Simulation of IT Service Processes with Petri-Nets

Christian Bartsch[1], Marco Mevius[1], and Andreas Oberweis[2]

[1] FZI Research Center for Information Technology, Software Engineering,
Haid-und-Neu-Str.10-14, 76131 Karlsruhe, Germany
{bartsch,mevius}@fzi.de
[2] Universität Karlsruhe (TH), Institute AIFB, Kaiserstraße 89, 76133 Karlsruhe, Germany
{andreas.oberweis}@kit.edu

Abstract. Due to a steadily increasing market for IT services, providers need to set apart from their competitors in order to successfully assert with their service offerings in the market. As a result the improvement of the quality of service process provisioning is becoming a major aspect. In this paper we propose a Petri-net based approach in order to model and simulate service processes in terms of availability levels. Supported by a tool service providers can perform a priori estimations during design time on the potential impact and interaction of availabilities of services involved in provisioning processes. We additionally show how obtained results can be taken into account for negotiating availability levels in Service Level Agreements (SLA).

Keywords: IT Service Process, Availability Pattern, Petri-Net, Process Modeling, Process Simulation.

1 Introduction

Service providers need to set apart from their competitors in order to successfully compete with their service offerings in the market. Besides classical selling propositions such as price, customer proximity or product quality, the quality of providing complete and flexible service processes is becoming a key differentiator from the competition [1]. This is motivated by the fact that the basic thing that matters to the customer is the usage of services based on agreed typical indicators within the domain of IT service management. This could be service availability, time to restore a service, response time or performance level. Furthermore the provisioning of IT services is increasingly based on the modularization of whole service processes. This characteristic offers certain potential for reducing costs and enhancing service quality at the same time [2].

Hidden from customers to a certain extent providers have to establish methods and procedures in order to manage service processes concerning quality aspects. One aspect could be the maximum service availability which can be provided to a customer. This knowledge is important as service level agreements (SLA) between respective stakeholders contain such metrics. Additional penalties might also be taken into account for being part of an SLA in case of nonfeasance. In practice the determination of service levels for availability during the SLA negotiation phase is often based on a

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 53–65, 2009.

rule of thumb. Providers offer a certain percentage for a complete service process not knowing about availabilities of partial services involved. A validation of availability values agreed in an SLA usually occurs after the signing of the contract. An a priori estimation for example based on simulation in order to reduce risks of nonfeasance can help providers with identifying feasible levels for service processes. They can "play around" with availability levels of services being part of an offered service process. The precise modeling of service processes and the resources [3] enabling certain process steps utilizing Petri-nets is supposed to support service providers to make assumptions about what availability levels are feasible for potential service offerings. The formal representation allows a systematic a priori simulation of availability values for IT service processes. In addition to this the quality of service processes is affected twofold. On the one hand side service providers are able to identify potential bottlenecks regarding the overall process availability by simulating the impact and interactions of all availabilities of IT services involved. On the other hand the ability to perform a priori estimations could support customers in identifying the required service levels in order to leverage and maximize the performance of their respective business processes. In this paper we introduce a new Petri-net based approach for modeling and simulating service processes in terms of availability levels in order to foster a priori estimations during design time. The approach to be introduced is based on High-level Petri-nets. In High-level Petri-nets, complex objects can be represented. IT service processes are modeled as a manipulation of these objects. High-level Petri-nets not only feature significant advantages with respect to IT service process modeling. Their precise foundation allows straightforward simulation and further extensive analysis. Moreover, High-level Petri-nets support the development and implementation of control and monitoring tools for a continuous improvement of the relevant IT service processes. It is highlighted how Petri-nets can be deployed for service modeling within the IT service management domain in order to improve quality aspects for potentially offered and provided service processes.

In Chapter 2 we discuss related work. We then introduce the basic concepts of Petri-nets and IT service processes in Chapter 3. Afterwards we present in Chapter 4 an approach for modeling the availability of IT services and show how IT service processes can be simulated in order to support contracting service level agreements. Subsequently we present in Chapter 5 two extensive simulation experiments and the findings which can be derived from the results of the simulations.

