An Aspect Oriented Method to Model and AnalyzeSelf-Recovery of Cloud Computing

Xiaoyun Zhang[1,2,3], Chanle Wu[1,2],Xue Su[4]

[1] School of Computer,Wuhan University,Wuhan 430072, Hubei,China

[2]National Engineering Research Center for Multimedia Software, Wuhan 430072, Hubei,China

[3] Wuhan Institute of Posts and Telecommunications Sciences, Wuhan 430074, Hubei,China

[4] Wuhan Railway Vocational College of Technology, Wuhan 430073, Hubei,China

Index Terms: Failure recovery; service composition; aspect orientation; Petri nets

Abstract. Cloud computing has attracted much interest recently from both industry and academic. More and more Internet applications are moving to the cloud environment. This paper proposes an aspect oriented method to model and analyze self-recovery of cloud application according to its characteristics. Petri nets are used as the formal description language for cloud application, and use it to describe its basic elements, such as, cloud module, resource service, physical machine, virtual machine, etc. Aspect oriented programming method is used to extract self-recovery process as the core and crosscutting concerns. On this basis, the self-recovery approach is presented, the related theories and tools of Petri nets are used to verify the correctness of proposed method. Experiments show that the proposed solutions have better performance in supporting the self-healing Web service composition.

Introduction

With the exponential growth of cloud computing as a solution for providing flexible computing resource, more and more cloud applications emerge in recent years. How to manage the failure in the cloud application has became an urgent and crucial research problem. Due to cloud application can be subject to the unexpected failures. It is difficult for the developers to plan recovery strategy at the design time and account for all the dynamic changes during execution. The failure of participants will lead to the cancelation of the whole process [1]. The existing protocol has offered a throw mechanism to deal with errors, but failures are handled and possibly recovered in a static way by employing pre-compiled compensation strategies and they over constrain the composition process making it rigid and not able to handle the dynamically changing situations. While many network applications such as financial services, online transactions or e-commerce are running in an unpredictable environment, and they require certain self-recovery ability. If the requests can’t be met, they will cause the loss of customers, economic loss, etc. Therefore, it requires that the cloud computing must have self-recovery ability, which means that cloud computing can heal itself if any execution problem occurs, in order to complete its execution successfully and meet the users’ constraints.

Consideration of these requirements in developing cloud computing is essential. In this paper, we make research on how to model and analyze self-recovery of cloud application based on the actual requirements. Petri nets are used to model for cloud module, resource service, physical machine, virtual machine, etc. We formally model the basic relationship between cloud modules, the interaction between physical machine and virtual machine, aspect oriented programming method is used to extract self-recovery process as the core and crosscutting concerns. And weaving rules are used to dynamically integrate these models into a whole model of cloud application. The effectiveness and feasibility of constructed model are analyzed based on the operation semantics and related theories of Petri nets. A case study demonstrates the approach can contribute to improvement of design quality, and has important scientific significance and value in developing highly reliable cloud application.

Modeling Cloud Computing

The function of cloud application is composed of a number of independent cloud module which is realized by virtual machine. Virtual machine can operate and migrate between physical machine based on the actual resource service and business process. The cloud module and physical machine has their attributes, such as, execution time, reliability, the required resource, etc. The system can control the execution and invocation of device by physical machine, and there is a certain logical relation between cloud modules. The classification of tasks which considers three different types of transaction properties: retry-able *r*, compensable *cp*, pivot *p*. Naturally, a service can combine properties, and the set of all possible combinations $\{r, cp, p, (r, cp)\}$.

The requirement of cloud computing is 7-tuple: $\Xi=(C, WS, CT, RL, TW, RT, RC)$. Where *C*, *WS* represent the set of tasks and available services; RL: $C \text{ x } C \rightarrow \{>, +, \|, cp, pre\}$ is the relation function between components, where >, +, *k*, *cp*, *pre* represent the sequence, choice, parallel, compensation and preference relationship. TW: $C \rightarrow WS^*$ is the available services of component; $RT : WS \rightarrow R \text{ x } TP$ is the *QoS* function of service, where *R* is the real number in (0,1), $TP = \{r, p, cp, \{r, cp\}\}$ is the transaction attributes of service; RC: $C \rightarrow \{v, nv\}$, *v*, *nv* represent the failure and non failure component.

