Maintainability and control of long-lived enterprise solutions

Oliver Daute Stefan Conrad

SAP Deutschland AG & Co. KG
Business Solution Technology
Homberger Straße 25
40882 Ratingen
oliver.daute@sap.com

Heinrich-Heine Universität Düsseldorf
Institut für Informatik
Universitätsstraße 1
40225 Düsseldorf
conrad@cs.uni-duesseldorf.de

Abstract: Maintainability of long-lived enterprise solutions is today's top challenge. The constantly increasing functionalities of information technology provided these days, including hardware and software products, leads to giant networked application infrastructures. Business processes glue applications and data sources to enterprise solutions to fulfill business needs. Large firms are more and more dependent on the permanent availability of their enterprise solution. A failure in one part of the landscape can rapidly impair the whole enterprise landscape and harm the business. Maintainability is a critical issue and new approaches are required to keep complex landscapes under control at all times. We introduced the *Real-Time Business Case Database*, RT-BCDB as a basic concept. This paper presents the next reasonable step. *Process Activity Control* describes a steering mechanism for complex enterprise application landscapes, to gain more stability, transparency and visibility of processes activities in order to improve maintainability and to support the system administration in their daily operations.

1 Introduction

More transparency and control inside complex application landscapes is required [KMP08] since concepts like cloud computing [Vo08], IT service management [ITIL], architecture frameworks [TOG09] or service-oriented architecture [SOA06] make it possible to build up giant enterprise application solutions. But mechanisms how to manage the underlying technology have been neglected. Long-lived enterprise solutions are subject to evolution [RB09], obsolescence and replacement. We have to find answers to questions regarding system maintenance, troubleshooting, availability, change management and the integration of new components.

Maintainability and control of complex software solutions is today's top challenge.

This paper presents mechanisms for increasing the maintainability and control of complex application landscapes. *Process Activity Control* (PAC) is the next step after the introduction of the *Real-Time Business Case Database* [Da09].

PAC concentrates on the control of business processes which are currently active within an application landscape. The goal is to avoid indeterminate processing states which can cause further incidents within the application environments.

Most enterprise or service frameworks are focused on business requirements which have improved the design of enterprise solutions significantly but often with too little consideration for the underlying information technology [Ro03]. Operation interests are neglected and little information about how to run a designed enterprise solution can be found. A single business process is able to trigger process activities across the whole landscape, uses different applications, servers or exchanges data. If a server fails or a database system stops its processing, then several business processes can be impaired (Fig. 2). In such situations it is quite difficult to determine the cause or impact on other process activities of the enterprise solution [KMP08].

The challenge for the system administration is to manage these complex application environments and to react as swiftly as possible to incidents [SW08]. Several tools are available to monitor large enterprise landscapes. Some of them just monitor the operating or the application level. Only few tools collect details about business processes and try to control them. But each tools has its own proprietary view to applications. There is no overall concept for heterogeneous application environments available.

Numerous business cases within an enterprise solution make it impossible to be aware of what each single business process does. The system administration must be able to analyze business processing like a technical issue. Thus, business processes are seen as objects which have a run-state and are requesting to run. This eliminates the need to collect information about what a specific business process does. The focus on essential information also standardizes business processes and eliminates the need to identify its run-state in the case of an incident. System administration can react more purposefully and the designers of business processes get detailed information about the behavior of their business cases.

Complexity leads to longevity of software solutions for several reasons; here we need only to think of done investments, cross application dependencies and efforts to exchange software products. This paper discusses the challenges and circumstances in complex enterprise solutions and why it is so difficult to control business processes. We will show how to set up a steering & control mechanism within the application environments and will discuss the benefits for the system management administration.

For the communication with PAC, we will introduce a C*ode of Business Processes*. It is a valuable step forward in defining the frame to identify business processes.

Finally, the use of PAC in collaboration with the RT-BCDB will be presented, preceded by a short introduction to it.