2 Related Work

Various publications such as [4, 5] contribute to the topic of business process management. Modern languages for business process management (e.g. the Business Process Execution Language (BPEL)[1], Business Process Modeling Language (BPML)[2] or ebXML Business Process Specification Schema (BPSS)[3]) focus on the tracking and execution of business processes by a business application. The formal analysis and monitoring of process performance is not considered. Different methods

[1] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
[2] http://www.omg.org/docs/dtc/06-02-01.pdf
[3] http://www.ebxml.org/

and description languages have been proposed for process modeling. Most of them are based on textual programming languages or graphical notations such as dataflow diagrams, UML state charts, Event-driven Process Chains (EPCs), Petri-nets or related notations [6, 7]. They can be divided into semi-formal methods, e.g., EPCs [8], and formal methods, e.g., Petri-nets [9, 10]. Semi-formal methods, especially EPCs, are very popular, widely disseminated and easy to use. They do, however, exhibit major shortcomings especially if they support the modeling and simulation of business processes with one integrated method [11]. These shortfalls can be overcome by the use of Petri-nets. On the basis of different definitions and interpretations of their static and dynamic modeling components different types of Petri-nets have been derived to date. High-level Petri-nets (Predicate/Transition nets [12] or Colored Petri-nets [13]) have proven to be suitable for modeling complex dynamic systems. They support an integrated description of structural and behavioral characteristics of dynamic systems. In recent years, enhancements have been made in order to improve the modeling of objects with more complex structures using High-level Petri-nets. Nested Relation/Transition nets combine the concept of complex structured objects with the Petri-net formalism [14]. Motivated by the increasing acceptance and dissemination of the XML standard [15], XML-nets were derived (for a detailed description we refer to [16]). The major advantages of the process-oriented High-level Petri-net-based modeling of IT service processes can be summarized as follows:

- High-level Petri-nets are capable of modeling and simulating availability levels of IT services in order to foster a priori estimations during design time.
- High-level Petri-nets provide capabilities for modeling complex objects with a hierarchical structure which can typically be found in IT service processes.
- By using standards, High-level Petri-nets support the intra- and inter-organizational exchange of service level information and thereby foster standardized design, implementation and monitoring of pre-defined performance levels.
- A High-level Petri-nets-based software prototype can be directly linked to simulation relevant information. It automatically simulates and executes pre-defined simulation runs of IT service processes.

For these reasons, High-level Petri-nets are not only suited for modeling business processes but also for the integrated process simulation and analysis especially in the domain of IT services and Service Level Management. SLM comprises proactive methodology and procedures to ensure that adequate availability levels of IT services are delivered to all IT users in accordance with business priorities and at acceptable cost [17]. One of the main ways for publishing services is by setting up a service level agreement (SLA) between the IT department and the business. The SLA is a document that defines and identifies the various services offered by a service provider to its clients. Since SLAs provide the term and conditions that define the services rendered by a provider, it then limits its scope of support, therefore minimizing production cost [18, 19]. It describes the services along with a regime for managing how well these are delivered. They are typically defined in terms of the availability, responsiveness, integrity and security delivered to the users of the service [17]. It states a commitment to range, quantity and quality of services of service providers. The size of service levels must be justifiable and benefit the business as a whole. If this is not the case then such service levels should be renegotiated [20]. Various research

approaches and results exist in the context of SLAs and quality of service (QoS) within respective domains such as Web Services [21, 22, 23], mobile communication [24], IT Management [25, 26] or supply chain management [27]. They primarily focus on the modeling, monitoring and analysis of SLAs but scarcely use the benefit of simulation for determining the impact of IT services on the service processes they support. This knowledge could be used – especially if considering service availability aspects – in order to enhance the definition of service levels during design time.

3 Petri-Nets and IT Service Processes

Petri-nets [10] are a formal graphical process description language that combines the advantages of the graphical representation of processes with a formal definition. They allow the representation of a class of real-world processes by utilizing a graphical process schema. Petri-nets consist of static components (places, depicted by circles) and dynamic components (transitions, depicted by rectangles) that are connected by edges. In the following we assume that the reader has basic knowledge about the functionality of Petri-nets. For illustration purposes we present a Place/Transition net in Figure 1 that describes a simplified example derived from the IT Service Management domain.

