February 2005

Model-Driven Process Configuration of Enterprise Systems

Alexander Dreiling
Queensland University of Technology

Michael Rosemann
Queensland University of Technology

Wil M. P. van der Aalst
Eindhoven University of Technology

Wasim Sadiq
SAP Research (SAP Australia Pty Ltd)

Sana Khan
BP Oil UK Limited

In: Ferstl, Otto K, u.a. (Hg) 2005. *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*; 7. Internationale Tagung Wirtschaftsinformatik 2005. Heidelberg: Physica-Verlag

ISBN: 3-7908-1574-8

Model-Driven Process Configuration of Enterprise Systems[*]

Alexander Dreiling, Michael Rosemann
Queensland University of Technology

Wil M. P. van der Aalst
Eindhoven University of Technology

Wasim Sadiq
SAP Research (SAP Australia Pty Ltd)

Sana Khan
BP Oil UK Limited

Abstract: The configuration of comprehensive enterprise systems to meet the specific requirements of an organisation up to today is consuming significant resources. The results of failing or delayed enterprise system implementation projects are severe and may even threaten the organisation's existence. One of the main drivers for implementing comprehensive enterprise systems is to streamline business processes. However, an intuitive conceptual support for business process configuration is insufficiently addressed by enterprise system vendors and inadequately researched in academia. This paper presents a model-driven approach to target this problem and proposes several configuration patterns that describe generic patterns of configuration alternatives, in order to understand what situations can occur during business process configuration. Based on these configuration patterns, a configuration notation is introduced that allows for visually highlighting configuration alternatives. Finally, we will sketch how configurable Event Driven Process Chains and the configuration of business processes can be supported using relational databases.

Keywords: Configuration, Customising, Configuration Patterns, Business Processes, Enterprise Systems

[*] The research results presented in this paper have been funded by SAP Research and Queensland University of Technology with the project "Modelling Configurable Business Processes". SAP is a trademark of SAP AG, Walldorf, Germany.

1 Introduction – Enterprise Systems and Business Processes

Business as a science, implicitly or explicitly formulates requirements for businesses supporting computer-based information systems. Since the 1920s, when Business as a science was put forward as a separation from Economics [Donh22], there have been several major shifts in business requirements which often resulted in changing requirements for information systems. One of these heavily impacting shifts in business requirements is business process orientation which has become a major topic in academia and for most companies since the 1990's [DaSh90]. Process orientation and thinking originated even earlier with one of the early examples provided by Taylor, when he revolutionised industrial engineering with ideas on work organisation, task decomposition, and job measurement [Tayl11]. Later examples were provided by Nordsieck who argued in 1934 that the structure of a company should be process-oriented [Nord34, p. 77] and compared the structure of a company to a stream, because it is an "uninterrupted value chain" [translated from Nord72]. Based on these ideas, Business Process Reengineering (BPR) became a popular management approach [Dave93, Gait83, HaCh93, Hamm97, Port85]. The main objective was a radical organisation-wide optimisation. Thus, BPR focused on enabling improvements in work processes and outputs [DaBe95]. Whereas BPR improves work processes in a bounded timeframe, Business Process Management (BPM) can be seen as a continuous approach [DaBe95]. Only since the notion of BPR and BPM emerged, has process orientation managed to significantly impact on the Information Systems field.

The term enterprise system came into fashion somewhat recently, but the concept behind it has been subject to academic discussion for a long time now and has evolved from an historic development in Business, Computer Science, and Information Systems. Computer-based systems became available for commercial use some decades after Business as a science had been developed. The idea of corporate wide integrated information systems was then developed [Beer66]. After massive technological and conceptual development, Enterprise Resource Planning (ERP) Systems seemed to have made this vision possible Examples of contributions to the ERP field cover, amongst others, the definition of ERP [KlRG00], configuration of ERP Systems [ArAn03, BrHM01, SoGD03], critical success factors of ERP Systems [HoLi99], modelling within the context of ERP [DKKS04], and possible developments of ERP in the future [MaPA00]. ERP focuses on the technical integration of different parts of the business such as financials, production, human resources, procurement, and distribution. ERP projects may vary in size and structure, each requiring careful management decisions during implementation [MaTv00]. In addition to size and structure of an ERP implementing organisation, its cultural background can dramatically influence an ERP project, as the typical Western understanding of conducting business is not valid in every part of the world [SoKT00].

Since the first discussions of ERP, several developments made a modification of the original ERP idea necessary. Although positive effects of inter-organisational information systems had been discussed before the notion of ERP emerged [JoVi88], the continuously increasing need for integrating not only internal functional departments but also ERP systems of organisations along the value chain hadn't been acknowledged until later. A good example of an inter-organisational business process would be the so-called vendor-managed inventory concept where stock replenishment is outsourced to the vendor of an organisation, and even demand forecasting is done by the vendor [SAP04]. Such scenarios require for effective behavioural integration [LeSH03] as opposed to a purely technical integration as traditional ways of conducting business are changed significantly.

Today, the term enterprise system is a label for what has been previously called an ERP system. The developments within Business, Computer Science, and Information Systems, out of which some have been highlighted above, led to a massive amount of requirements driving the complexity of enterprise systems. Accordingly, the scope and applicability of business areas that are supported by enterprise systems like SAP have been growing significantly over the past few years. Enterprise systems nowadays need to offer a lot of functionality in order to cope with a large amount of business requirements. This functionality needs to be aligned with the business in order to create value for the organisation, confronting the organisation with the options of either configuring the enterprise system, the organisation, or a combination of both. Especially the first option is very important because an organisation may not wish to change their processes, and also requirements may change over time making an adaptation of the enterprise system necessary.

