Model-Driven Evaluation of User-Perceived Service Availability

Andreas Dittrich and Rafael Rezende

ALaRI Advanced Learning and Research Institute
Università della Svizzera italiana (USI)
Via G. Buffi 13, CH-6904 Lugano, Switzerland
{andreas.dittrich,rafael.ribeiro.rezende}@usi.ch
http://www.usi.com/

Abstract. Service-oriented architecture (SOA) has emerged as an approach to master growing system complexity by proposing services as basic building elements of system design. However, it remains difficult to evaluate dependability of such distributed and heterogeneous functionality as it depends highly on the properties of the enabling information and communications technology (ICT) infrastructure. Moreover, every specific pair service client and provider can utilize different ICT components, constituting for the *user-perceived* view of a service.
We provide a model-driven methodology to automatically create reliability block diagrams of such views. Given a service description, a network topology model and a pair service client and provider, it identifies relevant ICT components and generates a user-perceived service availability model (UPSAM). We then use this UPSAM to calculate the steady-state availability of different views on an exemplary mail service deployed in the network infrastructure of University of Lugano, Switzerland.

Keywords: Service networks, Service dependability, Availability, Quality of service, Service network management, Modeling, Object oriented modeling, Design engineering

1 Introduction

Growing functional and non-functional requirements have increased IT system complexity significantly during the last decade. At the same time, modern business operation is relying ever more on IT services and thus, predictable service delivery with time, performance and dependability constraints. In order to tame complexity and enable efficient design, operation and maintenance, various modeling techniques have been proposed. *Service-Oriented Architecture* (SOA) proposes a formalism where services are the basic building elements of system design. [4]

Meeting non-functional property requirements is crucial for successful service provision. However, non-functional properties like service availability are highly dependent on the properties of the underlying information and communications

technology (ICT) infrastructure. This work focusses on *user-perceived service availability*: Given an ICT infrastructure with a set of providing service instances and a set of service clients. The user-perceived availability is the probability for a service provided by one or more of these instances to perform its required function when requested from a specific client.

To assess the user-perceived service availability for any client within the network, information about the system availability is not sufficient, because every specific pair service requester and provider can utilize different ICT components. This is why, although services are usually well-defined within business processes, assessing service availability remains uncertain. The underlying infrastructure varies according to the position of the *service requester* – represented by a person or even an information and communications technology (ICT) component – and the concrete *providing service instance*. Evaluation of user-perceived service availability should employ a model of the ICT infrastructure where service properties are linked to component properties.

One important concept in service-oriented architecture is composition, the possibility to combine the functionality of multiple services to provide more complex functionality as a composite service with a single interface. If the individual services within the composition are indivisible entities regarding their functionality, they can be called *atomic services*. For instance, an *email* service can be divided into atomic services *authenticate*, *send mail* and *fetch mail*. In this sense, *email* corresponds to a *composite service* constituted by the atomic services *authenticate*, *send mail* and *fetch mail*.

This paper provides a methodology to evaluate user-perceived service availability. Given a set of input models that describe the service network topology, its services and actors, it generates and solves specific *user-perceived service availability models* (UPSAM) for different user perspectives. These models are expressed as *reliability block diagrams* (RBD) and evaluate steady-state service availability. The evaluation can be useful when designing a service network to estimate the expected quality of service provision. After deployment, it can be used to detect bottlenecks in the network or to evaluate the impact of planned changes to the ICT infrastructure. The methodology could be extended to provide evaluation of different dependability properties that also cover dynamic network behaviour during service usage.

The following section provides an overview of related work. Section 3 states the scientific problem of evaluating user-perceived service availability, followed by an approach to solve it in Section 4. Given a service description, a network topology model and a pair service provider and requesting client, we employ a methodology to automatically identify relevant ICT components and from that generate the UPSAM. Finally, Section 5 demonstrates the feasibility of our approach in a representative case study by applying it to an exemplary email service within parts of the service network infrastructure of University of Lugano, Switzerland. We extract the UPSAM of that service for different service clients and calculate and compare their availability. Section 6 summarizes our work by pointing out the main contributions and remaining open issues.

