

Model-Based Software Configuration: Patterns and Languages

**Alexander Dreiling^{1,5†}, Michael Rosemann², Wil van der Aalst^{2,3},
Lutz Heuser⁴, Karsten Schulz⁵**

¹European Research Center for Information Systems, University of Münster
Leonardo-Campus 3, 48149 Münster, Germany
alexander.dreiling@ercis.de
+49 251 8338070

²Faculty of Information Technology, Queensland University of Technology
126 Margaret St, Brisbane QLD 4000, Australia
m.rosemann@qut.edu.au

³Department of Technology Management, Eindhoven University of Technology
GPO Box 513, NL-5600 MB Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

⁴SAP Research, SAP AG
Neurottstr. 16, 69190 Walldorf, Germany
lutz.heuser@sap.com

⁵SAP Research, SAP Australia Pty Ltd
Level 12 / 133 Mary Street, Brisbane QLD 4000, Australia
ka.schulz@sap.com

† Corresponding Author

Model-Based Software Configuration: Patterns and Languages

Abstract

The common presupposition of enterprise systems (ES) is that they lead to significant efficiency gains. However, this is only the case for well-implemented ES that are well-aligned with the organisation. The list of ES implementation failures is significant which is partly attributable to the insufficiently addressed fundamental problem of adapting an ES efficiently. As long as it is not intuitively possible to configure an ES, this problem will prevail because organisations have a non-generic character. A solution to this problem consists in re-thinking current practices of ES provision. This paper proposes a new approach based on configurable process models which reflect ES functionalities. We provide in this paper a taxonomy of situations that can occur from a business perspective during process model configuration. This taxonomy is represented via so-called semantic configuration patterns. In the next step we discuss so-called syntactic configuration patterns. This second type of configuration patterns implements the semantic configuration patterns for specific modelling techniques. We chose two popular process modelling languages in order to illustrate our approach.

Keywords

Adaptability, Configuration, Customising, Alignment, Process Model, Requirements Engineering

1 Introduction

Enterprise systems (ES) offer holistic support for intra-organisational business processes and inter-organisational supply chains (Klaus et al., 2000). ES vendors such as SAP or Oracle frequently refer to so-called success stories (SAP AG, 2004; Oracle Corp., 2006) in order to

highlight significant efficiency gains, cost reductions, quality improvements, and the like. Such success stories, however, are contrasted by other examples where ES implementations escalated or were abandoned, consumed tremendous resources and yet sometimes failed to achieve measurable success (Stein, 1998; Davenport, 1998; Key, 1998). Despite comprehensive research on the critical success factors of enterprise systems (Sumner, 1999; Hong & Kim, 2002; Holland & Light, 1999; Umble et al., 2003) and factor models targeting escalation and failure (e.g., Keil, 1995; Keil et al., 2000; Smith et al., 2001; Keil & Robey, 2001; Schmidt et al., 2001), the reasons for ES failures remain difficult to explain (Robey et al., 2002), because ES are complex socio-technical systems (Somers et al., 2000) (this was already acknowledged for Management Information Systems (Bostrom & Heinen, 1977)). ES projects involve a variety of parties and players, and they often require or are used (“technochange” (Markus, 2004)) for dramatic change within the organisation.

However, failure is at least partly attributable to insufficient means of adapting the ES to the organisation leading to an insufficient quality of the actual system configuration (Rosemann et al., 2004). This is enforced by the fact that escalated ES projects are turned around and brought back on track by abandoning ES configuration (Sumner & Hamilton, 2005). The customer then has to subscribe to the world-view of the ES vendor. A major problem is that ES configuration knowledge is typically not embedded in the ES by means of intuitive mechanisms, but tacitly held within certified experts (often consultants). This makes ES implementation success directly depended on these experts (Robey et al., 2002) and drives cost and dependence on external resources.

A proper solution to this problem consists in making the ES configuration process more intuitive. In turn, this will facilitate lowering the cost of configuration during initial ES implementation and during post-implementation adaptation resulting from organisational change. In effect, configuration decisions must be made more intuitive for a larger audience

including business users and management. This is of paramount importance, because managers must know how their systems work, which was acknowledged already in the 1960s (Ackoff, 1967) and became legal responsibility with the Sarbanes-Oxley act (Sarbanes & Oxley, 2002; Ribstein, 2002).

Our contribution to overcoming the entangled configuration process consists in the proposition of generic *configuration patterns* and their explication using two specific process modelling languages. Configuration patterns (Dreiling et al., 2005) are derived from workflow patterns (van der Aalst et al., 2003) and describe on a conceptual level which situations can occur during business process configuration. They refer to configurable parts within a process model, i.e., a part that can be adapted during implementation and after implementation. We will illustrate how these generic configuration patterns can be applied using two specific languages in which the configurable parts are explicitly highlighted. This methodology allows a business user or a manager to identify configuration points in a business process by graphical means and facilitate the decision making process regarding how the process should be implemented in the organisation. The patterns then explicate to the user the impact of this decision on the rest of the process. While we specifically address ES we certainly think that our approach can be generalised in order to serve the configuration of other types of process-aware information systems (as understood by Dumas et al., 2005) as well.

The remainder of this paper is structured as follows. First, we discuss the research objectives and relevant background on aligning business and IT. This will be followed by a differentiation between customising and configuration based on the relevant literature. Subsequently, we will introduce so-called semantic configuration patterns, i.e., patterns of configuration activities that are required from a business perspective. The semantic configuration patterns will then be applied to Event-driven Process Chains and Petri nets in

order to show their general applicability and to discuss the steps that are necessary in order to apply the semantic configuration patterns. We then discuss our contribution, limitations of our approach and implications for academia and practice before we close our paper with a short summary and outlook.

2 Motivation and Background

Our approach targets process configuration especially within the domain of enterprise systems for several reasons:

- First and foremost, ES, as of today, represent a bundle of technical and business expertise. This means that an ES cannot be viewed as a piece of technology that can be simply contextualised within an organisation. An ES is delivered with a myriad of pre-implemented business processes. The delivery of a comprehensive set of pre-packaged content clearly differentiates enterprise systems from other types of information systems. If a specific organisation wishes to support their processes with an ES, the standard ES processes must often be adapted. However, due to the complexity of the configuration and subsequent software maintenance processes, organisations often subscribe to the standard (“vanilla”) set provided by the ES (Davenport, 1998; Loonam & McDonagh, 2004).
- Secondly, ES form the backbone of an organisation. They are considered the norm for holistically supporting all operations in an integrated way (Volkoff et al., 2005) and for integrating an organisation into a supply chain (Lee et al., 2003). Therefore, in essence, an ES is an enormously complex socio-technical system which again contrasts it from other process-aware information systems. While this complexity is required for the support of organisational operations, it is difficult for organisational actors to embrace. The inherent and necessary complexity of ES and their importance

also prevents most organisations to build them from scratch. Therefore, configuring a pre-implemented ES seems to be the most promising way of achieving the desired organisational and managerial support through an information system.

- Thirdly, the notion of workflow management or process-awareness exists for a significant time both in academia and practice (zur Muehlen, 2004). However, within the domain of ES, especially large-scale ES, workflow management and process awareness is often not explicitly supported. Rather, many processes and procedures are coded in user interfaces and application logic with models on top describing the underlying processes. As a result the notion of process-awareness needs to be promoted and the usability of business process models needs to be enhanced.
- Fourth and finally, the comprehensiveness of ES requires significant investments making the ES selection and adaptation process one of the most costly IT investments in many organisations. Consequently, any applied research leading to more efficient and effective software configuration processes has a high potential for amortisation when applied to ES.

We therefore argue that it is especially important to consider ES and languages that are used in the ES context when examining model-based configuration and proposing an improved methodological support. Our results are certainly applicable for other types of process-aware information systems that feature a similar degree of complexity and coupling of business and technical expertise, but we have explicitly investigated into ES for the reasons given above.

The concept of ES is subject to academic discussions for several years now. Examples of recent contributions (typically under the name of Enterprise Resource Planning – ERP) cover among others the definition of ERP (Klaus et al., 2000), critical success factors of ERP Systems (Akkermans & van Helden, 2002; Holland & Light, 1999), modelling within the

context of ERP (Dalal et al., 2004), and possible future developments of ERP (Markus et al., 2000).

