Model-Driven Enterprise Systems Configuration

Jan Recker[1], Jan Mendling[2], Wil van der Aalst[1,3], and Michael Rosemann[1]

[1] Queensland University of Technology,
126 Margaret Street, Brisbane QLD 4000, Australia
{j.recker, w.vanderaalst, m.rosemann}@qut.edu.au
[2] Vienna University of Economics and Business Administration,
Augasse 2-6, 1090 Vienna, Austria
jan.mendling@wu-wien.ac.at
[3] Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract. Enterprise Systems potentially lead to significant efficiency gains but require a well-conducted configuration process. A promising idea to manage and simplify the configuration process is based on the premise of using reference models for this task. Our paper continues along this idea and delivers a two-fold contribution: first, we present a generic process for the task of model-driven Enterprise Systems configuration including the steps of (a) *Specification* of configurable reference models, (b) *Configuration* of configurable reference models, (c) *Transformation* of configured reference models to regular build time models, (d) *Deployment* of the generated build time models, (e) *Controlling* of implementation models to provide input to the configuration, and (f) *Consolidation* of implementation models to provide input to reference model specification. We discuss inputs and outputs as well as the involvement of different roles and validation mechanisms. Second, we present an instantiation case of this generic process for Enterprise Systems configuration based on Configurable EPCs.

1 Enterprise Systems and Reference Modeling

Over the last years, Enterprise Systems (ES) have evolved to comprehensive IT-supported business solutions that presumptively support and enhance organizations in their business operations. This, however, only holds true for such systems that are well-aligned with organizational requirements. As Enterprise Systems are developed in a generic manner in order to provide benefits to a wide variety of organizations, industry sectors and countries, their implementation entails the problem of aligning business and IT. Alignment, however, implies extensive configuration and customization efforts in the implementation process and may lead to significant implementation costs that exceed the price of software licenses by factor five to ten [1].

ES vendors are aware of these problems and try to increase the manageability of the implementation process. One respective measure is to deliver ES products

E. Dubois and K. Pohl (Eds.): CAiSE 2006, LNCS 4001, pp. 369–383, 2006.

along with extensive documentation and specific implementation support tools. *Reference models* play a central role within such documentation. Vendors provide a set of process models as reference models of their software package [2]. The SAP reference model as such an example includes a large number of process models representing the system processes [3]. However, research shows that reference models still are only of limited use to the ES configuration process [4]. This is mainly due to a lack of conceptual support for configuration in the underlying modeling language. In this context, a *configurable modeling language* should at least support the structured modification and exclusion of model elements or whole parts of a model as well as the definition of constraints on configurability [5]. This is of particular importance for leveraging the main objective of reference models, i.e., streamlining the adaptation of ES. Beyond conceptual support in terms of flexible or configurable modeling languages, see e.g. [5, 6], there is a need for a clearly structured configuration procedure. ES configuration based on configurable reference models is a multi-facetted task requiring guidance to the overall process. It comprises in particular model configuration, validation, translation, deployment, controlling, and consolidation; with each of these subtasks demanding not only profound knowledge of configurable reference modeling but also of the processes of the organization. A dedicated approach is needed to manage the process of model-driven Enterprise Systems configuration all the way from model design to deployment.

Following this line of argumentation this paper reports on the development and application of a generic engineering process for the design and usage of configurable reference models in a model-driven approach towards Enterprise Systems configuration. To be more concise, the *contribution of our paper* is two-fold: First, we introduce an *engineering process* covering the tasks of specification, configuration, transformation and deployment of configurable reference models and the two feedback loops of controlling and consolidation. The engineering process will be described on a generic level to allow for wider uptake in ES contexts beyond the limits of any given modeling language. Second, as an instantiation case, we report on the deployment of this generic engineering process in the development and application of *Configurable EPCs* (C-EPCs) [5, 7] in the context of model-driven ES configuration. We proceed as follows: Section 2 presents the generic engineering process for configurable reference models. Section 3 then reports on the application of the engineering process based on C-EPCs. After discussing related research in Section 4, we conclude the paper in Section 5.

2 A Generic Configurable Reference Modeling Process

This section defines a process for engineering and deploying configurable reference models in the context of Enterprise Systems implementation. This process is generic in that it is not dependent on a specific modeling technique or method. However, a requirement for the application of our engineering process is that the reference modeling language used throughout the process must be configurable

as defined in Section 1. Subsection 2.1 gives an overview of the process while subsections 2.2 to 2.7 introduce the six steps of model specification, configuration, transformation, deployment, controlling, and consolidation.