2 Background & Terminology

RT-BCDB stands for *Real-Time Business Case Database* and it is an approach to collecting and providing information about business process activities in heterogeneous application landscapes. RT-BCDB aims to improve the transparency of activities.

The knowledge acquired supports the administration during maintenance activities and is an important source of information for the business designers as well. RT-BCDB supports tasks, like updates, recovery, planning or optimization of the time schedules.

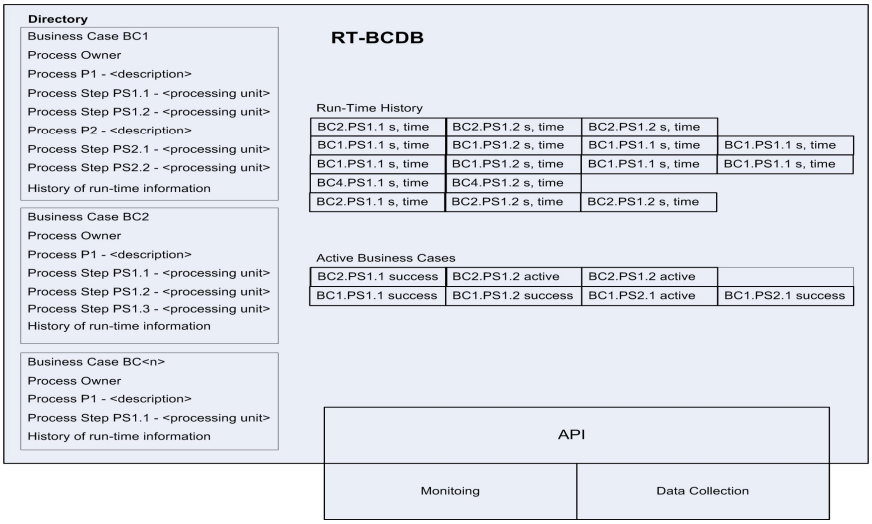

Figure 1: Real-Time Business Case Database

A system failure requires detailed investigations about the impact on business process activities before a system recovery can take place. This means processes which were impaired must be identified and must be included in the recovery process. The data of RT-BCDB is important during the analysis and restoration phase in order to bring back business processing to the latest consistent state. It is also indispensable for monitoring, reporting and changing the landscape.

RT-BCDB stores data about business cases, processes, owners, history of previous processing, execution frequencies, run-time, dependencies, as well as availabilities of processing units and applications. Knowledge about run-states of business processes is important information for maintaining and controlling business processes.

The term *enterprise solution* encapsulates a collaboration environment of hardware and software technologies with the common purpose of providing an infrastructure for business processes. A solution can consist of ERP software, various legacy systems, data warehouses, middleware for exchanging data and connecting software applications. Other expressions are *application landscape* or *application environment*.

Business cases are designed by the business requirements and needs. A business case consists of several processes which can be performed on different systems. Business cases determine the tasks of the customer's enterprise solution.

A *business process* consumes data or provides them and can be triggered by other processes or services. Business processes make use of different applications and data sources across an application landscape with regard to the enterprise needs.

System maintenance activity relates to ongoing tasks on the system administration level. Typical maintenance activities are updates, incident analysis, recovery, replacement and the like. (RT-BCDB supports active or reactive maintenance activities).

3 The Idea

Process Activity Control is required because of the continuously increasing complexity of long-lived enterprise solutions, driven by evolution, availability, growth, business requirements, modern tools and enterprise application framework methods [TOG09]. IT administration has to manage these solutions in any situation. New mechanisms are required to assist them.

The constantly increasing complexity of enterprise solution is the number one cause of cost-intensive system failures [EIU07].

Incidents interrupt business processes while they are performing a task. The malfunction of a processing unit or of an application can cause process activities to fail. Processes need to be restarted in order reach data consistency on business process level.

The idea of PAC is to minimize uncontrolled failure of business processes and reduce the amount of incidents. If problems within the application landscape are already known, e.g. a database stopped processing, then there is no reason for a process to start with the risk of halting in a failure situation. PAC acts proactively and thus avoids disruptions.

