A Model for Evaluation of User-Perceived Service Properties

Andreas Dittrich, Igor Kaitovic, Cristina Murillo, Rafael Rezende
Advanced Learning and Research Institute (ALaRI)
Università della Svizzera italiana (USI)
Lugano, Switzerland
{*andreas.dittrich,igor.kaitovic,murilloc,ribeiror*}*@usi.ch*

***Abstract*—An ever-increasing number of both functional and non-functional requirements has resulted in growing system complexity which demands new solutions in system modeling and evaluation. As a remedy, service-oriented architecture (SOA) offers services as basic building elements of system design. Service dependability is highly dependent on the properties of the underlying information and communications technology (ICT) infrastructure. This is especially true for the user-perceived dependability of a specific pair service client and provider as every pair may utilize different ICT components.**

We provide a model for the description of ICT components and their non-functional properties based on the Unified Modeling Language (UML). Given a service description, a network topology model and a pair service client and provider, we propose a methodology to automatically identify relevant ICT components and generate a user-perceived service infrastructure model (UPSIM). We demonstrate the feasibility of the methodology by applying it to parts of the service network infrastructure at University of Lugano (USI), Switzerland. We then show how this methodology can be used to facilitate user-perceived service dependability analysis.

***Keywords*-Service networks; Service network management; Service dependability; Availability; Quality of service; Modeling; Metamodeling; Object oriented modeling; Design engineering**

I. Introduction

In the last decades, increasing requirements for both functional and non-functional properties have lead to a significant rise in system complexity. As a parallel trend, modern business operation is relying ever more on IT services. For this reason, businesses are heavily dependent on predictable service delivery with time, performance and dependability constraints. Failing to meet these requirements can cause a loss of profits or business opportunities and in critical domains, can have an even more severe impact.

In order to tame complexity and enable efficient design, operation and maintenance, various modeling techniques have been proposed. They help to understand systems by describing their components, predict their behavior and properties by analysis, specify their implementation and finally, they may enable system validation and verification. *Service-oriented architecture* (SOA) proposes a model/formalism where services are the basic building elements of system design [1].

Meeting non-functional property requirements is crucial for successful service provision. However, non-functional properties like service dependability are highly dependent on the properties of the underlying *information and communications technology* (ICT) infrastructure. This is especially true for the *user-perceived* dependability of a specific pair service requester and provider as every pair can utilize different ICT components. To assess service dependability for any client within the network, information about the overall network dependability often is not sufficient. This is why, although services are usually well-defined within business processes, assessing service dependability remains uncertain. The underlying infrastructure varies according to the position of the requesting client – represented by a person or even an ICT component – and the concrete providing service instance. Evaluation of user-perceived service dependability should employ a model of the ICT infrastructure where service properties are linked to component properties.

For simplification purposes throughout the text, *service client* and *providing service instance* are also referenced as *requester* and *provider*, respectively. A mapping of two specific instances requester and provider to the ICT infrastructure, that defines the user-perceived scope, is referred to as *service mapping pair*.

The following section provides brief background information about service network modeling. Section III gives an overview of related work. Section IV clearly states the scientific problem examined in this paper, followed by our approach to solve it in Section V. We provide a model for the description of a network of ICT components and their non-functional properties based on the *Unified Modeling Language* (UML). Given a service description, a network topology model and a service mapping pair, we propose a methodology to automatically identify relevant ICT components and generate a *user-perceived service infrastructure model* (UPSIM). Finally, Section VI demonstrates the feasibility of our approach in a representative case study by applying it to parts of the service network infrastructure of University of Lugano (USI), Switzerland. We then outline how this methodology can be used to facilitate user-perceived service dependability analysis in Section VII. Section VIII summarizes our work by pointing out the main contributions and remaining open issues.

II. Background

In this work, we adopt and extend the service definition from Milanovic et al. [2]:

Definition 1. *"Service is an abstraction of the infrastructure, application or business level functionality. It consists of a contract, interface, and implementation."*

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. A composite service is composed of and only of two or more atomic services, while an atomic service can be part of any number of composite services. Ideally, atomic service functionality should not be redundant, that is, every atomic service provides a different functionality.