To support expectations that customers place on enterprise systems, these systems need to cater for a large number of diversified requirements. Generally, a customer is interested in deploying a subset of available features to support their specific needs. In order to be able to react to these customer demands it is of paramount importance to understand what generic configuration situations occur during process configuration and to explicitly address configuration in process modelling languages. SAP targets configuration with its Implementation Guide (IMG), a comparatively large tool resulting in projects that consume significant resources. But even if the business process management hype peaked years ago [DaSh90, Dave93, HaCh93], process configuration within SAP is not intuitively model-driven. Apart from SAP's inability to react adequately to some of the implications of the current BPM trend (as one practical example amongst many), academia has also not yet addressed process configuration within enterprise systems sufficiently.

Our paper is a first step towards overcoming this situation. We will first elaborate on the research methodology used in the underlying research. We then discuss configuration as a concept and highlight the configuration patterns that have been developed within our research. Subsequently, Configurable Event Driven Process Chains (CEPCs) will be introduced as an extension to Event Driven Process Chains (EPCs) [Aals99, Sche00] which is based on the configuration patterns. We also elaborate on how the configuration patterns can be realised using relational database technology. Finally, the paper concludes with a short summary and future prospects will be discussed.

2 Research Methodology

The research findings presented in this paper result from a design science approach. Design science in contrast to behavioural science creates a different type of artefacts. Whereas the latter is concerned with explaining and predicting behaviour in human-computer interaction [HMPR04], the former produces artefacts in a more engineering or construction-like approach. Hevner et al. [HMPR04] defined seven guidelines for design science in information systems research. We addressed each of them in the following way [citations obtained from HMPR04]:

1. *Design as an Artefact ("must produce a viable artifact"):* We provide configuration patterns based on a language for highlighting configuration alternatives as an extension to a commonly known process modelling language.
2. *Problem Relevance ("develop technology-based solutions to important and relevant business problems"):* The underlying business problem has been described in the introduction as an insufficient support for process configuration in order to align enterprise systems to the organisational requirements.
3. *Design Evaluation ("utility, quality, and efficacy of a design artifact must be rigorously demonstrated"):* The presented research results have been derived involving a number of researchers and practitioners. However, empirical validation is still outstanding and remains to be delivered during the last part of this research project. We plan to conduct focus groups and surveys in order to validate the results.
4. *Research Contributions ("must provide clear and verifiable contributions in the areas of the design artifact"):* This research is (to our knowledge) the first systematic approach to construct a configurable business process modelling language which is its main contribution.
5. *Research Rigor ("Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact"):* The configuration patterns have been rigorously derived from workflow patterns [AHKB03]. They are supposed to support as many requirements for a configurable reference modelling language as possible [RoAa03]. However, the evaluation of the design artefact is still to be evaluated in future research.
6. *Design as a Search Process ("requires utilizing available means to reach desired ends while satisfying laws in the problem environment"):* The research conducted was validated several times with researchers, practitioners and SAP

as the industry partner of this project. Some additional configuration patterns to the ones presented in this paper were discussed and abandoned again. In that, the research could be described as a *generate-and-test-for-appropriateness* approach.

7. *Communication of Research ("must be presented effectively both to technology-oriented as well as management-oriented audiences"):* This paper presents part of this aim as well as regular presentations and discussions at SAP and within SAP's environment.

3 Configuration of Business Process Models

3.1 What is Configuration?

Configuration of software in order to meet requirements of organisations has been subject to academic discussion for a significant period of time as early examples suggest [GSSD84, LSWG88]. Davenport [DaHC98] describes the process of configuration as a methodology performed to allow a business to balance their IT functionality with the requirements of their business. More specifically, Soffer et al. [SoGD03] describe configuration as an alignment process of adapting the enterprise system to the needs of the enterprise. Especially, if an organisation achieved competitive advantages in enacting a business process in a certain way, they usually will not wish to change this business process in order to fit into an enterprise system. In this case, the reference process within the enterprise system needs to be changed according to the real-world business process. Soffer et al.'s approach [SoGD03] allows for implementing process variants based on the values of certain attributes. Enterprise system configuration involves setting all the usage options available in the package to reflect organisational features [DaHC98]. Brehm et al. [BrHM01] define nine different change options for enterprise systems from predefined alterations (e.g. by marking checkboxes) within the enterprise system to alterations of the program code. Holland and Light [HoLi99] argue that a critical success factor of enterprise system implementation is to avoid program code changes and wherever possible using predefined change options. In terms of model configuration Becker et al.'s approach is one of the most advanced [BDKK02]. It features several mechanisms for transforming a reference model into a build time model. Becker et al.'s approach is very generic and differs from our research in that we, first, seek generic patterns that arise during model configuration and, second, that we propose a configurable modelling language with the CEPC.

Configuration and customisation are often used interchangeably Merriam-Webster's Collegiate Dictionary defines configuration as the "relative arrangement of parts or elements" whereas customising is defined as "to build, fit, or alter according to individual specifications" [Merr03]. With these definitions in mind we can only perform reconfiguration (alteration of relative arrangement of parts or elements within enterprise systems) or customisation (alteration of enterprise systems in order to meet the specification of the enterprise). The latter includes alterations of program code which, we do not pursue in our research. We are rather concerned with the configuration of enterprise systems. For the purpose of this paper, we define (re-)configuration of an enterprise system as the process of aligning business aspects such as functions, information, processes, or organisation with generic enterprise systems in order to meet the business requirements of the enterprise in the most efficient way. For the sake of simplicity we will use the term configuration instead of reconfiguration from here on.

