Activity Control in Application Landscapes

A Further Approach to Improving Maintainability of Distributed Application Landscapes

Oliver Daute and Stefan Conrad

SAP Deutschland AG & Co. KG, University of Düsseldorf, Germany
oliver.daute@sap.com, conrad@cs.uni-duesseldorf.de

Abstract. The system administration has been waiting for a long time for procedures and mechanism for more control over process activities within complex application landscapes. New challenges come up due to the use of linked up software applications to implement business scenarios. Numerous business processes exchange data across complex application landscapes, for that they use various applications, retrieve and store data. The underlying technology has to provide a stable environment maintaining diverse software, databases and operating system components. The challenge is to keep the distributed application environment under control at any given time. This paper describes a steering mechanism to control complex application landscapes, in order to support system administration in their daily business. *Process Activity Control*, PAC is an approach to get activities under central control. PAC is the next reasonable step to gaining more transparency and visibility to improving system maintenance of Cloud Computing environments.

Keywords: Cloud computing, complex application landscape, distributed infrastructure, process activity control, RT-BCDB, Code of business process.

1 Introduction

More transparency and control inside complex application landscapes is required [6] [9] since concepts like Cloud Computing [17], client-server architectures, service-oriented architecture [12], or IT service management [5] make it possible to build up giant networked applications environments. New mechanisms are required to ensure maintainability, evolution and data consistency in order to support the operation of the underlying distributed information technology. Cloud Computing infrastructures require control, virtualization, availability and recovery of their applications and data.

Process Activity Control (PAC) is the next step after the introduction of the *Real-Time Business Case Database* (RT-BCDB) [1]. The concept of PAC concentrates on the control of processes activities which are currently running within an application landscape. The goal is to avoid indeterminate processing states which can cause further incidents within a Cloud environment.

Most enterprise or service frameworks are focused on business requirements which have improved the design of enterprise solutions significantly but often with too little

D.R. Avresky et al. (Eds.): Cloudcomp 2009, LNICST 34, pp. 83–92, 2010.

consideration for the underlying information technology. Operation interests are neglected and little information about how to run a designed enterprise solution can be found. A sequence of application processes (e.g. a business case) is able to trigger process activities across the whole landscape, uses different applications, servers and exchanges data. The challenge for the system administration is to manage these complex Cloud environments and to react as swiftly as possible to incidents [11].

The missing outer control mechanism is the fundamental idea for *Activity Control in application landscape*. Activity Control is an approach to having power over processes in order to reduce incidents, to gain more stability and to improve maintainability. PAC and RT-BCDB are able to improving the system administration in Cloud application environments significantly.

2 Terms and Areas of Discussion

The term *RT-BCDB* [1] 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. In RT-BCDB information about run-states of active business processes are collected and stored synchronously. This information supports the system administration during maintenance activities of complex application environments and is an important source of information for the business designers as well. In detail, RT-BCDB stores information about business cases, business processes, process owner, history of previous processing, execution frequencies, runtime, dependencies and availabilities of processing units and applications. Knowledge about run-states of business processes is important for maintaining and controlling processes and applications [1].

A *Cloud computing environment* or *application landscape* or *application infrastructure* can consist of 'simple' applications, ERPs, legacy systems, data warehouses, as well as middleware for exchanging data and connecting software applications. Clouds are complex distributed application landscapes.

A *business case* combines *(cloud) applications* and describes a sequence of activities to fulfill specific tasks. Business cases make use of different applications and databases across a landscape with regard to the enterprise needs. A *business (application) process* consumes data or provides them and can trigger other processes or services. Processes which have a high importance, such as invoicing, are called *core business processes*. An *enterprise solution* is built up of several software components and information sources. It is designed by the business requirements. Business cases determine the tasks of the customer's enterprise solution.

3 The Idea

Process Activity Control is required because of the continuously increasing complexity of application landscapes driven by business requirements, modern tools and enterprise application frameworks which make it more comfortable to design enterprise application solutions [8]. The challenge for the IT administration is to manage these application environments in any situation. New mechanisms are required to assist the system administration in their work.