The indivisibility of an atomic service obviously is in the eye of the beholder. Usually, the granularity is defined by the re-usability within the business process model. For instance, an *email* service can be divided into the atomic services *authenticate*, *send mail* and *fetch mail*. These could also be split into finer grained services. However, if the complete business process is well described with the current granularity in such a way that *authenticate*, *send mail* and *fetch mail* can be reused within other services without modifications, there is no need for further reduction. In this sense, *email* corresponds to a *composite service* constituted by the atomic services *authenticate*, *send mail* and *fetch mail*.

The design and modeling of an ICT infrastructure is usually based on principles of *model-driven development* (MDD) [3], very often using the *Unified Modeling Language* (UML) [4]. MDD provides a better understanding of arising problems and their potential solutions through system abstraction.

The standardized UML provides a set of graphic notations to model structural and behavioral characteristics. In addition to a standard set of fourteen different diagrams, UML has two important customization mechanisms: profiles and stereotypes. These features enable designers to aggregate detailed information and better represent complex systems with diverse properties. Stereotypes specify new modeling elements, with properties called stereotype attributes. Profiles describe model semantics with stereotypes and constraints. Thus, profiles and class are additional instruments to tailor UML models and their elements (e.g. classes, associations, objects) for specific needs. To be applicable to UML diagrams, when designing a profile each of its stereotypes must extend a UML element. For instance, a stereotype S, extending the class element and containing the stereotype attribute A can only be applied to classes, which are then denominated stereotyped classes and inherit the attribute A.

III. Related work

Service-oriented architecture (SOA) [1] proposes the design and implementation of services as discrete system components. *Service-oriented system engineering* (SOSE) [5] specifies the development phases of service-oriented methodology: specification, analysis and validation. A business process model contains a clear specification of which services the system must provide. The service concept in this paper stems from [2], where composite services are described as a composition of atomic services. Additionally, there is a direct link between atomic services and the ICT infrastructure, its hardware and software components required to provide the specified functionality.

Service dependability properties such as, availability, performability [6] or responsiveness [7] depend directly on the properties of the ICT infrastructure on which the service is deployed. Milanovic et al. propose a methodology for the automatic generation of service availability models based on run-time monitoring [2], [8], [9]. Their methodology is focused on steady-state availability analysis and relies on a *configuration management database* (CMDB) system, which collects data from the network infrastructure in order to perform the deployment mapping. This way, the description of the ICT infrastructure can be completely automated. As a drawback, it requires an external tool, such as a network management tool, to provide this information. Also, the methodology can only be used to assess steady-state availability. Although the authors use *Business Process Model and Notation* (BPMN) [10] to model services, UML activity diagrams could be considered as an alternative.

A different definition of services is used in the *Availability Management Framework* (AMF) [11]. AMF specifies components as basic framework entities that consist of a set of software or hardware resources. In contrast to [8], where infrastructure and services are modeled independently, AMF components are intrinsic service providers, which can be grouped into bigger logical units called *service units* (SU).

Dependability analysis modeling (DAM) [12] also associates services and ICT components by defining the relation of a service and its underlying infrastructure. However, since the methodology proposes the modeling of the system entirely using UML, changes on the service mapping pair have to be reflected within the model. Even with clear annotation, this procedure can be tedious and error prone. Moreover, *DAM* is conceived as a complex UML Profile (extended UML model) aiming at dependability modeling, thus excluding other non-functional properties.

Models are abstractions of a system and/or its environment. Every model conforms to a metamodel, which defines the syntax of the model as an explicit specification describing the relevant concepts and relations between these

concepts [13]. The idea of obtaining one model from another characterizes a model to model transformation, where the input and output models can either share the same metamodel or have a completely different syntax. Usually, such transformations rely on identifying graph patterns as model elements and match them to given structures of the metamodel, as accurately described by Czarnecki et al. in an extensive survey on model transformation [14].