Enterprise systems stand at the end of a long development of organisational and managerial support systems, which began in the 1950's (Turban, 1995) building on the concept of cybernetics (Ashby, 1956; Beer, 1966; Wiener, 1948; Wiener, 1967; Beer, 1959). Since then, in essence, the scope of these applications and the intended user group has been continuously extended and the nature of their provision to an organisation has changed. Whereas early management support systems such as Inventory Control Systems were intended to support a small part of organisational activities and actors, today's ES target not only all major processes within various organisations, but also facilitate the execution of entire industry value chains. In addition, the development of a software industry and a sector for enterprise applications of a considerable size within this industry led to the provision of pre-implemented generic solutions that were delivered to organisations. With the advent of large-scale off-the-shelf ES such as SAP's R/3 system in the 1990s, an area of potential conflict between business and IT arose from the individual nature of organisations and the highly generic nature of ES as the most advanced form of off-the-shelf-software (Davenport, 1998). *Configuration is then seen as a structured process which transforms the generic package into a system individualised for the organisation-specific context.*

Significant discussions of *alignment* or *fit* reach back at least to the 1980s, when alignment was perceived as being increasingly important in a SIM Delphi study (rank 5) (Hartog & Herbert, 1986). The IS literature contains, since then, a broad spectrum of contributions on alignment (or misalignment) and fit (or misfit) (Reich & Benbasat, 2000; Brown & Magill, 1994; Segars & Grover, 1998; Chan et al., 1997; Sabherwal & Chan, 2001; Palmer & Markus, 2000; Segars & Grover, 1999; Robey et al., 2002; Henderson & Venkatraman, 1999; Dennis et al., 2001; Silver et al., 1995; Nelson, 1991; Srinivasan, 1985; Majchrzak et al.,

2000). Alignment is seen as either a dependent or an independent construct, i.e. alignment can be achieved by influencing variables or alignment itself is important as an influencing variable for another construct. Contributions of the latter type mainly agree that alignment is bound to efficiency gains or is a factor for implementation success.

Although there is an established body of research on alignment, there remains a considerable lack of research on *actual methods* that help achieving *operational alignment* within the field of Information Systems. Related work mainly resides in the discipline of Computer Science. *Requirements engineering* (*elicitation, analysis, and the like*) or *software engineering* are seen as approaches leading to the desired computer-based information systems. However, alignment research in Computer Science often fails to embrace the variety of approaches in management science and organisation theory. Especially symbolic-interpretive (Weick, 1969) and post-modern developments (Boje et al., 1996; Chia, 1996) in organisation theory, the abandonment of concepts such as rationality (Hirschheim & Newman, 1991; Styhre, 2003), the dissolution of organisational and managerial substance (Boje et al., 1996), and the abandonment of Simon's anthropological assumption of human beings as information-processing systems (Newell & Simon, 1972), requires for re-thinking what the role of requirements engineering is and how it can actually be supported. In fact, requirements engineering in practice, especially for ES selection and implementation, is significantly different from its academic understanding (Rosemann et al., 2005). However, alignment research does often not transcend rationality assumptions or a sociology of order (Burrell & Morgan, 1979) with consensus as a main theme. Alignment, by nature, is a socio-technical task and requirements do not exist per se, but are being socially constructed (Berger & Luckmann, 1966). Research on appropriate methodological support for alignment should therefore not be mainly in the hands of computer scientists but also requires contributions from the IS field (Eijnatten, 1993).

In order to develop an intuitive, graphical mechanism for model configuration as part of a requirements engineering phase, a layer of conceptual models abstracting from technical details and visualising the core processes as supported by the selected system must be added. Such a layer of conceptual models must be mutually dependent on the actual system, i.e., system or models change accordingly if models or system are configured. The field of Business Process Management (Hammer & Champy, 1993; Davenport, 1993; Davenport & Short, 1990; van der Aalst et al., 2000) (especially Workflow Management (Dumas et al., 2005; van der Aalst & van Hee, 2002; Fischer, 2003; Georgakopoulos et al., 1995; Jablonski & Bussler, 1996; Leymann & Roller, 1999; zur Muehlen, 2004)) which implies a separation of process logic—expressed by means of a process language—from application logic led to progress in this respect. It has also, both theoretically and practically, made software more flexible. The remaining gap is constituted by a lack of understanding on how to systematically adapt process models. Consequently, the question arises how the current software configuration process can be made more intuitive and efficient. Our approach discusses a model-based way towards configuring processes within ES based on extensions of popular process modelling techniques, which is significantly different from current practice.

3 Customising and Configuration

Configuration of software has been subject to academic discussion for a significant period of time (Gibson et al., 1984; Lucas Jr. et al., 1988). Davenport (1998) 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. (2003) describe configuration as an alignment process of adapting the enterprise system to the needs of the organisation. Especially, if an organisation achieves competitive advantages in

enacting a business process in a certain way, it usually does 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 (2003) 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 (Davenport, 1998). Brehm et al. (2001) 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 (1999) argue that a critical success factor of enterprise system implementation is to avoid program code changes and wherever possible using predefined change options. Becker et al.’s generic approach (2002) features several mechanisms for transforming a reference model into an individualised process model. Our research differs from the ones outlined here in that we seek generic configuration patterns that arise during process model individualisation in order to better understand and guide the process configuration. In particular, we build upon the notion of configuration patterns as described in (Dreiling et al., 2005), and extend this approach by explicitly separating *semantic* and *syntactic configuration patterns*. This explains what is necessary to implement such configuration patterns. It is primarily necessary for separating business-related configuration decisions from their technical execution. Furthermore, this distinction allows us to support configuration patterns that are defined from a business perspective within several languages.

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” (Merriam-Webster, 2003). With these definitions in mind we can only perform *re-configuration* (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. For the purpose of this paper, we define (re-)configuration of an enterprise system as *the process of aligning business requirements expressed as functions, information, processes, or organisational structures with generic enterprise systems capabilities*. For the sake of simplicity we will use the term *configuration* instead of *re-configuration* from here on.

Especially during process configuration, a simple configuration approach that can be described as switching on or off functionality (Bancroft et al., 1998) seems to be inappropriate. The SAP's implementation guide (IMG), for example, includes 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 requires (Bhattacharjee & Ramesh, 2000). However, there is no support on how explicit processes (i.e., processes that are represented by a process model) can be altered, which is imperative for answering questions such as to how and when should a process-related function be configured, and what interrelations a function has with another functions in a process context.

4 Model-based Software Configuration

It has been argued in the preceding section that there is a pressing need to be able to configure conceptual models as a proxy for system configuration tables. The results from this phase would be a set of configured conceptual models. The phase of an ES implementation project targeted by our approach is called *configuration-time* (Rosemann & van der Aalst, 2005). Configuration-time extends the commonly accepted two-stage model of built time and

run-time with a preceding step. This step is necessary for our approach for three main reasons:

- ES vendors equip their software with configuration mechanisms to adapt the system for a specific customer. If process models are used for configuration, i.e., if process models are adapted for a specific customer, these models consequently must incorporate such configuration mechanisms. Configuration-time process models differ from their build-time counterparts in that they must capture the wholeness of the system's capabilities. During configuration-time the total set of software functionality, as expressed by a superset of models, is reduced to the subset of models relevant for the organisation. The outcomes of this phase are build-time models, which are used as templates for the actual execution of business processes, i.e. run-time processes.
- Existing configuration mechanisms in ES highlight the parts of the ES that can be changed. SAP, for example, has a specific tool called Implementation Guide (IMG) for this purpose. The result of an action performed in the IMG is a built time environment that defines run-time instances. The IMG, so to say, is a configuration-time tool. Transferred to process model configuration, configuration-time is a decoupled step that defines a build-time process model, according to which instances at run-time will be executed. By incorporating configuration mechanisms into process models, vendors are enabled to equip their solutions with a specification tool for build-time models. Build-time models are derived in a methodologically assisted way as opposed to freely altering them. They remain in a predefined solution space, which allows for systematically avoiding configuration errors. ES vendors could furthermore be made liable, grant guarantees, and compliance to standards could be ensured.

- ES vendors face a continually increasing complexity of their solutions as a result of integrating more and more functionality. In the same instance, the market demands quick changes to recent trends such as the support of Customer Relationship Management, Supplier Relationship Management, or compliance with new interoperability standards and legislative requirements. ES vendors have increasing difficulties with reacting to such demands if they do not decompose their applications into manageable pieces. It becomes meaningful from a software engineering perspective to implement configurable cores that can be easily adapted not only by the customer, but also by the vendor in order to create, for instance, new industry-specific solutions.

We distinguish between semantic configuration patterns, which refer to the content of a process model and syntactic configuration patterns, which address syntactic correctness of process models. Both types of configuration patterns are described in this section in detail and by means of examples.

