Integrating Business Process Reengineering with Application Development under Architecture

H.A. Reijers[1] and R.A. van der Toorn[2,3]

[1]Eindhoven University of Technology, Department of Technology Management,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
h.a.reijers@tue.nl
[2]Eindhoven University of Technology, Department of Mathematics and Computing Science,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
r.a.v.d.toorn@tue.nl
[3] Deloitte & Touche Management and ICT Consultants,
P.O. Box 23103, NL-1100 DP, Amsterdam, The Netherlands
rvandertoorn@deloitte.nl

Abstract

Business process reengineering (BPR) is one of a company's major instruments to achieve improved performance. BPR projects almost always include major efforts on developing and integrating information technology. This paper addresses how BPR and application development may be methodologically integrated, building on a product-based design of business processes and a component-based IT development. An architectural framework inspired on Zachman's is used as a frame of reference. The need for a closer integration between BPR and application development is shown by a case study, which involves an actual BPR project for a large Dutch bank. The paper also includes an example to illustrate the presented methodology.

Keywords:

BPR, application development, systems architecture, formal modeling, case description.

Introduction

At the beginning of the 21st century, process thinking and Business process reengineering (BPR) (Hammer and Champy, 1993) have become mainstream thinking for business people and systems people alike. At the same time, methodologies for application development have matured and substantial standardization is taken place in the field of modeling techniques, e.g. UML (Kruchten, 2000). However, little attention has been paid so far how application development should be aligned with the new design of a business process. It is not difficult to imagine the consequences of such a lack of alignment. It is shown in this paper that these consequences are indeed encountered in practice.

The contribution of this paper is that it presents a methodology which links the redesign of a business process to the development of information systems that support such a design. The tightness of this link is accomplished by showing how the different proposed models used during the phases of a BPR project are related to each other. Models are placed within an architectural framework, which is a variant on that of Zachman (1999). The heart of the methodology is the Product Based Design (PBD) approach, which can be used to create efficient and effective business process designs (Reijers and Voorhoeve, 2000; Aalst, Reijers

and Limam, 2001). The part of the methodology which is concerned with the actual application development can be classified as component based (CB) (Booch, 1994; Garlan and Shaw, 1993; Szyperski, 1998). A CB approach in systems development has several advantages over more traditional ones, such as a clearer separation of concerns and an increased control of the development project.

Considerable attention within the presented methodology is paid to formal correctness of the derived models and their validation with end users. This is to ensure as much as possible that a new process – once supported by its aligned information systems – will indeed render the performance as envisioned at the start of the BPR project.

The structure of the paper is as follows. First, we will give a frame of reference for the presented methodology. Then, our practical experience with BPR and application development in a recent project for a Dutch bank is described. The methodology is given in some detail, distinguishing several steps and, for each step, its intent and its deliverables. We have included a non-trivial example of developing a new business process in a financial environment and an overview of related work. Finally, we present a summary and the intended extensions of the presented methodology.

Frame of reference

Product Based Design

Product Based Design (PBD) is an approach to derive business process designs in administrative settings. PBD is a prescriptive design method which basically translates a manufacturing concept to the world of administrative processes, such as found in banking, insurance, government, etc. Material Requirements Planning, often referred to as MRP-I, determines the production schedule based on the ordered quantifies, current stock, and the composition of a product as specified in a so-called *Bill-of-Material* (Orlicky, 1972). In other words, production is driven by the structure of the product. With PBD the structure of an informational product, such as a mortgage loan or a social insurance permit, is decomposed into a structure of informational elements which are used to derive a process design. Actual information elements of an administrative product may be related to each other in several ways.

Aalst et al. (2001) describe strategies for the derivation of an optimal process design on basis of the information element structure. The problem of deriving such a design is to select a proper set of production rules and subsequently order them in such a way that an optimal performance with respect to the optimization goals may be expected.

Actual application of PBD has rendered process designs that are radically different from the existing processes they replaced and render flow time reductions of up to 35 % and cost reductions of up to 75 % (Reijers and Voorhoeve, 2000; Crom and Reijers, 2001). Note that an important difference between PBD and traditional approaches is that PBD does not take the existing process as the starting point of the BPR initiative. Rather, it focuses on the very legitimization of the process: the products it should deliver.

An architectural framework

To discuss the use of the PBD deliverables in a system development effort it is useful to distinguish the architecture of an information system first. System architectures focus on the structure of a system, which comprises smaller *components*, the externally visible properties of those components, and the relationships among these components (Shaw and Garlan,

1996). An *architecture* can be defined as the fundamental organization of a system embodied in its components, their inter-relationships, the relations to their environment, and the principles guiding its design and evolution (IEEE, 2000). System architectures are important because they provide descriptions of the system at various levels of abstraction. Hence, system architectures enable various stakeholders to deal effectively with the complexity of a system and to reason about it at various levels of abstraction. Moreover, architectures are a means to integrate the various views of a system (Kruchten, 1995).

We will use an architectural framework that is a variant of the popular information systems architecture of Zachman (1999). Zachman defines an information systems architecture as a set of architectural representations (or models) placed within two dimensions: the *perspective* and the *description type*. The distinguished perspectives agree with the interests of different stakeholders in a system development effort. A manager's view on the system differs from that of an designer, a programmer's view may be completely different from both. Each of the participant's views is, however, relevant to develop the system successfully. We distinguish the following perspectives:

- the *business perspective* distinguishes the purpose of the system in terms of the objectives of the company,
- the *logical perspective* focuses on a logical description of the information system (its functionality) to support the business goals,
- the *technical perspective* is concerned with the software that realizes the desired functionality of the information system,
- the *infrastructural perspective* involves the hardware and general software required for the business-specific software to be executed.