2 Related Work

Service-Oriented Architecture (SOA) [4] provides a set of methodologies where system components are designed as interoperable services. In this paper, a service is defined according to [6] as "an abstraction of the infrastructure, application or business level functionality. It consists of a contract, interface, and implementation. [...] The service interface provides means for clients to connect to the service, possibly but not mandatory via network." The same authors define a composite service as a composition of basic indivisible services called atomic services, which are shaped according to their business functionalities. Their definition focuses on the optimal re-usability, in order to avoid redundant atomic services with similar or the same purpose. Milanovic et al. also propose a methodology for the automatic generation of service availability models based on run-time monitoring [7, 5, 6]. In their methodology, a configuration management database system collects information about the network topology for further service deployment and steady-state availability analysis.

A different definition of services is proposed by the Service Availability Forum (SAF) [11] through the *Availability Management Framework* (AMF). In their specification, AMF components are the basic entities of the framework and consist of a set of software or hardware resources. In contrast to Milanovic et al. in [5], where infrastructure and services are modeled independently, SAF describes AMF components as intrinsic service providers, which can be grouped into bigger logical units called service units (SU).

Salehi et al. [10] proposes a *UML-based AMF configuration language* (UACL) to facilitate the generation, analysis and management of the AMF configurations. The language has been implemented by means of a *Unified Modeling Language* (UML) [9] profile. *Dependability Analysis Modeling* (DAM) [2] consists also of a UML profile for dependability modeling. It correlates service and ICT components, and describes them with a complete set of properties, although no transformation is provided by the methodology.

The authors of [17] provide a stochastical model to assess user-perceived web servide availability and demonstrate that there can be significant differences between the system and user-perceived perspectives. In [12] a new status-based model to estimate user-perceived availability proposed. Both works do not model the providing infrastructure in detail, however.

In order to assess the service dependability from different user perspectives, an extraction of relevant network parts is presented in [3]. Given a model of the network topology, a service description and a pair service requester and provider, a model-to-model transformation is applied to obtain a *user-perceived service infrastructure model* (UPSIM): Given an ICT infrastructure that contains a providing service instance p_i and a service client c_j. The UPSIM is that part of the infrastructure which includes all components, their properties and relations hosting the atomic services used to compose a specific service provided by p_i for c_j. The approach in [3] uses a subset of UML elements as well as UML profiles and stereotypes to impose specific dependability-related attributes to ICT components.

The paper at hand builds on the work in [3] with a methodology to obtain as output a specific availability model expressed as *reliability block diagram* (RBD) to evaluate service availability for different user perspectives. The case study in Section 5 uses an implementation of that methodology that is extensively described in [8].

3 Problem Statement

The availability of a service, as any non-functional property, depends on the underlying ICT infrastructure required for service execution. Moreover, a service may require a different set of ICT components for each user perspective within the infrastructure, as the service can be invoked for different pairs of service requester and provider. Also, the topology and services may change due to reconfiguration, addition or removal of components, upgrades and so on.

This dynamicity represents one of the main challenges of availability evaluation, especially during run-time, when changes need to be instantly considered in the availability models. A methodology is needed to support the model-driven evaluation of user-perceived service availability. The methodology should include:

1. A model to describe the ICT infrastructure, including availability properties for each component.
2. A model to describe services in hierarchical manner.
3. A formalism to relate an abstract service to parts of a concrete deployed infrastructure for any specific user perspective.
4. A mechanism to generate and solve a *user-perceived service availability model* (UPSAM) for such a user perspective.

The complete methodology should be automated as much as possible to support quick model updates in dynamic environments and to eliminate human errors during update or upgrade procedures. Preferably, the methodology should be defined and implemented using well-known standards and open-source tools to support external verification and to facilitate its dissemination.

4 Methodology

Since user-perceived non-functional service properties depend on the underlying infrastructure, the methodology consists of generating a *user-perceived service availability model* UPSAM from a service description, a network topology and a mapping between them. The UPSAM is then evaluated using an external tool for availability analysis. Infrastructure and services are represented in UML models as in [3]:

– *Class diagrams* are used to describe structural units of the network (e.g.: routers, clients, servers), their properties and relations in distinct classes.

– *Object diagrams* describe a deployed network structure/topology composed of *class* instances, namely objects with all properties of the parent class, and *links* as instances of their relations.
– *Activity diagrams* are used for service description and represent the service as a flow of actions.