4.1 *Semantic Configuration Patterns*

Semantic configuration patterns allow for a formalised understanding of how configuration occurs from a business perspective and help to identify possible configuration alternatives. Semantic configuration patterns are defined as patterns which depict a configuration scenario and specify the potential implementation alternatives which can be derived from this pattern. Our work is anchored in a subset of the well-known *workflow patterns* (van der Aalst et al., 2003). For a carefully selected set of patterns we examined how they can be made configurable.

4.1.1 Task Related Patterns

We have focused on the active parts of processes, i.e., functionality (functions, tasks, transitions, and the like) and the control flow. We have not examined the configurability of events (or states) as more passive parts of processes since they cannot actively be influenced by the organisation in scope. It is the reaction to events that can be influenced and this reaction is covered by our work because it is typically an activity within the organisation that is being undertaken in order to react to an event.

We must, for the purpose of this paper, limit the discussion to examples. The discussion of the complete set of configuration patterns based on the workflow patterns would exceed the length of this paper and is furthermore not necessary in order to make our argument. We want to show that pattern-based approaches help to understand process configuration and can be used to guide process configuration. We discuss in more detail four semantic patterns that are sufficiently different from each another. This will show the variety of decisions that can be made during process configuration from a business perspective. For two of them, we will then focus syntactic aspects, when it comes to concrete language support. Again, the two chosen ones are sufficiently different from each other in terms of their technical realisation in order to depict our approach.

In light of the configurability of tasks, the first pattern called *optionality* is foundational and the most critical. During configuration-time, a task can be declared *configurable* by allowing switches *on*, *off* or *optional* choices to be made. If a task is switched on during configuration-time, it will remain in the build-time model. Hence, all instances of this process during run-time will include this specific task. If a task is switched off, then it will be permanently removed from the build-time model and with it from all instances at run-time. 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. Table 1 illustrates this pattern. We did not choose a specific

modelling technique in order to highlight the generalisability of our approach. The application of this pattern within specific modelling techniques will be discussed later in this section.

Table 1 approximately here

The configuration decision of any configurable task must be examined in a broader context. Thus, the closely related semantic pattern of *sequence inter-relationships* is founded on the principle that two or more functions may be dependable on each other during configuration. The resulting dependency is called a relationship. This relationship can be either *mutually exclusive* or *mutually dependent*. In case of a mutually dependent relationship, all tasks of the relationship must be selected in the same consistent way during configuration-time. For instance, two mutually dependent tasks A and B must either both be switched on or off. Mutually exclusive tasks must be configured opposite to each other. In the case of two tasks A and B, A must be switched on if B is off, and vice versa. As in the first two patterns, the decision might be deferred to run-time. In this case the build-time model must leave the decision as to whether switching a task on or off open. Mutual dependency and exclusiveness between tasks must then be ensured at run-time. Table 2 is an illustration of the semantic configuration pattern of sequence inter-relationship.

Table 2 approximately here

In some cases, the order of execution for a number of tasks within a process can be configured. This leads to the semantic configuration pattern of *interleaved parallel routing*. If an arbitrary sequential order of execution of a number of functions is allowed during build-time, i.e., if no semantic interrelationship exists between the tasks that is influenced by the order in which they are executed, the decision about the order of execution of these tasks is left open at configuration-time. In certain cases it might be necessary to defer the decision

about this order even to run-time. The set of functions must then remain within the build-time model. The semantic configuration pattern of interleaved parallel routing is depicted in Table 3.

Table 3 approximately here

4.1.2 Control-flow Related Patterns

The semantic configuration pattern of *parallel split* is the first of a series of control-flow-related patterns. It is focused on capturing configuration alternatives at instances where the AND connector is configurable. The only important implication in this case is the number of outgoing branches of the configured AND connector. This means that the connector itself must remain an AND connector with at least two outgoing branches, but further outgoing branches from this connector can be removed. Conversely, the pattern *synchronisation* handles the merging of branches from the configurable AND connector. As with the corresponding split, the connector itself cannot be changed but incoming arcs can be removed. The semantic configuration pattern of *exclusive choice* explains the instances where an XOR can be modified during configuration-time. This pattern caters for the possible logic that can be derived from a configurable XOR connector. Its corresponding pattern of *simple merge* focuses on the merging that happens on a configurable XOR connector. The final pattern that handles a connector is *multi choice*. This pattern captures the configuration alternatives found in a configurable OR split. As a result, this pattern can potentially support an OR, AND, XOR, and individual sequences during build-time. Table 4 illustrates the three splitting patterns, showing configurable alternatives and their build-time sequences. The corresponding join connectors lead to the same population of Table 4 highlighting the reversing joins for the splits.

Table 4 approximately here

4.2 *Syntactic Aspects of Configuration*

The semantic configuration patterns defined above are not language-specific. In order to actually apply the semantic patterns during configuration, they need to be embedded in a specific modelling technique. As examples, we discuss the application of semantic patterns to two popular process modelling languages, Event-Driven Process Chains (EPCs) and Petri nets. We will elaborate why in the case of EPCs it is meaningful to define syntactic patterns for explaining the configuration alternatives of business processes expressed in this language. In the case of Petri nets we apply a different approach which uses two steps: one step to obtain the desired logic and another step to clean up the model (using reduction rules). The examples show that the steps involved in transforming a lawful configurable business process model into another lawful configured one are significantly different for alternative languages. Moreover, some semantic patterns are not necessarily expressible within certain languages. Therefore, our approach can also potentially be used to evaluate the suitability of a process modelling language to be enhanced to a configurable process modelling language.

4.2.1 *Syntactic Patterns for Lawful Event-Driven Process Chain Configuration*

Event-Driven Process Chains (EPCs) (van der Aalst, 1999; Scheer, 2000) became popular with the success of the modelling solution ARIS and the use of EPCs for the documentation of SAP-enabled business processes. EPCs are perceived as being more useful from a business perspective than from a technical perspective because they have been designed with an intuitive notation in mind, and do to cater, in contrast to Petri nets, for executability. The missing formal semantics of EPCs have been identified as a problem and several attempts have been made to establish formal semantics for EPCs in retrospect (e.g., Kindler, 2004). Nevertheless, due to the informality of an EPC's semantics, the process models are kept very simple and can be understood relatively easy.

Within EPCs, *events* and *functions* occur strictly alternately. The alternation of functions and events may be extended using *connectors* (*splits* and *joins*). In-between an event and function there may be connectors specifying the routing logic. Splits (AND, XOR, OR) may require for a decision as to which branches are being executed. This decision is made by the function before the split and thus, no event is allowed to be preceded by an OR or an XOR split.

The semantic pattern of *optionality* requires for switching tasks on, off, or optional. Resulting from the decision made in an optionality scenario, changes to other parts of an EPC may be necessary, which results from the semantic pattern *sequence inter-relationships*. Both of these semantic patterns are addressed in the following discussion.

Tasks in EPCs are resembled by functions. In order to enable correct atomic configuration steps that consist of a semantic decision and a subsequent syntactical clean-up, event(s) must be removed from an EPC along with functions because otherwise events would follow each other, leading to syntactically incorrect EPCs. Table 5 presents a configurable function (one that can be switched in accordance with the semantic pattern of optionality) in all of its lawful environments. It must be both preceded and succeeded by an event, a split or a join leading to nine different syntactic configuration patterns for the semantic pattern of optionality within EPCs.

Table 5 approximately here

We use the example of the syntactic pattern EFE (a configurable function (F) is preceded and succeeded by an event (E)) in order to show the next necessary step for a deeper understanding of EPC configuration. In this example (depicted in Table 6), there are several configuration options. The configuration decision of leaving the function within the process leads to transforming the configurable function into a non-configurable function and both the preceding and succeeding events remain within the process.

Table 6 approximately here

The second group of decisions is concerned with switching off the configurable function (function A). A connection of two successive events is syntactically incorrect. Therefore, one configuration option removes the preceding event (E_P) along with the function, one configuration option the succeeding event (E_S), and the third one removes both events (E_S and E_P) and replaces them with a new one (X). This last option, of course, is not a purely syntactic decision, since the new event (X) is semantically different from the previously existing ones. However, a user configuring a process model must have this option in case the process after configuration would not make sense semantically with either one of the two original events.