Note that in distinguishing these perspectives we condensed Zachman's original five layers into merely four. In our consultancy practice we experienced that these layers are better recognized by all stakeholders than the complete Zachman framework, which is often too complex for the problem class.

The second dimension of the framework involves the different *types* of descriptions one can make of an information system. These types are applicable for each of the distinguished perspectives. Zachman proposes the universal 'what', 'how' and 'where' questions as the important types. At the same time, he admits that the 'who', 'when' and 'why' question may be just as important, but dismisses them from his framework. We believe that distinguishing the data, the functions, and the process type of descriptions on the one hand and components on the other hand can capture a comprehensive treatment of the most important issues:

- a *data* model involves a description of the relevant entities or objects,
- a *function* model focuses on a description of the involved functions or services,
- the *process* model expresses how data and functions are integrated into an ordered network,
- the *component* model introduces hierarchy and encapsulation and is therefore a mean to reduce and divide the complexity of a system.

Note that unlike the data, functional, and process model that focus on a *single* aspect of the *entire* system, a component integrates *all* aspects of a particular *part* of the system in its environment. Also note the similarities with the ARIS framework (IDS Scheer, 2001).

The complete framework is depicted in Table 1. For each combination of a certain perspective and a type of description, examples of common representations or types of models are given.

	Data	Function	Process	Component
Business	Lists of - customers - suppliers - resources	Lists of - products - services	Business processes overview (purchasing customer care, support processes, etc.)	Company structure: departments (back, mid and front office)
Logical	- class diagram (data) - ER-diagram - product structures - constraints - messages	- class diagram (methods) - data flow diagram - functional specifications	- detailed process model - use case descriptions	- logical component model - logical interaction model
Technical	- database model - file descriptions - message structures (XML / Custom)	- software specifications - component configurations - technical frameworks	Workflow Management System specifications and configurations	- software component model with interactions - interface specifications - distinctions of Software Packages (CRM, ERP, HEM, etc)
Infra-structure	storage capacity	- processor speed - network functionality - I/O-access - performance issues	operating system	Hardware model including - computers - network infrastructure

Table 1. An Architectural Framework

Practical experience

The development methodology as presented and illustrated in the following sections is based on our experiences in a large BPR project. During the years 2000 and 2001, we participated in a project to redesign a major bank's process for handling credit applications of commercial parties. The group to which the bank belongs is a global financial institution of Dutch origin, active in the field of banking, insurance, and asset management in 65 countries with more than 100,000 employees. The process that was redesigned is executed at all its Dutch offices and handles on a yearly basis some 25,000 applications for loans and credit facilities. The project also involved the development of new applications, systems integration with existing applications, and the introduction of a Workflow Management System to support the process execution.

At the start of the project in the beginning of 2000, the PBD methodology was selected for the technical redesign of the process. Simultaneously, the choice was made for a particular software development method. Directly from the start, two project teams were formed that respectively concerned themselves with the process design (the *process team*) and application development/systems integration (the *IT team*). Both teams started off simultaneously. The IT team started to analyze the existing situation with respect to the existing systems and

architecture, while the process team tried to understand the process that was to be redesigned. Subsequent activities of both teams focused on analyzing information requirements within the process, modeling found data dependencies, defining required services of applications, etc. These activities clearly overlapped, although the relations between the delivered models of the separate teams was unclear.

After four months, the process team finished its design and handed it over to the IT team. Some inconsistencies became directly obvious between the needs of the proposed business process on the one hand and the proposed services to be delivered by the new applications on the other. The IT team was given the responsibility of exactly finding and sorting out all the differences. However, the IT team found it difficult to break away from the form and the content of their initial models. As suspicion continued about the correctness and consistency of the various models, members of both teams proposed to build a prototype of the new process, including the rough functionality of the new applications. The general management could, however, not be convinced of the cost-effectiveness of such a prototype. Application development then took off without a clear point of understanding between the involved parties. After half a year of development, applications were build that could not support the needs of the new business process from the perspective of the process team. In response, a different development methodology was selected but this did not improve the quality of the delivered applications.

We identified the following two major issues that caused the troublesome course of the project:

- *the lack of alignment between the process redesign and application development* From the start of the project, it was unclear how the activities and the deliverables of the project teams were linked to each other. Although their perspectives are inevitably different, it was not clear how the various models were related during the entire course of the project.
- *the lack of validation and verification activities early in the project.* No points were introduced early in the project to test and assess the models rendered so far against the expectations of the various stakeholders.

In the next section, we present a methodology that counters these issues.

A development methodology on basis of PBD

The development methodology we propose is outlined in Figure 1. On the left-hand side of the figure from the top to the bottom of the page, the six main steps are listed as boxes with rounded corners. Some of these steps are elementary, while other steps are more complex (i.e., steps two, three, and four). Control steps to verify and validate the deliverables of the main steps are depicted in the middle of the figure as smaller boxes. They are assumed to directly follow up the main step they refer to.

At the right-hand side of the figure, the architectural framework as introduced in Table 1 is depicted several times. For each main step, the shaded boxes in the respective framework representation indicate which aspects and architectural levels are covered by the deliverables of the particular development step. The shaded boxes in Figure 1 illustrate that the presented design methodology involves all architectural levels and aspects that were introduced in Table 1.

Clearly, the emphasis of the methodology is on the design-phase. We will know discuss of the main and control steps of Figure 1 their intent and deliverables.