PAC also addresses another currently unsolved problem: the start and stop process of a complex application landscape or parts of it. It is still a complex matter to shutdown a single application without the knowledge of dependent business cases and without impairing the business. At the moment, there is no outer control for business cases available. Whenever a shutdown is required, PAC identifies active business processes to stop and will avoid the start of further business cases.

Business processes are triggered or started by different activators. A process can cause different activities across the whole application landscape and exchanges data. Various automated control instances start and stop processes, too. But there is no overall concept for heterogeneous application environments.

The figure (Fig. 2) depicts a well-known situation in application environments without control of processes. Uncontrolled failure of business processes may result in an unknown run-state or data inconsistencies on business level.

From the perspective of a business case, a consistent state requires more than data integrity on database level. Also dependent interfaces or single process steps must be taken into consideration. Those can halt in an inconsistent state anywhere in an application environment.

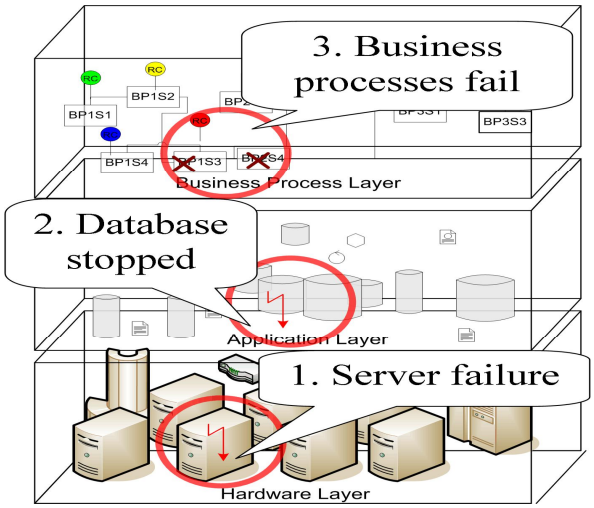

Figure 2: Failure within the application environment

PAC works as a steering mechanism for business processes and is especially valuable in the control of core business processes. To interact with business processes, PAC makes use of RunControl commands. We will discuss these later. PAC is able to collect run-state information of business processes and send them to RT-BCDB.

4 Code of Business Processing

Various situations arise in complex application landscapes because a form of identification for business processes is missing. These are not easy to handle or to overcome in case of incidents. Therefore, identification is required for business processes in the communication with PAC. The Code of Business Processing, CoBP covers some minimum requirements.

Traffic laws are simple and effective. They are necessary to control and steer the traffic within a defined infrastructure, the traffic network. Traffic laws and networks together describe a kind of code of conduct which participants have to accept in order to participate in the traffic. It is an appropriate steering mechanism for a complex environment. This example of a regulating mechanism gives us an idea of what is needed for application landscapes.

We will try to translate some elements of traffic laws and network into a code for business processes.

We propose a *Code of Business Processing* which contains general rules and requirements for using an application environment. The code should only be applied to processes which are of significance for the enterprise solution itself. That means only processes with a high impact in case of a failure have to meet the code. The CoBP consists of rules and requirements:

First CoBP: Each process must have a unique form of identification. This is required to identify and steer a process while it is active. Identification is needed to follow the tracks (footprints) during the processing lifetime and to refer to the process owner. Process IDs are unique across all applications to avoid false identification. A process ID can also be used to classify. The most important business processes have a distinctive ID.

Second CoBP: Each business process must be documented. It must belong to a business case and visualization must exist. A diagram contains the run-states a business process can run in. Procedures must be given for recovery purposes in case of a process failure.

Third CoBP: Each process must have a given priority. On the one hand it is important to decide which process should be given priority and also to determine a sequence of processing. On the other hand it is useful to mark processes regarding the impact on the application environment. The business processes with higher priority must process first, unless PAC decides differently.