Especially during process configuration, a simple configuration approach that can be described as switching on or off functionality [BaSS98], seems to be inappropriate. In SAP's IMG there are several thousand configuration tables. They define how the system should function, what a transaction screen looks like, how many transaction screens there are, or what kinds of information a process will require [BhRa00]. Some of the configuration decisions within SAP's IMG affect processes within SAP's enterprise system landscape. However, there is no explicit support on how the processes are altered, which is imperative for answering questions such as to how and when should a function be configured, and what configuration time inter-relations a function has with another function. Correspondingly, there is a lack in configurable reference process modelling languages in academia to highlight configuration alternatives and to understand situations in which configuration occurs. We argue that the configuration process needs to be guided with a configurable reference model in order to avoid scenarios where non-configurable models are provided that can be freely altered. Apart from the inability to make configuration decisions explicit, free alterations of process models may also lead to semantically bad process models as described by Kindler [Kind04]. In order to analyse how configuration occurs, configuration patterns were developed. These patterns are discussed in the following section.

3.2 Configuration Patterns

Configuration patterns are defined as patterns which depict a configuration scenario and highlight the potential implementation alternatives that are available. A configuration pattern shows the options that are available at *configuration time*. Configuration time is defined as the moment in time where configuration decisions need to be made. At configuration time, there may be a number of potential build time process alternatives. Configuration patterns capture the configuration time choices and the total subset of build time options.

Configuration patterns were developed using workflow patterns [for more information please refer to AHKB03] by analysing how they could be configured and examining all the possible build time scenarios. This research examined the configuration scenarios that may occur on a process, in particular focusing on configuration of functions, connectors, and control flow. Configuration of organisational and data structures is not part of the scope of this research. Configuration patterns were specified from workflow patterns. The workflow patterns served as a benchmark for completeness and accuracy. The configuration patterns developed served as a basis to derive a configuration notation.

Of the twenty workflow patterns only eight could be used to derive configuration patterns because of two reasons: Firstly, the EPC modelling notation restricted the accurate expression of the workflow patterns. Mainly, the workflow patterns that are concerned with process instances (i.e. workflow cases) cannot be expressed by EPCs because EPC models are at type level (e.g., a workflow pattern expressing a variation of the amount of instances at one point in a process cannot be considered here). Secondly, the subsequent expression of the workflow pattern in EPC models caused an overlap with some of the other existing configuration patterns and therefore, they were not identified as configuration patterns.

3.2.1 Configuration Pattern 1: Optionality (Table 1)

A function in an EPC can be configured during configuration time by switching it *on*, *off* or *optional*. Table 1 illustrates this pattern. It contains the two functions *Function 1* and *Function 2* and three events. Both functions can be switched off in order to remove them from the build time model. In order to establish syntactical integrity of the build time model either the preceding or succeeding event of, for example Function 1, needs to be removed from the model as well or both events need to be substituted by a new event. This decision should be based on the semantic meaning of the events in relation to this function. If a function is deemed optional, the decision about its execution is postponed to run time where it is made on a case-by-case basis. The naming of the events in this case will be based on the semantic meaning and relation of the configurable function. If a function remains optional, an additional function needs to be included in the model that makes the decision to perform it or not at run time. Furthermore, extra connectors and extra events have been included for syntactic correctness. This configuration pattern requires for a configurable function within a configuration notation.

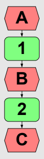
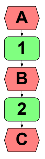
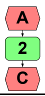
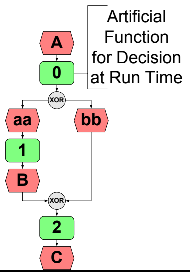
Configuration Pattern Optionality	Build time Configuration Possibilities		
	Combination *(Switched ON)*	Partial *(Switched OFF)*	Conditionally Skipped *(Switched OPT)*
		Syntactically, events A and B can trigger the process. The semantic assumption here is that event A triggers it. 	 Artificial Function for Decision at Run Time A 0 XOR aa bb 1 B XOR 2 C

Table 1: Configuration Pattern of Optionality

3.2.2 Configuration Pattern 2: Parallel Split (Table 2)

The Parallel Split comprises of the AND connector. This pattern captures the configuration alternatives that may exist if an AND split is configurable. This connector signifies a point where a single workflow splits into multiple workflows which must be executed in synchronisation. It is important to note that this configuration alternative implies that a configurable AND can only be configured into an AND. The only choice can be to reduce the amount of incoming or outgoing branches.

3.2.3 Configuration Pattern 3: Exclusive Choice (Table 2)

This pattern depicts a configuration case involving a configurable XOR connector in a split. If an XOR is configurable at configuration time it can support itself, a combination of XOR sequences (if there is more than one branch) and the individual sequences that either branch or merge into the XOR connector.

3.2.4 Configuration Pattern 4: Multi Choice (Table 2)

The Multi Choice configuration pattern captures the configuration alternatives present in a configurable OR split. This pattern potentially supports an OR, AND, XOR, and individual sequences at build time. In summary, these patterns can be set up to: (1) support a separate individual sequence of the branch; (2) allow a function at run time to exclusively choose between branches (XOR connector); (3) execute all branches after the split at run time (AND connector); and (4) allow a function to decide upon the execution of at least one branch after the split (OR connector).