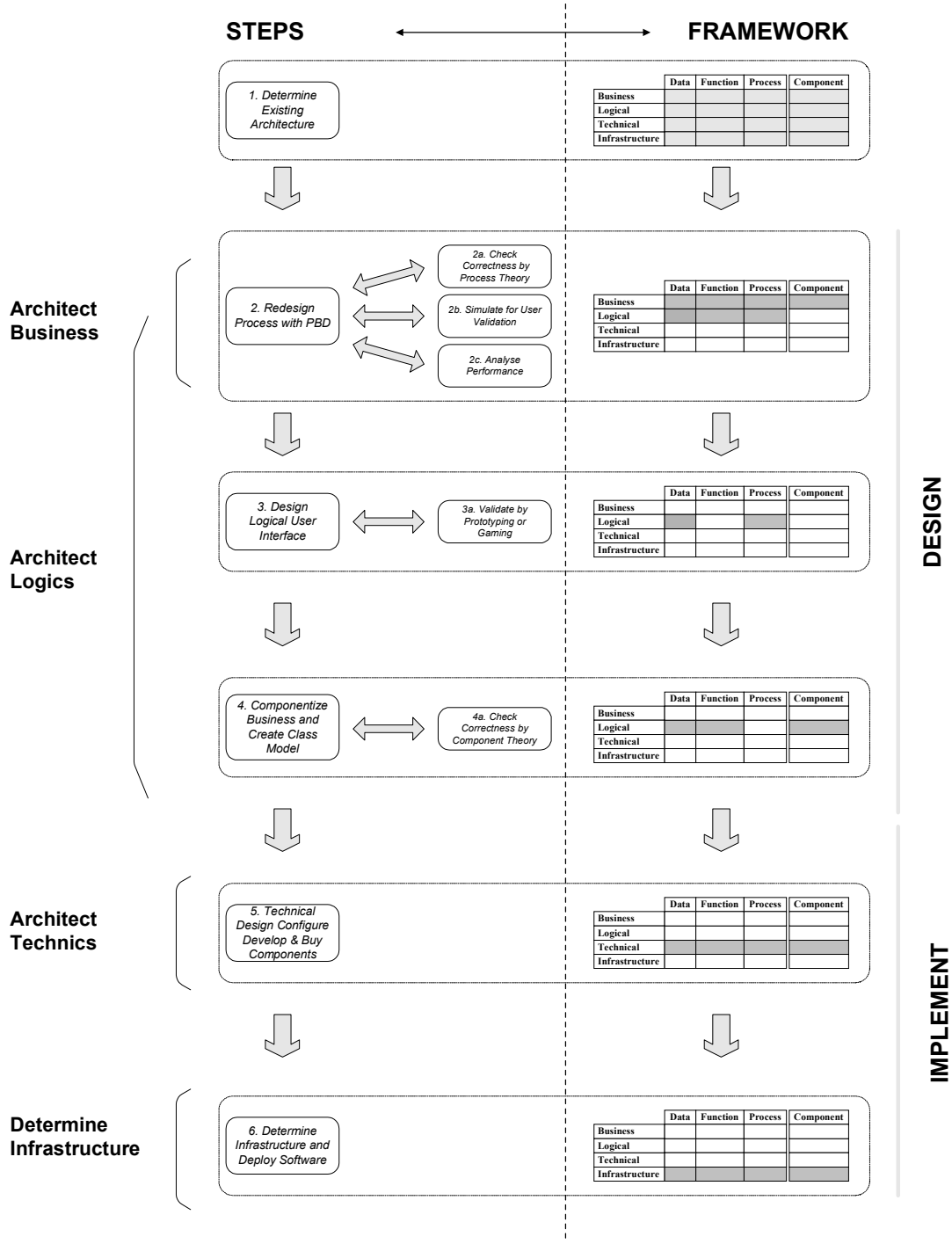

Figure 1. Architectural Design Methodology based on PBD

1. Determine Existing Architecture.

- *Intent:* The challenge of this first step is to analyze the current situation and to describe it in a number of consistent architectural models.
- *Deliverables:* The deliverable of this step is a complete architectural description of the information systems that needs to be changed, as well as its environment.

2. Redesign Process with PBD

- *Intent:* In the second step, a process redesign project is carried out. The goal is a new or an improved business process design. In this methodology, we propose the use of PBD.
- *Deliverables:* Applying PBD requires results in two detailed deliverables: (i) an information element structure including production rules and (ii) a process design. Both models relate directly to the *business* needs, but are also of interest from information system designers.

2c. Analyze Performance

- *Intent:* The performance of the design is determined with respect to performance indicators such as throughput time, service time, waiting time, occupancy rate, etc. Simulation or analytical approaches may be used for this purpose (see e.g. Desel and Erwin, 2000).
- *Deliverables:* If the design is satisfactory, then there are no new deliverables. In case the design is unsatisfactory, the design is altered. Another possibility is that a choice between various alternative designs is made after this step.

3. Design Logical User Interface

- *Intent:* In the third step of the methodology, the logical user interfaces of the systems that will support the process execution are defined.
- *Deliverables:* The design of logical user interfaces may result in an adapted process model, with tasks that are either fused or split up in sub-tasks. Moreover, for each task a logical user-interface is defined that can be used by GUI designers as starting point for further development.

3a. Validate by Prototyping or Gaming

- *Intent:* The purpose of this control step is the user validation of the *content of the tasks* within the process design. This validation step is done by prototyping the new system or by gaming (Guha, Kettinger and Teng. 1993). A methodical way of doing this on basis of PBD is presented by Crom and Reijers (2001), as well as a case description.
- *Deliverables:* The validation step renders an improved process model with respect to the content of the tasks. Also, different grouping of information elements on the user interface may be determined.