Frequently, incidents within application landscapes interrupt business processes while they are performing a task. The malfunction of a processing unit or of an application can cause business processes failure. Business processes need to be restarted or rolled back for completion to reach a consistent state within the business data logic. The increasing complexity of software solution is the number one cause of system failures [3].

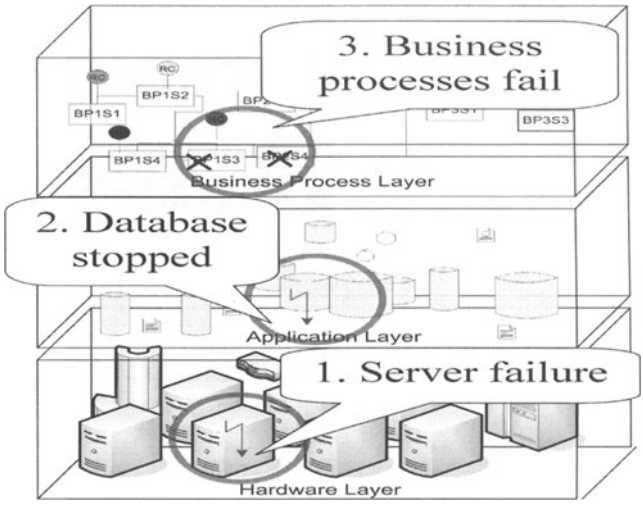

Fig. 1. Failure within the application environment

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

PAC also addresses another unsolved problem: the start and stop process of an application landscape or parts of it. It is still a challenge and complex matter to shutdown an application without the knowledge of dependent processes running within the environment. Business processes are triggered by different activators. At the moment, no outer control for business case in Cloud application environments is available.

The figure depicts a well-known situation in application environments without process control. When a server fails, all applications and database used to run on this processing unit will fail too. Business processes using these applications and databases will be impaired and must terminate immediately. In application environments without PAC this uncontrolled failure of business processes may result in unknown run-states or data inconsistencies.

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

state anywhere in an application environment. The challenge is to avoid these inconsistencies. The basis for this is the knowledge about business processes, dependencies, availabilities and run-states information. Our goal is to support the system administration in their work.

PAC works as an outer control mechanism for processes and is especially valuable in the control of core business processes. To interact with application processes, PAC makes use of RunControl commands. PAC is able to collect run-states and send them to RT-BCDB. PAC works best in collaboration with RT-BCDB.

4 Code of Business Processing

Various situations arise in distributed application landscapes because of missing form of identification. These are not easy to handle or to overcome in case of incidents. For activity control we propose a *Code of Business Processing*, CoBP. This code 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.

Traffic laws are simple and effective. They are necessary to control and steer the traffic within a defined infrastructure. Traffic laws describe a kind of code of conduct which participants (road users) have to accept. It is an appropriate mechanism for a complex environment with easily learnable rules. We will try to translate some elements of traffic laws and network into a code for business processes used for complex application environments.

First CoBP: Each process must have a unique form of identification. This is required to identify a process and to steer the process while it is active. **Second**: Each process must have a given priority. The higher business processes must process first, unless PAC decides it differently. **Third**: Each business process must be documented. It must belong to a business case and visualization must exist. Procedures must be given for recovery purposes in case of a failure. **Fourth**: The higher a priority is the higher the charge for a business process. A process with a high priority does have a significant impact on all other processes that run within that environment.

Ideally, communication between processes should always take place on traceable ways. **Fifth CoBP:** Business Processes should use defined and traceable ways for processing. This forces the use of known interfaces, improves the traceability and supports the maintainability of Cloud application landscapes.

5 Basic Elements of PAC

Process Activity Control is an approach to controlling process activities in complex Cloud application environments. PAC is aware of the function states of processing units and applications. PAC will stop further processing in case problems occur within the application environment. This will prevent business processes running into undefined processing states.

PAC has to consider several issues in order to control process activities. A major task is, for instance, determining the function state of processes, applications and processing units. PAC can take advantage of the agents introduced with RT-BCDB.

The tasks of the agents are dependent on the kind of source of information. The agents inspect the given sources and try to identify run-state and availability information. On the hardware and application level, agents can search for a specific pattern in a log file to determine the function state. Application processes on operating system can be monitored as well to identify availability or throughput. A premature termination of an application process may point to a failure.