We are using the input model specifications from [3] but enhance the described workflow. Instead of generating a *user-perceived service infrastructure model* (UPSIM), we output a *reliability block diagram* (RBD) for that part of the network that is relevant for service provision for a specific pair service requester and provider. Figure 1 presents an overview of the methodology.

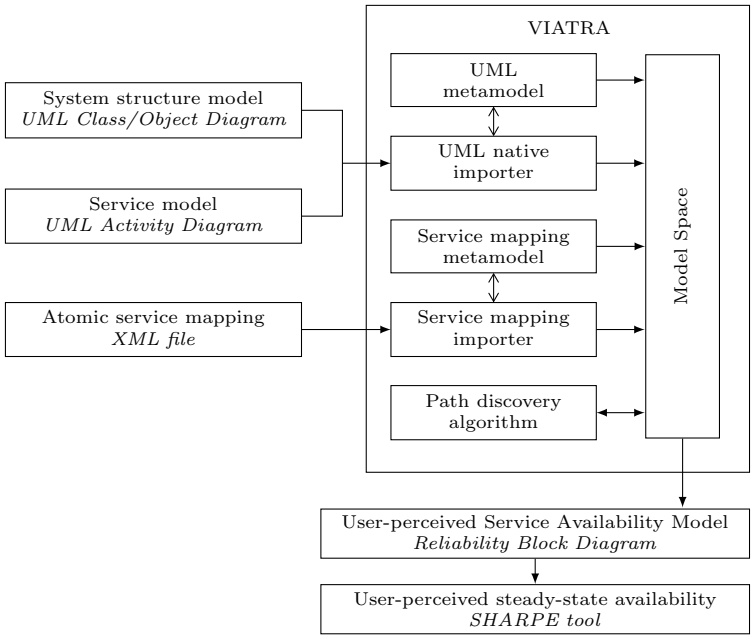

Fig. 1. Implementation of the model transformation

Following is a step-by-step description of the methodology. Steps 1 to 7 have been adapted from [3] for the scope of this work where necessary. Steps 8 to 10 are the main contribution of this work and are described in more detail in Section 4.1. Apart of the last Step 10, the workflow is based on the open-source development tool Eclipse [13], using both the UML2-compliant [9] modeling tool Papyrus [14] and the model transformation plug-in VIATRA2 [15]. Extensive details about the implementation of all steps can be found in [8].

1. Identify ICT components and create respective UML classes for each type. For subsequent availability analysis, an elementary UML availability profile

(see Figure 2) is applied to classes. This results in a class diagram containing the description of every ICT component.
2. Model the complete ICT infrastructure using UML object diagrams with instances of the classes from Step 1.
3. Identify and iteratively describe services using UML activity diagrams with atomic services as building blocks (Actions). This step results in a collection of service models with no correlation to the infrastructure.
4. Generate service mapping pairs by mapping atomic services from Step 3 to respective requester and provider ICT components from the infrastructure object diagram (Step 2).
5. Import ICT infrastructure and service UML models to the VIATRA2 model space. VIATRA2 creates entities for model elements, their relations and for atomic services.
6. Import service mapping pairs to the VIATRA2 model space using a custom service mapping importer.
7. For each atomic service, discover all acyclic paths between requester and provider, provided by the mapping in Step 4. Resulting paths are stored separately in the model space for further manipulation.
8. Generate atomic UPSAMs. For each atomic service, Paths extracted from Step 7 are merged into a single network topology, corresponding to the user-perceived service infrastructure. The atomic UPSAM is obtained as an RBD from that infrastructure.
9. Generate composite UPSAM. According to the service model from Step 3, the atomic UPSAMs are combined into a single RBD.
10. Calculate the user-perceived availability with the *Symbolic Hierarchical Automated Reliability and Performance Evaluator* (SHARPE)[16] using the composite UPSAM from the previous step.

Steps 1 to 3 are done manually using a UML modeling tool like Papyrus [14] and kept unaltered as long as the ICT infrastructure and services descriptions do not change. The mapping (Step 4) is a simple XML structure where changes will eventually be performed in order to analyze different user-perspectives on a service. This can be done manually or automated. Steps 5 through 10 are then fully automatable.

The availability profile presented in Figure 2 contains elementary properties required for stead-state availability analysis: *mean time between failures* (MTBF) and *mean time to repair* (MTTR). Additionally, the *redundantComponents* property specifies internal redundancy, which can be used to implicitly define a large set of ICT components into a single object in the infrastructure model.

