We used an extension of the Event-driven Process Chain to demonstrate the design of a configurable reference process modeling language. Furthermore, we gave first insights into how configurable models can be derived via process mining from executed business system-supported processes. In principle, other modeling languages could be extended in similar ways. It has been discussed how the idea of configuring process models can be applied to other views, such as the data perspective. We briefly reported on the development of a configurable data modeling language as an example.

We expect research on configurable reference modeling to give a stimulating input to both academic and practical work around reference models in the future. The development of generic, configurable languages such as the C-EPC and the establishment of tool-neutral interchange formats such as EPML (Mendling & Nüttgens, in press) or the XML Metadata Interchange (XMI) format (OMG, 2005) provide promising prototype examples that strive for practical adoption in the form of commercial solutions. Configurable reference models may be used to facilitate a model-driven implementation process of business systems (Recker, Mendling, van der Aalst, & Rosemann, 2006). Or, the usage of configurable reference models can lead to the cross-organizational consolidation of previous process configurations, thereby accumulating an evidence-based body of knowledge as to the configuration and enactment of business processes across multiple industry sectors, regions and cultures. These are just a few ideas but they already indicate that reference modeling and model configurability continue to emerge as a vibrant and influential research discipline in the future.

REFERENCES

Bancroft, N. H., Seip, H., & Sprengel, A. (1997). *Implementing Sap R/3: How to Introduce a Large System into a Large Organization* (2nd ed.). Englewood Cliffs, New Jersey: Prentice Hall.

Beer, S. (1966). *Decision and Control: The Meaning of Operational Research and Management Cybernetics*. London, UK: John Wiley & Sons.

Chen, P. P.-S. (1976). The Entity Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems, 1*(1), 9-36.

Curran, T., Keller, G., & Ladd, A. (1997). *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, New Jersey: Prentice Hall.

Daneva, M. (2000). Practical Reuse Measurement in ERP Requirements Engineering. In B. Wangler & L. Bergmann (Eds.), *Advanced Information Systems Engineering: 12th International Conference* (Vol. 1789, pp. 309-324). Stockholm, Sweden: Springer.

Daneva, M. (2004). ERP Requirements Engineering Practice: Lessons Learned. *IEEE Software, 21*(2), 26-33.

Davenport, T. H. (1998). Putting the Enterprise into the Enterprise System. *Harvard Business Review, 76*(4), 121-131.

Davenport, T. H. (2000). *Mission Critical: Realizing the Promise of Enterprise Systems*. Boston, Massachusetts: Harvard Business School Press.

Dreiling, A., Chiang, M., Rosemann, M., & van der Aalst, W. M. P. (2005). Towards an Understanding of Model-Driven Process Configuration and its Support at Large. In N. C. Romano (Ed.), *2005 Americas Conference on Information Systems* (pp. 2084-2092). Omaha, Nebraska: Association for Information Systems.

Dumas, M., van der Aalst, W. M. P., & ter Hofstede, A. H. M. (Eds.). (2005). *Process Aware Information Systems: Bridging People and Software Through Process Technology*. Hoboken, New Jersey: John Wiley & Sons.

Fettke, P., & Loos, P. (2003). Classification of Reference Models - a Methodology and its Application. *Information Systems and e-Business Management, 1*(1), 35-53.

Halmans, G., & Pohl, K. (2003). Communicating the Variability of a Software-Product Family to Customers. *Software and System Modeling, 2*(1), 15-36.

Jansen-Vullers, M. H., van der Aalst, W. M. P., & Rosemann, M. (2006). Mining Configurable Enterprise Information Systems. *Data & Knowledge Engineering, 56*(3), 195-244.

Keller, G., Nüttgens, M., & Scheer, A.-W. (1992). *Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)"* (Working Paper No. 89). Saarbrücken, Germany: Institut für Wirtschaftsinformatik, Universität Saarbrücken (in German).

Kesari, M., Chang, S., & Seddon, P. B. (2003). A Content-Analytic Study of the Advantages and Disadvantages of Process Modelling. In J. Ang & S.-A. Knight (Eds.), *14th Australasian Conference on Information Systems*. Perth, Australia: School of Management Information Systems.

