EPC verification in the ARIS for MySAP reference model database

EPC Verification in the ARIS for MySAP reference model database

B.F. van Dongen and M.H. Jansen-Vullers

Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
{b.f.v.dongen,m.h.jansen-vullers}@tm.tue.nl

Abstract. To configure a process-aware information system (e.g., a workflow system, an ERP system), a business model needs to be transformed into an executable process model. Due to similarities in these transformations for different companies, databases with reference models, such as ARIS for MySAP, have been developed. The models stored in such a database can be customized to generate an executable model. Since these customized models are typically used on an execution level, it is of the utmost importance that both the reference models and their customizations are free of erroneous constructs.

In this paper, we analyze a selection of the reference models for SAP R/3 that are stored in the ARIS for MySAP database, and we verify whether they are correct. Since these models are stored as Event-driven Process Chains (EPCs), we use a verification approach tailored towards the verification of this language to check for errors in the models. We show that the reference models in ARIS for MySAP indeed contain some errors and we present the implications of those errors, if these models would be used for the execution of business processes.

Keywords: Event-driven Process Chains, Verification, SAP, Reference Models.

1 Introduction

Nowadays, *process-aware information systems* such as Enterprise Resource Planning (ERP) [18] systems and Workflow Management (WFM) [4, 21] systems are used to support a wide range of operational business processes. On an operational level, these systems are often configured on the basis of a process model. The design of such a process model is a complicated and error prone task. Furthermore, the process models that are designed in difference companies are often very similar. For this reason, databases with process models for many different applications have been developed. These databases can be used as a reference during process design, hence the term *reference models*.

Together with the business model of a company, a reference model is selected that best fits the process under consideration. During the process model design phase, a designer customizes that reference model to fit the business model of the company. The result of this customization phase is an informal specification of

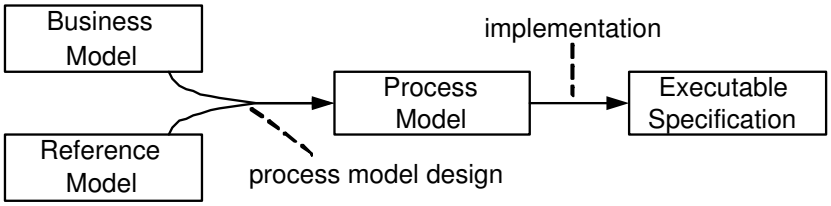

Fig. 1. Phases in the configuration of a PAIS

a process in terms of a customized process model. In the implementation phase, this model is used to implement an executable specification for a specific information system, such as SAP R/3. These phases are presented in Figure 1. Since all the steps between selecting a reference model and producing an executable specification are performed by humans, errors are likely to be introduced.

The use of reference models does not eliminate the possibility of introducing errors into the process model. It should, however, assist the designer in such a way that errors are less likely to be introduced. Therefore, it is of the utmost importance that the selected reference model is correct. Especially since usually, many processes are modelled independently of each other, even though, when considering the real life processes, process models are highly dependent. Furthermore, when errors in process models are implemented in an executable specification in the implementation phase, they can have severe operational consequences.

To find errors in process models, many authors have developed verification methods. Basically, all of these verification methods can be used to check whether a process model is *correct*, in other words, they can be used to check for *correctness* of a process model. In Section 2, we categorize verification methods and we show that some methods look on the level of the executable specification, some on level of the business model and some on process models or reference models.

In this paper, we focus on the correctness of reference models for a specific information system, SAP R/3. The reference models are available in the ARIS for MySAP database in the ARIS Toolset, a commercial product of IDS-Scheer. As a modelling language, the ARIS Toolset uses *Event-driven Process Chains* (EPCs) [17, 18, 31]. We selected SAP R/3, since EPCs are used in a large variety of systems, of which SAP R/3 is market leader. Furthermore, many verification approaches exist for EPCs. The verification method we chose looks at verification from a designers point of view and assumes the process designer to know what he intends to model.

We take the SAP reference models as a starting point, and use the verification approach presented in [13], as our verification method. We show that many of the SAP reference models are correct and can indeed be used without any problems. However, we also show that some of the models should be used with care, i.e., if the environment in which they are used satisfies certain conditions they are correct. Furthermore, we show that a small number of the reference models is

structurally incorrect, i.e., they need to be revised before they can be used as reference models. With respect to these errors, we investigate some common causes, and show how designers could avoid these errors.

The remainder of this paper is structured as follows. In Section 2 we discuss related work with respect to the verification of process models. In Section 3 we describe our domain of analysis: SAP R/3, the EPC modelling method and the reference models. Next, in Section 4, we describe the approach for the verification of these models as implemented in the ProM framework[1], and described in [13, 12]. Following this approach we are able to evaluate the SAP reference models in Section 5. This evaluation is based on two lines: the evaluation of one complete module (Section 5.1) and a guided search through the database with reference models (Section 5.2). Finally, in Section 6, we draw some conclusions.