For smaller environments this mechanism provides information which is sufficient enough to control process activities. For large application landscapes PAC must also be informed of run-states of business processes. Therefore PAC will benefit when using the knowledge base of RT-BCDB.

The information is used to react to current circumstances within the application environment. PAC will try to avoid any starting of processes which will make use of a malfunctioning processing unit or impaired application or process or service.

Basic elements of PAC: a decision-control mechanism, a Custom Rule Set, the CoBP, an interface to RunControl, and a communication process to RT-BCDB.

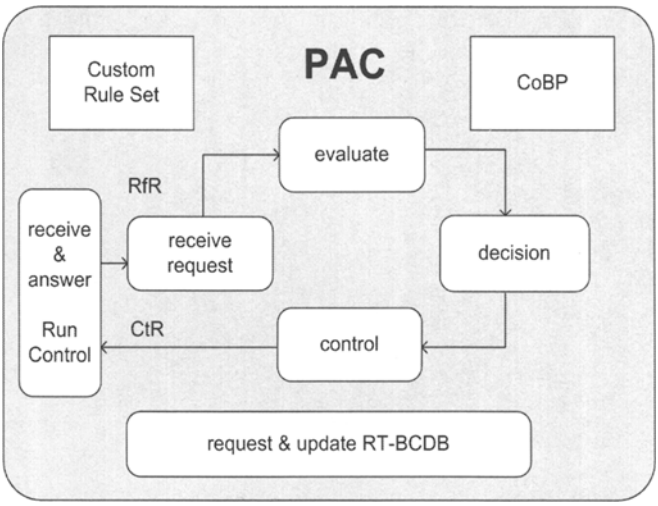

Fig. 2. Architecture of PAC

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 an application process starts or stops or changes its run-state, then RunControl will send an RfR. The RfR contains the ID and the state of running.

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 is an activity based on CoBP, Custom Rule Set and the evaluation of the previous activity. A final decision will be prepared to return a '*Confirmation to Run* (CtR)' or to stop or to halt a business process or application.

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

Further basic elements are CoBP, described previously and the application interface which is used to communicate with RT-BCDB.

PAC as a control instance must monitor its own availability. Therefore at least two instances of PAC must run within the application environment. This is necessary to prevent that PAC is becoming a *single-point of failure* for the application infrastructure. One instance of PAC is the master instance and the second is functioning as the backup instance. If PAC detects a malfunction with its master instance then it passes control to the second instance. In normal operation the second instance should also be used to answer RfR. This makes sense for the distribution of workload of PAC and will avoid delays in the steering of business process activities.

6 Run-Control

PAC introduces an extension to *RunControl* commands. RunControl commands are used to receive information about process run-state. They are also required for controlling the progress of process activities.

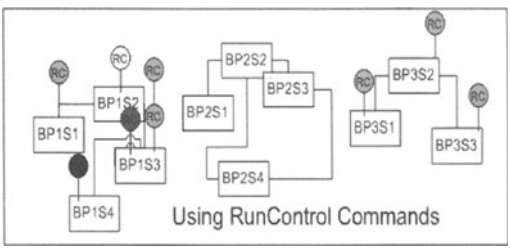

Fig. 3. Collecting Run-State

Whenever a process starts, stops or waits, the RunControl command will send a message with the process ID and the run-state. RunControl statements were first introduced with the architecture of RT-BCDB. There, RunControl statements are used to collect run-states and to store them immediately in the run-state table of RT-BCDB. Due to this an overview of current process activities is available at any time.

Several options are given to implement RunControl statements. One option is inserting RunControl statements into the source code. This makes sense especially for newly designed applications [2] [16] [14]. For existing applications adaptations are

possible for instance during migration projects [13]. For sure, reverse engineering should be the preferred discipline to enrich the resource code with RunControl statements.