Fourth CoBP: The higher a priority is the higher is the charge for a business process. Certainly, accounting and charging for processes is not very often done at customer site. But a process with a high priority does have a significant impact on all other processes that run within that environment.

For our purposes the first two rules are sufficient to control activities. Additional rules can be added. Ideally, communication between processes should always take place in traceable ways. As a result of this our last CoBP follows.

Fifth CoBP: Business Processes should use defined and traceable ways for interaction. This forces the use of known and open interfaces. The CoBP improves the traceability of business processing significantly and supports the maintainability. For existing application landscapes, the 5th CoBP requires reviewing the process interaction.

The rules of CoBP enable better steering of business processes, provide more transparency and thus improve maintainability.

5 Process Activity Control

Process Activity Control is an approach to controlling process activities in complex application environments. PAC will stop further business processing in case of problems occurring. This will prevent business processes running into undefined processing states and avoid further incidents.

PAC has to consider several issues in order to control process activities. A major task is, for instance, determining the function state of business processes, applications and processing units or the functioning of PAC itself. This can be done by RT-BCDB.

RT-BCDB uses agents to collect information about an application landscape. On the hardware and application level, agents search for a specific pattern to determine the function state and availability.

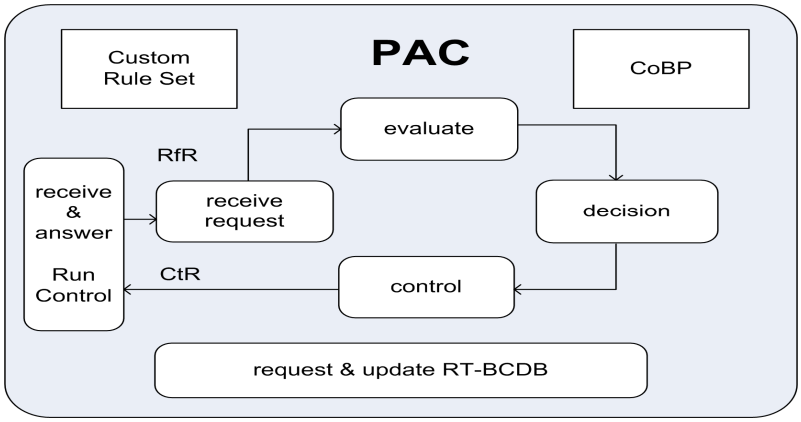

Figure 3: Architecture of PAC

Applications fork operating system processes, which can be monitored as well to identify throughput. A premature termination of an application process on operating systems can point to the failure of an application. The agents inspect the given sources of information and try to identify run-state and availability information. This information is used by PAC to react to current circumstances. PAC will try to avoid any starting or triggering of business processes which will make use of a malfunctioning processing unit or impaired applications.

For smaller software environments information about the availability of hardware and software application is sufficient enough to control and to avoid further incidents. But only few details about business process activities are visible.

For large application landscapes, such as long-lived enterprise solutions, PAC must also be informed of run-states of business processes. This information is provided by the knowledge base of RT-BCDB.

PAC consists of several basic elements: a decision-control mechanism, a Custom Rule Set, the CoBP, an interface to RunControl, and a communication interface to RT-BCDB. The decision-control mechanism is subdivided into four main activities: *receive request, evaluate, decision* and *control*. Each activity has one or more tasks.

Activity '*receive request'*, just receives the *Request for Run* (RfR) in sequence of income. Whenever a business process starts or stops or changes its run-state, then RunControl will send an RfR. The RfR contains the process ID and the state of running.

The activity '*evaluate'*, evaluates the RunControl request against the information stored in RT-BCDB. The run-state table of RT-BCDB always reflects the status of process activities within the application environment. Any known problems with the availability of applications or processing units are taken into consideration.

The '*decision'* process uses CoBP, Custom Rule Set, RT-BCDB and the evaluation of the previous activity. A final decision will be prepared to return a '*Confirmation to Run* (CtR)' or to stop a business process.