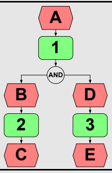
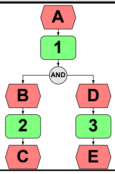
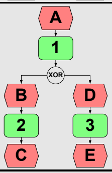
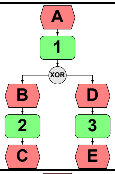
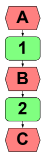
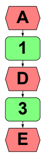
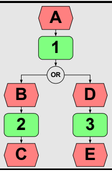
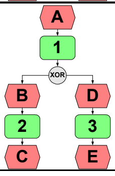
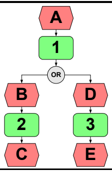
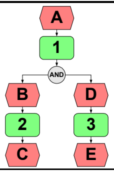
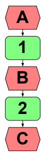
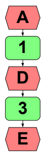
Configuration Pattern	Depiction of Configuration Pattern	Build Time Configuration Possibilities				
					Sequence	
		XOR	OR	AND	*Sequence 1*	*Sequence 2*
Parallel Split		*n.a.*	*n.a.*		*n.a.*	*n.a.*
Exclusive Choice			*n.a.*	*n.a.*		
Multi Choice						

Table 2: Configuration Patterns of Parallel Split, Exclusive Choice and Multi Choice

3.2.5 Configuration Pattern 5: Synchronisation

This pattern is similar to the Parallel Split configuration pattern, except this captures the number of alternatives that may exist in an AND join. Similar to the Parallel Split, it offers only limited configuration alternatives. Configuration pattern 5 is depicted as Configuration pattern 2 with at least two branches being joined instead of one being split.

3.2.6 Configuration Pattern 6: Simple Merge

Similar to the Exclusive Choice pattern, this pattern depicts a configuration case involving a configurable XOR connector in the merger of two or more processes. This configuration pattern has the same number and types of alternatives as present in the Exclusive Choice configuration pattern. Hence, as for configuration patterns 5 its depiction corresponds to configuration pattern 3, with the difference being that paths are joined instead of split.

3.2.7 Configuration Pattern 7: Synchronising Merge

Synchronising Merge configuration pattern captures the configuration alternatives present in a configurable OR merge. This pattern supports an OR, AND, XOR, and individual sequences at build time. Its representation is similar to configuration pattern 4.

3.2.8 Configuration Pattern 8: Interleaved Parallel Routing

It may also be possible to configure the order of execution for a number of functions in a process. This configuration scenario is captured in the Interleaved Parallel Routing configuration pattern. According to this pattern, Function 1 and 2 both have to be executed in an arbitrary order but not at the same time. Hence, at configuration time the decision is left open (denoted by the box around the EPC blocks; these EPC blocks must lead to at least syntactically correct EPCs). Table 3 identifies the configuration alternatives that would exist in this scenario with emphasis on the functions involved in the sequence.

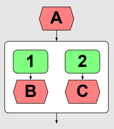
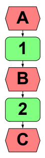
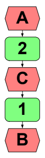
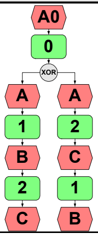
Configuration Pattern Interleaved Parallel Routing	Build Time Configuration Possibilities Assuming semantic definition of Event A is to initiate the process and either Event B or C terminates the process.		
	Sequence 1 fixed at build time	*Sequence 2 fixed at build time*	*build time model for decision made at run time*
			 A0 0 XOR A A 1 2 B C 2 1 C B

Table 3: Configuration Pattern of Interleaved Parallel Routing

3.2.9 Configuration Pattern 9: Sequence Inter-relationships

This pattern is comprised of two sequence workflow patterns. This configuration pattern is founded on the principle that two or more functions which can exist in isolation may be dependent on each other during configuration. This interdependency may enforce an ON, OFF or OPTIONAL status on another function or connector. This inter dependency is described as a relationship. There are many forms in which a relationship may occur: Equivalence or Conditional. Optionality levels [SoGD03] may also be employed to describe inter-relationships. Table 4 illustrates how two different functions 2 and 6 which occur in separate sequences have an underlying interdependency (in this case they are mutually dependent: if one of them is switched-off, then the other one needs to be switched-off as well. They can also be setup as mutually exclusive: switching-off one function means switching on the other and vice versa).

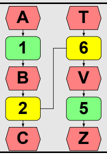
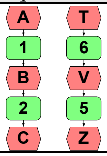
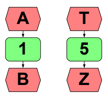
Configuration Pattern Sequence Inter-relationships	Build Time Configuration Possibilities Assuming that semantically Events C and V are output events of Function 2 and 6.	
	Sequences 1	*Sequences 2*
		

Table 4: Configuration Pattern of Sequence Inter-relationships

3.3 Configuration Notation – Configurable Event Driven Process Chains (CEPCs)

To describe configuration alternatives the EPC notation [Sche00] was manipulated with some extensions. In total this research proposes thirteen new notation constructs. These extensions can be classified as *configurable nodes* and as *configuration attributes.* A configurable node is a point where configuration alternatives may exist [RoAa03]. It can be described as a variation point [HaPo03]. Configurable nodes are described in Table 5 (the lines of the notation symbols are thicker than their non-configurable counterparts which becomes obvious if a CEPC contains both configurable and non-configurable functions or connectors).

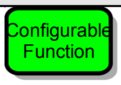
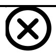
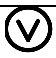
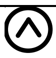
Name	Description	Notation	Configuration Pattern that captures the build time alternatives of a decision
Configurable Function**	The Function can be either turned *on*, *off*, or *optional*.		Pattern 1 *Optionality*
Configurable XOR**	Implications: It can remain the same, or consist of one sequence.		Pattern 3 *Exclusive Choice* and Pattern 6 *Simple Merge*
Configurable OR**	Implications: It can support an: OR, AND, XOR or a sequence.		Pattern 4 *Multiple Choice* and Pattern 7 *Synchronising Merge*
Configurable AND**	Implications: It can only remain the same (AND).		Pattern 2 *Parallel Split* and Pattern 5 *Synchronisation*

Table 5: Configurable Nodes (** Specified in [RoAa03])

A configuration attribute describes the potential set of build time alternatives that may exist at a configurable node. The aim of the configurable attribute is primarily to describe the configurable node. For an overview of the notation used to describe configurable nodes refer to Table 6.