4. 'Componentize' Business and Create Class Model

- *Intent:* This main step aims at creating component and class models for the system development effort. They cover the data and function aspects in detail on the logical level.
- *Deliverables:* The specific deliverables of this step are a component model, a class model, a component interaction model, and a cross-reference table. In the first two models, the business data and services are structured into logical entities. A component and class model at the business level define the relation between business components and classes on the one side and all the business processes on the other. The component models contain the following parts: a component structure, a class model, and a life cycle of the component itself (or of its principal class). In the third model – the cross-reference table – the interaction between these structures is described. Also, the relation between the redesigned process and the components is specified. In a cross-reference table, the tasks of the process are listed on the horizontal axis and the methods of the components are listed on the vertical axis. An intersection point of two elements from the list, indicates the usage of the method provided by the component in the respective task.

4a. Check Correctness by Component Theory

- *Intent:* The final control step within the proposed methodology aims at checking the interaction of logical components within their environment. First, the individual

component models that have been created are placed in a global model. In such a model the problem of the coordination of the processes or life cycles among components is addressed. Aalst, Hee, and Toorn (2002) have effectively solved this problem. Secondly, interaction scenarios, i.e., use cases, between components can be executed manually. This gives an insight in the collaboration of components. these steps may lead to improvement of the components and the business process design.

- *Deliverables:* The deliverables of this step are an improved component model, an improved class model, an improved component interaction model and, finally, an improved cross-reference table.

At the end of the fourth phase the design of the system is complete, consistent, and validated by all stakeholders. In the fifth step, the logical components and the process specifications are translated into a technical design. As a final step, hardware and network issues need to be addressed. The contents of both phases depend highly on the technical solutions that are selected. These issues are not within the scope of this article.

A credit facility example

We will illustrate the methodology as presented in the previous section with an example. The example concerns a Credit Facility process which is inspired by an actual application of the methodology. A Credit Facility is a product for existing customers of an imaginary bank.

Analysis of documents, systems, and daily work of employees that have to deal with the credit facility process determine the existing architecture (Step 1). The deliverables are at all levels of the model introduced in Figure 1. At the business level, we have a precise product definition which unambiguously defines the scope of the credit limit service of the bank, a description of the credit facility process, a description of the customers that are served by this process, a description of the services, and a list of departments, persons, roles and functions involved in the credit facility process. At the logical level we have Business Class models with classes concerning important business objects, Data Models including constraints, a detailed process description of the current credit facility process such as currently used within the bank, and descriptions of supportive processes. The technical models are not discussed for reasons of space.

The step 'Redesign Process with PBD' (Step 2) is an interesting part since it is the key of our development methodology. Figure 2 graphically depicts an information element structure (IES) of a Credit Facility. From the IES we learn that customers ought to have a salary account to obtain a credit facility at this bank. In a more formal sense, it specifies e.g. a value for information element *22.type* can be used to determine a value for *10.credit limit*, but that is also used (among other information elements) to determine *6.credit proposal*. In Figure 3 the corresponding *Credit Facility Process* is depicted; this is the other deliverable that results from applying PBD. The process is depicted with the Petri net modeling technique as a workflow net (Aalst, 1998). Milestones are depicted as circles and tasks as boxes. In each task of the process model a link is made to the information elements that are used in that particular task. In each task, the values of a set of information elements are used to produce values for a number of other information elements. In Figure 3, the respective production rules are depicted alongside the tasks. For example, the task *creditability check* incorporates a production rule that determines an outcome of the *12.creditability check* on basis of *29.customer* information. Note that this production rule exists in the IES of Figure 2. The production rule is applicable because the *intake* task has provided the required customer information.