The '*control'* activity is the steering process of PAC. It has two functions. The first is to answer the RfR and to send a CtR. In the case of a business process having to be paused, the *control* process waits to send the CtR until problems are solved. The second function is to stop business processes in the case of the application landscape having to be shut down. Vice versa '*control'* enables the start-up of business cases in a predefined sequence, for instance after maintenance activities or after the elimination of incidents.

The *Custom Rule Set* contains rules given for a customer's application landscape. The rule set can contain an alteration of priorities or a list of business cases which have to run with a higher priority. Also preferred processing units can be part of the rule set. Another element is CoBP, which was described previously. Finally, an application interface, which is used to communicate with RT-BCDB, is also part of PAC.

In operation, PAC as a control instance must monitor its own availability. At least two instances of PAC must run within the environment. This is necessary to prevent PAC is becoming *single-point of failure* for the enterprise solution. In normal operation the second instance should be used to answer RfR.

6 Run Control

PAC introduces an extension to *RunControl* [Da09] commands. RunControl commands are used to receive information about processes and required for controlling the progress of their activities. Whenever a business process starts, stops or waits, the RunControl command will send a message with the process ID and the run-state. The information will be stored immediately in the run-state table of RT-BCDB. Due to this, active business cases and their status can be made visible (Fig.4).

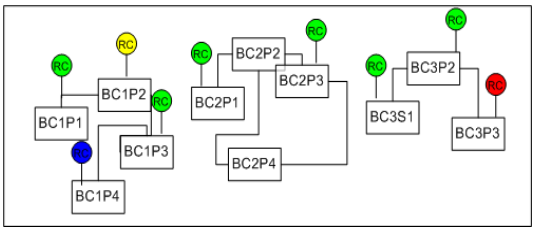

Figure 4: Run-States of active business cases

We adapted the 'RunControl' statement for use with PAC. Instead of sending the run-states information using the agents, this information is send to PAC immediately. PAC forwards the information to RT-BCDB. On the business process layer, PAC takes over the tasks of the agents. The second change is an extension of functionality. The RunControl function waits until it receives a '*Confirmation to Run*' from PAC. To distinguish between the two versions of RunControl statements, we will use **RunControlAC** when using PAC.

```
RunControlAC(BC1.PS1, "active")
call function PurchaseOrder;
if  return code <> 0 then
    call ExceptionHandling;
    RunControlAC(BC1.PS1, "exception")
end
RunControlAC(BC1.PS1, "successful")
```

Figure 5: Example for RunControl in source code or transparent layers

Several options are given to implement RunControl statements (Fig. 5). An option is to insert RunControl statements into the source code [UML] [Sv07]. For existing applications adaptations are possible during migration projects [St06]. Reverse engineering might be an appropriate discipline too. Consequently for the future design of business solutions, applications should be developed with regard to run-state information or the RunControl statements.

Instead of modifying the source code, processes can also be called up from a transparent layer in which functions are embedded in RunControl statements (Fig.6). This option is much easier to implement and to introduce for heterogeneous landscapes.

Certainly, some effort is needed for the implementation of the RunControl. But with the constantly increasing complexity of application landscapes, a mechanism as described is indispensable for keeping a long-lived enterprise solution under control.

7 Improving Maintainability and Control

The aim of our concepts is to gain more transparency of and control over process activities. To increase the overall availability, e.g. by the prevention of cost-intensive incidents, is one aim more to be listed. We will discuss next by means of a simple example, how PAC improves the maintainability of complex application landscapes.

An application stops its processing (Fig. 6). PAC recognizes this problem and stops further processing of business cases which will make use of the application. Here, a single business case is impaired and has to be stopped in order to prevent data inconsistencies. If a standby application is available, PAC can move the processing to it.

In the case of performance bottlenecks, PAC is able to stop a business process in order to prevent a problem from getting worse or shift an RfR to another application unit if possible.

Although users have to wait until the problem is solved, indeterminate processing states will be avoided. These can cause further incidents and additional efforts for solving.