A short version of the work presented here has been published at the International Conference on Business Process Management (BPM05) [14]. This paper extends [14] by presenting the verification results for the whole module from Section 5.1. Furthermore, we present a guided search trough the reference model database in Section 5.2.

2 Related work

Since the mid-nineties, a lot of work has been done on the verification of process models, and in particular workflow models. In 1996, Sadiq and Orlowska [27] were among the first to point out that modeling a business process (or workflow) can lead to problems like livelock and deadlock. In their paper, they present a way to overcome syntactical errors, but they ignore the semantical errors. Nowadays, most work that is conducted focusses on semantical issues, i.e. "will the process specified always terminate" and similar questions. The work that has been conducted on verification in the last decade can roughly be put into three main categories, namely "verification of models with formal semantics", "verification of informal models" and "verification by design". In this section, we present these categories and give relevant literature for each of them.

2.1 Verification of models with formal semantics

In the first category we consider the work that has been done on the verification of modeling languages with formal semantics. One of the most prominent examples of such a language are Petri nets [11, 23, 24]. Since Petri nets have a formal mathematical definition, they lend themselves to great extent for formal verification methods. Especially in the field of *workflow management*, Petri nets have proven to be a solid theoretical foundation for the specification of processes. This, however, led to the need of verification techniques, tailored towards Petri nets that represent workflows. In the work of Van der Aalst and many others [2, 6, 10, 16, 35] these techniques are used extensively for verification of different classes of

[1] See www.processmining.org for details.

workflow definitions. However, the result is the same for all approaches. *Given a process definition, the verification tool provides an answer in terms of "correct" or "incorrect".* However, not all modeling languages have a formal semantics. On the contrary, the most widely used modeling techniques, such as UML and EPCs are merely an informal representation of a process. These modeling techniques therefore require a different approach to verification.

2.2 Verification of informal models

Modeling processes in a real-life situation is often done in a less formal language. People tend to understand informal models easily, and even if models are not executable, they can help a great deal when discussing process definitions. However, at some point in time, these models usually have to be translated into a specification that can be executed by an information system. This translation is usually done by computer scientists, which explains the fact that researchers in that area have been trying to formalize informal models for many years now. Especially in the field of workflow management, a lot of work has been done on translating informal models to Petri nets. Many people have worked on the translation of EPCs to Petri nets, cf., [1, 3, 9, 20]. The basic idea of these authors however is the same: "Restrict the class of EPCs to a subclass for which we can generate a sound Petri net". As a result, the ideas are appealing from a scientific point of view, but not useful from a practical point of view.

Also non-Petri-net based approaches have been proposed for the verification of informal modeling languages. One of these ideas is *graph reduction*. Since most modeling languages are graph-based, it seems a good idea to reduce the complexity of the verification problem by looking at a reduced problem, in such a way that correctness is not violated by the reduction, i.e. if a model is not correct before the reduction, it will not be correct after the reduction and if the model is correct before the reduction, it will be correct after the reduction. From the discussion on graph reduction techniques started by Sadiq and Orlowska in 1999 [28, 29] and followed up by many authors including Van der Aalst et al. in [5] and Lin et al in [22], it becomes clear that again the modeling language is restricted to fit the verification process. In general this means that the more advanced routing constructs cannot be verified, while these constructs are what makes informal models easy to use.

The tendency to capture informal elements by using smarter semantics is reflected by recent papers, cf. [3, 9, 19]. In these papers, the problem is looked at from a different perspective. Instead of defining subclasses of models to fit verification algorithms, the authors try to give a formal semantics to an informal modeling language. Even though all these authors have different approaches, the goal in every case is similar: *try to give a formal executable semantics for an informal model.*

2.3 Verification by design

The last category of verification methods is somewhat of a by-stander. Instead of doing verification of a model given in a specific language, it is also possible to give a language in such a way that the result is always correct. An example of such a modelling language is IBM MQSeries Workflow [21]. This language uses a specific structure for modelling, which will always lead to a correct and executable specification. However, modelling processes using this language requires advanced technical skills and the resulting model is usually far from intuitive.

In this section, we have presented an overview of the literature on process model verification. We have categorized the various methods in three main categories and pointed out why many of them are not used in practice. In this paper, we use the technique presented in [13] that can be seen as a combination of first two categories. It assumes *the designer* to be able to decide whether or not a specification is semantically correct. This technique has been implemented in the Process Mining (ProM) Framework, that is able to import EPCs defined in the ARIS Toolset[2] and provides the designer with feedback about possible problems. Since SAP reference models are available in the ARIS Toolset format, and the users of these reference models are typically consultants that have a deep knowledge about the process under consideration, we found this to be the best approach for the verification of the SAP R/3 reference models.