4.1 User-Perceived Service Availability Model Generation

Since all the atomic services within a given composite service may be executed, the paths found in Step 7 are merged into one model which corresponds to the partial infrastructure required for proper service delivery of a given service

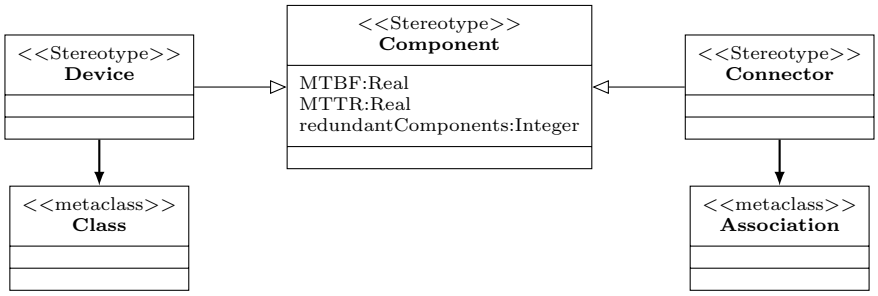

Fig. 2. Elementary availability profile

pair. The UPSAM is a transformation of that partial infrastructure. The *instanceSpecifications* of the components within that partial infrastructure have the same signature as in the original ICT infrastructure from Step 1. Therefore, they maintain the same set of properties as the classes they instantiate. It is thus guaranteed that a subsequent availability analysis will find specific required properties for every element of the UPSAM.

As a consequence of Step 7, each atomic service has its own set of paths. All ICT components forming the path are translated into serialized blocks inside the RBD, given that all of them must be working in order to traverse the path. If an ICT component has n redundant components, the RBD will have n parallel blocks with the same characteristics. This corresponds to the *redundantComponents* property of the profile.

An atomic service is available if all ICT components of at least one of its paths are available. This introduces path redundancy inside the service network, and is represented within the RBD by placing blocks related to these paths in parallel. Identical blocks within those parallel paths are then merged into a single block. Let us demonstrate this using Figure 3 as an example of an ICT infrastructure model.

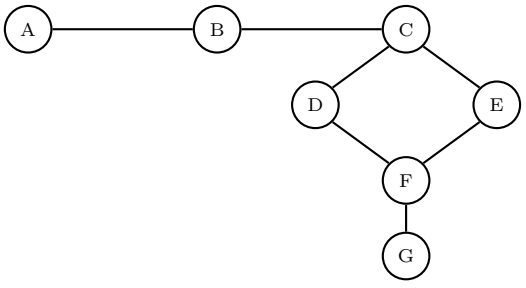

Fig. 3. ICT Infrastructure model example

All components have an internal redundancy of 0, only component B has an internal redundancy of 2 (`redundantComponent=2`). The following paths are identified from component A to G:

$A \rightarrow B \rightarrow C \rightarrow D \rightarrow F \rightarrow G$
$A \rightarrow B \rightarrow C \rightarrow E \rightarrow F \rightarrow G$

Components A, B, C and F, G are represented as a series of blocks, as they are common for both paths. Blocks D and E are in series within their respective paths but parallel to each other. Thus, they are represented as a pair of parallel blocks in between the two sequences obtained previously. Knowing that B has an internal redundancy of two extra components, it is represented as three parallel blocks B. The resulting RBD is presented in Figure 4. Note that the order of the blocks does not affect the resulting steady-state availability.

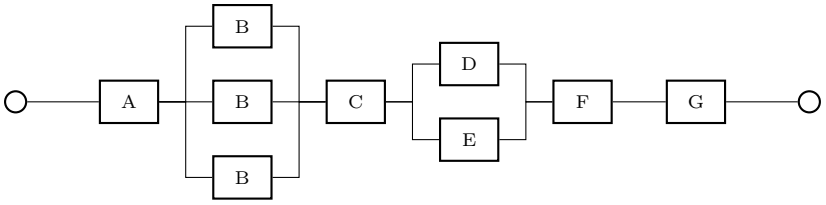

Fig. 4. Reliability Block Diagram of the example ICT infrastructure model

5 Case Study

We will now demonstrate the evaluation of user-perceived availability using the methodology from Section 4 with an exemplary *Send mail* service. It consists of resolving the *mail exchanger* (MX) address via the *domain name system* (DNS) and then sending an email message over that MX by means of the common *simple mail transfer protocol* (SMTP). During SMTP communication, the MX checks the credentials provided by the client with an external authentication server. This service represents a widespread use-case in today's service networks. In detail, the service is composed of three atomic services: *Resolve mail server address*, *Dispatch email via SMTP* and *Check authentication*. The UML activity diagram representing the *Send mail* flow of actions of the composite service is shown in Figure 5.