IV. Problem statement

Service networks enable a plethora of business processes and applications. Assessing non-functional properties like dependability of these processes poses several challenges. First, the dependability of a service depends on the ICT components it runs on. Second, the ICT components change for every pair requester and provider, possibly even for every service invocation of a specific pair due to the dynamicity of the environments: Network topologies change, components are upgraded or undergo maintenance after failure, services are migrated and so on.

Assuming the concepts of composite and atomic services from Section II, we introduce the definition of a *user-perceived service infrastructure model* (UPSIM):

Definition 2. *Given an ICT infrastructure N, a providing service instance S_p and a service client S_c with $S_p, S_c \in N$. A User-perceived service infrastructure model $N_{upsim} \subseteq N$ is that part of N which includes all components, their properties and relations hosting the atomic services used to compose a specific service provided by S_p for S_c.*

A methodology is needed to support the assessment of user-perceived non-functional properties. The methodology needs:

1) A model to describe ICT components including specific non-functional properties
2) A formalism to model networks as relational structures of those ICT components with the ability to assign roles (e.g. requester, provider) to specific components
3) A service model to describe composite and atomic services and map them to the relational structure that represents the network
4) Generation of a specialized UPSIM that includes only those ICT components specific for the communication between a given pair requester, provider during execution of a previously described service, according to Definition 2.

The Assessment of user-perceived non-functional properties can then be done on the UPSIM generated in the last step. All steps need to be automated as far as possible to support quick model updates in dynamic environments and to eliminate human errors during update/upgrade procedures.

As a side goal, the methodology should be defined and implemented using well known standards and freely available tools to support external verification and to facilitate its dissemination.

V. Methodology

Given a model of the network topology, the description of a service composed of atomic services and a pair of service requester and provider, we apply a model-to-model transformation to obtain the *user-perceived service infrastructure model* (UPSIM). The approach uses a subset of *Unified Modeling Language* (UML) elements as it is standardized, easily expandable, well supported by numerous tools and thus, widely accepted in both industry and academia. Moreover, *profiles* and *stereotypes* are applied to impose specific attributes to ICT components, resulting in a coherent model with a standardized description:

- *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. Object diagrams are used to describe both the complete network structure as well as the UPSIM.
- *Activity diagrams* are used for service description and represent the service as a flow of *actions*.

Open-source tools are chosen for the implementation. Eclipse [15] is a multi-language software development environment with numerous extension plug-ins. This includes the UML2-compliant [4] modeling tool Papyrus [16] and the model transformation plug-in VIATRA2 [17] with embedded support for UML models. VIATRA2 complements the Eclipse framework with a transformation language based on graph theory techniques and abstract state machines. It also provides its own model space, so it can import models to an intermediate representation where they can be manipulated before a subsequent model generation.

A. Model specifications

Figure 1 depicts the context of a UPSIM as a UML class diagram. The following subsections explain the different parts of the diagram in detail.

1) ICT Infrastructure model: The *ICT infrastructure* in the left part of Figure 1 aggregates a set of interconnected *ICT components*. Network nodes and the communication between them exhibit very distinct properties. For this reason, *ICT components* are subdivided into *Device* and *Connector* types. Following standard UML notation, Figure 1 shows that every *Connector* must be associated to two *Devices*, which may have any number of *Connectors*. *Devices* and *Connectors* are respectively modeled as *classes* and *associations* in a UML class diagram. The ICT infrastructure is then presented in a UML object diagram, where network