2.1 Overview of the Process

Reference model configuration contrasts with the traditional software development process: during implementation, the scope of the ES system is continuously narrowed down to finally meet the requirements of the organization. This process starts with the overall system capabilities which are then reduced to a relevant subset. Reference models can be used as semi-formal descriptions of such overall capabilities [2] and a configurable reference modeling language provides the means to express configuration alternatives. The lifecycle model introduced by Rosemann and van der Aalst [5] illustrates this continuous 'narrowing down' process by defining different "time" notions: At *design time* the overall capabilities of the ES are captured as a (configurable) reference model. At *configuration time* capabilities that are deemed desirable before the background of organizational requirements are selected from the reference model. This means that irrelevant parts of the model are excluded. At *build time* the configured model is deployed on an ES to serve as a 'template' for how the system support for business will look like during execution. Finally, at *run time* single instances are created for specific cases. Our generic process for model-driven ES configuration is related to these "time" notions, however, we extend this lifecycle with *feedback loops* as described below. The overall process defines four major stages comprising reference model *specification*, *configuration*, *transformation*, and *deployment* (see Fig. 1).

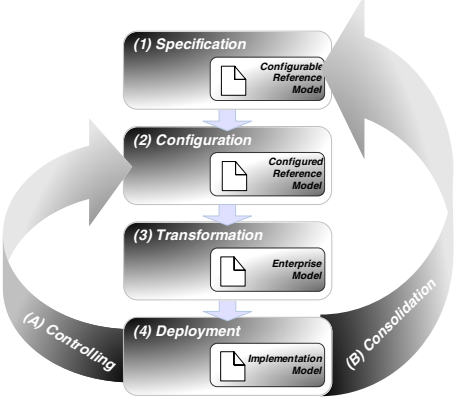

Fig. 1. Engineering process for model-driven ES configuration

The four stages need to be continuously assessed as to their contribution towards fulfilling organizational requirements, which in turn may be subjected to modification due to internal or external changes. As reference models capture

knowledge in the form of current best practice descriptions, they form part of an *organizational learning cycle* by (a) being affected by changes within the organizational setting and (b) effectuating such changes via technological or organizational developments. Organizational learning in general can be differentiated in single- and double-loop learning [8]. Single-loop learning can be understood as error minimization in accordance to given objectives and assumptions. Double-loop learning includes a reflection upon these assumptions and may result in completely new objectives, processes and outcomes. Applying these insights to the task of model-driven configuration of Enterprise Systems, we argue that single-loop learning comprises the reflection on a configuration as to its contribution to given organizational requirements. Double-loop learning then is the reflection on the presupposed best practice knowledge captured in the reference models as to whether or not it sufficiently enables organizations to fulfill their objectives. In order to facilitate single- and double-loop learning with configurable reference models, our generic process is extended by two feedback mechanisms, namely *controlling* and *consolidation*. Controlling is understood as the reflection on the implementation of the "best practice" knowledge described in the reference models within the organizational setting, viz., a diagnosis of how well the selected configuration aligns with organizational requirements. Controlling in this sense provides a means to facilitate single-loop learning. Consolidation is understood as a reflection on the specification of the "best practice" knowledge described in the reference models based on current implementation in several organizational settings, viz., a diagnosis of whether the reference model itself (and the ES described within) has to be subjected to refinement or extension due to evolution of technological and/or organizational factors in its domain. Based on this understanding consolidation provides a means to facilitate double-loop learning.

The different stages and loops are explained in the following subsections. In contrast to the lifecycle model used by Rosemann and van der Aalst [5] that merely offers a conceptual distinction of the phases, our engineering process provides guidance for those involved in an ES configuration project by giving detailed recommendations for each of the four stages and the feedback loops. In particular, we will describe for each stage the **inputs** and **outputs**, the different **steps**, **responsibilities**, and **validation mechanisms**.

2.2 Step (1): Specification of Configurable Reference Models

The first step is concerned with model development. The goal is to produce a **configurable reference model** as an output. This configurable reference model captures system functionality, capabilities and structure on a conceptual level (as does a traditional reference model) [2] and furthermore defines *variation points* within the model that capture configurable aspects of an ES. A variation point captures the place of a configuration decision together with the related possible choices and consequences, and thereby serves the concept of variability [9], which empowers constructive model reuse and facilitates the derivation of model variants from the initial model. Concerning input there are basically two options: (1) *Development from scratch*. This means selecting an appropriate configurable