3 SAP R/3 Reference models

Several authors researched the area of reference models before, see e.g. [15, 34, 26, 32, 33, 7, 25, 30, 8]. In this section we introduce reference models based on [26] and then explain Event-driven Process Chains (EPCs).

3.1 Reference models

Reference models are generic conceptual models that formalize recommended practices for a certain domain [15, 34]. Reference models accelerate the modelling process by providing a repository of potentially relevant business processes and structures. With the increased popularity of business modelling, a wide and quite heterogenous range of purposes can motivate the use of a reference model. These purposes include software development, software selection, configuration of Enterprise Systems, workflow management, documentation and improvement of business processes, education, user training, auditing, certification, benchmarking, and knowledge management [26].

What we learn from previous authors is that we can distinguish two types of reference models: industry models and application models. Industry reference models are generally higher level models and they aim to streamline the design

[2] See www.ids-scheer.com for information about the ARIS toolset.

of enterprise-individual (particular) models by providing a generic solution. Application reference models describe the structure and functionality of business applications including Enterprise Systems. In these cases, a reference model can be interpreted as a structured semi-formal description of a particular application. This application can then be seen as an existing off-the-shelf-solution that supports the functionality and structure described in the reference model.

Rosemann and van der Aalst explain in [26] that application reference models tend to be more complex than industry reference models. They explain that the SAP reference model is one of the most comprehensive models [8]. Its data model includes more than 4000 entity types and the reference process models cover more than 1000 business processes and inter-organizational business scenarios. In the early nineties, two companies called SAP and IDS Scheer, have developed an intuitive process modelling langauge, which resulted in the process modelling language Event-driven Process Chains (EPCs). This language has been used for the design of the reference process models in the ARIS for MySAP database that we consider in this paper. EPCs also became the core modelling language in the Architecture of Integrated Information Systems (ARIS) [30, 17].

3.2 Event-driven Process Chains (EPCs)

The SAP R/3 reference models are modelled as Event-driven Process Chains, or EPCs, in the ARIS Toolset. An EPC consists of three main elements. Combined, these elements define the flow of a business process as a chain of events. The elements used are:

Functions, which are the basic building blocks. A function corresponds to an activity (task, process step) which needs to be executed. A function is drawn as a box with rounded corners.

Events, which describe the situation before and/or after a function is executed. Functions are linked by events. An event may correspond to the position of one function and act as a precondition of another function. Events are drawn as hexagons.

Connectors, which can be used to connect functions and events. This way, the flow of control is specified. There are three types of connectors: $\wedge$ (and), $\times$ (xor) and $\vee$ (or). Connectors are drawn as circles, showing the type in the center of the circle.

Functions, events and connectors can be connected with edges in such a way that (i) events have at most one incoming edge and at most one outgoing edge, but at least one incident edge (i.e. an incoming or an outgoing edge), (ii) functions have precisely one incoming edge and precisely one outgoing edge, (iii) connectors have either one incoming edge and multiple outgoing edges, or multiple incoming edges and one outgoing edge, and (iv) in every path, functions and events alternate (no two functions are connected and no two events are connected, not even when there are connectors in between).

In the ARIS for MySAP reference databases, there are hundreds of EPCs that can be used in many different situations, from "asset accounting" to "procurement" and "treasury". Since we cannot discuss all these models here, we focus on one of the modules that can be considered to be a representative subset of all reference models, namely "procurement". This is a set of some 40 EPCs, all in the area of procurement. They describe processes for (i) internal procurement, (ii) pipeline processing (iii) procurement of materials and external services, (iv) procurement on a consignment basis, (v) procurement via subcontracting, (vi) return deliveries, and (vii) source administration.

All 40 models were analyzed using the approach described in [13]. Before we show the results of this verification process in Section 5, we first briefly introduce this verification approach in Section 4.

4 Verification approach

For the verification of the EPCs in our reference model database, we use the approach described in [13]. This verification approach is tailored towards the verification of Event-driven Process Chains and it assumes *the designer* of an EPC to be able to decide whether or not the EPC is correct. The approach is implemented in the ProM framework ([12]) and it is freely available for download.

The verification process described in [13] consists of several steps. In the first step, the designer of the EPC has to provide the tool with all combinations of initial events that could initiate the modelled process. Using this, the tool calculates all the possible outcomes of the process (in terms of events that occurred and have not been dealt with). Then, the tool requires the designer to divide those outcomes in two groups, the first of which contains all the outcomes that represent the desired behavior of the process. The second group contains the undesired behavior. Clearly, depending on the model, either of the two groups can be empty.