Kindler, E. (2005). On the Semantics of EPCs: Resolving the Vicious Circle. *Data & Knowledge Engineering, 56*(1), 23-40.

Mendling, J., & Nüttgens, M. (in press). EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). *Information Systems and e-Business Management*.

Mendling, J., Recker, J., Rosemann, M., & van der Aalst, W. M. P. (2005). Towards the Interchange of Configurable EPCs: An XML-based Approach for Reference Model Configuration. In U. Frank & J. Desel (Eds.), *Enterprise Modelling and Information Systems Architectures 2005* (Vol. P-75, pp. 8-21). Klagenfurt, Germany: German Computer Society.

Mendling, J., Recker, J., Rosemann, M., & van der Aalst, W. M. P. (in press). Generating Correct EPCs from Configured CEPCs. In *21th Annual ACM Symposium on Applied Computing*. Dijon, France: ACM.

Misic, V. B., & Zhao, J. L. (2000). Evaluating the Quality of Reference Models. In A. H. F. Laender, S. W. Liddle & V. C. Storey (Eds.), *Conceptual Modeling - ER 2000* (Vol. 1920, pp. 484-498). Salt Lake City, Utah: Springer.

Neumann, G., & Zdun, U. (2000). XOTcl, an Object-Oriented Scripting Language. In *7th USENIX Tcl/Tk Conference* (pp. 163-174). Austin, Texas.

OMG. (2005). *MOF 2.0/XMI Mapping Specification, v2.1*. Retrieved January 17, 2006, from http://www.omg.org/docs/formal/05-09-01.pdf

Oracle. (2006). *Oracle Consulting Business Solutions*. Retrieved January 13, 2006, from http://www.oracle.com/consulting/solutions/index.html

Recker, J., Mendling, J., van der Aalst, W. M. P., & Rosemann, M. (2006). Model-driven Enterprise Systems Configuration. In E. Dubois & K. Pohl (Eds.), *Advanced Information Systems Engineering - CAiSE 2006* (pp. forthcoming). Luxembourg, Grand-Duchy of Luxembourg: Springer.

Recker, J., Rosemann, M., van der Aalst, W. M. P., & Mendling, J. (2006). On the Syntax of Reference Model Configuration. Transforming the C-EPC into Lawful EPC Models. In C. Bussler & A. Haller (Eds.), *Business Process Management Workshops* (Vol. 3812, pp. 497-511). Berlin, Germany et al.: Springer.

Rosemann, M. (2000). Using Reference Models within the Enterprise Resource Planning Lifecycle. *Australian Accounting Review, 10*(3), 19-30.

Rosemann, M. (2002). Application Reference Models and Building Blocks for Management and Control (ERP Systems). In P. Bernus, L. Nemes & G. Schmidt (Eds.), *Handbook of Enterprise Architecture* (pp. 595-616). Berlin, Germany: Springer.

Rosemann, M., & Shanks, G. (2001). Extension and Configuration of Reference Models for Enterprise Resource Planning Systems. In G. Finnie, D. Cecez-Kecmanovic & B. Lo (Eds.), *12th Australasian Conference on Information Systems* (pp. 537-546). Coffs Harbour, Australia: School of Multimedia and Information Technology.

Rosemann, M., & van der Aalst, W. M. P. (In Press). A Configurable Reference Modelling Language. *Information Systems*.

Rosemann, M., Vessey, I., & Weber, R. (2004). Alignment in Enterprise Systems Implementations: The Role of Ontological Distance. In *25th International Conference on Information Systems* (pp. 439-448). Washington D.C.: Association for Information Systems.

SAP. (2006). *SAP Business Maps: Solution Composer*. Retrieved January 13, 2006, from http://www.sap.com/solutions/businessmaps/composer/

Scheer, A.-W. (2000). *ARIS - Business Process Modeling* (3rd ed.). Berlin, Germany et al.: Springer.

Seubert, M., Schäfer, T., Schorr, M., & Wagner, J. (1994). Praxisorientierte Datenmodellierung mit der SAP-SERM-Methode. *EMISA Forum, 4*(2), 71-79 (in German).