modeling technique to develop the reference models. As to methodical guidance, traditional reference model engineering approaches may be followed. The only additional concern here is to place emphasis on the conceptual description of variation points and configuration-related information within the models. (2) *Extension of existing models.* This option refers to the fact that, often, reference models are already available. As an example, the SAP reference model (Version 4.6) [3] covers more than 1,000 business processes. Such existing reference models are, however, usually depicted using traditional reference modeling techniques that do not allow for the description of configuration-related information, for instance the highlighting and selection of different process alternatives [5]. Hence, a configurable modeling language is needed to extend the existing model in order to express variation points and configuration information. It is efficient to stick to the language in which the reference model is expressed and to extend it by annotating the model with configuration concepts, rather than redefining the model in (yet) another modeling language. A potential solution for re-engineering the existing reference model based on process mining techniques is described in [7].

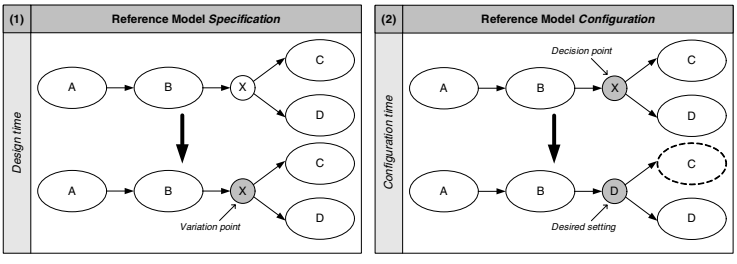

Fig. 2. Specification and configuration of reference model

Part (1) of Fig. 2 illustrates how input and output of the specification step are related. If there is a reference model available, configurable aspects of the system being modeled have to be made obvious in the model by extending it with variation points. In Fig. 2, we exemplarily highlighted such a configurable element by a grey background color.

Concerning responsibilities, the specification step has to rely on ES experts who are familiar both with the functionality of the ES and the support capabilities for an organization's business processes it provides. Furthermore, expertise is required in terms of reference modeling. Usually, such experts are employees of the ES vendor who are responsible for system documentation. If such documentation is not provided by the ES vendor itself, a configurable reference model of an ES might be defined by a consulting company or by an organization using the ES.

Concerning validation mechanisms, existing model quality frameworks (e.g., [10]) can be used in order to ensure the quality of the configurable reference model. This early step and the quality of its output is of crucial importance since as conceptual models used in the requirements specification phase of a system development process determine the acceptability and usability of the

product to be built [11]. Not only the configuration alternatives have to be made explicit, but also constraints in terms of interrelations between certain configuration alternatives. Due to this delicate nature, it definitely calls for a deeper investigation in terms of methodical guidance, which in turn we must consider out of scope for this paper. We nevertheless suggest that the result of this task should be validated by at least a second domain expert.

2.3 Step (2): Configuration of Configurable Reference Models

The second step deals with the configuration of a configurable reference model. Taking the reference model defined in the previous step as input, this task defines a set of *configuration decisions* for all configuration aspects of the model and yields a **configured reference model** as an output. Hence, in the configurable reference model for each configurable node a decision on the desired setting has to be taken. Each variation point in the configurable reference model defines a *decision point* at which the reference model user has to specify a configuration parameter while adhering to potential constraints and requirements. Part (2) of Fig. 2 demonstrates this problem in a simple example. The configurable reference model depicts two mutually exclusive alternatives of conducting business, depicted by a circled X for a logical either-or split: either the sequence $A-B-C$ or $A-B-D$ is allowed. A particular organization has to select one of these two alternatives of conducting their business processes via the Enterprise System. Hence, the X split in this case represents a decision point, e.g., to select the option $A-B-D$ (highlighted by changing the circled X to a circled D), with the consequence of excluding C from the model.

Concerning responsibilities, this configuration step builds on the knowledge of ES experts who are familiar both with the functionality of the ES, the requirements of the organization, and the configuration of reference models. In this context, these are most likely members of a configuration/implementation project team involving consultants and experts of the organization itself.

Concerning validation mechanisms, at this stage, the desired configuration needs to be validated against the constraints defined in the configurable reference model. If these constraints have been specified in a formal manner, this task can be conducted automatically. Consider the following example: an organization chooses for its sales & distribution software package not to offer credit card payment to customers. Conclusively, the accounting software package neither needs to provide functionality for credit card authorization and payment. The first configuration decision has a consequence onto the second variation/decision point in that it restrains the possible set of configuration alternatives. Hence, validation at this stage refers to the evaluation of configuration decisions against constraints or configuration requirements.