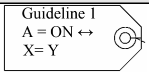
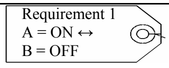
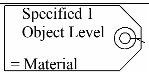
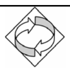
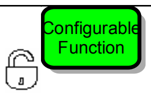
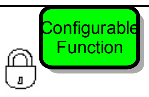
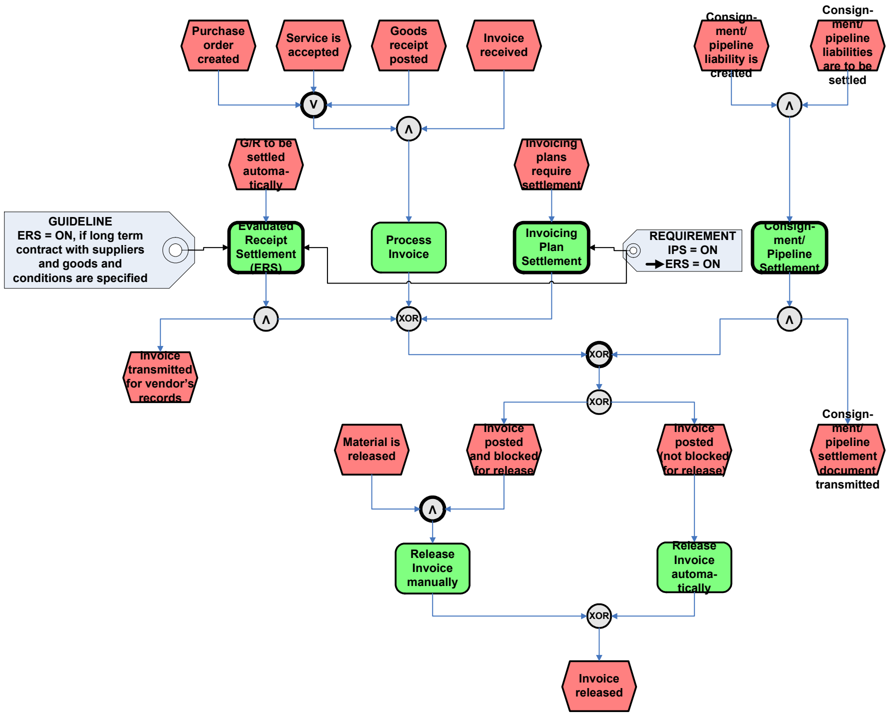
Name	Description	Notation	Configuration Pattern or Configuration Requirement [RoAa03]
Guideline**	Soft Recommendation: guides possible configuration decisions		Pattern 9 *Sequence Inter-relationships*
Requirement**	Hard Recommendation: used to describe a system constraint		
Specification Level	This notation element is used to specify the level at which a configurable node needs to be specified. This can be either at System, Object or Occurrence Level		
Routing Container	Order of configurable functions can be changed arbitrarily at configuration time. However, a decision can be made at runtime to specify which order the functions can be executed in		Pattern 8 *Inter-leaved Parallel Routing*
Non-Critical Configurable Function / Connector	The manner in which the Node is configured is a non-critical decision (by default)		Configuration Requirement: *Critical* and *Non-Critical Decisions*
Critical Configurable Function / Connector	The manner in which the Function is configured is a critical decision. The configuration decision is only hardly reversible		
Default	A configurable connector can have a default like a particular sequence	e.g., *default= ON* or *default= seq 1*	Configuration Requirement: *Mandatory* and *Optional Decisions*
Optional Node	If no explicit configuration decision was made a configurable function is switched on or off by by default. Connectors can have default configuration values as well.	e.g., *opt, default= ON* or *opt, default=seq1*	

Table 6: Configuration Attributes Used to Describe Configurable Nodes (** Specified in [RoAa03])

4 Business Example Invoice Verification

The introduced configurable EPC will now be used to briefly outline a business example. Figure 1 depicts a generic application reference model for invoice verification. In our scenario, we assume that a company acquires this application reference model and configures it to the company's specific needs. The reference model includes three configurable functions, *Evaluated Receipt Settlement (ERS)*, *Invoicing Plan Settlement*, and *Consignment/Pipeline Settlement* and three configurable connectors.



Purchase order created
Service is accepted
Goods receipt posted
Invoice received
Consign-ment/ pipeline liability is created
Consign-ment/ pipeline liabilities are to be settled
V
Λ
Λ
G/R to be settled automa-tically
Invoicing plans require settlement
GUIDELINE
ERS = ON, if long term contract with suppliers and goods and conditions are specified
Evaluated Receipt Settlement (ERS)
Process Invoice
Invoicing Plan Settlement
REQUIREMENT
IPS = ON
→ERS = ON
Consign-ment/ Pipeline Settlement
Λ
XOR
Λ
Invoice transmitted for vendor's records
XOR
XOR
Material is released
Invoice posted and blocked for release
Invoice posted (not blocked for release)
Consign-ment/ pipeline settlement document transmitted
Λ
Release Invoice manually
Release Invoice automa-tically
XOR
Invoice released

Figure 1: Invoice Verification (derived and adapted from SAP's application reference models for SAP R/3 version 4.6c)

Our example company identified *Consignment/Pipeline Settlement* as not necessary since they do not run any consignment warehouses. The guideline for *ERS* recommends keeping ERS in case there are long-term contracts with suppliers and goods and conditions are specified. The organisation identified some supplier-goods combinations where this is the case and keep ERS. Additionally, the organisation identified *Invoicing Plan Settlement* as necessary and keeps it which in itself already would have led to keeping ERS since a requirements for Invoicing Plan Settlement is that ERS must remain in the model if Invoicing Plan Settlement remains in the model. The three configurable connectors are accepted as they are delivered in the reference model. The configured model is depicted in Figure 2.