The requirement of cloud computing is mainly used to explain the execution process of cloud computing, physical machine, virtual machine and their basic attributions.

A. Syntax and semantics of model

Based on the concept of *AOP*, the requirement of cloud computing is the core concern, while the failure recovery process is viewed as the crosscutting concerns, which includes the failure warning concern, service selection concern and recovery concern. The model is divided into the composition net, advice net, introduction net and aspect net.

Definition 1: A 7-tuple $\Sigma = (N, IO, \lambda,)$ is called Base net (*BN*) model if: $N = (P, T, F, D, A_T, A_F, M_0)$ is a colored Petri net, *P*, *T*, *F* represent the place, transition and arc; *D* is the non-empty finite individuality set. f_D, f_S is the predicate set and symbol set. $A_T : T \rightarrow f_D$, for $t \in T$, the free variable in $A_T(t)$ must be the free variable in the directed arc with one end of the arc is t. $A_F : F \rightarrow f_S$, if $(p, t) \in F$ or $(t, p) \in F$, then $A_F(p, t)$ or $A_F(t, p)$ is the n-symbol set, the default value is empty. $M_0 : P \rightarrow f_S$ is the initial marking of Σ. $IO \subset P$ is a special type of place. λ: $T \rightarrow N^*$ is the priority of transition t_i, the default value is 0, the smaller of λ, the greater of transition's priority. $\forall x \in (P \cap T)$, we denote the pre-set of x as ${}^\bullet x = \{y | y \in (P \cup T) \wedge (y, x) \in F\}$ and the post-set of *x* as $x^\bullet = \{y | y \in (P \cup T) \wedge (x, y) \in F\}$. $\forall x \in (P \cup T)$, the input/output arc of t_i and the free variable in $A_T(t_i)$ are denoted by $FV(t_i)$.

Definition 2: A 6-tuple $\Omega = (\Sigma, \Gamma, TI, TA, PI, PA)$ is called Composition Net model. Where Σ is a *BN* model for the basic structure of Ω. $\Gamma = \{\Gamma_i \mid i \in N\}$ is the finite set of page. $TI \subset T$ is the set of substituted node. TA: $TI \rightarrow \Gamma$ is used to allocate the page to the substituted node. $PI \subset P$ is the set of interface node. *PA* is the mapping function of interface, which maps the interface into the input and output of the page. The individuality set *D* is mainly used to describe the resource service in cloud computing. In this paper, we abstract the element as the individuality $d_i=(it, i, RW_i)$, where $it \in \{w, f, d\}$ represents the described object of individuality, *w*, *f*, *d* represent service, fault and data packet respectively, I represents the position of individuality. Let $k\#(d_i)$ be the *k*th element of individuality d_i. While the common data packet is abstracted as individuality φ, and all individualities in *CN* model are φ. In order to distinguish the transition and place in each net, the element *x* in net N_i is denoted by $N_i{\bullet}x$, and the corresponding component C_i, service $WS_{i,j}$ and fault fa_i are denoted by individuality $d^c{}_i$, $d^w{}_{i,j}$, $d^f{}_i$.

Definition 3: The mapping $CutN$:$\{N_i{\bullet}p_j, N_m{\bullet}p_n, \ldots, N_f{\bullet}p_k\}$ is called the point-cut of the system, where *CutN*, $N_i{\bullet}p_j$ are the name and join-point of point-cut. A triple $Asp = (CutN, AN, IN)$ is called an aspect of cloud computing, where *CutN*, *AN*, *IN* represent the point-cut, advice net and introduction net. All the feasible replacement of *t* under *M* are denoted by set $VP(M, t)$. If $VP(M, t) \neq \Phi$, then *t* is enable, denoted by $M[t >$. The t_i is effective under marking M if and only if t_i is enable and it doesn't exist transition in $ET(M)$ whose priority is greater than t_i. The process that *M* reaches *M'* by firing a feasible replacement $t_i < d_1, d_2, \ldots, d_n >$ is denoted by $M[t_i < d_1, d_2, \ldots, d_n >> M'$. All the concurrent transitions under marking M are denoted by set $MT(M)$. The set $H(M)=\{t<d_1, d_2, \ldots, d_n>|t \in MT(M), t<d_1, d_2, \ldots, d_n> \in VP(M, t)\}$ is called the greatest concurrent set of M. If there exists firing sequence $H_1, H_2, \ldots, H_k$ and state sequence $M_1, M_2, \ldots, M_k$, which make $M[H_1>M_1[H_2>M_2 \ldots M_{k-1}[H_k>M_k$, then M_k is a reachable state from M. All the possibly reachable states of M are denoted by $R(M)$ and $M \in R(M)$. All the firing sequences from M to M_k are denoted by set $\delta(M, M_k)$.