2.4 Step (3): Transformation of Configured Reference Models

The third step is concerned with the transformation of a configured reference model as input to an **enterprise model** as output. This enterprise model describes conceptually the way the organization will conduct business with the

support of the Enterprise System once implemented and running. In short, a "traditional" individual model has to be derived from the configured reference model. If the configuration semantics of the configurable reference modeling language have been defined in a formal way and the activities are supported by applications, this task can be automated by a transformation program. Otherwise, the transformation has to be done manually by an ES expert with modeling expertise. It is recommended to automate the transformation, as a manual execution of this task is both time-consuming and error-prone. Furthermore, instead of validating the enterprise model against the configured model, a validation of the correctness of the transformation program is sufficient, which is much more efficient. As an example, modeling languages that are specified via an XML schema can easily be validated and transformed. Still, at least one ES and business expert should inspect the resulting models to validate that the models (still) meet the requirements of the organization. An automated transformation is especially beneficial when both the configuration decisions have to be translated to the output model and the re-establishment of syntactical correctness of the model becomes necessary [12]. For illustration purposes, consider the example given in Part (3) of Fig. 3. It is assumed that an organization has chosen to implement the sequence $A-B-D$ instead of implementing the sequence $A-B-C$. Thus, the option C - which still exists in the configured reference model - needs to be excluded from the enterprise model. Furthermore, the decision point has to be excluded from the model in order to re-establish syntactical correctness.

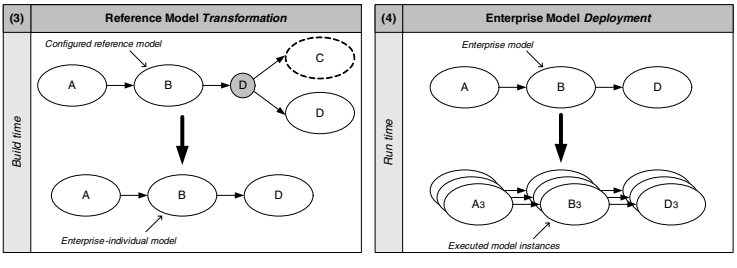

Fig. 3. Transformation to and deployment of enterprise model

2.5 Step (4): Deployment of Enterprise Model

The fourth step is concerned with the deployment of the enterprise model and yields an implement and running enterprise system (which can be understood as an **implementation model**) as output. Part (4) of Fig. 3 shows the principle. There are basically two questions that are important in this context.

First, does a process engine or similar system exist that is able to execute models, in particular the enterprise model, given the modeling language used? It would be desirable if a reference process model that has been transformed to an enterprise model would be directly executable in a workflow engine. A popular example for such an executable process specification is BPEL4WS [13]. If the

model is not directly executable, the enterprise model has to be transformed to a modeling language that runs on a dedicated execution engine. If the semantics of the used modeling language are defined in a formal way, this task can be automated by a transformation program. Otherwise, the transformation has to be done manually by an ES or IT expert with modeling expertise.

Second, does the enterprise model already include run time information about data flow and interfaces to applications? If not, the enterprise model or the transformed enterprise model need to be enriched with technical information, and can only be deployed afterwards. Depending on how much technical information still needs to be added to the model, the deployment has to be done by an IT expert or may also be done by an ES expert. Furthermore, testing of the enterprise models is of crucial importance before deployment, especially when run time information is manually added by IT experts. The implementation models are supposed to be instantiated in order to support the operations of the organization. Accordingly, errors in the models may have a direct impact on business performance.

2.6 Loop (A): Controlling of Instance Models

The single-learning feedback loop stems from the notion of process monitoring and controlling. For the purpose of this paper, process *monitoring* deals with the collection of data about workflow instances at run time, mostly in audit trail logs, i.e., an observation of the processes as they are executed in the organization at hand [14]. Process *controlling*, also referred to as *process mining* [15] or *business process intelligence* [16], deals with the ex-post analysis of logged audit trail data of process enactment. It aims at reviewing process performance as to whether and how processes fulfill organizational requirements and support organizational objectives. As process performance is determined by the support provided by the implemented Enterprise System, we argue here that poor process performance is an indicator for an Enterprise System configuration that does not entirely support all organizational requirements and objectives. Based on noted deviations in process performance, the process, as it is being supported or enacted by the ES, needs to be re-configured in order to improve overall performance. Hence, the feedback loop of controlling provides ex-post evaluation of the customized implementation of the Enterprise System based on actual process enactment performance.