We simplify the fault model by taking only the steady-state availability of ICT components into account. This means we assume that all faults from classes *fail stop* to *byzantine*[1] are combined in the steady-state availability of the individual ICT components. We also disregard service discovery: The DNS server

[1] An ordered fault classification can be found in [1].

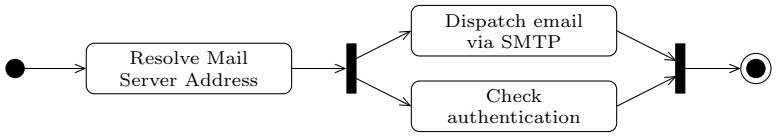

Fig. 5. Send mail service represented in UML activity diagram

address is known a priori to the client as is the authentication server address to the MX.

The underlying network on which the service is deployed is based on the network of University of Lugano, Switzerland. The network core consists of the central switches with redundant connections and is nearly identical to the real infrastructure, while the tree-formed peripheral parts connected to the core have been reduced for demonstration purposes. As described in Section 4, the ICT infrastructure is represented by a UML object diagram, where each node is an instance of a specific ICT component class described in a UML class diagram. The links between nodes are also represented as instances of associations from the UML class diagram. For simplification purposes, associations are given the maximum availability of 1 – meaning that they are always available – so that their respective RBD blocks can be omitted in latter illustrations without affecting the steady-state availability. The full topology is shown in Figure 6.

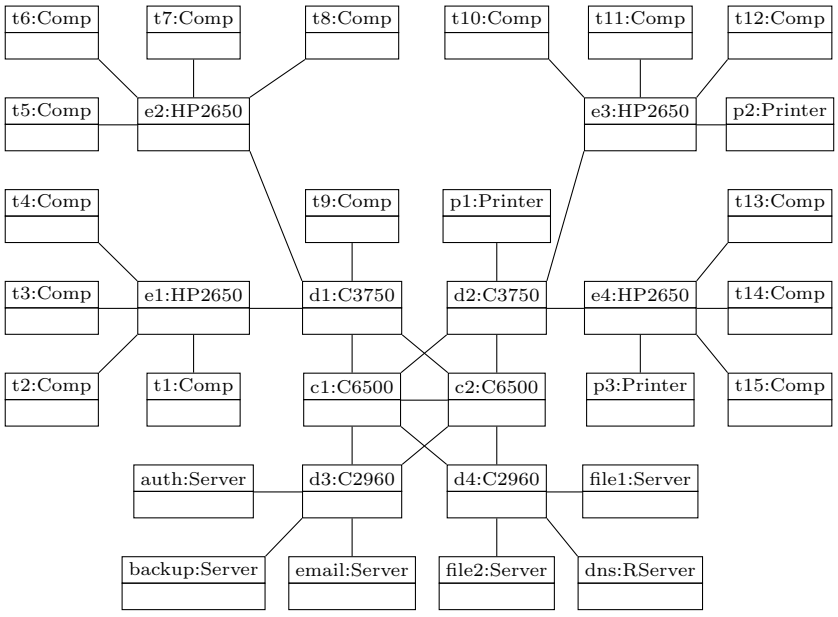

Fig. 6. Network infrastructure presented in UML Object Diagram

As an example for an ICT component description in UML, Figure 7 shows a fraction of the UML class diagram (Step 1 of the methodology) containing the description of the devices and their connections as, respectively, classes and associations. The type `RServer` represents a server containing an internal redundancy of one extra component (`redundantComponents=1`) which signifies that there are actually two servers with one server to fail over. Type `C2960` represents a switch.