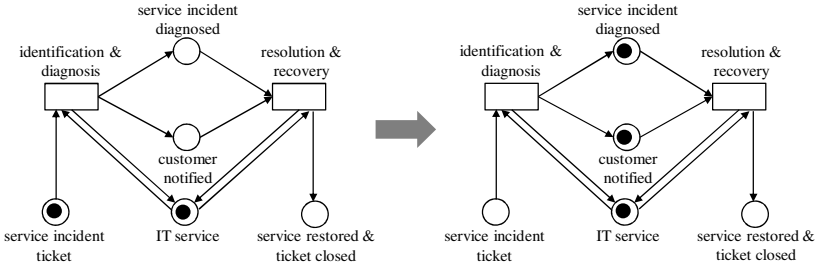


Fig. 1. Example for Place/Transition net

A ticket, represented as an individual object in place "service incident ticket", can be processed by transition "identification&diagnosis" if an object is available in place "IT service". The transition can fire and because of a branch the ticket object is fired to the places "service incident diagnosed" and "customer notified". As there is a loop between place "IT service" and transition "identification&diagnosis" the same object is sent back to its origin place.

The major task of business performance analysis on the basis of availability levels is the assessment of alternative business process designs with respect to a given set of objectives. In addition to methods commonly applied in Business Process Management (BPM), in particular High-level Petri-nets support analysis based on simulation. Simulation in process-oriented management of IT service processes aims, for example, at validating the defined exceptional states and the corresponding customizing activities. Due to their formal foundation Petri-nets can directly be executed and therefore be used to simulate the described process [28]. A simulation engine

interprets the formal syntax and semantics of the Petri-net and transforms it into a machine-readable code [6]. A widespread technique to formally model services, is based on Petri-nets [3].

4 Availability Modeling and Simulation of IT Service Processes

In this section we present an approach to model and configure the availability of IT services and the service processes they support. The availability of an IT service is the percentage of possible time that the service is online or available to use. A service should be available on demand as often as possible in order to be able to serve users needs when they need them to be served [29]. We assume that in case of an IT service enabling the execution of certain activities within service processes it is mandatory that a service is available as soon as the processing of a certain object through an activity requires it. Figure 2 illustrates the states of service availability we differentiate within the IT service management domain.

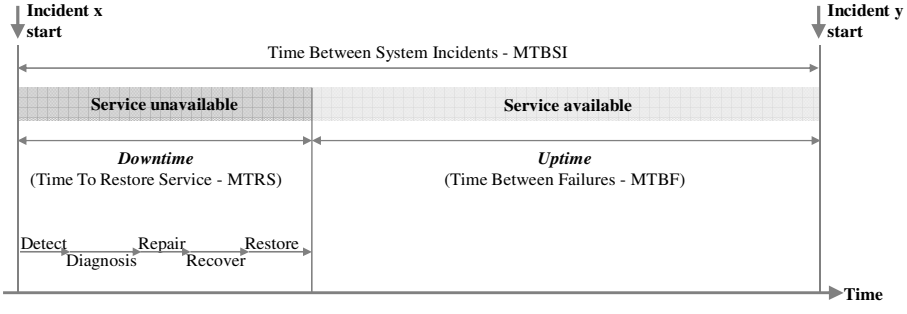


Fig. 2. States of service availability

As soon as any internal or external event triggers an incident x of an IT service and is detected by respective tools it passes into the state *unavailable*. The total period that an IT service is unavailable for operational use within an agreed service time is called *downtime* $dt(s_n)$.

$$dt(s_n) = (1 - a(s_n)) * ast(s_n) \quad (1)$$

In order to finally restore the failed IT service several steps such as diagnosing the cause of fault as well as repairing and recovering the service must be processed. All steps can take a different time. The period from detecting to restoring the IT service is the *time to restore a service* (MTRS). The service is now available again.

The determination of service availability always requires an *agreed service time* $ast(s_n)$ being the value during which a particular IT service s_n is agreed to be fully available. The total period that an IT service is operational within an agreed service time is called *uptime* and takes the probability $a(s_n)$ into account – being identical with key figure a*vailability* – to which a service will not fail.

$$ut(s_n) = a(s_n) * ast(s_n) \quad (2)$$

The time until the next incident y causes another failure of the same IT service is the *time between two failures* (MTBF). The *Availability* $a(s_n)$ of an IT service is the ability to perform its required function over a stated period of time. The availability of a service considers all occurred downtimes during an agreed service time. We define:

$$a(s_n) = \frac{ast(s_n) - \sum dt(s_n)}{ast(s_n)} \tag{3}$$

Availability pattern for IT services

For simulation purposes the formal modeling of service processes and the respective IT service availabilities with Petri-nets requires closer examination on the representation of service downtimes and the corresponding *Time To Restore Service* $MTRS(s_n)$ involved. With the following pattern shown in Figure 3 we can model the aforementioned metrics.