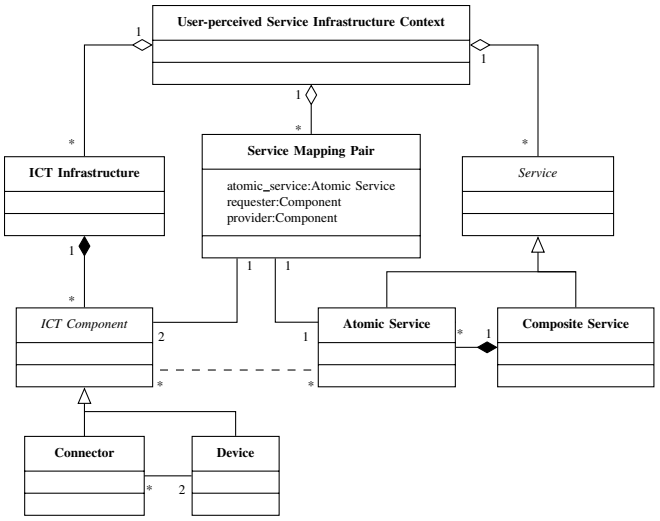

Figure 1. Context of a user-perceived service infrastructure model (UPSIM).

nodes are *instanceSpecifications* of those *classes*, and communication is represented by the corresponding *links*, which are instances of *associations*. To ensure that two different instances of the same *class* have also the same properties, every *class* may only have static attributes.

The same set of attributes can be applied to common model elements through *profiles* to facilitate dependability analysis that requires specific properties to be present for each model element. A *stereotype* containing these specific properties (e.g. average response time, failure rate) can be employed to guarantee that every *ICT component* inherits them and thus, meets the requirements of the analysis.

2) Service model: Composite services are modeled with UML activity diagrams using *atomic services* as building blocks, as depicted in the right part of Figure 1. A composite service consists of initial and final nodes, *atomic services* and *join* and *fork* figures. In this particular diagram, for instance, *atomic service 2* and *atomic service 3* are executed in parallel. According to Milanovic et at. [2], atomic services are abstractions of the ICT infrastructure, application or business level functionality. At this point, atomic services are still considered abstract functionalities and are not yet related to a set of concrete ICT components. This relation exists later during service execution, where atomic services map to a set of ICT components including requester, provider and connecting ICT components and inherit their properties.

Figure 2 presents the UML activity diagram of a simple composite service. It is assumed that each atomic service is being executed – in series or in parallel. Instead of using decision nodes, separate decision branches are modeled as separate services.

3) Mapping of atomic services to ICT components: Since service properties strongly depend on the underlying infrastructure, a correct mapping between services and the

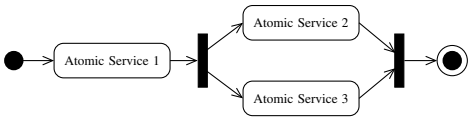

Figure 2. Service description as UML activity diagram.

ICT components that enable them is required for further analysis. This mapping is represented by the dashed line in Figure 1. Atomic services are instantiated by a service mapping pair when defining requester and provider. The mapping, provided as an XML file (see Figure 3), contains a unique description of the service mapping pair requester and provider for every atomic service.

The service mapping pair in the center of Figure 1 gives the initial and final boundaries of the ICT infrastructure used by a specific atomic service. Other related ICT components depend on the possible paths between the ICT infrastructure and provider nodes. The methodology thus uses a path discovery algorithm to identify these paths as initially proposed in [8].

The reasons for system changes are manifold: user mobility, network topology changes due to new or failing components, service migration and so on. Separating the infrastructure model, the service description and the mapping allows to efficiently handle dynamic system changes by updating only individual models. For instance, the UPSIM can be generated for different user perspectives with only minor changes to the mapping of atomic services while the abstract service model and network model remain untouched. In a mobile scenario, where users can be at different positions within the network but still use the same service, the network model and mapping need to be updated while the service description remains the same. In general, changes to the network topology require updating only the network model and mapping but not the service description. Migrating a service from one provider to another requires updating only the mapping while substituting a service – replacing one service composition with another one that provides the same functionality – requires changing only the service description and mapping but not the network model.

B. Methodology overview

As output, the UPSIM is also presented as UML object diagram and includes the ICT components relevant for the specific service mapping pair. All redundant paths between

```
<atomicservice id="atomic_service_1">
    <requester id="component_a"></requester>
    <provider id="component_b"></provider>
</atomicservice>
```

Figure 3. Example of the XML code representing the mapping of a single atomic service.

requester and provider are included. A methodology to generate the UPSIM is presented in Figure 4 while each step is described in detail below. Steps 1 to 4 provide the input models specified in Section V-A for later transformation and UPSIM generation. The three input models are in the left part of the figure. All model tranformations and their auxiliary measures are depicted within the center rectangle labeled *VIATRA2* and described in Steps 5 to 7. The UPSIM generated in Step 8 can be seen at the bottom of the figure.