To support the single-loop learning feedback look we use recent achievements in process mining [15]. To illustrate the relationship between process mining and reference models we refer to Fig. 4. Essential for process mining is the presence of an *event log* (also referred to as audit trail or transaction log), which log refers to some event, e.g., the start or completion of some activity. The event may bear a timestamp or refer to the person/application executing it. The event may also hold data, e.g., the outcome of a decision activity. Clearly, an information system that is supporting or controlling an operational process is able to monitor such events. We distinguish between two forms of process mining: *process discovery* and *conformance checking* (see Fig. 4).

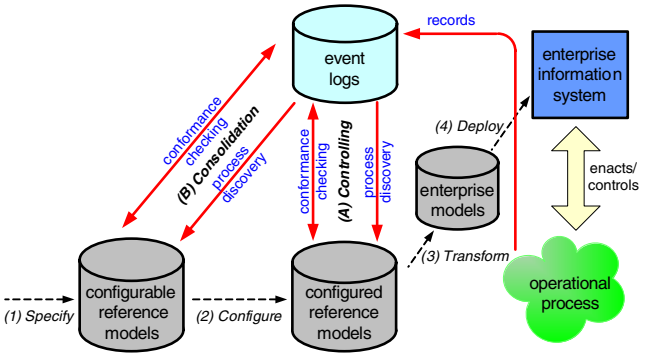

Fig. 4. Process mining approach and relation to configurable reference models

The goal of process discovery is to extract knowledge from event logs in the form of models. These may be process models, e.g., an EPC or Petri net, but also other models such as social networks or time-charts describing the performance (e.g., flow times). Process discovery does not require an a-priori model (such as a reference model), however, the discovered model may be used for delta analysis, i.e., comparing the mined model representing the *actual* process with the reference model representing the *predefined* process. Delta analysis can be used to find parts of the process that are never used or find parts where users deviate from the prescribed procedure. Moreover, the discovered models may refer to other aspects such as time, data and resources. For example, the discovered model may highlight the bottlenecks in the process, reveal the social network (e.g., which people are working together on a frequent basis), or relate properties of cases to their execution (e.g., cases involving more than 1000 euro and handled by the team in Paris tend to be late).

Unlike process discovery, conformance checking *does* require an a-priori model to which it compares the observed behavior as recorded in the log. Using conformance checking one can detect discrepancies but it is also possible to see which parts of the process are really used, where bottlenecks are etc. Clearly, this is very useful for measuring (and quantifying) the "fit" between the real process and some reference model and to pinpoint typical deviations.

To actually measure conformance and to discover a variety of models, we have developed the *ProM framework*[1]. In the context of this framework, several process discovery tools have been developed, e.g., the well-know alpha algorithm [15]. Moreover, the framework offers a *Conformance Checker*, a *Social Network Analyser*, and a variety of other analysis tools.

The dashed lines in Fig. 4 refer to the steps identified in Fig. 1. First, the reference models are specified and then for a particular context (organization and process) they are configured. The configured model is then transformed and deployed. The configured reference model can be compared with the derived models (process discovery) or directly with the event logs (conformance check-

[1] Both documentation and software can be downloaded from `www.processmining.org`

ing). This way it is possible to find different types of problems that may lead to a re-configuration. For example, analysis may show that in reality, the execution of the process does not match with the configured reference process model. This may imply an incorrect implementation, office workers not following the proper procedures, or a misalignment that needs to be addressed by reconfiguring the system. The analysis may also highlight parts of the configured reference model that are rarely active (or over-active), which, too, indicates a suboptimal configuration of the system. Moreover, conformance checking may pinpoint bottlenecks and other performance-related issues. These diagnostics may assist in improving the configuration of the reference model.

Responsibilities for this task are multi-fold. The monitoring step of this stage is best performed by IT experts that capture relevant process performance data in audit trails and have experience in applying process mining techniques. The actual analysis should be done by an analyst having knowledge of process mining and the application domain. It is definitely possible to automate this analysis and offer a kind of "business cockpit" to managers and end-users. Then, the step of controlling is a rather managerial task and merely includes decisions as to how to re-configure the processes in order to increase their performance. Still, based on the assumption that process performance is determined by the support provided by the Enterprise System, an ES expert is recommended to be consulted for this task in order to elicit possible alternatives for supporting existing processes through alternative ES configurations.