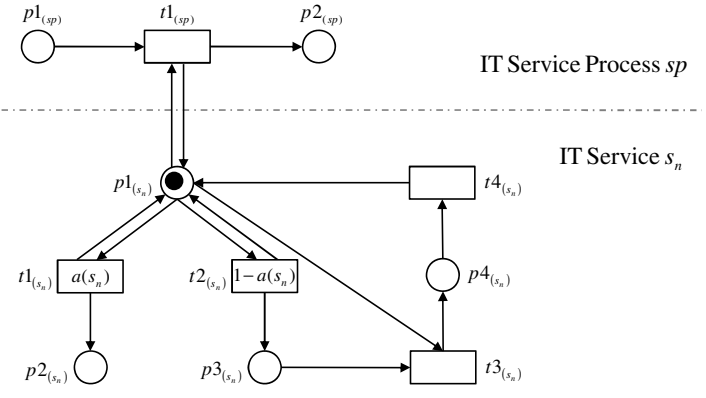


Fig. 3. Availability pattern for IT services

It can therefore be interpreted as representative of a respective IT service s_n in terms of availability. The proposed availability pattern based on Petri-nets can be used as a type of resource in order to enable the execution of a (semi-) automatic process step within a service process. A transition $t_m(sp)$ is connected to the respective supporting IT service s_n. A marking M of the considered IT service process sp and IT services s_n enables a transition $t_m(sp)$ if it marks every place in $\bullet t_m(sp)$. If $t_m(sp)$ is enabled within M, then it can fire, and its firing leads to the successor marking M' ($M \xrightarrow{t_m(sp)} M'$). In this depiction we simplified the IT service process and assume a linear process executing process steps sequentially. For simulation purposes each pattern – representing a single IT service s_n – needs an individual initial configuration in terms of probability of service failure $1 - a(s_n)$ and the average period from detecting an IT service failure to restoring the service for operational $MTRS_{sim}(s_n)$. Especially the last metric, represented by transition $t3(s_n)$, requires closer examination as its value is comprised of further data. The *checking interval* $ci(s_n)$ is a defined repetitive timing after which transitions $t1(s_n)$ and $t2(s_n)$ is enabled to fire an object.

The introduction of transition $t4(s_n)$ is supposed to illustrate the option for integrating additional activities such as accounting, reporting, penalty handling etc.

The checking cycle $cc(s_n)$ is a supporting construct in order to determine the total number of potential failures within an agreed service time. Due to the structure of the availability pattern we use uptime $ut(s_n)$ for further calculation instead of $ast(s_n)$. Transitions $t1(s_n)$ and $t2(s_n)$ can only fire – and therefore check for availability – if $p1(s_n)$ is marked. This means that the IT service is up and running.

$$cc(s_n) = \frac{ut(s_n)}{ci(s_n)*100} \tag{4}$$

Running a cycle equates to a hundred proceeded availability checks. This means for example that for a given service availability of 98.0 percent transition $t1(s_n)$ fires 98 objects (= ping for availability) to $p2(s_n)$ and $t2(s_n)$ 2 objects (= service failure) to $p3(s_n)$ on average. In order to calculate $dtast(s_n)$, the total amount of "Downtime objects" in $p3(s_n)$ per $ast(s_n)$ as shown in (6), we need to identify the number of potential *downtimes per checking cycle* $dtcc(s_n)$ at first.

$$dtcc(s_n) = (1 - a(s_n)) * 100 \tag{5}$$

$$dtast(s_n) = dtcc(s_n) * cc(s_n) \tag{6}$$

The value for $MTRS_{sim}(s_n)$ which is represented in the pattern by transition $t3(s_n)$ allows the modeling of the maximal allowed downtime per agreed service time. We calculate the minimum as shown in (7.1) because the value of $MTRS_{sim}(s_n)$ can't be higher than the maximum downtime derived from the given service availability.

$$MTRS_{sim}(s_n) = \min\left(dt(s_n), \frac{dt(s_n)}{dtast(s_n)}\right) \tag{7.1}$$

On closer examination of $MTRS_{sim}(s_n)$ it can be seen that the value is individual for every given service availability $a(s_n)$ considering respective values for $ast(s_n)$ and $ci(s_n)$ which leads us to following:

Thesis.