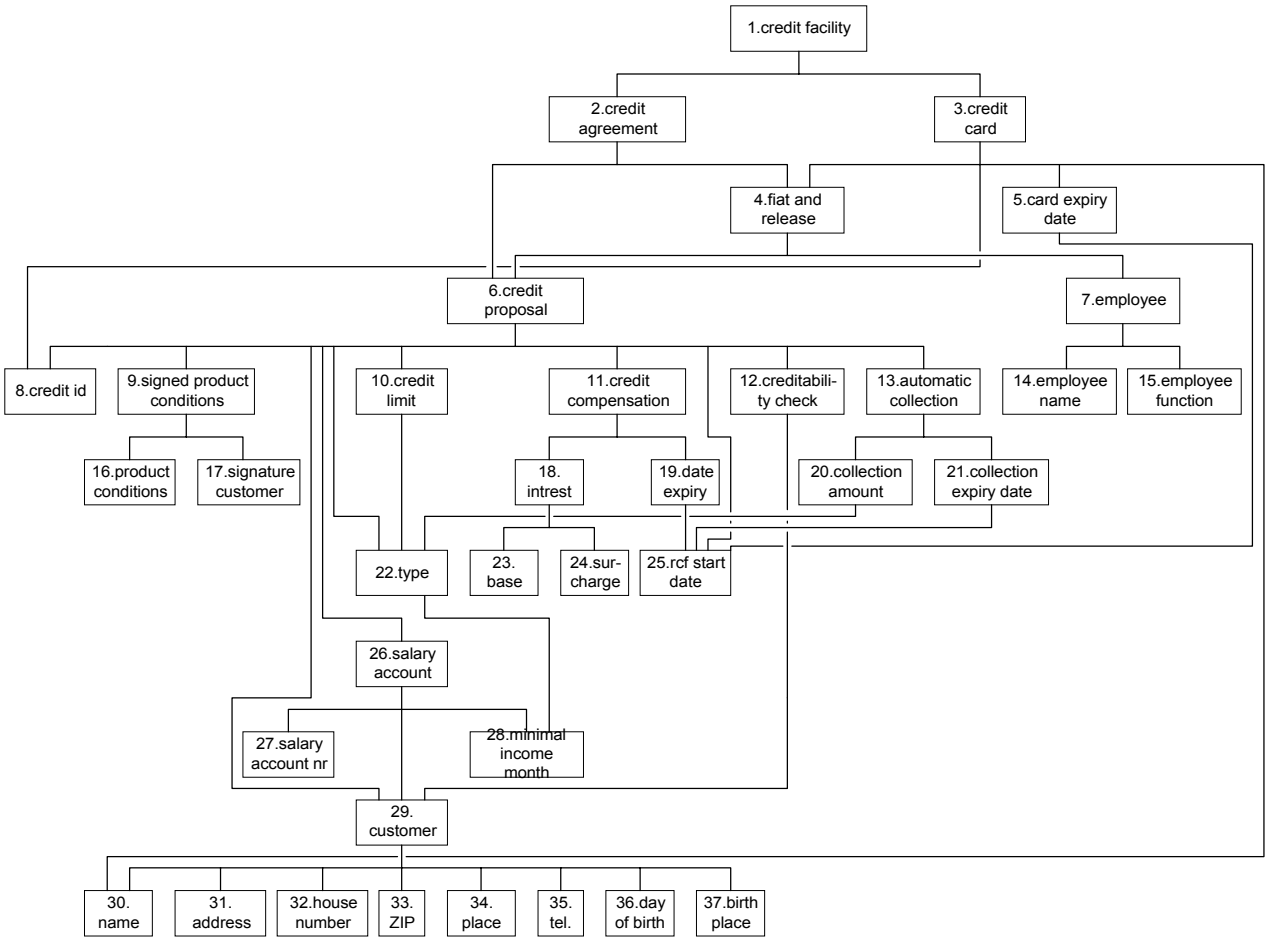

Figure 2. The Information Element Structure of the Credit Facility

The designed process is optimal in the sense that service and response times are minimized. This optimization is obtained by smartly ordering so called *knock-outs,* i.e., decision points between direct termination of the process or continuation (Aalst, 1998). The knock-outs in Figure 3 are just after the salary and creditability check and just after the execution of the fiat. These points are ordered in such a way that early termination of the process at the lowest possible cost is most likely.

Typically a PBD process specification is situated in the architectural model in Table 1 on both the business and the logical level. It is present in the business level since it is easy to interpret from it the flow of the business process at a high level. It is recognizable by end-users and managers. The logical level is involved since each task also contains the production rules for the information elements. Therefore, the process specification can be used to communicate the new process to managers, end-users and business analysts, but also with system analysts and architects. For the latter group not only the process specification but also the IES is of interest, as it is the starting point for the integration of the respective data and functions into a system architecture.

With respect to the logical user interface (Step 3), a single logical window could be made with which an end user could determine the *credit limit, the collection amount, collection expiry date, automatic collection, interest date expiry, credit compensation,* and the *card expiry date*. Note that if information elements of different tasks can be satisfactorily grouped together in

one window, then the structure of the IES tree must still be respected within the window. Grouping of information elements of several small tasks that are either causally independent or are placed in succession can now be fused in the process design. A detailed treatment of deriving logical user interfaces on basis of the PBD deliverables is described by Crom and Reijers (2001).