The third set of configuration alternatives defers the decision as to whether switching a function is switched on or off to run-time where it is made on a case by case basis. This requires for a split of the process branch before the function (A) and a join after it with the second branch excluding the function. The syntactic restrictions of EPCs lead to four different configuration options with variations depending on the direct environment (i.e., one step behind and ahead in the process) of the configurable function. Variant 1 splits before the preceding event (E_P) of the configurable function and rejoins immediately after it (A). The split in this variant, as in the following ones, must be of type XOR, because it must show at build-time that there are two distinct possibilities at run-time with either executing the function or not. Variant 2 also splits before the preceding event (E_P) of the configurable function, but it rejoins after the succeeding event (E_S) of the same function. This variant requires a new (artificial) event (E_X) because otherwise the alternative process branch of the configurable function would include two functions following each other which is syntactically incorrect. Variants 3 and 4 are problematic because they split before the configurable function (A), hence after its preceding event (E_P). Since the split must be an

XOR, it must not succeed an event. This syntactic rule denotes that the decision as to which process branch will be performed must be made by a function and cannot be made by an event. However, we included these variants for completeness purposes and also because this syntactic rule is more of an informal nature than of a formal one. Variant 3 rejoins after the succeeding event (E_S) of the configurable function and variant four immediately after it. Variant 4 requires for an artificial (empty) function (A') as an alternative to the configurable one (A), because without this artificial function the alternative branch to the one including the configurable function would directly combine two events with each other which would be syntactically incorrect.

The configuration alternatives for the remaining EPC syntactic configuration patterns from Table 5 are constructed in a similar way. We examined the lawful environments of the configurable function and constructed configuration alternatives for all combinations of predecessors and successors. In general, the configuration alternatives look more complicated because of the splits and joins but are similar in principle. They switch-off or switch-optional preceding or succeeding sets of events. However, as already mentioned we cannot discuss them in detail here.

Some syntactical patterns require for a second syntactical configuration step in order to perform meaningful configuration. This necessity arises from the possible wider environments in which some of the syntactic patterns are embedded. Our configuration approach explicitly does not remove a process branch if all of its elements are removed. This leads to shortcuts from a split to a join. We argue that—for syntactic configuration—it is illegitimate to remove such a shortcut because this would constitute a semantic change. Hence, the shortcut remains and typically leads therefore to no problems with removing connectors if the content of a branch has been removed. Despite from this, Table 7 depicts two scenarios, where the examination of the wider environment of two syntactic patterns

becomes necessary. The patterns of EFJ and SFE can be in an environment where the configurable function is in one of two branches and the second branch already is a shortcut between the connectors that frame the configurable function. In this case, we must look one additional step ahead or back in order to be able to detect such a situation. It is obvious, that in such a case we need to remove the branch and with it the framing connectors because otherwise there would be two shortcuts between the same set of connectors which would be meaningless.

Table 7 approximately here

The connector-related semantic patterns need to be handled differently from the function-related ones in terms of their syntactically correct application within a modelling language. As an example, we use well-structured EPCs (every split has a corresponding join) and discuss the case of a configurable OR split that divides the process into n branches and rejoins them later via a configurable OR join. The configuration options are a logical subset of the different behaviours an OR can express, hence, the OR option itself, the XOR, and the AND. The OR and XOR options would defer the decision as to which branch to execute at run-time. If at build-time a decision for one of the branches can be made, the configuration option would be to decide for a sequence of this branch in the context of its preceding and succeeding process segments. Table 8 depicts these configuration options. It is furthermore necessary to consider the predecessor of the configurable OR join. As already mentioned, one of the informal rules of EPCs is that an XOR and OR split must not succeed an event. Hence, if a configurable OR is placed after an event the OR and XOR configuration options must be enhanced by adding an artificial decision function before the split and artificial reacting events for the decision function directly after it.

Table 8 approximately here

4.2.2 Graph Reduction for Lawful Petri net Configuration

The classical Petri net model was one of the first models adequately describing concurrency (Desel & Esparza, 1995; Murata, 1989; Reisig & Rozenberg, 1998). It is a simple, but also rather expressive, language consisting of just two objects: *places* and *transitions*. Places can be connected to transitions and transitions can be connected to places. Together they form a directed graph. One can think of places as being *passive* and transitions as being *active*. The *state* of a Petri net, also named *marking* is determined by the distribution of *tokens* over places. At any point in time a place contains a given number of tokens. Transitions *consume* tokens from their input places and *produce* tokens for their output places. To be more precise, we need to introduce two concepts: *enabling* and *firing*. A transition is enabled if each of its input places contains at least one token. Enabled transitions may fire. Firing a transition implies the removal of one token from each input place and the addition of one token to each output place. This way the state of the Petri net can change while its network structure is fixed.

Petri nets are graphical and have formal semantics. This allows for intuitive models and a wide variety of analysis techniques. Although few modelling tools and information systems directly apply Petri nets, the fundamental ideas in Petri nets have been adopted by many languages and commercial systems such as COSA, Protos, Income, and Baan-DEM. Note that for example the EPCs discussed in the previous section have been inspired by Petri nets.

In order to apply the semantic configuration patterns introduced above to Petri nets, we must discuss the impact of the configuration choices proposed by the semantic patterns for the syntactic validity of Petri nets. We will do this in terms of syntactic patterns similarly to the EPC discussion and furthermore use *reduction rules* to transform configured Petri nets into smaller ones. The reduction rules simplify the resulting Petri net without changing the tangible behaviour.

Let us first consider the pattern of optionality. Tasks in a Petri net are represented by transitions. Transitions are represented by squares. To indicate that they are optional at configuration-time, they are shown using thick lines. In Table 9, the transition on the left is optional. If the transition is configured "on", then the transition is replaced by a normal transition. If the transition is configured "off", then the transition is replaced by a so-called silent transition. Silent transitions are transitions that do not represent tasks but that are merely added for the routing of tokens. To indicate that a transition is silent, it is labelled with τ . If the decision is postponed until run-time, there is a choice between a transition representing the task (i.e., A) and a silent transition (i.e., τ). Note that a place with multiple output arcs represents a choice, i.e., if both input places contain one token then both A and τ are enabled. However, if one of them fires, the other one is disabled. This way it is enforced that exactly one of the two is executed.

Table 9 approximately here

Table 9 shows that it is easy to represent the first configuration pattern in terms of Petri nets. However, this may lead to the addition of many silent transitions making the model unreadable. Fortunately, we can apply so-called *reduction rules* to get rid of superfluous silent transitions. We propose the τ -reduction rules shown in Table 10. These rules are inspired by (Verbeek et al., 2004; Desel & Esparza, 1995; Murata, 1989), however, unlike the classical rules they differentiate between silent and non-silent transitions. The first rule shows that a transition connecting two places may be removed by merging the two places, provided that tokens in the first place can only move to the second place. The rules are self-explanatory. However, when applying the rules one should clearly differentiate between silent and non-silent transitions. For example, in the fourth rule at least one of the transitions should be silent; otherwise the rule should not be applied (as indicated). The τ -reduction rules shown in Table 10 do not preserve the moment of choice and therefore assume trace

semantics rather than branching/weak bisimulation. In this paper, we do elaborate on this and simply refer to (van der Aalst & Basten, 2002) for a more detailed discussion of the various notions of equivalence in the context of (workflow) processes. For this paper, trace semantics with silent tasks suffices. However, for more advance applications involving automated support, a more refined notion is needed.

Table 10 approximately here

Table 11 shows the syntactical Petri-net-based pattern for Parallel Split and Synchronisation. Again we use thick lines to indicate the configurable parts. Note that the patterns are similar to EPCs. The main difference is that there are no explicit AND-split and AND-join connectors. In a Petri net, a single transition can act as an AND-split and AND-join.

Table 11 approximately here

Table 12 shows the syntactical Petri-net-based pattern for Exclusive Choice and Simple Merge. Again we use thick lines to indicate the configurable parts. Note that a single place can act as an XOR-split and XOR-join. If the configuration decision is "XOR", the whole net is copied. Again we allow for selecting a subset of all possibilities. In the extreme case (sequence), only one possibility remains. The latter case is presented by the right most Petri net in Table 12.

Table 12 approximately here

Compared to EPCs, AND/XOR-splits and joins are very simple; they directly correspond to transitions and places and there is no need to introduce additional connectors. Unfortunately, it is more difficult to model OR-splits (Multi Choice) and OR-joins (Synchronising Merge) as shown in Table 13.

Table 13 approximately here

To explain the Multi Choice and Synchronising Merge patterns in terms of Petri nets, we assume (for simplicity) that there are two tasks (or sub-processes) A and B. There are three possibilities: (1) just A is executed, (2) just B is executed or (3) both A and B are executed. In an EPC it suffices to simply use OR connectors. This may lead to semantic problems as mentioned before. Therefore, it needs to be specified in more detail in a Petri net as shown in Table 13. There are three split (s_A, s_B, s_{AB}) and three join (j_A, j_B, j_{AB}) transitions. These transitions need to be connected through additional places to ensure that e.g. s_{AB} is not followed by j_A and j_B instead of j_{AB} . If one understands the left-most un-configured Petri net, it is easy to understand the configured Petri nets on the right.