$$\frac{dt(s_n)}{dtast(s_n)} = \frac{ci(s_n)}{a(s_n)}$$

Proof.

$$\frac{dt(s_n)}{dtast(s_n)} = \frac{(1 - a(s_n)) * ast(s_n)}{dtcc(s_n) * cc(s_n)} = \frac{(1 - a(s_n)) * ast(s_n)}{(1 - a(s_n)) * 100 \ * \frac{ut(s_n)}{ci(s_n) * 100}}$$

$$= \frac{(1 - a(s_n)) * ast(s_n)}{\frac{(1 - a(s_n)) \ * ut(s_n)}{ci(s_n)}} =$$

$$\frac{ci(s_n) * ast(s_n)}{ut(s_n)} = \frac{ci(s_n) * ast(s_n)}{a(s_n) * ast(s_n)} = \frac{ci(s_n)}{a(s_n)}$$

As a result we can define (7.2):

$$MTRS'_{sim}(s_n) = \min\left(dt(s_n), \frac{ci(s_n)}{a(s_n)}\right) \tag{7.2}$$

This implication eases the identification of values for $MTRS'_{sim}(s_n)$ because the checking interval $ci(s_n)$ as well as service availability $a(s_n)$ serve as direct input into the model and don't need to be calculated separately compared to (7.1).

Representation of IT service processes

For our purposes we model service processes using the duration d of a single process step $sp_{l,x}$ and the IT services enabling respective step. The service process is depicted as a sequence of tupels $[d(sp,x); a(s_n)]$, where $d(sp,x)$ is the duration of process step x within a service process sp and $a(s_n)$ indicates the availability of IT services s_n involved in the execution of (sp,x). Exemplarily an IT service process A containing three process steps and supported by two IT services could then be coded as: $A = [d(A,1); a(s_1)] \rightarrow [d(A,2); a(s_1), a(s_2)] \rightarrow [d(A,3)]$.

$(A,1)$ is supported by s_1 and the respective service availability $a(s_1)$ whereas $sp_{A,3}$ is not supported by an IT service at all. In order to finally gain the overall availability of a service process $a(sp)$ we first need to identify the maximum number of objects the service process can go through if IT services involved provide a hundred percent availability. The number of objects in the end place of the service process modeled with Petri-nets completely depends on the duration of each process step and therefore defines an upper bound not being constrained by any IT service availability. We then determine the impact of single IT service availability to the overall availability of the supported IT service process by comparing the number of objects in the end place after a simulation period in relation to the predetermined upper bound.

5 Evaluation

In Figure 4 we present a simplified use case derived from a conducted IT service management project[4].

For this simulation example we assume that all IT services perform independent from each other. The checking interval within each IT service was set to 60 minutes. The scenario includes three IT services supporting two incident management processes with different execution times. It basically differs in the number of IT services involved. As the example shows, an IT service is available as long as an object lies within a place $p1(s_n)$light turns into a green state or red state as soon as a. The availability pattern, as depicted in Figure 3, was modeled with a simulation tool being under ongoing development[5].

As soon as parallel and alternative paths will become part of the simulation complexity will increase noticeably. As a consequence more simulation cycles will be necessary in order to keep significance of the simulation results.

[4] Please note that other service process configurations containing alternatives and parallelism would also be possible but are not mentioned in this example.

[5] A deployable version of the simulation tool will be open to the public soon.