Fig. 7. Predefined network elements represented in UML Class Diagram

The complete list of ICT components including relevant availability data is presented in Table 1. In the UML object diagram of the real topology in Figure 6 (Step 2 of the methodology) each node is represented by a unique identification and the respective type, in the format `id:Type`. Types *HP2650*, *C3750*, *C6500* and *C2960* are switches, the other types should be self-explanatory. Given that `RServer` has `redundantComponent=1`, the *dns* is then known to have redundancy although represented by a single node.

Table 1. Specification of ICT components

Type	Manufacturer	Model	MTBF(hours)	MTTR(hours)	RC*
C2960	Cisco	Catalyst 2960-48FPD-L	183498	0.5	0
C6500	Cisco	Catalyst 6500	61320	0.5	0
C3750	Cisco	Catalyst 3750G-24TS	188575	0.5	0
HP2650	Hewlett-Packard	ProCurve 2650	199000	0.5	0
Server	Dell	PowerEdge T620	60000	0.1	0
RServer	Dell	PowerEdge T620	60000	0.1	1
Comp	HP Compaq	DC7800	3000	24.0	0
Printer	Canon	IR3245N	2880	1.0	0

*redundantComponent

In this case study, the ICT components *t1* and *backup* were chosen as clients to compare two views on a composite service as perceived by different clients. Components *dns*, *email* and *auth* play the roles of dns server, mail server and authentication server. The mappings between atomic services and the ICT infrastructure (Step 4 of the methodology) for the clients *t1* and *backup* are given

Table 2. Service mapping pairs of the *Send mail* service for client *t1*.

Atomic Service	Requester	Provider
Resolve mail server address	t1	dns
Dispatch email via SMTP	t1	email
Check authentication	email	auth

Table 3. Service mapping pairs of the *Send mail* service for client *backup*.

Atomic Service	Requester	Provider
Resolve mail server address	backup	dns
Dispatch email via SMTP	backup	email
Check authentication	email	auth

in Table 2 and Table 3, respectively. It can be seen that only minor changes to the input models are necessary to change the user-perceived view on a service: Only the requesting instance in the mapping is changed, the network model and service description remain untouched.

In the following, we will demonstrate how to generate the UPSAM for the first atomic service, *Resolve mail server address*. Generation for the subsequent atomic services is omitted but will follow the exact same procedure. Starting from *t1* in the network shown in the infrastructure model of Figure 6, the path discovery algorithm (Step 7 in the methodology) identifies eight acyclic ways to reach *dns*:

t1→*e1*→*d1*→*c1*→*d4*→*dns*
t1→*e1*→*d1*→*c1*→*c2*→*d4*→*dns*
t1→*e1*→*d1*→*c1*→*d2*→*c2*→*d4*→*dns*
t1→*e1*→*d1*→*c1*→*d3*→*c2*→*d4*→*dns*
t1→*e1*→*d1*→*c2*→*d4*→*dns*
t1→*e1*→*d1*→*c2*→*c1*→*d4*→*dns*
t1→*e1*→*d1*→*c2*→*d2*→*c1*→*d4*→*dns*
t1→*e1*→*d1*→*c2*→*d3*→*c1*→*d4*→*dns*

Paths are then merged and transformed into a single reliability block diagram, the UPSAM, shown in the upper part of Figure 8. Basically, this procedure – corresponding to Step 8 of the methodology – reduces common nodes of different paths and excludes those which do not affect the overall availability of the service. For instance, in order to pass through *d2*, nodes *c1* and *c2* must be available, in addition to the common nodes *t1*, *e1*, *d1*, *d4* and *dns*. However, their availability implies that there is already at least one path guaranteed to be available between *d1* and *d4*. This is because associations have an availability of 1 and there are associations between *c1* and *d1*,*d4* as well as between *c2* and *d1*,*d4*. For this reason, the node *d2* does not affect the overall availability and is

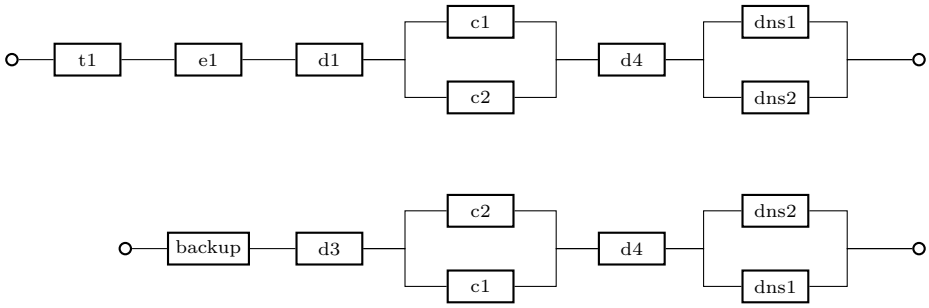

Fig. 8. *User-perceived service availablity models* (UPSAM) of atomic service *Resolve mail server address* for requesters *t1* and *backup*.