5 Discussion

The preceding section has introduced semantic and syntactic configuration patterns that in conjunction allow for transforming a semantically meaningful and syntactically correct process model base into another model base that is semantically meaningful and syntactically correct. Semantic configuration patterns are language-independent and refer to the semantic consequences of a model decision on model base. Syntactic configuration patterns, in a second step, re-establish syntactical correctness that may have arisen while a semantic configuration pattern was applied. This section discusses our contribution, its limitations and its implications for research and practice.

5.1 Contributions

We have argued in the beginning of this paper that a layer of conceptual models bound to enterprise system functionality is necessary in order to facilitate a more intuitive ES configuration process. We have also argued why an extension of the two-stage model of build-time and run-time is necessary in order to specify build-time models which remain in a predefined solution space.

The combination of semantic and syntactic configuration patterns transfers and extends the idea of ACID transactions (atomicity, consistency, integrity, durability) known from database management to model configuration. In the first step, a configuration decision is examined towards its semantic impact on the remaining model base. All semantic consequences are taken into account and are handled appropriately in the way exemplarily described in the previous section. In a second step a syntactic clean-up re-establishes syntactic correctness of the model base.

The introduced configuration patterns do not only contribute to the body of knowledge on configuration in that they highlight which situation can occur during configuration, but also add description to the business process. Semantic configuration patterns establish a link between different parts of a process model that are not evidentially connected by control or data flow or other means. In that they provide an additional layer of description of a business process that did not exist before.

Furthermore, we argued that from a vendor perspective, the implementation of configurable cores is meaningful as a tool for handling increased solution complexity. Many aspects of an ES can be implemented by the vendor once and then applied within a certain situation. A certain business process can, for instance, look different in two different countries due to legal requirements. In such a case it can be meaningful to implement a generic business process and add configuration rules for the two different countries as opposed to providing different hard-coded solutions for the two countries. If a third country is added, the configuration rules for this country must be defined as opposed to implementing yet another solution. The same applies for industries and corresponding industry-specific solutions and other examples where generic solutions can be configured.

5.2 *Limitations*

Several decisions limit our contribution and need to be discussed. First and foremost, we chose to examine configuration of tasks (or functions or actions) as active parts of business processes as opposed to events. We implied that a process can be conceived of as a complex event-reaction pattern and that it is mainly the actions that an organisation can influence. In this conception an event is something that just happens outside of or within the organisation and the organisation must decide on how to react to it. However, this is not necessarily the only way of addressing process model configuration. It is also possible to examine configurability of events. An organisation may as well define a certain amount of escalation guidelines and define in a second step to which events these escalation principles constitute reactions to. We still think that the majority of configuration decisions within ES rather refer to the active parts of the process. Moreover, some process modelling techniques do not even consider the notion of events but just focus on the task flow of a process. In this respect we believe that we provided a more generically applicable approach. However, we acknowledge that this will only solve the problem of process model configuration to a certain extent. We will focus on event configuration in our future work.

Another limitation arises from the fact that we use the paradigm of fixed build-time models. Some authors believe that this already limits organisational reality to an illegitimate extent because the process changes mainly occur during run-time (e.g., Bernstein, 2000). While we readily acknowledge this limitation we nevertheless point to the areas of applicability. Many processes within ES are absolutely fixed over quite a significant period as a reaction to legal requirements, organisational values or the like. Nevertheless, they must be implemented in the first place and this is where our suggestions contribute. Processes that are subject to frequent change may escape our contribution.

5.3 *Implications*

Several implications for research stem from our discussion. Alignment between business and IT has been a research topic for within Information Systems and Computer Science for many years (examples include Reich & Benbasat, 2000; Brown & Magill, 1994; Segars & Grover, 1998; Chan et al., 1997; Sabherwal & Chan, 2001; Palmer & Markus, 2000; Segars & Grover, 1999; Robey et al., 2002; Henderson & Venkatraman, 1999; Dennis et al., 2001; Silver et al., 1995; Nelson, 1991; Srinivasan, 1985; Majchrzak et al., 2000). However, research in the area of conceptual modelling as a means for bridging the gap between business and IT has widely ignored the context of large off-the-shelf-solutions such as enterprise systems. The methodology proposed in this paper provides just one example for the additional or at least different requirements of such software solutions. In the chosen context, enterprise systems, configuration is a common activity within many organisations. However, many comprehensive ES packages only insufficiently allow lifting this task to a level, where non-technical staff of an organisation can be involved as well. We proposed a method that caters for configuration at a model-level. ES vendors are continuously setting the stage for model-driven process configuration. Academia should respond and put more emphasis on investigating issues with conceptual models in turn. It has been argued that relevance is vital within IS research (Applegate, 1999; Davenport & Markus, 1999; Benbasat & Zmud, 1999; Lyytinen, 1999), and in the case of model-driven configuration ES vendors continually provide relevance (e.g., SAP's (2004) Enterprise Service Architecture strategy).

This paper hopefully stimulates the research community in the domain of conceptual modelling to further explore the application and adaptation of existing modelling techniques and methods in the context of enterprise systems. Several research projects build on our work. Event-based process configuration as opposed to function-based process configuration,

technical support for the conceptual ideas, or empirical studies as to how usable the concepts outlined here are in large real-world settings are just a few examples.

In terms of implications for practice, the proposed methodology has potential for significant “real world” impact as it provides a reasonably pragmatic approach towards a model-based configuration process. The increased interest of large enterprise systems vendors such as SAP and Oracle to convert their solutions into more process-aware information systems including a decoupling of processes from the application layer provides fertile ground for an increased demand for such model-based configuration methodologies. Furthermore, the idea of configurable models can also be applied to a wide range of available reference models such as eTOM, ITIL or SCOR.

6 Conclusions & Outlook

Enterprise systems provide a wide range of pre-defined solutions for typical intra- and inter-organisational business processes. Appropriate process variants have to be derived from this set of processes during the system configuration process. As such, the adoption of off-the-shelf solutions can be seen as a materialisation of the alignment process, in which IT solutions of strategic importance are configured so that they correspond to the business. The current practice of this alignment process largely relies on expensive external resources.

It has been proposed in this paper to utilise extended business process models for a more intuitive and structured configuration process. Various configuration patterns have been presented and exemplary insights into the challenge of deriving syntactically correct models have been given. By contrasting those patterns in two popular modelling techniques, i.e. Event-driven Process Chains and Petri nets, it became clear how different the requirements for supporting these patterns can be.

Our future research has three main directions. First, we will work on further developing and empirically testing notations for visualising configuration patterns. Second, we will extend the focus on a configurable control flow by studying configurability of related data and organisational entities. Third, we will explore how configurable models can be derived by mining enterprise systems.

7 Acknowledgements

We would like to thank the two anonymous reviewers and associate editor for providing extensive and valuable feedback for amending the paper.

8 References

- AALST WMPVD (1999) Formalization and verification of event-driven process chains. *Information & Software Technology* 41(10), 639-650.
- AALST WMPVD and BASTEN T (2002) Inheritance of workflows: An approach to tackling problems related to change. *Theoretical Computer Science* 270(1-2), 125-203.
- AALST WMPVD, DESEL J and OBERWEIS A (Eds.) (2000) *Business process management: Models, techniques, and empirical studies*. Springer, Heidelberg, Germany et al.
- AALST WMPVD and HEE KMV (2002) *Workflow management: Models, methods, and systems*. MIT Press, Cambridge, MA.
- AALST WMPVD, HOFSTEDE AHMT, KIEPUSZEWSKI B and BARROS A (2003) Workflow patterns. *Distributed and Parallel Databases* 14(3), 5-51.
- ACKOFF RL (1967) Management misinformation systems. *Management Science* 14(4), B-147-B-156.
- AKKERMANS H and VAN HELDEN K (2002) Vicious and virtuous cycles in ERP implementation: A case study of interrelations between critical success factors. *European Journal of Information Systems* 11(1), 35-46.
- APPLEGATE LM (1999) Rigor and relevance in MIS research-introduction. *MIS Quarterly* 23(1), 1-2.