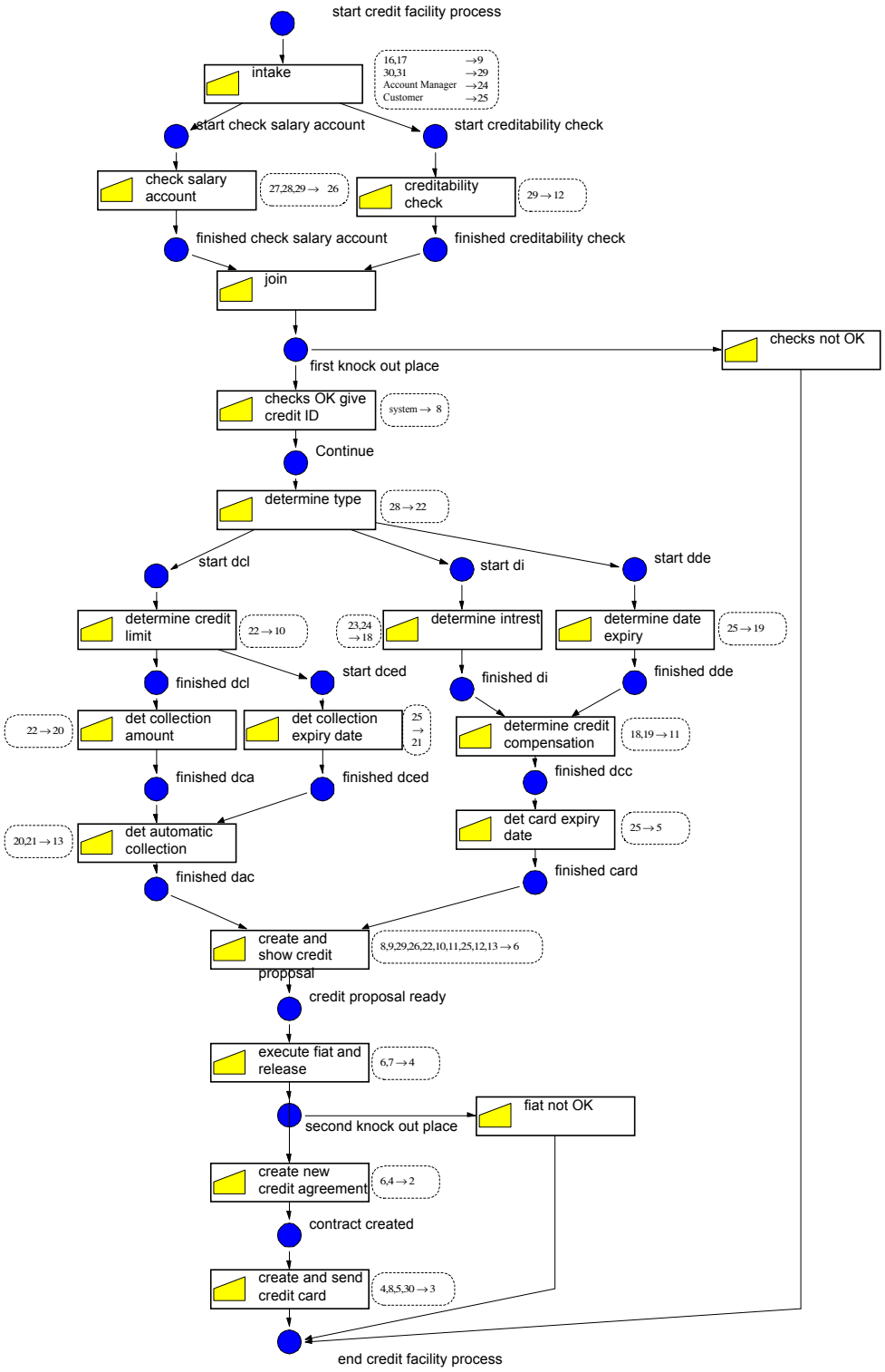

Figure 3: The Credit Facility Process

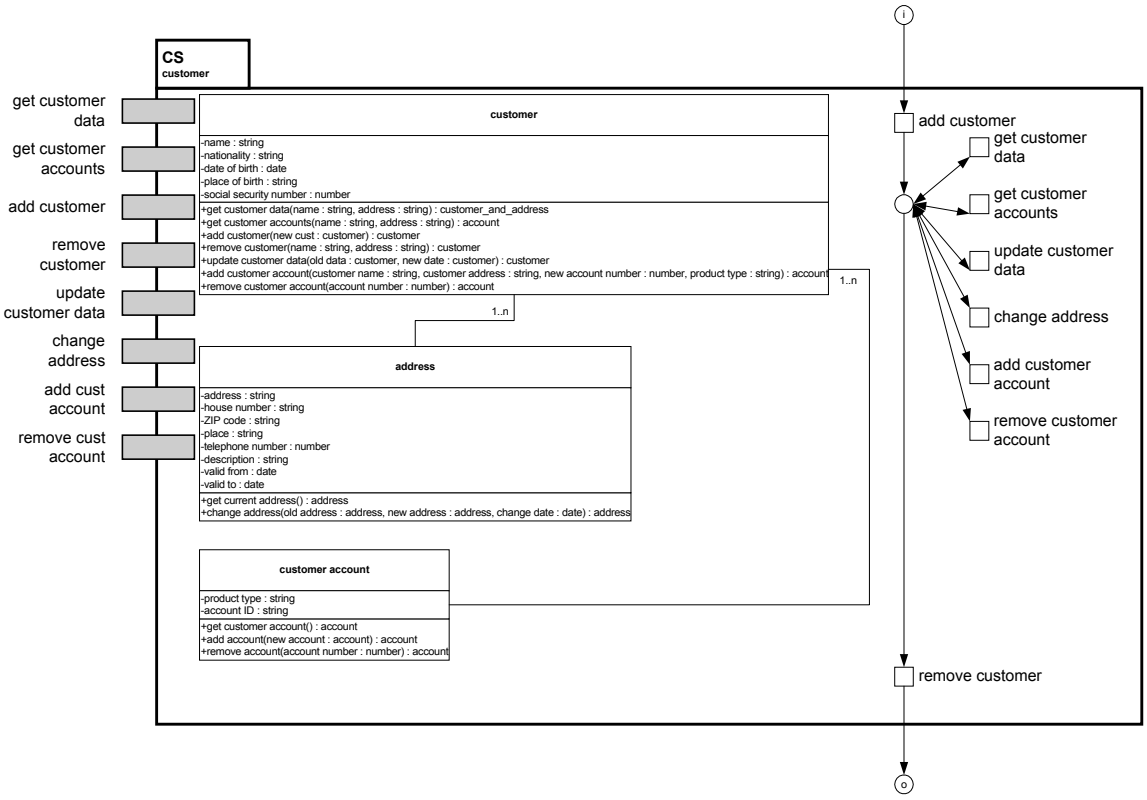

Figure 4. The Customer Component Specification

A component model, such as presented in Figure 4, is created in Step 4 from the deliverables obtained by the previous steps in the following way:

- Firstly, all information elements are mapped onto the necessary business components. The necessity of these components depends on the scope and environment of the information system of which the process is a part. There are two possibilities: (1) business components may already exist, since they are already used for other information systems or (2) they have to be invented.
- Secondly, inside these components these information element are clustered in classes. If necessary, new classes are created.
- Thirdly, operations that are carried out in the tasks of the process model are introduced in the component model and its classes. If possible they linked to already existing methods, otherwise new methods are defined. A method should always be present when there is an information element that occurs in both the task and the class.

An example of the component and class model is given by Figure 4. This figure depicts the *CS Customer,* i.e., the component specification of the customer component. At the left-hand side of the figure the *methods* of the component are depicted. These methods are called directly from tasks within the designed business processes. Therefore, they also occur in the task-component cross-reference table. (This table is important but straightforward; therefore an example is omitted.) Clearly, the methods present at the component interface may be services used by more than one business process alone. Furthermore, each interface method is also attached to a class inside the component. Vice versa, not all methods of classes are methods of the component. Note that there is a distinction between private and public methods, which we will not discuss for the lack of space.