1) Identify ICT components and create the respective UML classes for each class type. According to the subject of a subsequent dependability analysis, a UML profile can be applied to classes in this step. Results in a class diagram containing the description of every *ICT component*. See also Section V-A1.
2) Model the *ICT infrastructure* using UML object diagrams with instances of the classes from Step 1. Results in an object diagram of the complete infrastructure. See Section V-A1.
3) Identify and iteratively describe services using UML activity diagrams with *atomic services* as building blocks (Actions). Results in a collection of service models with no correlation to the infrastructure. See Section V-A2.
4) Generate *service mapping pairs* by mapping *atomic services* from Step 3 to *ICT components*. Results in an external XML file (see Figure 3). See Section V-A3.
5) Import ICT infrastructure and service UML models to the VIATRA2 model space using its native UML importer. VIATRA2 creates entities for model elements and their relations. Also, atomic services are transformed into entities of the model space.
6) Import *service mapping pairs* to the *VIATRA2 model space* using a custom service mapping importer. See Section V-C.
7) Discover all possible paths between *requester* and *provider*. Given a composite service, its atomic services and their *service mapping pairs*, an algorithm discovers all paths between the ICT components contained in these mappings. Resulting paths are stored separately in the model space for further manipulation. This step is described in V-D.
8) Generate the UPSIM. Paths extracted from Step 7 are merged into a single network topology, corresponding to the user-perceived service infrastructure. The UPSIM is obtained as a UML object diagram. See Section V-D. See Section V-E.

Steps 1 to 3 are done manually using a UML modeling tool like Papyrus [16] 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 8 are then fully automated.

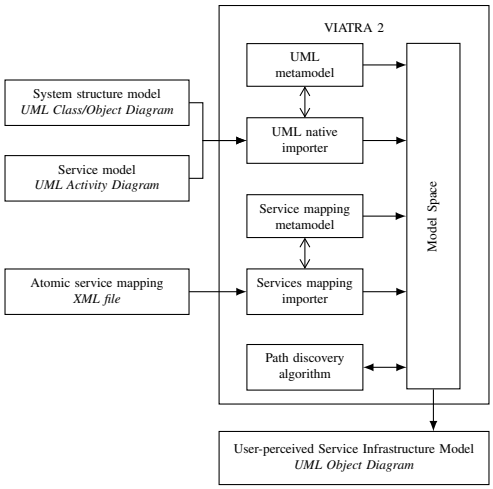

Figure 4. Implementation of the model transformation.

C. VIsual Automated model TRAnsformations tool

The approach relies on a *model-to-model transformation* (M2M) using VIATRA2, that receives multiple input models conforming to specific source metamodels and produces a single target model conforming to a target metamodel (see Figure 4).

A metamodel describes the abstract syntax of a model, its elements, their properties and relationships and modeling rules [14]. In VIATRA2 [17], a metamodel is described using a specific syntax provided by the *VIATRA textual metamodeling language* (VTML). Models and metamodels are stored in the *Visual and Precise Metamodeling* (VPM) model space, which provides a flexible way to capture languages and models from various domains by identifying their entities and relations. To import the UML models into this model space, VIATRA2 provides a UML native importer and a UML metamodel.

In order to import the service mapping pair into the model space, we developed a custom service mapping metamodel and importer plug-in based on the hierarchy presented in Figure 3. The task of the services mapping importer is to parse the XML file, traverse the content tree and find appropriate VPM entities in the metamodel corresponding to the type of each element. It is implemented in the Java programming language and added to the project workspace as an Eclipse plug-in.