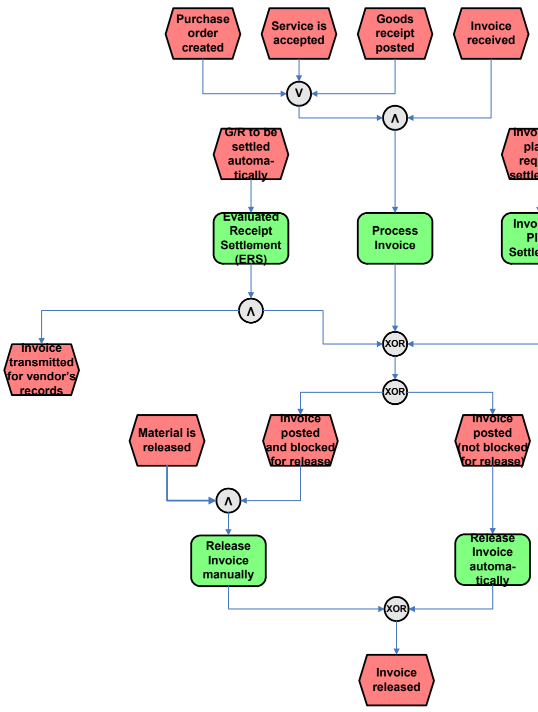

Purchase order created
Service is accepted
Goods receipt posted
Invoice received
V
∧
G/R to be settled automa-tically
Invoicing plans require settlement
Evaluated Receipt Settlement (ERS)
Process Invoice
Invoicing Plan Settlement
∧
XOR
Invoice transmitted for vendor's records
XOR
Material is released
Invoice posted and blocked for release
Invoice posted (not blocked for release)
∧
Release Invoice manually
Release Invoice automa-tically
XOR
Invoice released

Figure 2: Configured Invoice Verification Process

5 Realisation of Reference Model Configuration

Within the project, we used relational database technology to perform the configuration of business process models. In order to do so, first a meta model was created that was able to capture the notation introduced in the previous section. This design process started with a simple meta model for EPCs, and later the requirements for a configurable reference modelling technique (nine requirements from [RoAa03]) led to extensions of the base meta model. We tried to minimise changes to the meta model and added attributes to existing constructs wherever this was possible (for seven requirements, attribute discussion will follow after the meta model introduction). However, adding new meta model constructs was unavoidable (for example the relationship type *Process Object Interrelationship (POI)* in Figure 3). The last remaining requirement (consideration of the impact on the perceived model complexity) is out of scope at this stage of the research and has been considered during the design of the configuration notation. Additional requirements are posed by the configuration notation introduced in the last section. Some of them go beyond the requirements which led to the extensions described so far, and again we tried to minimise the impact on the meta model by extending it mainly in terms of attributes. However, the configuration notation element *Routing Container* required the introduction of a new entity type and a relationship type for assigning process objects to a routing container. The current meta model for CEPCs is shown in Figure 3.

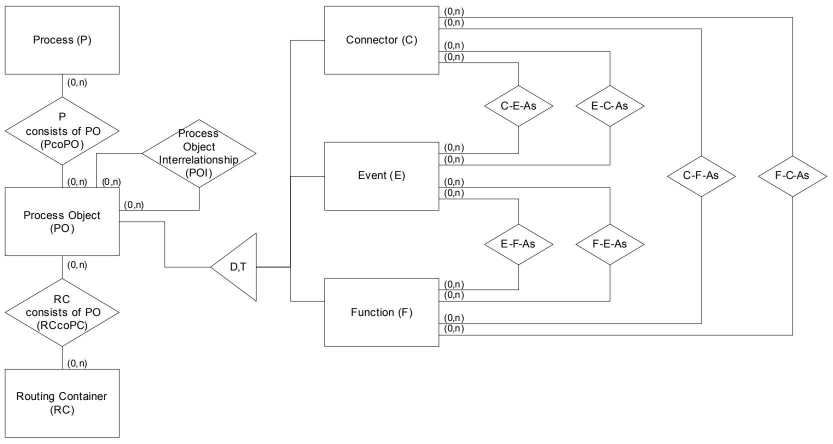


Figure 3: Meta Model of Configurable Event Driven Process Chains

We chose to introduce a relation for each entity type and for each (0,n)-(0,n)-relationship type. By this design choice, the specialisations of Process Object become relations as well. However, we can avoid NULL-Values for attributes that are defined for the specialisations only. Generally every relation resulting from the transformation of the meta model includes the three generic attributes *Primary Key*, *Name*, *Version* and may include the attribute (group) *Foreign Key(s)*. Foreign Keys, apart from expressing relationship types, were used to point from the specialised process object (function, event, and connector) to the process object itself. The version attribute allows for reflecting an element's point in the life cycle (e.g., configured after business analysis, configured after technical analysis).

For seven of the requirements from [RoAa03] we introduced attributes and assigned them to the appropriate relations. Certain attribute values imply others. E.g., if *Configurability* is set as *'no'* then all others discussed in this section necessarily will be *'NULL'*. *Configurability* refers to the question as to whether the element can be switched on or off (for Functions) or transformed into a build time construct (for connectors). By setting *Importance* to *'mandatory'* a user has to make a decision at configuration time whereas *'optional'* does not require a decision at configuration time (in case of no decision the value from the attribute *Default* will be accepted for the build time model). *Scope* lets a reference modeller highlight if, e.g., the configuration decision impacts locally (subsidiary level) or globally (company level). *Criticality* makes a statement on how easy it is to change a certain configuration decision. *Level* allows for making statements about whether the configuration will be done at, e.g., company level or subsidiary level. The configuration of business processes will impact certain parts of the system. *VariationPoint* includes references to these parts, e.g., transaction codes within SAP's IMG. Finally, *Guideline* and *Requirement* are attributes for recommending configuration decisions, in terms of soft and hard recommendations respectively. The configuration notation introduced in the last section additionally requires for introducing the attribute *Type* for expressing that a function can be configurable or not, or that a connector can be either XOR, OR, AND, cXOR, cAND, or cOR.