4.1 Semantically correct models

Models that are semantically correct are models of processes that, when started in any allowed state, will *always* terminate in one of the allowed termination states. In other words, routing constructs do not have to be synchronized. Choices can be made locally, without any knowledge of the execution history.

4.2 Syntactically correct models

Models that are syntactically correct are models of processes that, when started in any allowed state, will always *have the possibility to* terminate in one of the allowed termination states. In other words, routing constructs have to be synchronized. Not all choices can be made locally, instead, the execution history limits the available options. An example of such a construct can be found in Figure 2, where the choices after functions A and B have to be synchronized in order to allow function C or D to execute.

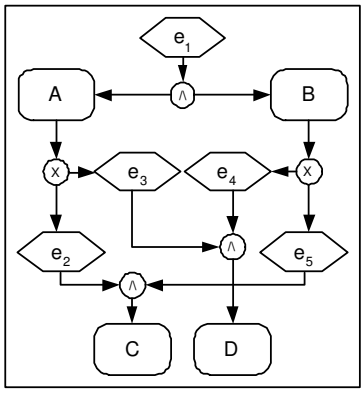

Fig. 2. EPC with choice synchronization

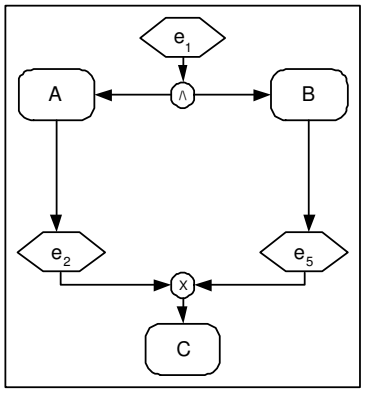

Fig. 3. EPC with erroneous routing

4.3 Incorrect models

The final class of models are the incorrect ones. These models contain syntactical errors, such as an AND-split followed by an XOR-join or the other way around. An example of such an incorrect model is shown in Figure 3, where functions A and B originate from an AND-split, and are later joined by an XOR-join. As a result, function C will be carried out twice based on the same case in event e_1.

5 Verification of the reference models

The application of the verification approach presented in Section 4 is based on a basic assumption: It assumes that the designer of a model has a good understanding of the actual business process that was modelled, and he knows which combinations of events can actually initiate the process in real life. Typically, reference models are used by consultants that do indeed have a good understanding of the process under consideration. Besides, they know under what circumstances processes can start, and which outcomes of the execution are desired and which aren't. Therefore, the approach seems to be well suited for the verification of the SAP reference models.

5.1 Procurement module

As stated in Section 3 we focus on the procurement module of the ARIS for MySAP reference model database, since it can be seen as a representative subset of all reference models. The procurement module contains several sub-modules and we analyzed all the models from these modules using the approach presented in Section 4. Surprisingly, already in the first model (Internal Procurement) there were structural errors. In Figure 4, we show a screenshot of the verification tool used. It shows part of an EPC in which an AND-split is later joined by an

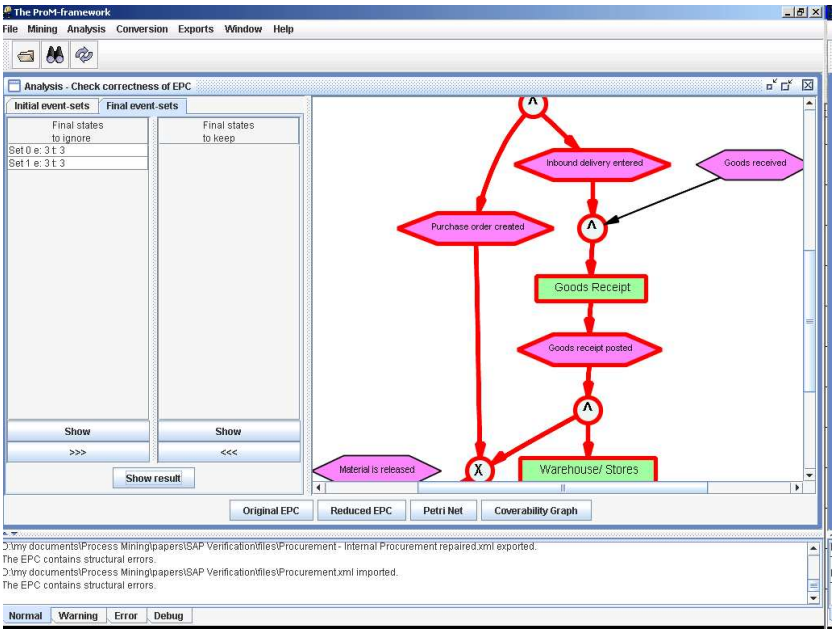

Fig. 4. Erroneous "Internal Procurement"