Silverston, L. (2001a). *The Data Model Resource Book, Volume 1: A Library of Universal Data Models for All Enterprises*. New York, New York: John Wiley & Sons.

Silverston, L. (2001b). *The Data Model Resource Book, Volume 2: A Library of Data Models for Specific Industries* (2 ed.). New York, New York: John Wiley & Sons.

Stein, T. (1998, August 31). SAP Sued Over R/3. *Information Week,* p. 134.

van der Aalst, W. M. P. (1999). Formalization and Verification of Event-driven Process Chains. *Information and Software Technology, 41*(10), 639-650.

van der Aalst, W. M. P., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G., & Weijters, A. J. M. M. (2003). Workflow Mining: A Survey of Issues and Approaches. *Data & Knowledge Engineering, 47*(2), 237-267.

van der Aalst, W. M. P., Weijters, A. J. M. M., & Maruster, L. (2004). Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering, 16*(9), 1128-1142.

van Dongen, B. F., Alves de Medeiros, A. K., Verbeek, M., Weijters, A. J. M. M., & van der Aalst, W. (2005). The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo & P. Darondeau (Eds.), *Applications and Theory of Petri Nets 2005* (Vol. 3536, pp. 444-454). Berlin et al.: Springer.

van Dongen, B. F., & van der Aalst, W. M. P. (2004). Multi-phase Process Mining: Building Instance Graphs. In P. Atzeni, W. W. Chu, H. Lu, S. Zhou & T. W. Ling (Eds.), *Conceptual Modeling - ER 2004* (pp. 362-376). Shanghai, China: Springer.

Welti, N. (1999). *Successful SAP R/3 Implementation: Practical Management of ERP Projects*. Reading, MA: Addison-Wesley.

Wisse, P. (2000). *Metapattern: Context and Time in Information Models*. Boston, Massachusetts: Addison-Wesley.

Mr. Jan Recker (1979) is a PhD student at the Business Process Management research group at the Faculty of Information Technology at Queensland University of Technology Brisbane,

Australia. His research interests include Business Process Modeling, Conceptual Model Evaluation, Process Configuration and Reference Modeling for Enterprise Systems. He is also a part-time teacher of Business Process Management-related units at the School of Information Systems at Queensland University of Technology.

Prof. Michael Rosemann, PhD (1967) is a full professor of Information Systems and co-leader of the Business Process Management Research Group at the Faculty of Information Technology at Queensland University of Technology. He is also a member of the Australian Research Council College of Experts. His research interests include business process management, information systems, process models, workflow management systems, enterprise systems and ontologies. He is the author and editor of five books and published more than 120 refereed journal papers, book chapters and conference papers on these topics.

Prof. Wil van der Aalst, PhD (1966) is a full professor of Information Systems and head of the Information Systems sub-department of the department of Technology Management at Technische Universiteit Eindhoven. He is also an adjunct professor at the Faculty of Information Technology of Queensland University of Technology. He directs the Eindhoven Digital Laboratory for Business Processes (EDL-BP) and is a fellow and management team member of the research institute BETA. His research interests include business process management, information systems, simulation, Petri nets, process models, workflow management systems, process mining, verification techniques, enterprise resource planning systems, computer supported cooperative work and interorganizational business processes. He published more than 200 books, journal papers, book chapters, conference papers and reports on these topics.

Ass. Prof. Monique Jansen-Vullers, PhD (1969) is an assistant professor at the Department of Technology Management at Eindhoven University of Technology (TUE), and member of the BETA research group. Currently she is working on (i) configurable reference models, (ii) process mining in Enterprise Resource Planning environments and (iii) business process redesign. She is the author of several academic publications in the mentioned research fields.

Alexander Dreiling, PhD (1975) is a researcher at SAP Research in Brisbane, Australia. Prior to joining SAP he worked as a research assistant at the European Research Center for Information Systems (ERCIS) in Münster, Germany and at the Centre for Information Technology Innovation in Brisbane, Australia. His research interests comprise conceptual data warehouse modeling, conceptual process modeling, and process configuration. His research so far led to approx. 25 refereed journal and conference publications.