Additionally, the *VIATRA2 textual command language* (VTCL) provides a flexible syntax to access the VPM model space. It is based on mathematical formalisms and provides declarative model queries and manipulation [18]. This language is especially useful in this methodology to implement the path discovery algorithm.

D. Path discovery algorithm

As stated, the service mapping pair gives the initial and final boundaries of the ICT infrastructure used by a specific atomic service. A path discovery algorithm is then used to identify all possible paths between requester and provider. This approach implies that topology changes on the ICT infrastructure do not require adjustments to the service model or the atomic service mapping, given that requester and provider are still running on the same ICT components.

For every service mapping pair, the algorithm discovers a set of paths between the respective requester and provider, and stores the visited entities in a reserved tree structure inside the model space.

The complexity of such algorithms grows significantly with the size of the ICT infrastructure. In order to find all possible paths, every node must be visited through all available edges. For this reason, the time complexity of the algorithm is even more sensitive to the number of edges, reaching $O(n!)$ for a fully interconnected graph of n nodes. However, real networks usually contain few loops, while most clients are located in tree-like structures with a low number of edges.

We chose to implement an depth-first search (DFS) algorithm [19] with a path tracking mechanism to avoid live-locks within cycles.

E. User-perceived service infrastructure model generation

The final step of the methodology is the generation of the output model by merging the paths obtained from the algorithm. Since all the atomic services within a given composite service are executed, their paths are also merged into one UML object diagram which corresponds to the partial infrastructure required for proper service delivery.

The *instanceSpecifications* of the UPSIM object diagram have the same signature as in the original ICT infrastructure. Therefore, they maintain the same set of properties as the classes they instantiate. It is thus guaranteed that a subsequent service dependability analysis will find specific required properties for every element of the user-perceived ICT infrastructure.

VI. Case Study

In this section, the feasibility of the methodology is demonstrated by generating the UPSIM of a printing service in a real network from different user perspectives. The ICT infrastructure is modeled with focus on service dependability assessment, more specifically, steady-state availability. The topology depicted in Figure 5 is based on the network of University of Lugano (USI), Switzerland. The network core, consisting of the central switches with redundant connections, is nearly identical to the real infrastructure while the tree-formed peripheral parts connected to the core have been reduced for demonstration purposes.

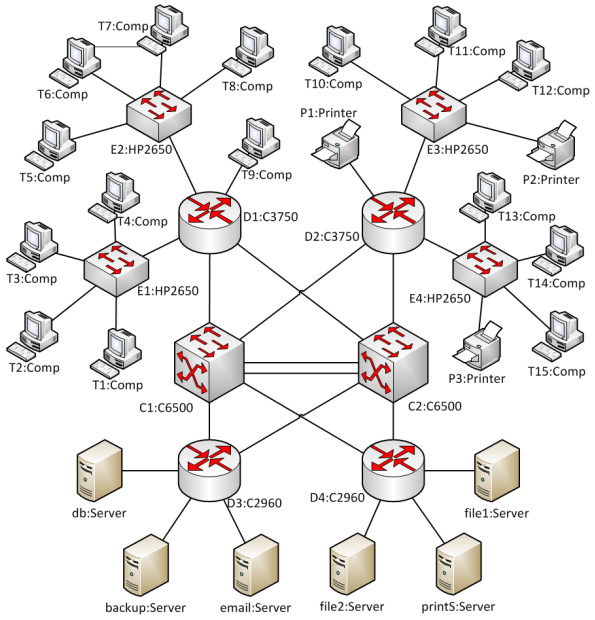

Figure 5. Network infrastructure based on university campus network.

The network provides a number of atomic services (e.g.: authenticate, print document, request backup) where each service has at least one provider. Atomic services can compose composite services (e.g. printing, backup), which are requested by different clients at various positions within the network.

The following sections explicitly reflect the steps of the proposed methodology defined in Section V-B.