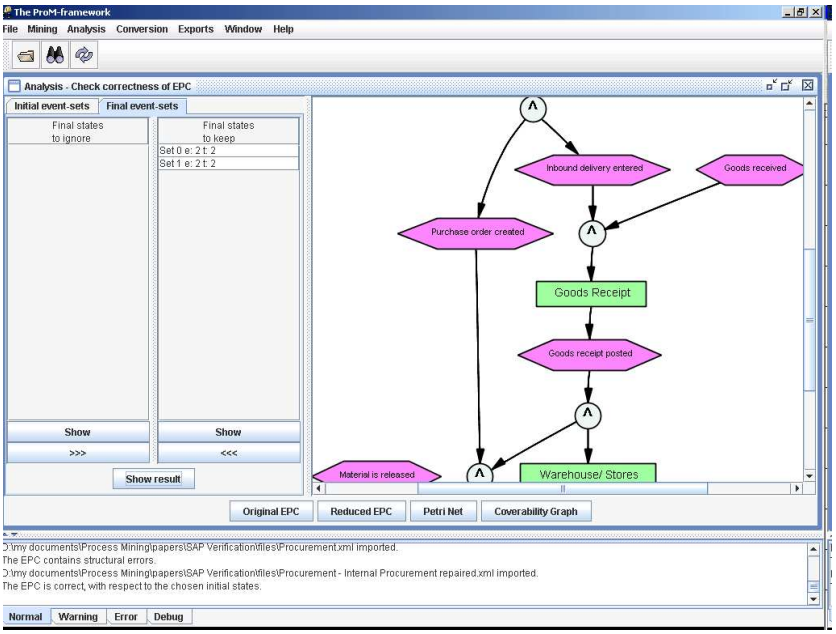

Fig. 5. Repaired "Internal Procurement"

XOR join. Recall Figure 3, where we have shown that this is clearly incorrectly modelled. As a result, if this model would *not* be repaired, payments could be made for goods that were never received. Obviously, this is not desirable. In Figure 5 we show the repaired model, i.e. the XOR-join has been changed into an AND-join. Now, the model is semantically correct, which means that it can be used in a business environment without problems.

The results of our analysis of the whole procurement module are presented in Table 1, which contains three columns. The first column shows the name of the module. The second contains the verification result. We use "I" for incorrect models, "S" for syntactically correct models, and "C" for semantically correct ones. The final column gives the business-wise implication of the error found if this model would be translated into an executable specification, if applicable.

Table 1. Table of results for the procurement module

Module name	Result	Implication of the problem
Internal Procurement	I	Payments can be done for goods never received.
↪ Goods Receipt	C	
↪ Invoice Verification	C	
↪ Purchase Requisition	C	
↪ Purchasing	C	
↪ Warehouse stores	C	
Pipeline Processing	C	
↪ Invoice Verification	C	
↪ Pipeline Withdrawal	C	
Materials and External Services	S	An invoice can be paid for ordered goods (not services) that have not yet been delivered.
↪ Goods Receipt	C	
↪ Invoice Verification	C	
↪ Purchase Requisition	C	
↪ Purchasing	C	
↪ Service Entry Sheet	C	
↪ Transportation	C	
↪ Warehouse/Stores	C	
Procurement on a Consignment basis	C	
↪ Goods Receipt	C	
↪ Invoice Verification	C	
↪ Purchase Requisition	C	
↪ Purchasing	C	
↪ Warehouse/Stores	C	
Procurement via Subcontracting	I	An invoice that is received twice will be paid twice.
↪ Goods Receipt	C	
↪ Invoice Verification	C	
↪ Provision of Components	C	
↪ Purchase Requisition	C	
↪ Purchasing	C	
↪ Transportation	C	
↪ Warehouse/Stores	S	When materials are simultaneously placed into the stock and removed from it, erroneous behavior occurs. Operational procedures should avoid this.
Return Deliveries	C	
↪ Invoice Verification	C	
↪ Outbound Shipments	C	
↪ Quality Notification	C	
↪ Shipping	C	
↪ Warehouse	C	
Source Administration	C	
↪ Outline Purchase Agreements	C	Redundant objects are present.
↪ RFQ/Quotation	C	

5.2 Guided model selection

From the previous section it seems that we can conclude that most errors are made in the higher level models. Using this as a guide, we tried to find problems in the reference models. In fact, in the high level models, it is not hard to find these mistakes. These high level models are usually more complex then the lower level models (i.e. they contain more functions, events and connectors). Therefore, errors are more likely to be introduced there. We would like to mention two observations that we made during this guided model selection.