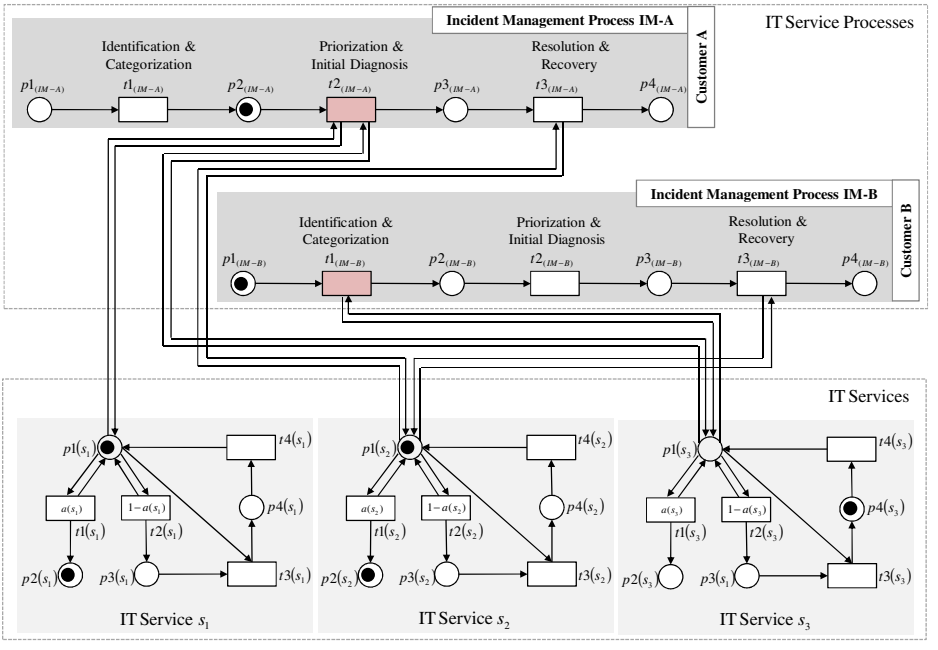


Fig. 4. IT service process availability simulation

Simulation Scenario A: 3 process steps | 3 IT services (2+1)
The first scenario as shown in Figure 5(a) simulates a service process $IM - A$ supported by IT services s_1, s_2 and s_3 whereas $(IM - A, 2)$ is supported by two services s_2 and s_3 simultaneously. The configuration for simulations $SimA1 - SimA5$ is as follows:

$$IM - A = [d(IM - A, 1) = 3] \rightarrow [d(IM - A, 2) = 5; a(s_1), a(s_3)] \rightarrow [d(IM - A, 3) = 10; a(s_2)]; \quad \mathrm{d}(IM - A) = 18;$$

The parameters for $a(s_1), a(s_2)$ and $a(s_3)$ – represented by dark grey bars in the figure – are changed for each simulation in order to find out the availability of the service process – represented by the light grey bars – at the end of simulation period of 60 days. Results show that $a(IM - A)$ of $IM - A$ is rather proportional to $a(s_n)$. A service provider must therefore increase the availability of selected IT services in order to increase the availability of the service process. The demand for high process availability usually implies a high (monetary) effort to ensure high IT service availability.

If we assume that the duration of service process steps themselves plays a role for respective service process availability we reconfigure sp_{IM-A} to:

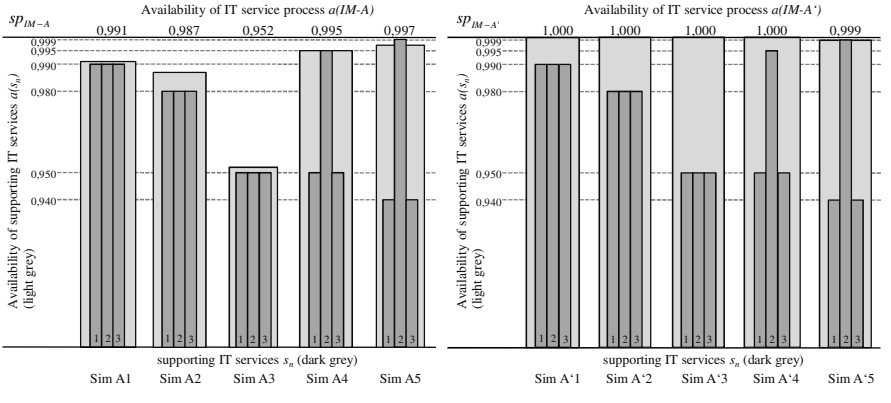
$$IM - A' = [d(IM - A', 1) = 3] \rightarrow [d(IM - A', 2) = 8; a(s_1), a(s_3)] \rightarrow [d(IM - A', 3) = 7; a(s_2)]; \quad d(IM - A) = 18;$$

We notice, that slight changes in $d(IM - A', 2)$ and $d(IM - A', 3)$ cause the effect that $a(IM - A)$ stays on a continuous high level of a hundred percent availability for all simulations $SimA'1 - SimA'5$. As a consequence this means that for service