The first set of relations captures the meta model constructs for *Process*, *Process Object*, *Routing Container* and directly connected relationship types. As for the requirements posted so far, *Process* and *Process Object*, in particular, do not need to be configurable. However, adding the configurability attribute to these relations enables us to state that, e.g., an entire process is not configurable, instead defining this for all of its components. The fact that *Process Object Interrelationship* and *Routing Container* feature the configurability attribute refers to the necessity to express that a reference model which allows for highlighting configuration decisions must be configurable before it is transformed into a build time model during configuration time (e.g., for adopting a reference model to the company needs). We have defined the following relations for this part of the meta model:

- $P = (\underline{pID}, pName, pConfigurability, pVersion)$
- $PO = (\underline{poID}, poName, poType, poConfigurability, poVersion)$
- $POI = (\underline{poiID}, poiName, poiType, poiConfigurability, poiVersion)$
- $RC = (\underline{rcID}, rcName, rcConfigurability, rcVersion)$
- $RCcoPO = (\underline{rccopoID}, rcID, poID, rccopoName, rccopoConfigurability, rccopoVersion)$
- $PcoPO = (\underline{pcopoID}, pID, poID, pcopoName, pcopoConfigurability, pcopoVersion)$

Regarding the requirements that have been discussed above, we have introduced the following relations for Connector, Event, and Function (Event is not configurable):

- $C = (\underline{cID}, poID, cName, cType, cConfigurability, cImportance, cDefault, cScope, cCriticality, cLevel, cVariationPoint, cGuideline, cRequirement, cVersion)$
- $E = (\underline{eID}, poID, eName, eVersion)$

- $F = (\underline{fID}, poID, fName, fType, fConfigurability, fImportance, fDefault, fScope, fCriticality, fLevel, fVariationPoint, fGuideline, fRequirement, fVersion)$

Finally, the relations for connections among events, functions, and connectors are introduced below. Adding the configurability attribute to each of them makes it possible to state, e.g., which event, i.e. the preceding or the succeeding event of a function, will be switched-off in case this function is switched-off by defining that either the incoming or outgoing connector of the function is configurable (can be switched-off). If both incoming and outgoing connectors of a function are configurable, the user that switches-off the function has to make a decision as to which event to switch-off with this function.

- $CEAs = (\underline{ceasID}, cID, eID, ceasConfigurability, ceasVersion)$
- $ECAs = (\underline{ecasID}, eID, cID, ecasConfigurability, ecasVersion)$
- $EFAs = (\underline{efasID}, eID, fID, efasConfigurability, efasVersion)$
- $FEAs = (\underline{feasID}, fID, eID, feasConfigurability, feasVersion)$
- $CFAs = (\underline{cfasID}, cID, fID, cfasConfigurability, cfasVersion)$
- $FCAs = (\underline{fcasID}, fID, cID, fcasConfigurability, fcasVersion)$

6 Summary and Outlook

Enterprise systems' processes need to be configured in order to meet requirements of organisations, but process configuration lacks a sound conceptual foundation that supplies established modelling techniques. We have tried to overcome this problem by introducing configuration patterns which aim at highlighting generic situations that occur during process model configuration. We also introduced a configuration notation based on the configuration patterns and sketched how this notation together with business process configuration can be supported using relational databases.

We consider this work only as the starting point towards mature configuration languages and thus a stepping stone for the next generation of truly configurable process driven ERP systems. Therefore, we envision many extensions and future research building upon it. Firstly, the configuration patterns themselves need to be extended in order to highlight different aspects such as data or organisational units within processes. Secondly, process configuration needs to be integrated into the configuration process of contemporary enterprise systems, since process configuration cannot be separated from structural configuration of the organisation or required data. Thirdly, different user groups such as management, business analysts or technical analysts have different perspectives on business processes. These perspectives need to be addressed and configuration needs to be supported in an integrated way amongst them. We will address some of these issues in our further research. We will also empirically test the proposed notation.

References

[Aals99] Aalst, W. M. P. v. d.: Formalization and Verification of Event-driven Process Chains. Information & Software Technology 41(10), 1999: pp. 639-650.

[AHKB03] Aalst, W. M. P. v. d.; Hofstede, A. H. M. t.; Kiepuszewski, B.; Barros, A. P.: Workflow Patterns. Distributed and Parallel Databases 14(1), 2003: pp. 5-51.

[ArAn03] Arinze, B.; Anandarajan, M.: A Framework for Using OO Mapping Methods to Rapidly Configure ERP Systems. Communications of the ACM 46(2), 2003: pp. 61-65.

[BaSS98] Bancroft, N.; Seip, H.; Sprengel, A.: Implementing SAP R/3: How to introduce a large system into a large organisation. 2nd ed. Manning Inc: Greenwich, 1998.

[BDKK02] Becker, J.; Delfmann, P.; Knackstedt, R.; Kuropka, D.: Konfigurative Referenzmodellierung. In: Becker, J.; Knackstedt, R. (Hrsg.) Wissensmanagement mit Referenzmodellen. Konzepte für die Anwendungssystem- und Organisationsgestaltung. Heidelberg, 2002, pp. 25-144.

[Beer66] Beer, S.: Decision and control: the meaning of operational research and management cybernetics. Wiley: London, 1966.