The first observation is that often, one particular initial event is applied in several (sub)models. Take, for example, the event "Deliveries need to be planned". This event occurs in 15 different models. Every time it occurs, it is joined with the event "delivery is relevant for shipment". However, in some models this is done via an XOR-join, and in some models via an AND-join. In Figure 6, we show these two events, used in the "Consignment Processing" module, where they are joined by an XOR-join. However, in Figure 7, we show the same two events in an AND-join configuration. Since these two events are always followed by something that refers to transportation, it seems that they should always appear in an AND-join configuration. However, only a designer with deep knowledge of the process that is modelled can decide if that is the case.

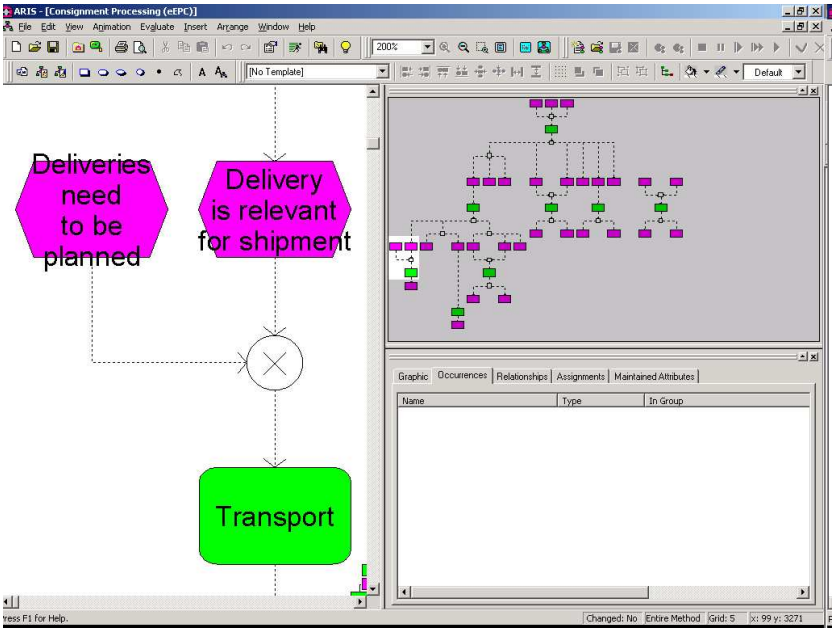

Fig. 6. Events joined as XOR ($\times$)

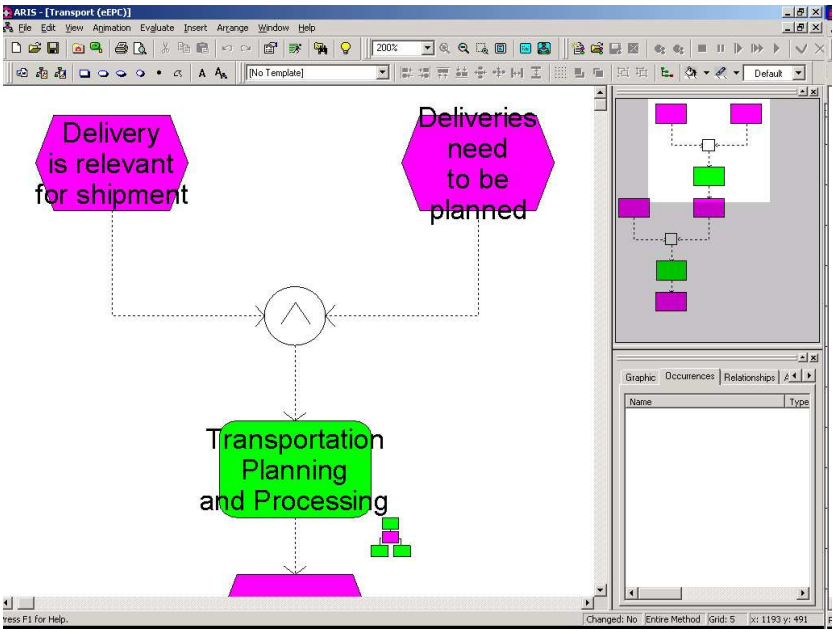

Fig. 7. Events joined as AND ($\wedge$)

The second observation, that seems to be a common one, is the effect of customization. Typically, many different organizations have very similar processes. Therefore, when building reference models, it is a good idea to use one model to create another one. The new model is then customized in such a way that it fits the needs of the new organization better. Figure 8 shows a screenshot of the ARIS toolset, showing two models, namely "Q-notification with Complaint Against Vendor" on top and "Internal Quality Notification" below. These two models are exactly alike, except that in the top-model, a vendors complaint score can be updated. Here, customization has been applied correctly.

In Figure 9, two models are shown for which the customization was performed incorrectly. The model on the left hand side represents the handling of a "Service Order" and on the right hand side it represents the handling of a "Maintenance Order". They are very similar, except that the latter does not make a distinction between maintenance at a customer site and at an internal site. Both models however, contain the same mistake. When services are to be entered, the rightmost event called "Services are to be Entered" occurs. However, when that is the case, due to the XOR-split in front of it, the function "Overall Completion Confirmation" will never be able to execute. Solving this problem requires a good understanding of the modelled situation since many correct solutions are possible.