PAC adapts the concept of RunControl statements to its needs. The first change is to the business information flow [2]. Instead of sending run-states information using the agents, the RunControl statements send this information to PAC. PAC forwards the information to RT-BCDB. The second change is the extension of functionality. Each RunControl statement sends, in addition to run-state information, a *'Request to Run'*. 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 an extended version for PAC and call it **RunControlAC**. The RunControlAC commands send the business case ID, run-state and an RfR.

```
RunControlAC(process-ID,run-state)
```

Fig. 4. RunControl for Activity Control

Certainly, some effort is needed for implementation of the RunControlAC. But with the constantly increasing complexity of Cloud application landscapes, a mechanism as described is indispensable for keeping distributed infrastructures under control. Consequently for the future design of business solution, applications should be developed with regard to run-state information or RunControl statements.

7 Improving System Maintainability

The aim of the concept is to gain more control over Cloud applications, as well as the prevention of incidents.

An example depicts how PAC is able to avoid incidents due to known problems. A failure of a server (processing unit) occurs and therefore an installed database must stop its processing. PAC recognizes this problem and stops further processing of business processes using the failed unit. Two business cases requesting to run are stopped by PAC and avoid indeterminate processing states. The application processes have to wait until the problem is solved. If a shadow database is available, PAC can move business processing to it.

PAC will make use of RT-BCDB information to decide the confirmation of a 'Request for Run'. If incidents to applications, processing units or business cases are known, then PAC will determine if a 'Request for Run' will make use of them. The run-states and availability information, stored in RT-BCDB, provides this important knowledge, as well as dependencies within the application infrastructure.

How to measure improvements in terms of *Return of Investments*? Some benefits are already shown and we will try to answer this question with regard to time, quality or money. We will start with time.

Time: Each incident which was prevented saves time. An incident costs time to identify the cause and time to solve. Additional time is needed for reporting and

documentation of the solution process progress, and several persons of different departments are involved. Users are hindered in their work and will lose time. We assume that each incident costs in sum an average of 6 hours.

Money: Costs arise due to incident handling, software for incident tracking and support staff. Downtimes can cause less productivity and can result in fewer sales. In the worst case, especially in the area of institutional banks, an unsolved incident can cause bankruptcy within a few days [3].

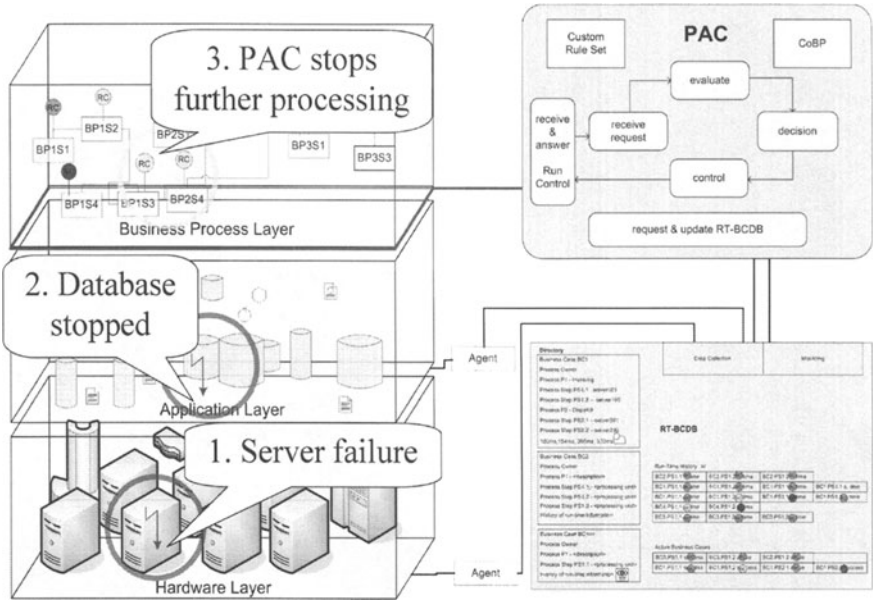

Fig. 5. Avoid indeterminate run-states

Quality is often not easy to measure. For Cloud application landscapes quality means availability, reliability, throughput and competitiveness. 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.

Maintenance tasks like updates or upgrades of the Cloud landscapes also require detailed information about the business processes possibly involved. PAC can prevent business process activities while parts of the application landscape are under construction. In case of performance bottlenecks, PAC is able to stop a business process in order to prevent that a problem from getting worse. Or PAC decides to shift an RfR to another Cloud application if possible. These are examples of how PAC is able to improving the maintainability of a Cloud application landscapes.