B. Modeling corn concern

The base net of component C_i is shown in Fig.1, the available service set of C_i is WS_i. If $M(p^I_i) \neq \Phi$, then firing t_{in} to choose the individuality x from p_w as the invoked service and begin to p_a. t_e is the successfully operate. If the component doesn't have the retriable service, and its available services have failed, then output the fault info. We can compose the BN model of each component based on their relationships. The steps of constructing base net - are: (1) Constructing the BN model of component based on its constructed method; (2) Introducing t_s, t_e , p_s , p_e to describe the beginning and termination of whole application.

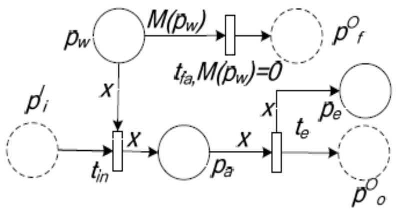

Fig 1 Base net of Component C_i

Failure Recovery Rules and Its Analysis

According to the characteristics of cloud computing, the failure of service and component can be divided into: (1)Failure of available service, which is caused by the failure of available service; (2) Failure of component, there is no available service can be called or all the available services of component have failed; (3) Failure of the operating environment, which refers to the connection between two components has interrupted, it makes the communication between components operate abnormally.

According to the different function, we can divide the failure recovery process into three concerns: failure warning concern, service selection concern and recovery concern.

(1) Failure warning concern: The system will give failure type and position when the composition process fails. So the failure warning concerns include two cut-points: p_{fc1}: $\{CN_i \bullet p_a \mid C_i \in C\}$, p_{fc2} : $\{P^i_f \mid C_i \in C, RL(C_i, C_j) \in \{cp\} \vee RC(C_i) = nv\}$, the introduction net is shown in Fig3(a)-(b). Cut-point p_{fc1}, p_{fc2} are used to process the failure of available service and component, where place p_{fw} is used to store the failed service. Transition t_{fw} is used to describe the failure of service, the firing probability is 1-$SP_{i,j}$. The specific weaving rules are: the system will weave p_{cf1} into the CN model of all components, while p_{cf2} will be woven into the base net, the priority of the introduced transition is set to 0, that is, it has the highest priority. Let the model of component C_i be CNF_i after weaving the failure warning concern.

(2)Service selection concern: Cut-point f_{cw}: $\{CN_i \bullet F(p_w, t_{in}) | C_i \in C\}$, the introduction net is shown in Fig.2(c). Transition t_{fc} is used to store the available service that meets the conditions to place p_{ews}, then the components can select the services. The weaving rules are: for each component, the system will introduce the cut-point and its' afterward element, the priority of introduced transition is set to 0.

(3)Recovery concern: it is mainly used to coordinate composition process with different handling rules according to the type and position of failure. Therefore, the cutting of recovery concern is based on the failure warning concern. Because retry rule has been included in the modeling process of component, the recovery concern mainly refers to theignore, replacement, compensation and preference handing rule, the cut-points are: t_{ig}, p_{tr}, $\{p_{tc1}, p_{tc2}, p_{tc3}\}$, p_{tf}, the definition is: t_{ig} : $\{CNF_i \bullet t_{fa} \mid C_i \in C, RC(C_i) = v\}$, p_{tr} : $\{CNF_i \bullet p_{wfc} \mid C_i \in C\}$, p_{tc1} : $\{CNF_i \bullet p_e \mid C_i \in C$, cp $\in M(C_i \bullet p_e)\}$, p_{tc2} : $\{P^i_f \mid C_i \in C, RL(C_i, C_j) = cp \cap RC(C_i) = v\}$, p_{tc3} : $\{p_{wf} \mid RC(C_i) = nv\}$, p_{tf}: $\{P^i_f \mid C_i \in C, RL(C_i, C_i) = pre\}$. The introduction net is shown in Fig.2(d)-(k). Cut-point p_{ti} describes the handling process of ignore rule:

If the component can fail, the system will invoke transition t_{fa} to ignore the failure of component and output the individuality to P^{O}_{o} when the component fails, then the system can continue to operate. Cut-point P_{tr} is used to describe the handling process of compensation rule, the system will compensate for the invoked service if component C_i has got the compensation command (p^{I}_{tr} has token) and the corresponding service has the transaction attributes of compensation. Cut-point p_{tf} is used to describe the process of preference rule.