PAC makes use of RT-BCDB knowledge to decide a '*Request for Run*'. If incidents to databases, applications, processing units or business cases are known, then PAC will determine if a '*Request for Run*' will make use of them. The run-state and availability information, stored in RT-BCDB, provides a virtual image of activities within the application infrastructure.

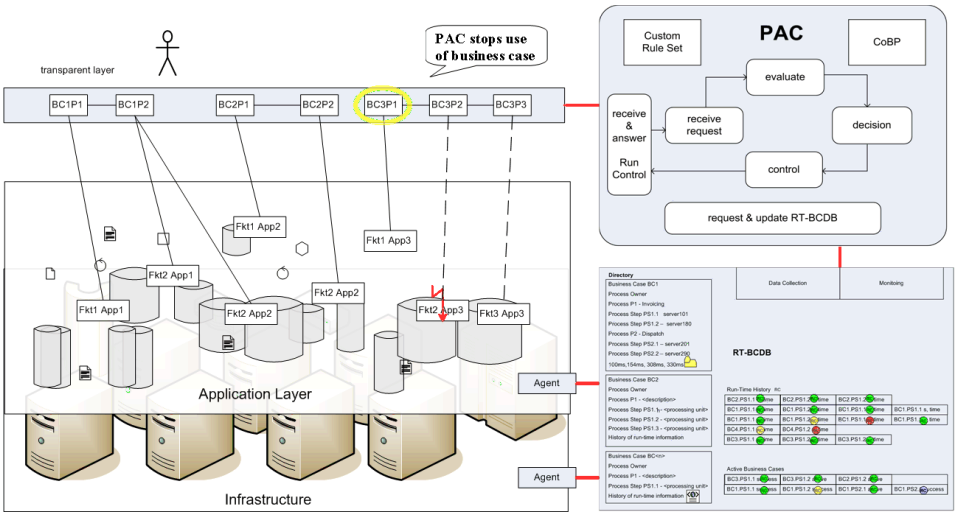

Figure 6: Avoiding indeterminate run-states and inconsistencies

Maintenance tasks like updates or upgrades of the applications also require detailed information about the business cases possibly involved. The constant availability of long-lived enterprise solution requests to maintain the landscape while numerous business processes are running. PAC can prevent process activities while parts of the application landscape are under construction. RT-BCDB provides the dependencies.

How to measure improvements in terms of *Return on Investments*? We will try to answer this question with regard to time, money and quality.

Time: Each incident which was prevented saves time. Identifying the cause of and solution to an incident cost time. Additional time is needed for reporting and documenting the solution, and several persons of different departments are involved. Users are hindered in their work and will lose time. PAC prevents cost-intensive incidents and improves the overall-availability of the enterprise solution.

Money: Costs are incurred by incident handling, software for incident tracking and support staff. Downtimes can cause less productivity and can result in fewer sales. An overall estimation is difficult to provide. In the worst case, especially in the area of institutional banks, an incident can cause bankruptcy within a few days [EIU07].

Quality is often not easy to measure. In enterprise application landscapes quality means availability, reliability, throughput, maintainability and competitiveness. Quality can save money but can increase costs. The introduction of PAC and RT-BCDB requires investments. But more visibility and transparency lead to more purposeful reactions to incidents within the landscape. This saves time.

And fewer incidents, however, increase the quality and the availability of an enterprise solution. We assume that for large environments the investment in regard to the increase in quality will save money in the end. In smaller environments our concept will at least improve quality. We expect that PAC & RT-BCDB reduce the TCO [Ga87], but further investigations are required to determine the quantity and quality of improvements.

8 Evolution and Obsolescence

Long-lived enterprise solutions are subject to evolution, obsolescence and replacement. Typical, for long-lived solutions is that the interaction between software components last longer than the components themselves. Therefore knowledge about activities is essential to maintain the landscape and to support the change management process.