A. Identify and model ICT components

In order to use the UPSIM for user-perceived service availability assessment, we developed a simple UML profile containing relevant properties for such analysis. We then applied it to the infrastructure. Each ICT component (device or connector) has intrinsic dependability attributes: *Mean-time-between-failures* (MTBF), *mean-time-to-repair* (MTTR) and *redundant components*. MTBF and MTTR refer to component failure and repair, respectively. This means that MTBF encapsulates all possible failure causes within the scope of a component, e.g. hardware or software. As can be seen in the availability profile (Figure 6), although *Device* and *Connector* inherit the same attributes from *Component*, they are distinguished in order to be applied – respectively and exclusively – to *Class* and *Association* elements.

A network profile (see Figure 7) has also been developed to guarantee the uniformity of the infrastructure presented in Figure 5. It defines an abstract stereotype *Network Device* to capture common properties of identified network components: *Router*, *Switch*, *Printer* and *Computer*. The latter one is abstract and specializes into *Client* and *Server*. *Communication* is a stereotype dedicated to the association

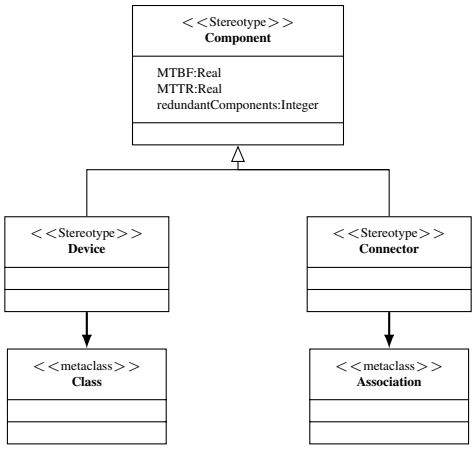

Figure 6. Simple profile to extend UML for availability modeling.

between classes, and has *channel* and *throughput* attributes. In practice, it corresponds to communication links between devices. With this profile, a set of attributes can consistently be imposed to stereotyped classes for later model transformation. Albeit only an illustrative representation in this case study, the network profile of Figure 7 can support a variety of attributes depending on the application.

The different types of ICT components identified in the infrastructure model depicted in Figure 5 are modeled in a set of stereotyped classes (see Figure 8). This UML class diagram contains all network elements with their predefined availability properties. At this point, we evaluate all constituting devices and their possible connections. For instance, *D1* and *D2* are different instances of the same device labeled *C3750*, which is known to be a switch. Therefore, the corresponding class is created, with *Component* and *Switch* stereotypes applied from the availability and network profiles, respectively. When all devices are represented as classes, their possible links are represented by associations stereotyped as *Component* and *Communication*.

B. Model the ICT infrastructure

The infrastructure object diagram (see Figure 9) is built with instances of classes and associations (namely *instanceSpecification* and *links*) from the previous step, and reflects the topology of the network in Figure 5.

Since attributes are statically defined in classes, those instances are already well-defined components. Also, given that links are instances of associations, the possibility for connections is ruled by those existing associations.

C. Identify and model services

In the exemplary printing service of this case study, a centralized print server holds all printing requests from authenticated clients. Using the same authentication credentials, a person is then able to conclude the requests by printing the physical documents on any printer connected to

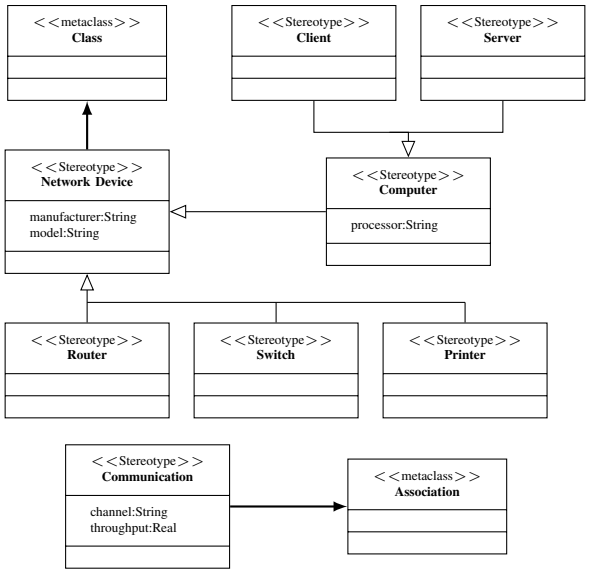

Figure 7. Network profile with types of elements and their basic properties.