2.7 Loop (B): Consolidation of Instance Models

The single-loop learning approach focuses on a specific context (i.e., a given organization and process) and can only result in a reconfiguration. Therefore, it does not aim at improving "best practice" in a broader setting, i.e., it does not reflect on the qualities of the configurable reference model. The double-loop learning approach that we refer to as *consolidation* has a wider scope than controlling. The input of the consolidation feedback loop is a set of instances originating from different configurations, i.e., experiences from multiple applications of the reference model are used as a starting point for the analysis of the reference model itself and not (just) one selected configuration. The result of this analysis can be used to modify the reference model itself. For example, analysis may show that although it is possible to configure a variation point in multiple ways, in reality always the same configuration decision is taken, thus leading to unnecessary configuration work. It is also possible that analysis shows that certain problems (e.g., performance or quality issues) typically occur when a certain configuration is being used. This knowledge can be used to revise the original reference model and the variation points within.

The consolidation phase consists of three smaller steps. First, process mining techniques as described in Section 2.6 are applied in a variety of situations where the reference model has been configured and deployed. For example, situations in different organizational units in the same enterprise or in comparable organizational units across different organizations may be used as input. For each

situation, process mining techniques are used to do process discovery and/or conformance checking. This gives insights into the way the system is really being used, helps to identify problems and is used to quantify the performance of the process. Each of these aspects is linked to the selected configuration and external factors such as load and resource availability. Note that compared to Section 2.6 these results are more likely at an aggregate level. The second step uses the results of this first step and compares all situations to discover patterns. This can be done in a qualitative way ("It seems that configuration A only works properly if combined with configuration B.") or in a quantitative way ("There is a positive correlation between the flow time and a particular configuration setting."). In the third and final step these patterns are used to modify the reference model (see Fig. 4). Note that the structure of the reference model may change. However, we envision that more changes will be made to relationships between the different configuration decisions. Moreover, the use of soft constraints in addition to hard constraints seems to be important. Soft constraints can serve as guidelines based on empirical evidence gained from the feedback loop of consolidation.

The responsibility of this task lies with the developers of the reference models guided by input from the organizations involved.

3 An Instantiation Case Using Configurable EPCs

So far, we outlined a generic process that covers the overall reference model lifecycle and applies it to the area of Enterprise Systems configuration. In the following, we will illustrate the technical feasibility of this process by applying it to the case of C-EPCs in the context of ES configuration. C-EPCs have been developed with the clear intention in mind to facilitate a model-driven approach towards ES configuration. In the following, we assume the reader to have some basic knowledge of EPCs. For an introduction, refer to [17].

Event-Driven Process Chains (EPCs) are a frequently used business process modeling language, especially for describing processes on a conceptual level. EPCs have been developed in a joint project by University of Saarland and SAP [17] and SAP has used them as a modeling language for their SAP R/3 reference model [3]. *Configurable EPCs* (C-EPCs) [5] extend EPCs to allow for the specification of variation points, configuration requirements and configuration guidelines in a reference model, including configurable functions that can be switched *on*, *off* or *optional*; configurable connectors that subsume possible build time connector types, which are less or equally expressive; configuration requirements (must-constraints) and guidelines (should-constraints); and an order relation over the configurable nodes [5]. EPCs have been chosen because they facilitate the usage of the SAP model in *step 1* for the specification of configurable reference models. Respective tool support is available as the ARIS Toolset of IDS Scheer AG is shipped with the SAP model. As a basis for steps 2 to 4, an XML representation of (a) configurable EPCs and (b) configured EPCs based on the EPC Markup Language (EPML) [18] has been specified [19].

This EPML extension serves as an input format for a C-EPC validation tool that has been implemented as a prototype [19]. This tool generates a report on whether a configurable EPC is correct with respect to the C-EPC definition, and whether configuration requirements and guidelines are met. The formal C-EPC definition allows the automation of the validation and, therefore, supports the configuration step (*step 2*). The transformation of C-EPCs to EPC process models bears some challenges which are specific to the syntax and semantics of EPCs [12]. An algorithm has been defined in [20] and implemented to automate this transformation step (*step 3*). This is supposed to speed up the development and grant the correctness of the resulting models. This algorithm is driven by a minimality criterion in order to generate an EPC with as little structure as necessary [20]. In the beginning, it had been an assumption that the generated EPC models can be directly deployed on the ES (*step 4*). As this might not always be the case, a transformation concept from EPCs to executable BPEL [13] process definitions has been developed [21]. As BPEL is a generic language for Web Service composition, this step can only be automated if the data flow and the Web Service endpoints are made explicit in the EPC model. Basically, such information can be included in the configurable model and preserved in the transformation step, so it is still available for deployment. The *two feedback loops* can both be supported by process mining techniques. The ProM framework introduced in Section 2.6 is able to rebuild EPC models from SAP event logs. Also, in [7] it has been shown how process mining can be used to generate C-EPCs from running workflows for controlling or consolidation purposes.