excluded from the UPSAM. Furthermore, redundant components are expanded: The *dns* component is converted into a pair of parallel blocks *dns1* and *dns2*. Figure 8 shows the UPSAM for requester *t1* side by side with the analogously created UPSAM for requester *backup*. We see only minor differences in the two models because to reach *dns*, both requesters have to use almost the same part of the network. Both have to traverse the network core, only the entry points are different. The next atomic service *Dispatch email via SMTP* paints a different picture. To reach the mail exchanger, requester *backup* does not need to traverse the network core, drastically reducing the number of blocks in the reliability block diagram. The UPSAM are depicted in Figure 9.

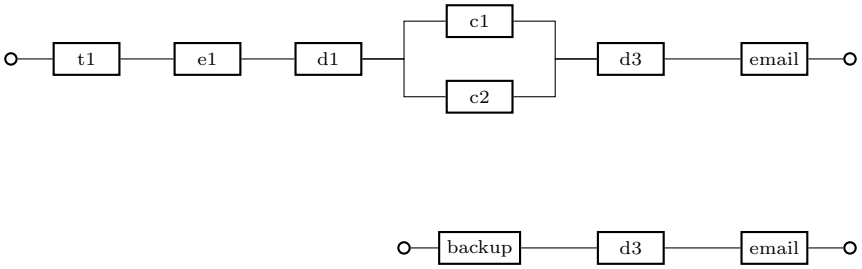

Fig. 9. *User-perceived service availablity models* (UPSAM) of atomic service *Dispatch email via SMTP* for requesters *t1* and *backup*.

Now, a composite UPSAM is created from the atomic UPSAMs according to the service description in Figure 5. Although the *Send mail* service described in the activity diagram contains a parallel execution, every single atomic service must be concluded in order to accomplish the execution of the composite service. For this reason, the resulting UPSAMs of the individual atomic services are put in series to compose the overall UPSAM of the composite service *Send mail*, as presented in Figure 10. This corresponds to Step 9 of the methodology. For

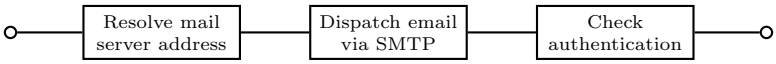

Fig. 10. Service availablity model of the *Send mail* service

the sake of clarity, atomic services have been combined into single blocks in the figure.

As the last step, we use SHARPE [16] to solve the obtained UPSAM to calculate the steady-state availability for the composite service *Send mail*. Results are shown in Table 4. We included results for the same service as requested by client *backup* to show how two different user perspectives on the same service differ in their availability.

Table 4. Service availability of *Send mail* service from different user perspectives

Service	Requester *t1*	Requester *backup*
Resolve mail server address (atomic)	0.999912118	0.999992884
Dispatch email via SMTP (atomic)	0.999910452	0.999993942
Check authentication (atomic)	0.999993942	0.999993942
***Send mail* (composite)**	**0.999816521**	**0.999980768**

In fact, altough the availability is reasonably high for both clients, it is ten times higher when the same service is requested by client *backup* instead of client *t1* (1.6 hours downtime per year for client *backup* versus 10 minutes for client *t1*). These differences are expected to be of a much higher magnitude in more heterogeneous networks with a significant variability in availability of the various component types, especially when taking into account different link qualities. This justifies the approach of considering user-peceived service availability.

6 Conclusion and Outlook

Assessing non-functional service properties like availability remains challenging. This is because service dependability depends highly on the properties of the providing ICT infrastructure. This infrastructure, however, changes for every different client. Thus, every client has another view on the service's availability. This is especially true in today's heterogeneous and widespread networks where the variability of availability among clients can be very high. Assessing system or service availability with aggregation functions might give an overview but falls short of providing a realistic picture of a service's dependability for specific clients.