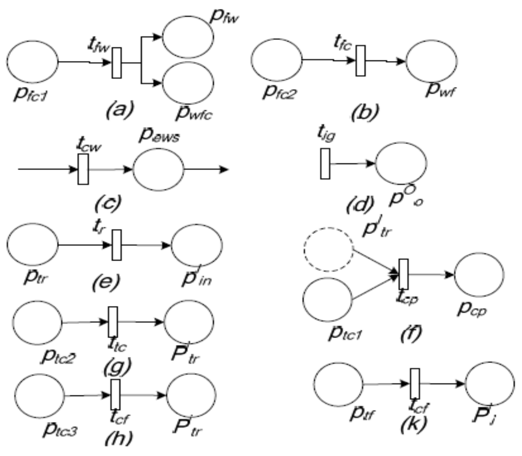

Fig 2 Modeling Concerns

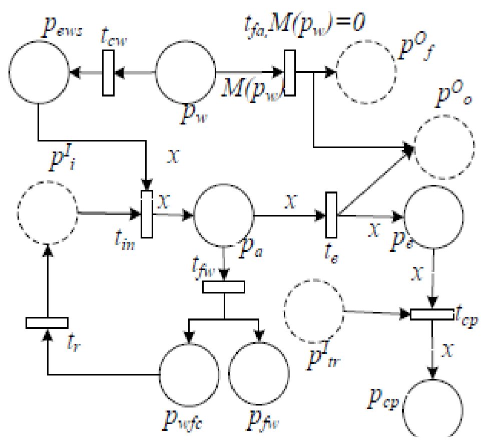

Fig 3 Failure recovery model of component Ci

Based on the definition of cut-point, advice net and introduction net, we will give the weaving rules of each concern:

(1) If $RC(C_i) = v$, then inserting the point-cut p_{ti} into the base net CN_i, otherwise, the system will insert the point-cut p_{tr}, p_{tc1} into the base net CN_i; (2) If $\exists C_j \in C$, $RL(C_i, C_j) = cp \cap RC(C_i) = v$, then inserting the point-cut p_{tc2} into the base net CN_i; (3) If $\exists C_j \in C$, $RL(C_i, C_j) = pre$, then inserting the point-cut p_{tf} into the base net CN_i, if $RC(C_i) = nv$, then inserting the point-cut p_{tc3} into the base net CN_i. Let Ω_f, Ω_c, Ω_t be the model got by weaving the failure warning concern, service selection concern, recovery concern into the base net, and the model got by weaving three concerns into the base net is called failure recovery model Ω_s, which is shown in Fig.3. Let M be a marking, $M(p_e) = \varphi$, then M is a normal termination marking, that is, it has realized the required function.

Theorem 1: $\forall C_i \in C$; $\forall M \in R(M_0)$, if $M(C_i{\bullet}p^{O}_{fa}) = \varphi$, and $M_f \in R(M) \cap FT(M_f) = \varphi$, δ is a firing sequence from M to M_f, $\forall C_k \in C(k \neq i)$, there are:

(1) If $RC(C_i) = v$:$(RL(C_i, C_k) = cp \rightarrow C_k{\bullet}t_{cp} \in \delta) \cap (RL(C_i, C_k) \in \{>, pre\} \rightarrow C_k{\bullet}t_{in} \in \delta)$;

(2) $RC(C_i) = nv$: $M(C_k{\bullet}p_e) = \varphi \rightarrow$ Ck$\bullet$tcp $\in \delta$.

Proof: (1) $M(C_i{\bullet}p^{O}_{fa}) = \varphi$, that is, the component C_i fails. Because $RC(C_i) = v$, that is, C_i is the failure component, and because $\forall Ck \in C_{(k \neq i)}$, if $RL(C_i, C_k) = cp$, according to the definition of cut-point p_{tc1} and p_{tc2}, it has $t_{tci,j}$.