[BhRa00] Bhattacharjee, S.; Ramesh, R.: Enterprise Computing Environments and Cost Assessment. Communications of the ACM 43(10), 2000: pp. 75-82.

[BrHM01] Brehm, L.; Heinzl, A.; Markus, M. L.: Tailoring ERP Systems: A Spectrum of Choices and their Implications. In: Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS). IEEE: Hawaii, USA, 2001.

[DaBe95] Davenport, T. H.; Beers, M. C.: Managing information about processes. Journal of Management Information Systems 12(1), 1995: pp. 57-80.

[DaHC98] Davenport, T.; Harris, J.; Cantrell, S.: Putting the Enterprise into the Enterprise System. Harvard Business Review 76, 1998: pp. 121-131.

[DaSh90] Davenport, T. H.; Short, J. E.: The New Industrial Engineering: Information Technology and Business Process Redesign. Sloan Management Review 31(4), 1990: pp. 11-27.

[Dave93] Davenport, T. H.: Process Innovation: Reengineering Work Through Information Technology. Harvard Business School Press: Boston, MA, USA, 1993.

[DKKS04] Dalal, N. P.; Kamath, M.; Kolarik, W. J.; Sivaraman, E.: Toward an Integrated Framework for Modeling Enterprise Processes. Communications of the ACM 47(3), 2004: pp. 83-87.

[Donh22] Donham, W. B.: Essential Groundwork for a Broad Executive Theory. Harvard Business Review 1(1), 1922: pp. 1-10.

[Gait83] Gaitanides, M.: Prozessorganisation. Entwicklung, Ansätze und Programme prozessorientierter Organisationsgestaltung. Vahlen: München, 1983.

[GSSD84] Gibson, C. F.; Singer, C. J.; Schnidman, A. A.; Davenport, T. H.: Strategies for making an information system fit your organization. 1984.

[HaCh93] Hammer, M.; Champy, J.: Reengineering the Corporation. A Manifesto for Business Revolution. HarperBusiness: New York, NY, USA, 1993.

[Hamm97] Hammer, M.: Beyond Reengineering: How the Processed-Centered Organization is Changing Our Work and Our Lives. HarperBusiness: New York, NY, USA, 1997.

[HaPo03] Halmans, G.; Pohl, K.: Communicating the Variability of a Software-Product Family to Customers. Software and System Modeling 2(1), 2003: pp. 15-36.

[HMPR04] Hevner, A. R.; March, S. T.; Park, J.; Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28(1), 2004: pp. 75-105.

[HoLi99] Holland, C. P.; Light, B.: A Critical Success Factors Model For ERP Implementation. IEEE Software 16(3), 1999: pp. 30-36.

[JoVi88] Johnston, H. R.; Vitale, M. R.: Creating Competitive Advantage With Interorganizational Information Systems. MIS Quarterly 12(2), 1988: pp. 152-165.

[Kind04] Kindler, E.: On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In: Proceedings of the Business Process Management: Second International Conference (Lecture Notes in Computer Science, Vol. 3080 / 2004). Springer: Potsdam, Germany, 2004, pp. 82-97.

[KlRG00] Klaus, H.; Rosemann, M.; Gable, G. G.: What is ERP? Information Systems Frontiers 2(2), 2000: pp. 141-162.

[LeSH03] Lee, J.; Siau, K.; Hong, S.: Enterprise Integration with ERP and EAI. Communications of the ACM 46(2), 2003: pp. 54-60.

[LSWG88] Lucas Jr., H. C.; Stern, L. N.; Walton, E. J.; Ginzberg, M. J.: Implementing Packaged Software. MIS Quarterly 12(4), 1988: pp. 536-549.

[MaPA00] Markus, M. L.; Petrie, D.; Axline, S.: Bucking the Trends: What the Future May Hold for ERP Packages. Information Systems Frontiers 2(2), 2000: pp. 181-193.

[MaTv00] Markus, M. L.; Tanis, C.; van Fenema, P. C.: Multisite ERP implementations. Communications of the ACM 43(4), 2000: pp. 42-46.

[Merr03] Merriam-Webster: Merriam-Webster's Collegiate Dictionary (online version available under http://www.m-w.com). 11th ed. Merriam-Webster: Springfield, MA, USA, 2003.

[Nord34] Nordsieck, F.: Grundlagen der Organisationslehre. Poeschel: Stuttgart, 1934.

[Nord72] Nordsieck, F.: Betriebsorganisation. Lehre und Technik. Schäffer Verlag: Stuttgart, Germany, 1972.

[Port85] Porter, M. E.: Competitive Advantage. The Free Press: New York, NY, USA, 1985.

[RoAa03] Rosemann, M.; Aalst, W. M. P. v. d.: A Configurable Reference Modelling Language. Technical Report, Queensland University of Technology, Brisbane, Australia, 2003.

[SAP04] SAP AG: Collaborative Business Scenario "Vendor Managed Inventory". http://www.sap.com/businessmaps/459B5C21A96411D3870B0000E820132C.htm, 2004, Download 2004-03-24.

[Sche00] Scheer, A.-W.: ARIS - Business Process Modeling. 3 ed. Springer: Berlin, 2000.

[SoGD03] Soffer, P.; Golany, B.; Dori, D.: ERP modeling: a comprehensive approach. Information Systems 28(6), 2003: pp. 673-690.

[SoKT00] Soh, C.; Kien, S. S.; Tay-Yap, J.: Cultural fits and misfits: is ERP a universal solution? Communications of the ACM 43(4), 2000: pp. 47-51.

[Tayl11] Taylor, F. W.: Principles of Scientific Management. Harper and Row: New York, NY, USA, 1911.