We provided an automated methodology that evaluates user-perceived service availability. Given a set of input models – representing the network topology, the service description and a pair requester, provider – that part of the

ICT infrastructure providing the service for the given service pair is extracted and transformed into a reliability block diagram which is solved to obtain the steady-state availability of the given service. The methodology uses a hierarchical service model where on the highest level there is a composite service composed of atomic services which in turn map to ICT infrastructure components. The reliability block diagram for the client-specific infrastructure providing a composite service consitutes the *user-perceived service availability model* (UPSAM).

The case study demonstrates the feasibility of the approach by applying it to an exemplary mail service deployed on parts of the network of University of Lugano, Switzerland. We showed how the availability of the same service can differ considerably even in such a high-availability network when requested from two different users. The methodology is thus able to provide a fine-grained view on service availability as experienced from different points of the network.

Future work will focus on the complexity of the various methodology steps. Especially, the path discovery algorithm and creation of the composite reliability block diagram need optimization when applied to networks with a high degree of connectivity, such as wireless mesh networks. Also, combining sets of components with a low variability in user-perceived availability to reduce the size of the topology graph will be considered. Finally, extending the methodology to evaluate different dependability properties like interval availability, performability or responsiveness remains an open issue.

References

1. Barborak, M., Dahbura, A., Malek, M.: The consensus problem in fault-tolerant computing. ACM Computing Surveys 25(2), 171–220 (June 1993)
2. Bernardi, S., Merseguer, J., Petriu, D.: An UML profile for dependability analysis and modeling of software systems. Tech. Rep. RR-08-05, University of Zaragoza (May 2008)
3. Dittrich, A., Kaitovic, I., Murillo, C., Rezende, R.: A model for evaluation of user-perceived service properties. In: International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW). IEEE Computer Society (May 2013), accepted for publication
4. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. The Prentice Hall Service Technology Series from Thomas Erl, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edn. (August 2005)
5. Malek, M., Milic, B., Milanovic, N.: Analytical availability assessment of IT services. In: Nanya, T., Maruyama, F., Pataricza, A., Malek, M. (eds.) Service Availability, Lecture Notes in Computer Science, vol. 5017, pp. 207–224. Springer Berlin Heidelberg (2008)
6. Milanovic, N., Milic, B.: Automatic generation of service availability models. IEEE Transactions on Services Computing 4(1), 56–69 (Jan-Mar 2011)
7. Milanovic, N., Milic, B., Malek, M.: Modeling business process availability. In: Congress on Services - Part I. pp. 315–321. IEEE Computer Society (July 2008)
8. Murillo, C.: Model-Driven Evaluation of User-Perceived Service Availability. Master thesis, Università della Svizzera Italiana (USI), Lugano, Switzerland (January 2013)

9. Object Modeling Group: Unified Modeling Language Infrastructure (August 2011), version 2.4.1
10. Salehi, P., Hamoud-Lhadj, A., Colombo, P., Khendek, F., Toeroe, M.: A UML-based domain specific modeling language for the availability management framework. In: 12th International Symposium on High-Assurance Systems Engineering (HASE). pp. 35–44. IEEE Computer Society (November 2010)
11. Service Availability Forum: Application Interface Specification (2011), http://www.saforum.org
12. Shao, L., Zhao, J., Xie, T., Zhang, L., Xie, B., Mei, H.: User-perceived service availability: A metric and an estimation approach. In: International Conference on Web Services (ICWS). pp. 647–654. IEEE Computer Society (July 2009)
13. The Eclipse Foundation: Eclipse development environment (March 2013), http://www.eclipse.org
14. The Eclipse Foundation: Papyrus UML modeling tool (March 2013), http://www.eclipse.org/modeling/mdt/papyrus
15. The Eclipse Foundation: VIATRA2, VIsual Automated model TRAnsformations (March 2013), http://www.eclipse.org/gmt/VIATRA2
16. Trivedi, K.S.: SHARPE (symbolic hierarchical automated reliability and performance evaluator) (February 2010), http://www.ee.duke.edu/~kst
17. Xie, W., Sun, H., Cao, Y., Trivedi, K.S.: Modeling of user perceived webserver availability. In: International Conference on Communications (ICC). vol. 3, pp. 1796–1800. IEEE Computer Society (May 2003)