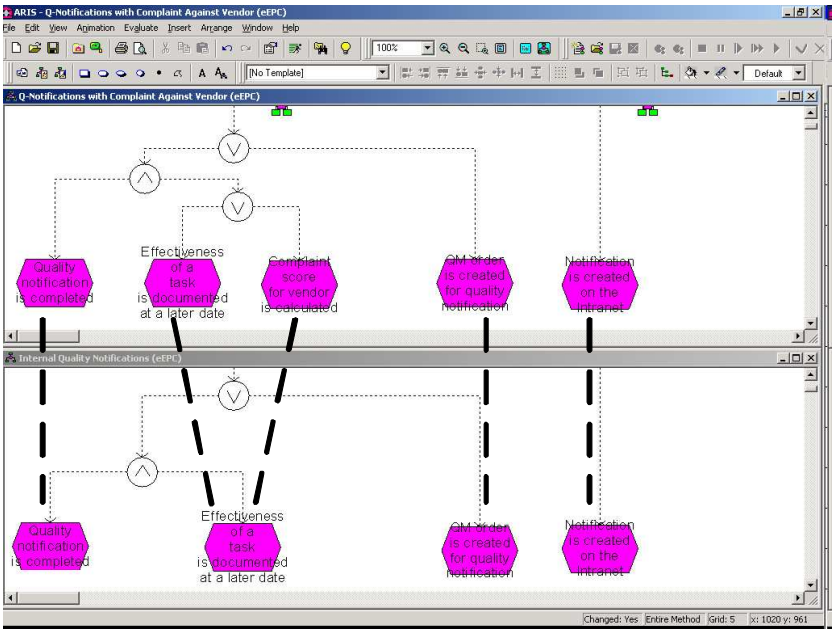

Fig. 8. Correct customization

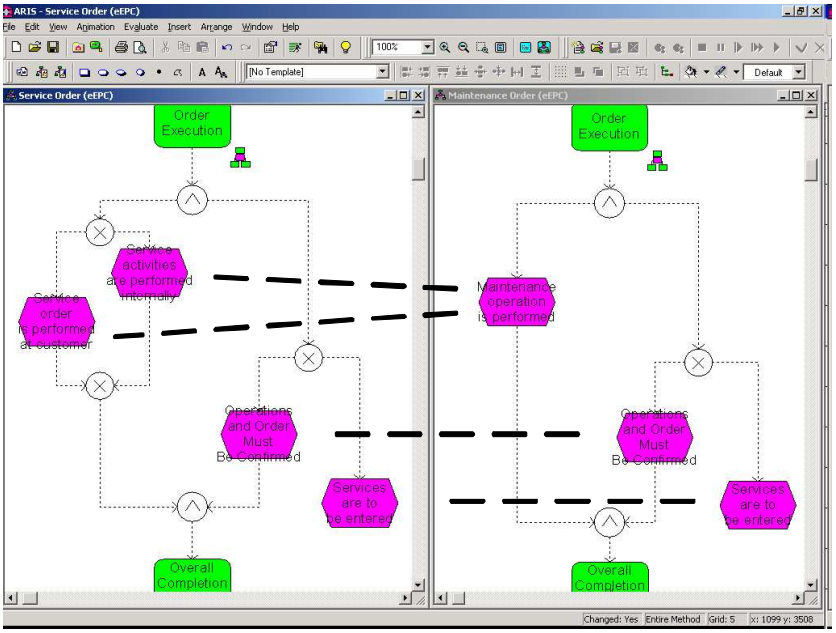

Fig. 9. Erroneous customization

6 Conclusion

Although we only looked at a small subset of the reference model database, we can draw some important conclusions. First of all, it seems that problems are more easily introduced into larger models than into smaller ones. The reason that we did not find many problems in low level models can probably be explained by the fact that these models are typically very small. However, when these models are connected by higher level models, errors are easily introduced. As we saw in Section 5, these errors can lead to severe complications, such as invoices being paid twice. Furthermore, when the same, or similar events are used in several modules, special care has to be taken. As we saw for the events with respect to shipments, there was no consensus about the use of them in different modules.

Finally, the errors we found with our verification approach were all trivial to repair. Therefore, we feel that the use of such a verification tool in the early stages of process modelling, or reference model development would greatly improve the effectiveness and applicability of these models in later stages.