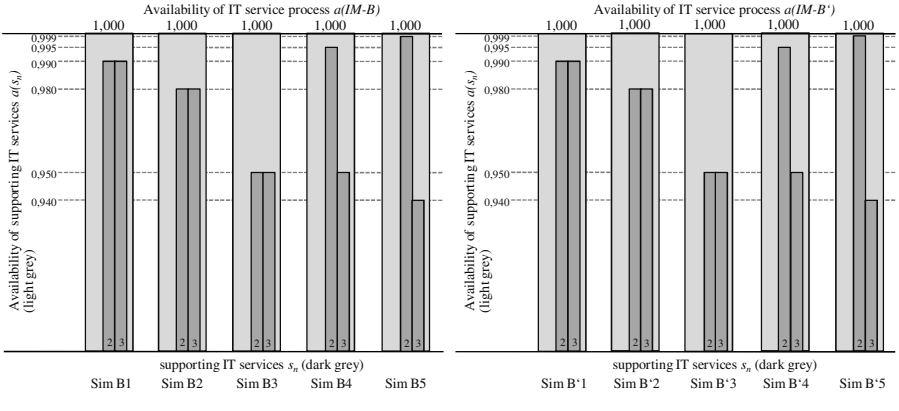
process $sp_{IM-A'}$ an availability level for the supporting IT services $a(s_1), a(s_2)$ and $a(s_3)$ of 95,00% would be sufficient in order to enable an availability of the supported IT service process of 100,00%. This finding is helpful for negotiating SLAs as there is no difference to the overall performance of the service process if paying for a 99,90% or for a 95,00% IT service availability and can save a customer money.

Simulation Scenario B: 3 process steps | 2 IT services (1+1)
As we can see from figure 5(b), simulations $SimB1 - SimB5$ as well as $SimB'1 - SimB'5$ for scenario B show the same results although we changed the duration of $(IM - B, 2)$ and $(IM - B, 3)$ in $IM - B$:

$$IM - B = [d(IM - B, 1) = 3; a(s_4)] \rightarrow [d(IM - B, 2) = 5] \rightarrow [d(IM - B, 3) = 10; a(s_2)]; \quad d(IM - B) = 18;$$
$$IM - B' = [d(IM - B', 1) = 3; a(s_4)] \rightarrow [d(IM - B', 2) = 3] \rightarrow [d(IM - B', 3) = 8; a(s_2)]; \quad d(IM - B') = 14;$$



(a)



(b)

Fig. 5. (a) Simulation results for service processes $\boldsymbol{IM - A}$ and $\boldsymbol{IM - A'}$. (b) Simulation results for service processes $\boldsymbol{IM - B}$ and $\boldsymbol{IM - B'}$.

Different availability levels for supporting IT services don't have any influence to the maximum service process availability of 100,00%. As in Scenario A 95,00% IT service availability would be sufficient in order to support the process best. The additional finding is that, even though we changed the duration of process steps all availabilities remained the same. This could be a hint of a much more *stable* service process $IM - B$ compared with $IM - A$ as slight changes in the duration in combination with different IT service availabilities don't affect the overall process performance at all.

6 Conclusion and Outlook

In this paper we presented a simulation-based approach in order to identify potential impact and interdependencies in terms of availability between IT services and the service process supported. This can assist service providers and their customers when negotiating SLAs during design time. The stakeholders involved now have a possibility to simulate a more realistic service availability level which can support their processes at most. The current approach can't handle the effect that as soon as an object is processed by a process step (transition) and the supporting IT service (place) fails along the way the affected object should actually be considered lost. At the moment the simulation only considers the incoming moment for a transition to be locked or active. As soon as an object "enters" a transition, it will be processed in any case. We need further research on complexity and performance aspects of current simulations as they are very resource intensive already, even though the examples are quite simple. We are working on an extension of modeling and simulating IT services by utilizing XML-nets [16] in order to exemplarily include rules and additional information about an object to cope with the current constraint. The objective is the Petri-net based simulation of whole SLAs described as XML-Documents.

References

1. Bartsch, C., Shwartz, L., Ward, C., Grabarnik, G., Buco, M.J.: Decomposition of IT service processes and identification of alternatives using ontogoly. In: IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 714–717. IEEE Computer Society Press, Los Alamitos (2008)
2. Böhmann, T., Junginger, M., Krcmar, H.: Modular Service Architectures - A Concept and Method for Engineering IT Services. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, pp. 74–83. IEEE Computer Society Press, Los Alamitos (2003)
3. Reisig, W.: Towards a Theory of services. In: Kaschek, R., et al. (eds.) UNISCON 2008. LNBIP, vol. 5, pp. 271–281. Springer, Heidelberg (1974)
4. van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.): Business Process Management. LNCS, vol. 1806. Springer, Heidelberg (2000)
5. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
6. Desel, J., Erwin, T.: Modeling, Simulation and Analysis of Business Processes. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 129–141. Springer, Heidelberg (2000)