Consider for instance the process task *intake* of the process design in Figure 3. The information element *29.customer* is retrieved by calling the method *get customer data* of the *CS customer* component with the parameters *30.name* and *31.address*. The inner working of the *get customer data* method is hidden for the outside of the component, however the method *get customer data* of the *customer* class uses its link to the *address* class to retrieve from that class the address data of a customer.

Clearly, the inner structure of a component comprises a number of business classes. Typically these classes capture the static structure of the component and a number of operations upon them. Apart from the list of methods components typically have an interface protocol. Such a protocol is an abstraction of the life cycle of a component or of its principal class and is the observable behavior of a component from the outside. In Figure 4 this life cycle is depicted at the right-hand side of the component. Obviously, this life cycle is very basic: a customer can be created and deleted. Only between creation and deletion of a customer object a number of operations can be executed. Including the life cycle of a component into the design can avoid many mistakes.

Related work

On a general level, the relation between information technology and BPR is much investigated. An important distinction in this respect is the one between change technologies and support technologies (Childe, Maull and Mills, 1996). Change technologies support the analysis, modeling, and mapping of existing processes, assessing their efficiency and effectiveness, measuring performance, and providing structured support for the change project's management and associated planning and control functions. Support technologies relate to implementing information systems to support the process configuration needed. Obviously, support technologies are linked to application development; we refer to change technologies when mentioning tools such as ExSpect (Aalst et al., 2000) and Woflan (Verbeek and Aalst, 2000). Various advantages of applying IT within the setting of BPR are recognized, such as cost reduction, time elimination, and error minimization (see e.g. Al-Mashari and Zairi, 2000; Sharp and McDermott, 2001). Gunasekaran and Nath (1997) and Lyons (1997) present overviews of the various types of IT typically applied in a redesigned business processes. Several IT issues are seen to be of influence on the success or failure of a BPR projects, such as an effective IT infrastructure (Al-Mashari and Zairi, 1999; Broadbent, Weill, and St. Clair, 1999) and sufficient technical IT competence (Teng, Fiedler and Grover, 1998).

With respect to the *separate* topic of business process reengineering, several methodologies exist (e.g. Hammer and Champy, 1993; Manganelli and Klein, 1994)). As we mentioned earlier, few BPR approaches exist (e.g. Aldowaisan and Gaafar, 1999; Hofacker and Vetschera, 2001) that are comparable with PBD in aim and depth. The sheer absence of prescriptive methodologies for redesigning business processes is also recognized by Gerrits (1994) and Sharp and McDermott (2001). With respect to application development an abundance of methodologies exist (e.g. Martin (1991); Kruchten (2000).

The main issue of this paper, the lack of synchronization between the application of BPR and IT, is also identified as problematic by Donovan (1994) and Murray and Lynn (1997). Bond (1999) recognizes similarities between models for BPR on the one hand and systems development on the other, without addressing their combined development. Sharp and McDermott, 2001 give an approach that does treat BPR and application development in an integrated manner. However, the relations between the various proposed models is only superficially discussed. Furthermore, it gives no attention to validation and verification. A

more formal treatment of the same subject is given by Hee (1994) and Bruno and Torchiano (1999). Similar to the methodology presented in this paper, formal models are discussed, as well as their relations in defining business processes and mapping them to information systems. In contrast to our proposal, the actual redesign of a business process is not treated. Bruno and Torchiano (1999) take the process design as one of the various views – the function view – on an enterprise. Just as it is possible to take an organization, information, or resource view. (Note that these views are similar to the aspects of Zachman). Because Bruno and Torchiano (1999) do not make the redesign the start point of developing these models, there is also no obvious flow of creation for the various models which we feel is a strength of our proposed methodology. The relations between the formal models as put forward by Hee (1994) are the inspiration for the methodology presented in this paper.

Conclusion

The incorporation of the PBD approach in a component-based systems development methodology enhances the link between a redesigned business process on the one hand and the applications that have to support this process on the other. We have experienced the need for such a methodology in an actual BPR project, but we think that the need is more general. However, we do not think that a tighter formal link will effectively solve all thinkable design and development problems. This is why we emphasize ample room for verifying and validating designs during a BPR effort. The application of verification techniques and the generation of validation instruments is strongly simplified by taking a formal design approach.

Future practical work will be aimed at developing tools that integrate with the presented methodology, to be used for validation purposes. An interesting research direction is the development of standard process and component constructions to facilitate correctness-by-construction. This will considerably reduce verification efforts.

References

Aalst, W.M.P. (1998): The Application of Petri Nets to Workflow Management, *The Journal of Circuits, Systems and Computers*, Vol. 8, No. 1, pp.21-66.

Aalst, W.M.P. (2000): Reengineering Knock-out Processes, *Decision Support Systems*, Vol. 30, No. 4, pp.:451-468.

Aalst, W.M.P., P. de Crom, R. Goverde, K.M. van Hee, W. Hofman, H.A. Reijers, and R.A. van der Toorn (2000): ExSpect 6.4: An Executable Specification Tool for Hierarchical Colored Petri Nets, in *Application and Theory of Petri Nets 2000*, *Lecture Notes in Computer Science* 1825, pp.455-464.

Aalst, W.M.P., K.M. van Hee, and R.A. van der Toorn (2002): Component-Based Software Architectures: A Framework Based on Inheritance of Behavior, *Science of Computer Programming*, Vol. 42, No. 2-3, pp.129-171.

Aalst, W.M.P., H.A. Reijers and S. Limam (2001): Product-driven Workflow Design. In *Proceedings of the Sixth International Conference on Computer Supported Cooperative Work in Design - Ontario, Canada*, pp.397-402.