References

1. W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.
2. W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, Berlin, 2000.
3. W.M.P. van der Aalst, J. Desel, and E. Kindler. On the Semantics of EPCs: A Vicious Circle. In M. Nüttgens and F.J. Rump, editors, *Proceedings of the EPK 2002: Business Process Management using EPCs*, pages 71–80, Trier, Germany, November 2002. Gesellschaft für Informatik, Bonn.
4. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
5. W.M.P. van der Aalst, A. Hirnschall, and H.M.W. Verbeek. An Alternative Way to Analyze Workflow Graphs. In A. Banks-Pidduck, J. Mylopoulos, C.C. Woo, and M.T. Ozsu, editors, *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02)*, volume 2348 of *Lecture Notes in Computer Science*, pages 535–552. Springer-Verlag, Berlin, 2002.
6. W.M.P. van der Aalst and A.H.M. ter Hofstede. Verification of Workflow Task Structures: A Petri-net-based Approach. *Information Systems*, 25(1):43–69, 2000.
7. P. Bernus. *GERAM: Genrealised Enterprise Reference Architecture and Methodology*.
8. T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.
9. J. Dehnert and W.M.P. van der Aalst. Bridging the Gap Between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems*, 13(3):289–332, 2004.

10. J. Dehnert and P. Rittgen. Relaxed Soundness of Business Processes. In K.R. Dittrich, A. Geppert, and M.C. Norrie, editors, *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, volume 2068 of *Lecture Notes in Computer Science*, pages 157–170. Springer-Verlag, Berlin, 2001.
11. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
12. B.F. van Dongen, A.K.A. de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A new era in process mining tool support. In *accepted tool presentation at ATPN 2005*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2005.
13. B.F. van Dongen, H.M.W. Verbeek, and W.M.P. van der Aalst. Verification of EPCs: Using reduction rules and Petri nets. In *accepted full paper at CAiSE 2005*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2005.
14. B.F. van Dongen and M.H. Vullers-Jansen. Verification of SAP reference models. In *accepted short paper at BPM 2005*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2005.
15. P. Fettke and P. Loos. Classification of Reference Models - a methodology and its application. *Information Systems and e-Business Management*, 1(1):35–53, 2003.
16. K. van Hee, N. Sidorova, and M. Voorhoeve. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In W.M.P. van der Aalst and E. Best, editors, *Application and Theory of Petri Nets 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 335–354. Springer-Verlag, Berlin, 2003.
17. G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.
18. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
19. E. Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 82–97. Springer-Verlag, Berlin, 2004.
20. P. Langner, C. Schneider, and J. Wehler. Petri Net Based Certification of Event driven Process Chains. In J. Desel and M. Silva, editors, *Application and Theory of Petri Nets 1998*, volume 1420 of *Lecture Notes in Computer Science*, pages 286–305. Springer-Verlag, Berlin, 1998.
21. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
22. H. Lin, Z. Zhao, H. Li, and Z. Chen. A Novel Graph Reduction Algorithm to Identify Structural Conflicts. In *Proceedings of the Thirty-Fourth Annual Hawaii International Conference on System Science (HICSS-35)*. IEEE Computer Society Press, 2002.
23. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
24. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
25. M. Rosemann. *Application Reference Models and Building Blocks for Management and Control (ERP systems)*, pages 595–616. Springer-Verlag, Berlin, 2003.

26. M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. QUT Technical report, FIT-TR-2003-05, Queensland University of Technology, Brisbane, 2003.
27. W. Sadiq and M.E. Orlowska. Modeling and verification of workflow graphs. Technical Report No. 386, Department of Computer Science, The University of Queensland, Australia, 1996.
28. W. Sadiq and M.E. Orlowska. Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models. In M. Jarke and A. Oberweis, editors, *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE '99)*, volume 1626 of *Lecture Notes in Computer Science*, pages 195–209. Springer-Verlag, Berlin, 1999.
29. W. Sadiq and M.E. Orlowska. Analyzing Process Models using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.
30. A.-W. Scheer. *Business Process Modelling*. 3rd edition, 2000.
31. A.W. Scheer. *Business Process Engineering, Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1994.
32. L. Silverston. *The Data Model Resource Book, Volume 1, A Library of Universal Data Models for all Enterprises*. revised edition, 2001.
33. L. Silverston. *The Data Model Resource Book, Volume 2, A Library of Data Models for Specific Industries*. revised edition, 2001.
34. U. Frank. Conceptual Modelling as the Core of Information Systems Discipline - Perspectives and Epistemological Challanges. In *Proceedings of the America Conference on Information Systems - AMCIS '99*, pages 695–698, Milwaukee, 1999.
35. H.M.W. Verbeek and W.M.P. van der Aalst. Woflan 2.0: A Petri-net-based Workflow Diagnosis Tool. In M. Nielsen and D. Simpson, editors, *Application and Theory of Petri Nets 2000*, volume 1825 of *Lecture Notes in Computer Science*, pages 475–484. Springer-Verlag, Berlin, 2000.