The C-EPC case illustrates that respective tool support for each step of the engineering process has already been established on a prototype basis. The next challenge is to combine the different implementations into a comprehensive configuration framework that can be used by practitioners.

4 Related Work

A number of academic contributions discussing issues related to Enterprise Systems aim at understanding the challenges of ES configuration. For instance, a number of contingencies that potentially impact such projects have been revealed in critical success factor models [22]. Other research claims that ES implementation project failures are likely due to difficulties arising while using specified requirements in the implementation process [23]. Empirical studies, too, tell failure stories [1].

While vendors aim at increasing the chance of ES implementation success by distributing reference models as part of system documentation, these models are at best partly deployed in the configuration of Enterprise Systems. Daneva [4] measured the level of reuse of the SAP reference models in a number of case studies and indicated that full reuse was not achieved in any of them, although sometimes the level of reuse was quite substantial. Some research has focused the field of configurable modeling, good collections of related approaches can be found in [5] and [24]. Some of the discussed approaches are closely related to our

ideas of configurable modeling; worthwhile mentioning here is the approach by Reinhartz-Berger et al. [25], who leverage the re-use of reference models for domain engineering using model specialization mechanisms based on generalization and UML stereotypes.

Concerning limitations, model-driven configuration is well suited for deployment of commercial-off-the-shelf software packages but not as a general approach to software engineering, which cannot entirely be described as a 'scoping' exercise. Also, the notion of re-usable models in the software engineering discipline refers to the employment of building blocks of software fragments in multiple contexts rather than the depiction of best practice patterns. There is, however, some related work. As an example, Haugen et al. [26] present an approach to leverage configurable models for system family engineering. In order to capture model variability, they utilize mechanisms of UML 2.0 composite structures and UML association multiplicities. Yet, their approach focusses more on the derivation of individual software systems from system families than on deriving variants from a given models.

5 Contributions and Limitations

This paper reported on the development and application of a generic engineering process for configurable reference modeling. We first presented a process for model-driven Enterprise Systems configuration consisting of the steps *specification*, *configuration*, *transformation*, and *deployment*, as well as the feedback loops *controlling* and *consolidation*. The second contribution of this paper was the application of this generic process to the development and application of C-EPCS for the purpose of configuring Enterprise Systems. We showed how C-EPCs conceptually facilitate a model-driven configuration process in all of our stages.

Our research has a few limitations. First, our approach needs to be empirically validated with business practitioners. This task is currently being conducted. We have already conducted a pilot laboratory experiment with postgraduate IT students on the perceived usefulness and perceived ease of use of C-EPCs in comparison to EPCs, showing that C-EPCs are in fact perceived as more useful and easier to use for the task of reference model configuration [27]. Second, our approach does not strongly consider the challenge of linking configurable models to Enterprise Systems functionality, i.e., how to link model configurations to actual modifications of programmed code. Third, we applied our generic engineering process to a configurable *process* modeling approach. It would be interesting to link it other perspectives such as a data view, refer, for instance, to [28].

Future work will concentrate on (a) an evaluation of our approach via case study application and (b) the development of a sophisticated configuration framework based on our proof-of-concept implementations. The ultimate goal is then to provide comprehensive tool support towards model-driven systems configuration.

Acknowledgement. The research on C-EPCs has been partly funded by SAP Research and Queensland University of Technology with the Strategic Link with

Industry project "Modelling Configurable Business Processes". SAP is a trademark of SAP AG, Germany.