Aldowaisan, T.A. and L.K. Gaafar (1999): Business Process Reengineering: an Approach for Process Mapping, *Omega: International Journal of Management Science*, Vol. 27, No. 5, pp.515-524.

Al-Mashari, M. and M. Zairi (1999): BPR Implementation Process: an Analysis of Key Success and Failure Factors, *Business Process Management Journal*, Vol. 5, No. 1, pp.87-112.

Al-Mashari M. and M. Zairi (2000): Creating a Fit Between BPR and IT Infrastructure: A Proposed Framework for Effective Implementation, *International Journal of Flexible Manufacturing Systems*, Vol. 12, No. 4, pp.253-274.

Bond, T.C.(1999): Systems Analysis and Business Process Mapping: a Symbiosis, *Business Process Management Journal*, Vol. 5, No. 2, pp.164-178.

Booch, G. (1994): *Object Oriented Analysis and Design with Applications*, Benjamin/Cummings.

Broadbent. M., P. Weill, and D. St. Clair (1999): The Implications of Information Technology Infrastructure for Business process reengineering, *MIS Quarterly*, Vol. 23, No. 2, pp.159-182.

Bruno, G. and M. Torchiano (1999): Making CIMOSA Operational: the Experience with the PrimeObjects Tool, *Computers in Industry*, Vol. 40, No. 2-3, pp. 279-291.

Childe, S., R. Maull, and B. Mills (1996): UK Experience in Business Process Reengineering. URL: http://bprc.warwick.ac.uk/rc-rep-9.html.

Crom, P.J.N. and H.A. Reijers (2001): Using Prototyping in a Product-driven Design of Business Processes, in *Proceedings of the Open Enterprise Solutions: Systems, Experiences, and Organizations Conference, Rome*, pp.41-47.

DeMarco, T. (1979): *Structured Analysis and Systems Specification*, Prentice Hall.

Desel, J. and T. Erwin (2000): Modeling, Simulation and Analysis of Business Processes, in *Business Process Management: Models, Techniques, and Empirical Studies, Lecture Notes in Computer Science* 1806, pp.129-141.

Donovan, J.J. (1994): *Business Re-engineering with Information Technology*. Prentice Hall.

Garlan, D. and M. Shaw (1993): An Introduction to Software Architecture, in *Advances in Software Engineering and Knowledge Engineering*, *Singapore*, pp.1-39.

Gerrits, H. (1994): Business Modeling Based on Logistics to Support Business Process Re-Engineering, in B.C. Glasson, et al., editors, *Business Process Re-engineering: Information Systems Opportunities ad Challenges*, pp.279-288..

Guha, S., W. Kettinger, and J.T.C. Teng (1993): Business Process Reengineering: Building a Comprehensive Methodology, *Information Systems Development*, summer, pp.13-22.

Gunasekaran, A. and B. Nath (1997): The Role of Information Technology in Business Process Reengineering, *International Journal of Production Economics*, Vol. 50, No. 2-3, pp.91-104.

Hammer, M. and J. Champy (1993): *Reengineering the Corporation: a Manifesto for Business Revolution*, HarperBusiness.

Hee, K.M (1994): *Information Systems Engineering: A Formal Approach*, Cambridge University Press.

Hofacker, I. and R. Vetschera (2001): Algorithmical Approaches to Business Process Design, *Computers & Operations Research*, Vol. 28, No. 13, pp.1253-1275.

IDS Scheer (2001): Broad Spectrum of Solutions for the E-organization: ARIS 6. URL: http://www.ids-scheer.com/sixcms_upload/media/169/aris_6_produkt_wp_en.pdf.

IEEE (2000): *Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE Standard 1471*., The Institute of Electrical and Electronics Engineers.

Kruchten, P. (1995): The 4+1 View Model of Architecture, *IEEE Software*, Vol. 12, No. 6, pp.42-50.

Kruchten, P. (2000): *The Rational Unified Process: An Introduction*, Addison Wesley.

Lyons, G. (1997): The Role of Information Technology in Enterprise Re-engineering, *Knowledge and Process Management*, Vol. 4, No. 4, pp.268-277.

Manganelli, R. and M. Klein (1994): *The Reengineering Handbook: a Step-by-step Guide to Business Transformation*, American Management Association.

Martin, J. (1991): *Rapid Application Development*, MacMillan.

Murray, M.A. and M.P. Lynn (1997): Business Process Re-engineering/Information System Development to Improve Customer Service Quality, *Business Process Management Journal*, Vol. 3, No. 1, pp.9-16.

Orlicky, A. (1972): Structuring the Bill of Materials for MRP, *Production and Inventory Management*, December, pp.19-42.

Reijers, H.A. and K. Voorhoeve (2000): On the Optimal Design of Processes and Information Systems: a Manifesto for Product Focus, *Informatie*, Vol. 42, No. 12, pp.50-57 (in Dutch).

Szyperski, C. (1998): *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley.

Sharp, A. and P. McDermott (2001): *Workflow Modeling: Tools for Process Improvement and Application Development*, Artech House.

Shaw, M. and D. Garlan (1996): *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall.

Teng, J.T.C., K.D. Fiedler, and V. Grover (1998): An Exploratory Study of the Influence of the IS Function and Organizational Context on Business Process Reengineering Project Initiatives, *Omega: International Journal of Management Science*, Vol. 26, No. 6, pp.679-698.

Verbeek, H.M.W. and W.M.P. van der Aalst (2000): Woflan 2.0: A Petri-net-based Workflow Diagnosis Tool, in *Application and Theory of Petri Nets 2000, Lecture Notes in Computer Science* 1825, pp. 475-484.

Zachman, J.A. (1999): A Framework for Information Systems Architecture, *IBM Systems Journal*, Vol. 38, No. 2-3, pp.454-470.