- ASHBY WR (1956) *An introduction to cybernetics*. Chapman and Hall, London, UK.
- BANCROFT N, SEIP H and SPRENGEL A (1998) *Implementing SAP R/3: How to introduce a large system into a large organisation*. Manning Inc, Greenwich.
- BECKER J, DELFMANN P, KNACKSTEDT R and KUROPKA D (2002) Konfigurative Referenzmodellierung. In *Wissensmanagement mit Referenzmodellen. Konzepte für die Anwendungssystem- und Organisationsgestaltung* (BECKER J and KNACKSTEDT R, Eds), pp 25-144, Heidelberg.
- BEER S (1959) *Cybernetics and management*. Wiley, New York, NY, USA.
- BEER S (1966) *Decision and control*. John Wiley & Sons, London, UK et al.
- BENBASAT I and ZMUD RW (1999) Empirical research in information systems: The practice of relevance. *MIS Quarterly* 23(1), 3-16.
- BERGER PL and LUCKMANN T (1966) *The social construction of reality: A treatise in the sociology of knowledge*. Doubleday, Garden City, NY, USA.
- BERNSTEIN A (2000) How can cooperative work tools support dynamic group processes? Bridging the specificity frontier. In *ACM 2000 Conference on Computer Supported Cooperative Work*, pp 279-288, ACM Press, Philadelphia, PA, USA.
- BHATTACHARJEE S and RAMESH R (2000) Enterprise computing environments and cost assessment. *Communications of the ACM* 43(10), 75-82.
- BOJE DM, GEPHART JR. R and THATCHENKERY TJ (Eds.) (1996) *Postmodern management and organization theory*. SAGE Publications, Thousand Oaks, CA, USA et al.
- BOSTROM RP and HEINEN JS (1977) MIS problems and failures: A socio-technical perspective. Part 1: The causes. *MIS Quarterly* 1(3), 17-32.
- BREHM L, HEINZL A and MARKUS ML (2001) Tailoring ERP systems: A spectrum of choices and their implications. In *34th Hawaii International Conference on System Sciences (HICSS)*, IEEE, Hawaii, USA.
- BROWN CV and MAGILL SL (1994) Alignment of the IS functions with the enterprise - toward a model of antecedents. *MIS Quarterly* 18(4), 371-403.
- BURRELL G and MORGAN G (1979) *Sociological paradigms and organisational analysis: Elements of the sociology of corporate life*. Heinemann, London, UK.

- CHAN YE, HUFF SL, BARCLAY DW and COPELAND DG (1997) Business strategic orientation, information systems strategic orientation, and strategic alignment. *Information Systems Research* 8(2), 125-150.
- CHIA RKG (1996) *Organizational analysis as deconstructive practice*. Walter de Gruyter, Berlin, Germany et al.
- DALAL NP, KAMATH M, KOLARIK WJ and SIVARAMAN E (2004) Toward an integrated framework for modeling enterprise processes. *Communications of the ACM* 47(3), 83-87.
- DAVENPORT TH (1993) *Process innovation: Reengineering work through information technology*. Harvard Business School Press, Boston, MA, USA.
- DAVENPORT TH (1998) Putting the enterprise into the enterprise system. *Harvard Business Review* 76(4), 121-131.
- DAVENPORT TH and MARKUS ML (1999) Rigor vs. Relevance revisited: Response to Benbasat and Zmud. *MIS Quarterly* 23(1), 19-23.
- DAVENPORT TH and SHORT JE (1990) The new industrial engineering: Information technology and business process redesign. *Sloan Management Review* 31(4), 11-27.
- DENNIS AR, WIXOM BH and VANDENBERG RJ (2001) Understanding fit and appropriation effects in group support systems via meta-analysis. *MISQ Quarterly* 25(2), 167-193.
- DESEL J and ESPARZA J (1995) *Free choice petri nets*. Cambridge University Press, Cambridge, UK.
- DREILING A, ROSEMAN M, AALST WMPVD, SADIQ W and KHAN S (2005) Model-driven process configuration of enterprise systems. In *Wirtschaftsinformatik 2005. eEconomy, eGovernment, eSociety* (FERSTL OK, SINZ EJ, ECKERT S and ISSELHORST T, Eds), pp 691-710, Physica-Verlag, Heidelberg, Germany.
- DUMAS M, AALST WMPVD and HOFSTED E AHMT (2005) *Process-aware information systems*. Wiley & Sons.
- EIJNATTEN FMV (1993) *The paradigm that changed the work place*. Van Gorcum, Assen, The Netherlands.
- FISCHER L (Ed.) (2003) *Workflow handbook 2003*. Workflow Management Coalition, Lighthouse Point, FL.

- GEORGAKOPOULOS D, HORNICK, M. and SHETH A (1995) An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases* 3, 119-153.
- GIBSON CF, SINGER CJ, SCHNIDMAN AA and DAVENPORT TH (1984) Strategies for making an information system fit your organization. *Management Review* 73(1), American Management Association, 8-14.
- HAMMER M and CHAMPY J (1993) *Reengineering the corporation. A manifesto for business revolution*. HarperBusiness, New York, NY, USA.
- HARTOG C and HERBERT M (1986) 1985 opinion survey of MIS managers - key issues. *MISQ Quarterly* 10(4), 351-361.
- HENDERSON JC and VENKATRAMAN N (1999) Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal* 38(2/3), 472-484.
- HIRSCHHEIM RA and NEWMAN M (1991) Symbolism and information systems development: Myth, metaphor and magic. *Information Systems Research* 2(1), 29-62.
- HOLLAND CP and LIGHT B (1999) A critical success factors model for ERP implementation. *IEEE Software* 16(3), 30-36.
- HONG K-K and KIM Y-G (2002) The critical success factors for ERP implementation: An organizational fit perspective. *Information & Management* 40(1), 25-40.
- JABLONSKI S and BUSSLER C (1996) *Workflow management: Modeling concepts, architecture, and implementation*. International Thomson Computer Press, London, UK.
- KEIL M (1995) Pulling the plug: Software project management and the problem of project escalation. *MIS Quarterly* 19(4), 421-447.
- KEIL M, MANN J and RAI A (2000) Why software projects escalate: An empirical analysis and test of four theoretical models. *MIS Quarterly* 24(4), 631-664.
- KEIL M and ROBEY D (2001) Blowing the whistle on troubled software projects. *Communications of the ACM* 44(4), 87-93.
- KEY P (1998) SAP America hit with a \$500m suit. *Philadelphia Business Journal*.
- KINDLER E (2004) On the semantics of EPCs: A framework for resolving the vicious circle. In *Business Process Management: Second International Conference (Lecture Notes in Computer Science, Vol. 3080 / 2004)* (DESEL J, PERNICI B and WESKE M, Eds), pp 82-97, Springer, Potsdam, Germany.

- KLAUS H, ROSEMANN M and GABLE GG (2000) What is ERP? *Information Systems Frontiers* 2(2), 141-162.
- LEE J, SIAU K and HONG S (2003) Enterprise integration with ERP and EAI. *Communications of the ACM* 46(2), 54-60.
- LEYMANN F and ROLLER D (1999) *Production workflow: Concepts and techniques*. Prentice-Hall, Upper Saddle River, NJ.
- LOONAM J and MCDONAGH J (2004) Principles, foundations & issues in enterprise systems. In *Managing business with sap: Planning implementation and evaluation* (LAU L, Ed), pp 1-32, IDEA Group Publishing, Hershey, PA, USA.
- LUCAS JR. HC, STERN LN, WALTON EJ and GINZBERG MJ (1988) Implementing packaged software. *MIS Quarterly* 12(4), 536-549.
- LYYTINEN K (1999) Empirical research in information systems: On the relevance of practice in thinking of is research. *MIS Quarterly* 23(1), 25-27.
- MAJCHRZAK A, RICE RE, MALHOTRA A, KING N and BA SL (2000) Technology adaptation: The case of a computer-supported inter-organizational virtual team. *MISQ Quarterly* 24(4), 569-600.
- MARKUS ML (2004) Technochange management: Using it to drive organizational change. *Journal of Information Technology* 19(1), 4-20.
- MARKUS ML, PETRIE D and AXLINE S (2000) Bucking the trends: What the future may hold for ERP packages. *Information Systems Frontiers* 2(2), 181-193.
- Merriam-Webster (2003) Merriam-Webster's Collegiate Dictionary, 11th ed, Springfield, MA, USA.
- MURATA T (1989) Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, pp 541-580.
- NELSON RR (1991) Educational-needs as perceived by is and end-user personnel - a survey of knowledge and skill requirements. *MISQ Quarterly* 15(4), 503-521.
- NEWELL A and SIMON HA (1972) *Human problem solving*. Prentice-Hall, Englewood Cliffs, NJ, USA.
- ORACLE CORP. (2006) Oracle business benefits series. Oracle Corporation.
<http://www.oracle.com/customers/business-benefits/index.html>, accessed on 2006-02-13