References

1. Davenport, T.H.: Mission Critical: Realizing the Promise of Enterprise Systems. Harvard Business School Press, Boston, MA (2000)
2. Rosemann, M.: Using Reference Models within the Enterprise Resource Planning Lifecycle. Australian Accounting Review **10** (2000) 19–30
3. Curran, T., Keller, G., Ladd, A.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model. Enterprise Resource Planning Series. Prentice Hall PTR, Upper Saddle River, NJ (1997)
4. Daneva, M.: Practical Reuse Measurement in ERP Requirements Engineering. In Wangler, B., Bergmann, L., eds.: 12th International Conference on Advanced Information Systems Engineering. Volume 1789 of Lecture Notes In Computer Science., Stockholm, Sweden, Springer (2000) 309–324
5. Rosemann, M., van der Aalst, W.: A Configurable Reference Modelling Language. Information Systems **In Press, also available from** `www.BPMCenter.org` (2006)
6. Soffer, P.: Scope Analysis: Identifying the Impact of Changes in Business Process Models. Software Process Improvement and Practice **10** (2005) 393–402
7. Jansen-Vullers, M.H., van der Aalst, W., Rosemann, M.: Mining Configurable Enterprise Information Systems. Data and Knowledge Engineering **56** (2006) 195–244
8. Argyris, C., Schön, D.: Organizational Learning II. Theory, Method, and Practice. Addison-Wesley, Reading, MA et al. (1996)
9. Halmans, G., Pohl, K.: Communicating the Variability of a Software-Product Family to Customers. Software and Systems Modeling **2** (2003) 15–36
10. Lindland, O.I., Sindre, G., Sølvberg, A.: Understanding Quality in Conceptual Modeling. IEEE Software **11** (1994) 42–49
11. Lauesen, S., Vinter, O.: Preventing Requirement Defects: An Experiment in Process Improvement. Requirements Engineering **6** (2001) 37–50
12. Recker, J., Rosemann, M., van der Aalst, W., Mendling, J.: On the Syntax of Reference Model Configuration. Transforming the C-EPC into Lawful EPC Models. In Bussler, C., Haller, A., eds.: Business Process Management Workshops. Volume 3812 of Lecture Notes in Computer Science. Springer, Berlin, Germany et al. (2006) 497–511
13. Andrews, T. et al.: Business Process Execution Language for Web Services. Version 1.1 (2003)
14. zur Muehlen, M.: Workflow-based Process Controlling. Foundation, Design and Application of workflow-driven Process Information Systems. Logos, Berlin, Germany (2004)
15. van der Aalst, W., Weijters, A., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering **16** (2004) 1128–1142
16. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M.: Business Process Intelligence. Computers in Industry **53** (2004) 321–343
17. Keller, G., Nüttgens, M., Scheer, A.W.: Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Technical Report 89, Institut für Wirtschaftsinformatik der Universität Saarbrücken, Saarbrücken, Germany (1992)

18. Mendling, J., Nüttgens, M.: EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). Information Systems and e-Business Management **In Press, also available from** `wi.wu-wien.ac.at/home/mendling` (2006)
19. Mendling, J., Recker, J., Rosemann, M., van der Aalst, W.: Towards the Interchange of Configurable EPCs: An XML-based Approach for Reference Model Configuration. In Desel, J., Frank, U., eds.: Enterprise Modelling and Information Systems Architectures. Volume P-75 of Lecture Notes in Informatics. German Informatics Society, Klagenfurt, Austria (2005) 8–21
20. Mendling, J., Recker, J., Rosemann, M., van der Aalst, W.: Generating Correct EPCs from Configured CEPCs. In: 21st Annual ACM Symposium on Applied Computing, Dijon, France, ACM (2006) forthcoming
21. Ziemann, J., Mendling, J.: EPC-Based Modelling of BPEL Processes: a Pragmatic Transformation Approach. In: 7th International Conference MITIP 2005, Genova, Italy (2005)
22. Holland, C.P., Light, B.: A Critical Success Factors Model for ERP Implementation. IEEE Software **16** (1999) 30–36
23. Rolland, C., Prakash, N.: Bridging The Gap Between Organisational Needs And ERP Functionality. Requirements Engineering **5** (2000) 180–193
24. Puhlmann, F., Schnieders, A., Weiland, J., Weske, M.: Variability Mechanisms for Process Models. PESOA-Report TR 17/2005, DaimlerChrysler Research and Technology and Hasso-Plattner-Institut, Ulm and Potsdam, Germany (2005)
25. Reinhartz-Berger, I., Soffer, P., Sturm, A.: A Domain Engineering Approach to Specifying and Applying Reference Models. In Desel, J., Frank, U., eds.: Enterprise Modelling and Information Systems Architectures. Volume P-75 of Lecture Notes in Informatics. German Informatics Society, Klagenfurt, Austria (2005) 50–63
26. Haugen, Ø., Møller-Pedersen, B., Oldevik, J., Solberg, A.: An MDA-based Framework for Model-driven Product Derivation. In Hamza, M.H., ed.: Software Engineering and Applications, Cambridge, MA, ACTA Press (2004) 709–714
27. Recker, J., Rosemann, M., van der Aalst, W.: On the User Perception of Configurable Reference Process Models - Initial Insights. In: 16th Australasian Conference on Information Systems, Sydney, Australia, Australasian Chapter of the Association for Information Systems (2005)
28. Rosemann, M., Shanks, G.: Extension and Configuration of Reference Models for Enterprise Resource Planning Systems. In Finnie, G., Cecez-Kecmanovic, D., Lo, B., eds.: Proceedings of the 12th Australasian Conference on Information Systems. Southern Cross University, Coffs Harbour, Australia (2001) 537–546