Evolution is driven by new requirements, growth and replacements of hardware or software, including general maintenance. Evolving a long-lived application solution can be difficult. Lack of information about dependencies of process interactions, the way the solution works and the integration of new components are some examples. Moreover the enterprise solution requires close to 100% availability. RT-BCDB is oriented to the information about the processing-unit, business cases and process activities. PAC supports the maintenance process while the enterprise solution is in use.

Innovation can lead step by step to obsolescence of applications of the enterprise solution. Often, so called legacy systems are difficult to exchange for some reasons. Problems are missing documentations, designers left the organization and too little information about the purpose and frequency of usage. Our concepts can explain their usage and improve the replacement. They are able to detect monolithic applications. A constantly decreasing usage can indicate obsolescence.

As we pointed out, most frameworks are focused on business requirements and neglect the maintenance interest. Thus, insufficient transparency in operation and missing control mechanism are the results to name only some of the gaps. Maintainability and availability (incl. avoidance of cost-intensive incidents) are the main really challenging tasks. But also recovery is an interesting issue to ensure data consistency across large application landscapes. Our concepts are steps forward to improving maintainability and control of long-lived software-solutions.

9 Conclusion

The concepts presented support maintenance tasks and the evolution of long-lived enterprise solutions. System administration can react more purposefully. PAC improves the control over business process activities, avoids incidents and makes better utilization of resources. This leads to a higher availability and a better quality of the complex application landscape. RT-BCDB is the knowledge basis and provides transparency; an image of process activities and their dependencies.

Certainly, some work needs to be done to implement such concepts, especially if software applications need to be adapted. But, with the constantly increasing complexity of enterprise solutions, mechanisms as described are indispensable for keeping long-lived enterprise application landscapes maintainable in the future.

References

[Da09] Daute, O.: Introducing Real-Time Business CASE Database, Approach to improving system maintenance of complex application landscapes, ICEIS 11th Conference on Enterprise Information Systems, 2009

[Da04] Daute, O.: Representation of Business Information Flow with an Extension for UML, ICEIS 6th Conference on Enterprise Information Systems, 2004

[EIU07] Economist Intelligence Unit: Coming to grips with IT risk; A report from the Economist Intelligence Unit, White Paper 2007

[Ga87] Gartner Research Group: TCO, Total Cost of Ownership, 1987, Information Technology Research, www.gartner.com

[ITIL] ITIL, IT Infrastructure Library, ITSMF, Information Technology Service Management Forum, www.itsmf.net

[KMP08] Kobbacy, Khairy A. H.; Murthy, D. N. Prabhakar, 2008, Complex System Maintenance Handbook, Springer Series in Reliability Engineering

[PH07] Papazoglou, M.; Heuvel, J.: Service oriented architectures: approaches, technologies and research issues, Paper, International Journal on Very Large Data Bases (VLDB), 16:389–415, 2007

[Ro03] Rosemann, M: Process-oriented Administration of Enterprise Systems, ARC SPIRT project, Queensland University of Technology, 2003

[SW08] Schelp, J.: Winter, Robert: Business Application Design and Enterprise Service Design: A Comparison. In: Int. J. Service Sciences 3/4 (2008)

[SOA06] SOA: Reference Model for Service Oriented Architecture Committee Specification, www.oasis-open.org, 2006

[St06] Stamati, T: Investigating The Life Cycle Of Legacy Systems Migration, European and Mediterranean Conference on Information Systems (EMCIS), Alicante Spain 2006

[Sv07] Svatoš, O.: Conceptual Process Modeling Language: Regulative Approach, Department of Information Technologies, University of Economics, Czech Republic, 2007

[RB09] Riebisch, M; Bode, S.: Software-Evolvability, GI Informatik Spektrum, Bd 32 4, Technische Universität Ilmenau, 2009

[TOG09] TOGAF, 9.0: The Open Group Architecture Framework, Vendor- and technology-neutral consortium, The Open GROUP, www.togaf.org, 2009

[UML] UML: Unified Modeling Language, Not-for-profit computer industry consortium, Object Management Group, www.omg.org