8 Extensions for Frameworks

Most enterprise or service frameworks are focused on the business requirements and neglect the operation interest. Concepts like SOA [12], IT Service Management [5] or

TOGAF [15] improve the design of application solutions but often with too little consideration for the underlying information technology. Business cases can be created easily by orchestrating services (composite application). But there is no information how to control them. No mechanisms are described how to react to problems within an application landscape. An active steering process is also not part of the frameworks. PAC is able to extend these frameworks and can reduce the TCO [4] significantly.

Virtualization, in the sense of representation, is one of the enablers of Cloud Computing infrastructures. Servers are pooled together acting like a large computing resource. Virtualization is the basis for new application platforms for managing distributed computing resources efficiently. Also process activities and their representation must be taken into consideration as presented in this paper. The goal is to gain more transparency and control over processes in order to reduce cost-intensive incidents and to avoid data inconsistencies on business process level.

Computing Clouds and the concepts, as mentioned above can benefit from the ideas of PAC & CoBP & RT-BCDB for gaining better maintainability and higher availability of an application landscape.

9 Conclusion

Maintenance and control of constantly increasing complexity of Cloud Computing environment are challenging tasks. New mechanisms as described are indispensable for keeping a distributed application infrastructure maintainable in the future.

PAC is a concept for gaining control, higher availability and better visibility of activities within Cloud application environments. Application processes will run into fewer incidents. The system administration can react more purposefully due to better transparency.

PAC is a further step to getting distributed application infrastructures landscapes under control. The concept works best in collaboration with the RT-BCDB [1]. Our ideas should encourage future research to invest more on these topics [7].

References

1. Daute, O.: Introducing Real-Time Business CASE Database, Approach to improving system maintenance of complex application landscapes. In: ICEIS 11th Conference on Enterprise Information Systems (2009)
2. Daute, O.: Representation of Business Information Flow with an Extension for UML. In: ICEIS 6th Conference on Enterprise Information Systems (2004)
3. Economist Intelligence Unit: Coming to grips with IT risk, A report from the Economist Intelligence Unit, White Paper (2007)
4. Gartner Research Group: TCO, Total Cost of Ownership, Information Technology Research (1987), `http://www.gartner.com`
5. ITIL, IT Infrastructure Library, ITSMF, Information Technology Service Management Forum, `http://www.itsmf.net`
6. Kobbacy, Khairy, A.H., Murthy, Prabhakar, D.N.: Complex System Maintenance Handbook. Springer Series in Reliability Engineering (2008)

7. Mei, L.: More Tales of Clouds: Software Engineering Research Issues from the Cloud Application Perspective. In: 33rd Annual IEEE International Computer Software and Applications Conference (2009)
8. 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)
9. Rosemann, M.: Process-oriented Administration of Enterprise Systems, ARC SPIRT project, Queensland University of Technology (2003)
10. Sarkar, S., Kak, A.C., Nagaraja, N.S.: Metrics for Analyzing Module Interactions in Large Software Systems. In: The 12th Asia-Pacific Software Engineering Conference, APSEC 2005 (2005)
11. Schelp, J.: Winter, Robert: Business Application Design and Enterprise Service Design: A Comparison. Int. J. Service Sciences 3/4 (2008)
12. SOA: Reference Model for Service Oriented Architecture Committee Specification (2006), `http://www.oasis-open.org`
13. Stamati, T.: Investigating The Life Cycle Of Legacy Systems Migration. In: European and Mediterranean Conference on Information Systems (EMCIS), Alicante Spain (2006)
14. Svatoš, O.: Conceptual Process Modeling Language: Regulative Approach, Department of Information Technologies, University of Economics, Czech Republic (2007)
15. TOGAF, 9.0: The Open Group Architecture Framework, Vendor- and technology-neutral consortium, The Open GROUP (2009), `http://www.togaf.org`
16. UML: Unified Modeling Language, Not-for-profit computer industry consortium, Object Management Group, `http://www.omg.org`
17. Vouk, M.: Cloud Computing – Issues, Research and Implementations. In: Proceedings of the 30th International Conference on Information Technology Interfaces (ITI 2008), pp. 31–40 (2008)