- PALMER JW and MARKUS ML (2000) The performance impacts of quick response and strategic alignment in specialty retailing. *Information Systems Research* 11(3), 241-259.
- REICH BH and BENBASAT I (2000) Factors that influence the social dimension of alignment between business and information technology objectives. *MISQ Quarterly* 24(1), 81-113.
- REISIG W and ROZENBERG G (Eds.) (1998) *Lectures on petri nets I: Basic models*. Springer, Berlin, Germany et al.
- RIBSTEIN LE (2002) Market vs. Regulatory responses to corporate fraud: A critique of the Sarbanes-Oxley act of 2002. *Journal of Corporation Law* 28(1), 1-67.
- ROBEY D, ROSS JW and BOUDREAU M-C (2002) Learning to implement enterprise systems: An exploratory study of the dialectics of change. *Journal of Management Information Systems* 19(1), 17-46.
- ROSEMANN M and AALST WMPVD (2005) A configurable reference modelling language. *Information Systems (in press)*.
- ROSEMANN M, VESSEY I and WEBER R (2004) Alignment in enterprise systems implementations: The role of ontological distance. In *Twenty-Fifth International Conference on Information Systems*, pp 439-448, Association for Information Systems, Washington, DC.
- ROSEMANN M, VESSEY I, WEBER R and WYSSUSEK B (2005) Reconsidering the notion of requirements engineering for enterprise system selection and implementation. In *16th Australasian Conference on Information Systems*, Manly, Australia.
- SABHERWAL R and CHAN YE (2001) Alignment between business and its strategies: A study of prospectors, analyzers, and defenders. *Information Systems Research* 12(1), 11-33.
- SAP AG (2004) Consumer product companies run SAP.
http://www.sap.com/industries/consumer/pdf/CSBook_Consumer_Products.pdf ,
 accessed on 2006-02-13, SAP AG, Walldorf, Germany.
- SAP AG (2004) Enterprise services architecture: An introduction. *SAP White Papers*, SAP AG, Walldorf, Germany.
- SARBANES P and OXLEY MG (2002) Sarbanes-Oxley act. *Report by the House Financial Services committee*, US Congress, Washington, DC, USA.
- SCHEER A-W (2000) *ARIS - business process modeling*. Springer, Berlin.

- SCHMIDT R, LYYTINEN K, KEIL M and CULE P (2001) Identifying software project risks: An international delphi study. *Journal of Management Information Systems* 17(4), 5-36.
- SEGARS AH and GROVER V (1998) Strategic information systems planning success: An investigation of the construct and its measurement. *MISQ Quarterly* 22(2), 139-163.
- SEGARS AH and GROVER V (1999) Profiles of strategic information systems planning. *Information Systems Research* 10(3), 199-232.
- SILVER MS, MARKUS ML and BEATH CM (1995) The information technology interaction-model - a foundation for the MBA core course. *MISQ Quarterly* 19(3), 361-390.
- SMITH HJ, KEIL M and DEPLEDGE G (2001) Keeping mum as the project goes under: Toward an explanatory model. *Journal of Management Information Systems* 18(2), 189-228.
- SOFFER P, GOLANY B and DORI D (2003) ERP modeling: A comprehensive approach. *Information Systems* 28(6), 673-690.
- SOMERS TM, NELSON K and RAGOWSKY A (2000) Enterprise resource planning (ERP) for the next millennium: Development of an integrative framework and implications for research. In *Americas Conference on Information Systems*, pp 998-1004, Association for Information Systems, Long Beach, CA.
- SRINIVASAN A (1985) Alternative measures of system effectiveness - associations and implications. *MISQ Quarterly* 9(3), 243-253.
- STEIN T (1998) SAP sued over R/3. *InformationWeek* 1998(698), 134.
- STYHRE A (2003) *Understanding knowledge management - critical and postmodern perspectives*. Marston Book Services, Abingdon, UK.
- SUMNER M (1999) Critical success factors in enterprise wide information management systems projects. In *Proceedings of the 1999 ACM SIGCPR conference on Computer personnel research*, pp 297-303, ACM Press, New Orleans, LA.
- SUMNER M and HAMILTON J (2005) The turnaround ERP project: Strategies and issues. In *11th Annual Americas Conference on Information Systems (AMCIS)*, pp 2093-2098, AIS, Omaha, NE, USA.
- TURBAN E (1995) *Decision support and expert systems: Management support systems*. Prentice-Hall, Englewood Cliffs, NJ.

- UMBLE EJ, HAFT RR and UMBLE MM (2003) Enterprise resource planning: Implementation procedures and critical success factors. *European Journal of Operational Research* 146, 241-257.
- VERBEEK HMW, AALST WMPVD and KUMAR A (2004) XRL/Woflan: Verification and extensibility of an XML/petri-net-based language for inter-organizational workflows. *Information Technology and Management Journal* 5(1-2), 65-110.
- VOLKOFF O, STRONG DM and ELMES MB (2005) Understanding enterprise systems-enabled integration. *European Journal of Information Systems* 14(2), 110-120.
- WEICK KE (1969) *The social psychology of organizing*. Addison-Wesley, Reading, MA, USA.
- WIENER N (1948) *Cybernetics*. J. Wiley, New York.
- WIENER N (1967) *The human use of human beings: Cybernetics and society*. Avon Books, New York, NY, USA.
- ZUR MUEHLEN M (2004) *Workflow-based process controlling: Foundation, design and application of workflow-driven process information systems*. Logos, Berlin, Germany.