Because $((t_{tci,j})^{\bullet})^{\bullet} = t_{cp}$, ${\bullet}t_{cp} = \{P^{j}_{tr}, C_k{\bullet}p_e\}$, and $FT(M_f) = \Phi$, therefore $C_k{\bullet}t_{cp} \in \delta$. Similarly, we can get $RL(C_i, C_k) \in \{>, pre\} \rightarrow C_k{\bullet}t_{in} \in \delta$ according to the semantics of failure recovery model. Similarly, the sub-proposition (2) is established.

Theorem 1 explains that the system can properly handle the failure when the component fails, we can verify the related properties by weaving the concerns into the base net. From the analysis process, we can get that AOP can reduce the complexity of verification under the premise of improving the flexibility of model.

Examples

This section shows the analysis process through a simplified Export Service. First, the system will look up the relevant information and select the destination (C_1), and packaging services (C_2) responses for processing and packaging export products. The system will implement the operation of export transport ordering, commodity inspection bureau inspection (C_3) and insurance processing (C_4) in parallel, where Export transport ordering include water transport ordering(C_5) and airport ordering (C_6), the corresponding results will feedback to the exporter. All the information is confirmed by exporter and will be sent to the regulatory authorities (C_7) for checking. Finally, financial services (C_8) is used to check the related instruments of product and feedback the related tax rebate to exporters, the process of a simplified Export Service is finished. The composition process can be represented by expression $C_1 > C_2 > (C_3 \parallel C_4 \parallel (C_5 + C_6)) > C_7 > C_8$, where C_2 is the failure component, the other components are non failure, $RL(C_5, C_6) = pre$. According to the requirements of Export Service, we can weave one or more concerns into the base net of Export Service. Using the related tools of Petri net to compute the state space of Ω, Ω_f, Ω_c, Ω_t and Ω_s. We can also verify the related properties of failure recovery model, which includes the correctness of composition process, the consistency of requirements, and the effectiveness of failure recovery process. We can get the invoked services of Export Service is well structured based on its definition. Due to the space limitations, we only simulated for the simplified Export Service. But it is enough to illustrate the correctness of analysis process.