7. Karagiannis, D., Juninger, S., Strobl, R.: Introduction to Business Process Management Systems Concepts. In: Scholz-Rieter, S. (ed.) Business Process modeling, pp. 81–106 (1996)
8. Scheer, A.-W.: ARIS – Business Process Modeling, 3rd edn. Springer, Berlin (2007)
9. Reisig, W.: Place/Transition Systems. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 254, pp. 117–141. Springer, Heidelberg (1987)
10. Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets I: Basic Models. LNCS, vol. 1491. Springer, Heidelberg (1998)
11. van der Aalst, W.M.P.: Formalization and Verification of Event-driven Process Chains. Information and Software Technology 41(10), 639–650 (1999)
12. Genrich, H.J.: Predicate/Transition Nets. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) Petri Nets: Central Models and Their Properties, Advances in Petri Nets, pp. 207–247. Springer, Heidelberg (1986)
13. Jensen, K.: Coloured Petri Nets. In: EATCS Monographs on Theoretical Computer Science. Basic Concepts, vol. 1. Springer, Berlin (1992)
14. Oberweis, A.: An integrated approach for the specification of processes and related complex structured objects in business applications. Decision Support Systems 17(1), 31–53 (1996)
15. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F., Cowan, J.: Extensible Markup Language (XML) 1.1 (Second Edition). W3C Recommendation 16, edited in place 29, World Wide Web Committee (W3C) (August 2006)
16. Lenz, K., Oberweis, A.: Inter-organizational Business Process Management with XML Nets. In: Ehrig, H., Reisig, W., Rozenberg, G., Weber, H. (eds.) Petri Net Technology for Communication-Based Systems. LNCS, vol. 2472, pp. 243–263. Springer, Heidelberg (2003)
17. Sturm, R., Morris, W., Jander, M.: Foundations of Service Level Management. Sams (2000)
18. Blokdijk, G.: Service Level Agreement 100 Success Secrets. Lightning Source UK Ltd. (2008)
19. Johnston, R., Clark, G.: Service Operations Management – Improving Service Delivery. Pearson Education Limited, Essex (2005)
20. Bruton, N.: Managing the IT services process. Butterworth-Heinemann, Burlington (2004)
21. Ludwig, H.: Web Services QoS: External SLAs and Internal Policies – Or: How do we deliver what we promise? In: Proceedings of the 4th IEEE International Conference on Web Information Systems Engineering Workshops, pp. 115–120. IEEE CS Press, Los Alamitos (2003)
22. Hudert, S., Ludwig, H., Wirtz, G.: Negotiating SLAs - An approach for a generic negotiation framework for WS-Agreement. In: Proceedings of the 20th International Conference on Software Engineering & Knowledge Engineering (SEKE 2008), pp. 587–592 (2008)
23. Wohlstadter, E., Tai, S., Mikalsen, T., Rouvellou, I., Devanbu, P.: GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions. In: Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), pp. 189–199. IEEE Computer Society, Los Alamitos (2004)
24. Song, M., Chang, R., Song, R., Song, J.: Research on the SLA-based Service Management in Mobile Communication Network. In: Canadian Conference on Electrical and Computer Engineering (CCECE 2004), vol. 2, pp. 1017–1020 (2004)

25. Anders, T.: Development of a generic IT service catalog as pre-arrangement for Service Level Agreements. In: 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005), vol. 2, pp. 567–573 (2005)
26. Wen-Li, D., Hang, Y., Yu-Bing, Z.: Testing BPEL-based Web Service Composition Using High-level Petri Nets. In: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), pp. 441–444 (2006)
27. Karhunen, H., Eerola, A., Jantti, M.: Improving Service Management in Supply Chains. In: Proceedings of the International Conference on Service Systems and Service Management, vol. 2, pp. 1415–1420 (2006)
28. Mochel, T., Oberweis, A., Sänger, V.: INCOME/STAR: The Petri net simulation concepts. Systems Analysis – Modeling – Simulation. Journal of Modeling and Simulation in Systems Analysis 13(1-2), 21–36 (1993)
29. Addy, R.: Effective IT Service Management. Springer, Heidelberg (2007)