9 Figures and Tables

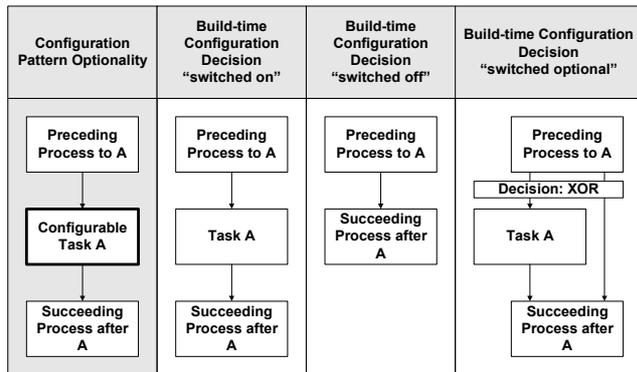


Table 1. Configuration Pattern of Optionality

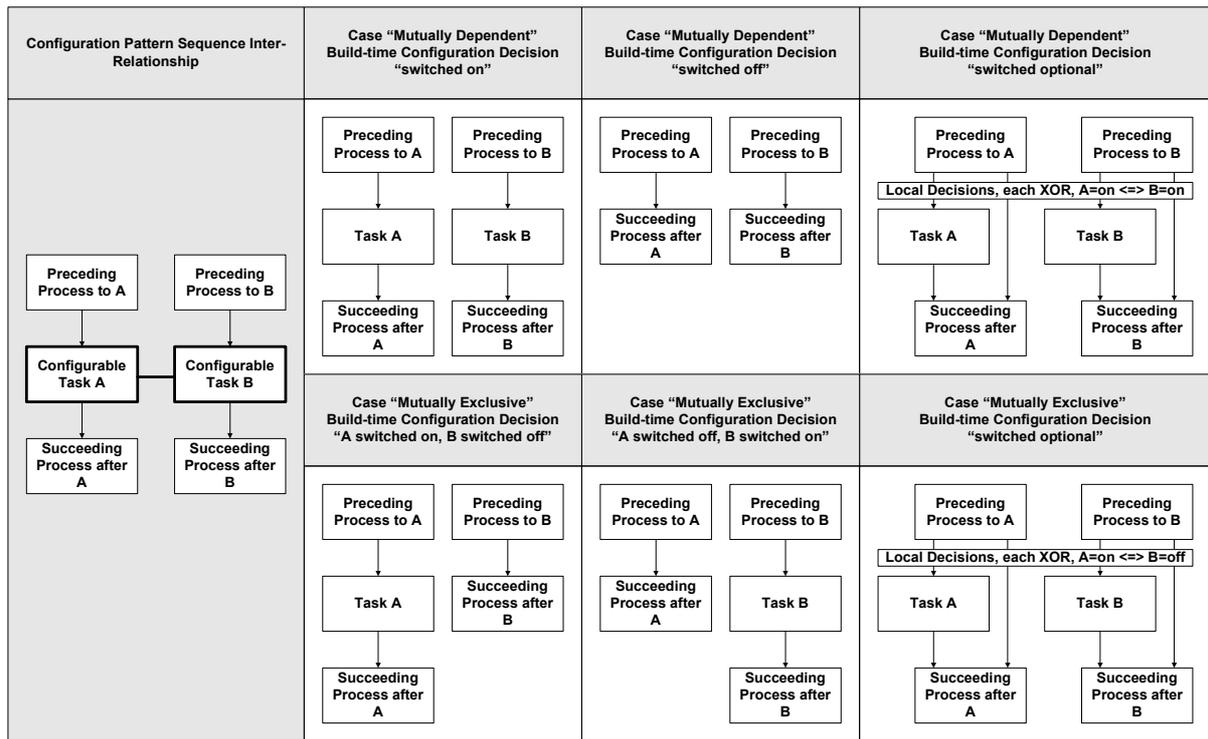


Table 2. Configuration Pattern of Sequence Inter-relationship

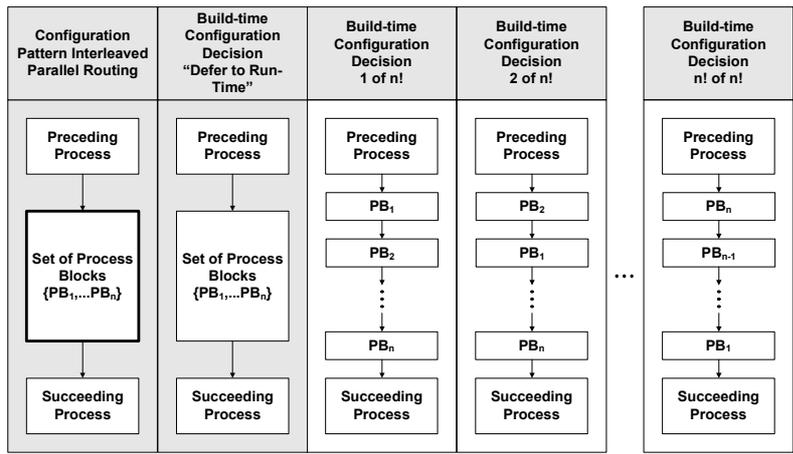


Table 3. Configuration Pattern of Interleaved Parallel Routing

Configuration Pattern Parallel Split	Build-time Configuration Decision "XOR"	Build-time Configuration Decision "OR"	Build-time Configuration Decision "AND"	Build-time Configuration Decision "Sequence 1"	Build-time Configuration Decision "Sequences i" ($1 \leq i \leq n$)	Build-time Configuration Decision "Sequence n"

Table 4. Semantic Configuration Patterns of Parallel Split, Exclusive Choice, and Multi Choice

	All possible scenarios where function succeeds an event	All possible scenarios where function succeeds a join	All possible scenarios where function succeeds a split
All possible scenarios where function is succeeded by and event	Syntactic Pattern EFE	Syntactic Pattern JFE	Syntactic Pattern SFE
	Syntactic Pattern EFJ	Syntactic Pattern JFJ	Syntactic Pattern SFJ
All possible scenarios where function is succeeded by join			
	Syntactic Pattern EFS	Syntactic Pattern JFS	Syntactic Pattern SFS
All possible scenarios where function is succeeded by split			

Table 5. Syntactic EPC Pattern as a result of all lawful contexts of a configurable Function A within an EPC

Syntactic Pattern EFE	Build-time Configuration Decision							
	"on"		"off"			"optional"		
	Variant 1	Variant 1	Variant 2	Variant 3	Variant 1	Variant 2	Variant 3	Variant 4

Table 6. Configuration Decisions (Choices) for Syntactic EPC Pattern EFE

Syntactic Pattern EFJ	Possible context of EFJ	Syntactic Pattern SFE	Possible context of SFE

Table 7. Possible Contexts of Syntactic Configuration Patterns EFJ and SFE

Semantic Patterns in an EPC environment	Build-time configuration Decisions for Semantic Patterns <i>Multi Choice/Synchronizing Merge, Exclusive Choice/Simple Merge, Parallel Split/Synchronization</i> , Case "Choice/Split succeeds Function"					
	Build-time Configuration Decision "XOR" (<i>Exclusive Choice/Simple Merge, Multi Choice/Synchronizing Merge</i>)	Build-time Configuration Decision "OR" (<i>Multi Choice/Synchronizing Merge</i>)	Build-time Configuration Decision "AND" (<i>Exclusive Choice/Simple Merge, Multi Choice/Synchronizing Merge, Parallel Split/Synchronization</i>)	Sequence 1 (<i>Exclusive Choice/Simple Merge, Multi Choice/Synchronizing Merge</i>)	Sequence i ($1 \leq i \leq n$) (<i>Exclusive Choice/Simple Merge, Multi Choice/Synchronizing Merge</i>)	Sequence n (<i>Exclusive Choice/Simple Merge, Multi Choice/Synchronizing Merge</i>)
Multi Choice and Synchronizing Merge 						
Exclusive Choice and Simple Merge 						
Parallel Split and Synchronization 						

Table 8. Configuration Decisions for Semantic Patterns *Multi Choice/Synchronisation, Exclusive Choice/Simple Merge, and Parallel Split/Synchronisation*

Semantic Configuration Pattern of Optionality	Build-time Configuration Decision		
	"on"	"off"	"optional"

Table 9. Petri-net-based Configuration Semantic Pattern of Optionality

Possible Pre-Reduction Petri Nets (τ as Silent Steps where applicable)	Reduced Petri Nets

Table 10. Reduction Rules for Removing Silent Steps in Petri Nets

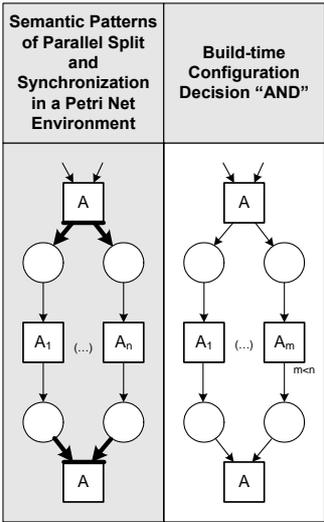


Table 11. Petri-net-based Configuration Patterns for Parallel Split and Synchronisation

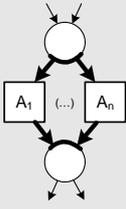
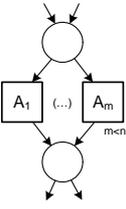
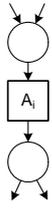
Semantic Patterns of Exclusive Choice and Simple Merge in a Petri Net Environment	Build-Time Configuration Decision "XOR"	Build-Time Configuration Decision "Sequence i"
		

Table 12. Petri-net-based Configuration Patterns for Exclusive Choice and Simple Merge

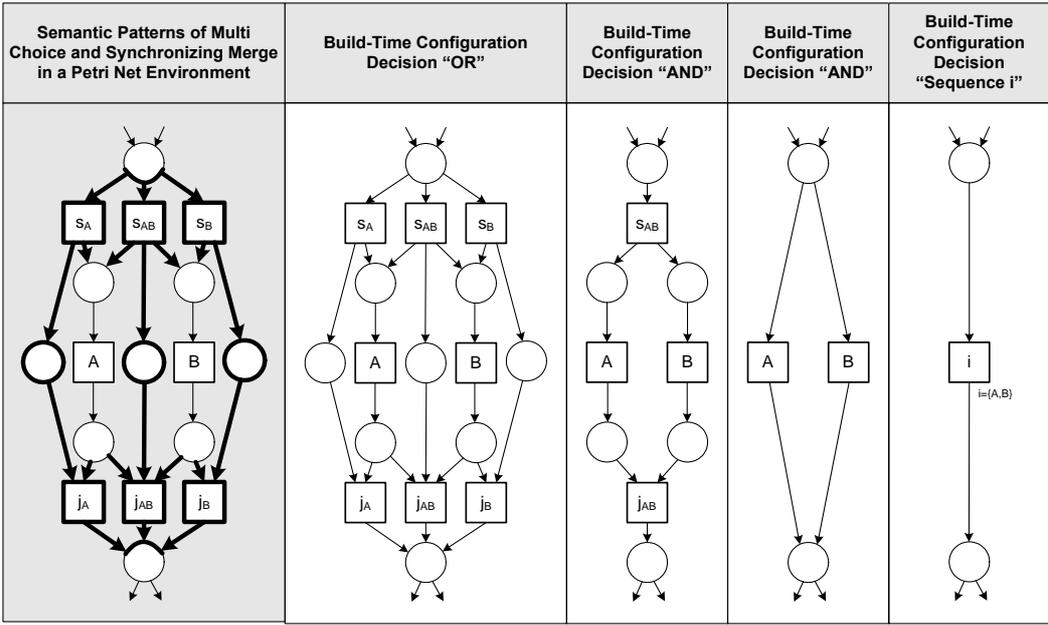


Table 13. Petri-net-based Configuration Patterns for Multi Choice and Synchronising Merge