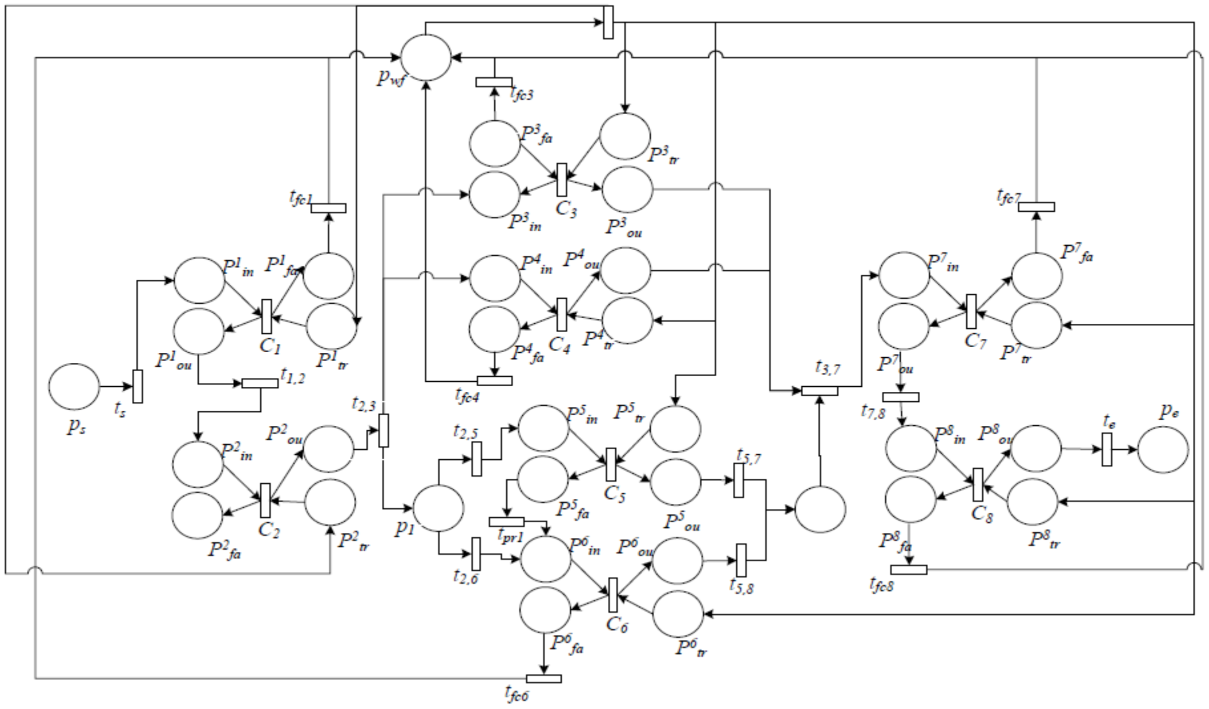

Fig 4 Failure recovery model of Export Service

The example and simulation results show that the proposed method can achieve the following results: (1) Correct characterizing the different components of cloud computing. The constructed model can ensure the correctness of failure recovery; (2) The flexibility of composition process, we can add or subtract the component based on the actual requirements; (3) The effectiveness of failure recovery: according to the semantic of failure recovery model, we can get that all components are processed in parallel when the component fails, therefore, the steps that the system reaches the terminate state is fixed when the component fails, which is unrelated with the number of component and their relationship, thus reducing the complexity of state space; (4) The advantages of using *AOP* ideas, the state space of base net and failure recovery model will non-linear grow with the number of available service and task increasing. While we only use base net when analyzing the basic properties. Therefore, the use of *AOP* ideas can reduce the analysis complexity.

Related Works

The work closes to ours is presented in [2]. In this work, the authors present a Petri net-based approach for supporting aspect-oriented modeling, but they didn't analyze the effectiveness of constructed model. The authors in [3] present some preliminary results in applying aspect orientation principle to secure software architectures design. A static strategy is proposed in [4] for modeling and analyzing fault tolerant service composition. And reference [5] introduces a modular monitoring mechanism that is able to observe these criteria and trigger a more advanced service selection procedure. Few of previous work focuses on the problem of handling failure. And the experience of Petri net for analyzing cloud computing has not been reported in detail yet.

Conclusions

As a new distributed computing mode, cloud computing is different from the traditional distributed computing: loose organization, high scalability, heterogeneity, and so on. And all these make the self-recovery under the cloud environment different from that under the traditional distributed computing environment. In this paper, we have proposed a method to model and analyze failure model for cloud application according to its characteristics. Petri nets are used as the formal description language for cloud application, and use it to describe the its basic elements, such as, cloud module, resource service, physical machine, virtual machine, etc. We formally model the basic relationship between cloud module, the interaction between physical machine and virtual machine, and composition rules are used to dynamically integrate these models into a failure model of cloud application. The operational semantics and related theories of Petri nets help prove the effectiveness and correctness of the proposed method. So the failure in cloud computing can be managed more efficiency, which can help the designers correct the software design.

Acknowledgment

The work is partially supported by the NSF of China under grants No.60672051.

References

[1] D. H.Shina, K. H.Leea, T. Sudab. Automated generation of composite Web services based on functional semantics. Web Semantics: Science, Services and Agents on the World Wide Web. 2009,7(4): 332-343.

[2] L. W. Guan, X. Y. Li, H. Hu, et al. A Petri net-based approach for supporting aspect-oriented modeling. Processing of the 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering. USA: IEEE Computer Society, 2008: 83-90.

[3] H. Q. Yu, D. M. Liu, X. D. He, et al. Secure software architectures design by aspect orientation. Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems. Washington, USA: IEEE Computer Society, 2005: 47-55.

[4] G. S. Fan, H. Q. Yu, L. Q. Chen and D. M. Liu. A method for modeling and analyzing fault-tolerant service composition. Processing of the 16th Asia-Pacific Software Engineering Conference, IEEE Computer Society, 2009: 507-514.

[5] B. Verheecke, M. A. Cibran, V. Jonckers. Aspect-oriented programming for dynamicWeb service monitoring and selection. Proceedings of the 2nd European Conference on Web Services(ECOWS 2004), Springer-Verlag, 2004: 15-29.