the network. This implementation is particularly suitable for universities or similar organizations to permit load balancing on printers and, consequently, reduce the execution time of this service. The printing service is composed of five atomic services, used in sequential order as depicted in the activity diagram in Figure 10. Following is a description of the atomic services:

1) Request printing. *Client login to print server and send documents to be printed.*
2) Login to printer. *User login to printer. Authentication credentials are sent from printer to print server.*
3) Send document list. *After successful authentication, the print server sends a list of queued documents for the specific user to the chosen printer.*
4) Select documents. *User selects document(s) to print from the list. Printer requests specified document(s) from the print server.*
5) Send documents. *Print server sends requested document(s) to the printer. Document(s) are in turn processed by the printer.*

The printing service description (see Figure 10) remains generic and abstract. It is a composition of exclusively atomic services, which are not yet mapped to ICT components. This means that in case of changes to the ICT infrastructure, the printing service description remains identical. Thus, the same service description can be used to describe a service for arbitrary pairs in any network that provides the atomic services.

D. Generate service mapping pairs

Each atomic service from the service description is now mapped to infrastructure elements by means of service mapping pairs, which associate each atomic service with

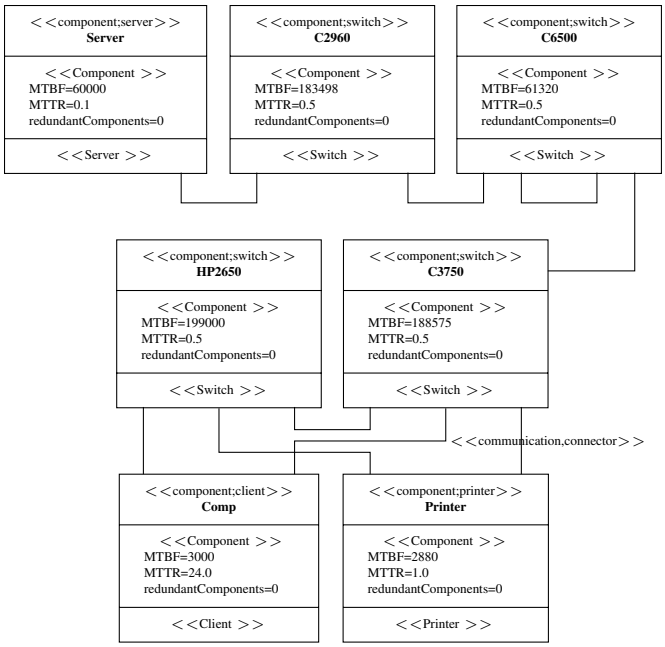

Figure 8. Predefined network elements classes.

requester and provider ICT components. Mapping is the key mechanism to support dynamicity as it allows to change service requesters and providers with minimal effort.

According to the printing service description of Figure 10, the service mapping should contain at least five pairs with their atomic service as unique key. Additional service mapping pairs could be listed in the mapping file to support other services. However, they will be ignored when the corresponding atomic service is irrelevant for the analyzed service.

To illustrate the proposed methodology for a specific user-perspective, we explicitly select requesters and providers of all service mapping pairs: We chose an actor requesting the printing service from client *T1* and printing the document on printer *P2* (see Figure 5). The print server used is *printS*. The corresponding mapping pairs are listed in Table I.

E. Import ICT infrastructure and service UML models to VIATRA2 model space

Next, we are running the VIATRA2 native UML2.2 importer on all files containing the set of input diagrams

Table I
MAPPING OF ATOMIC SERVICES (AS) AND ICT COMPONENTS TO DEFINE SERVICE MAPPING PAIR REQUESTER(RQ), PROVIDER(PR)

AS	***RQ***	***PR***
Request printing	T1	printS
Login to printer	P2	printS
Send document list	printS	P2
Select documents	P2	printS
Send documents	printS	P2

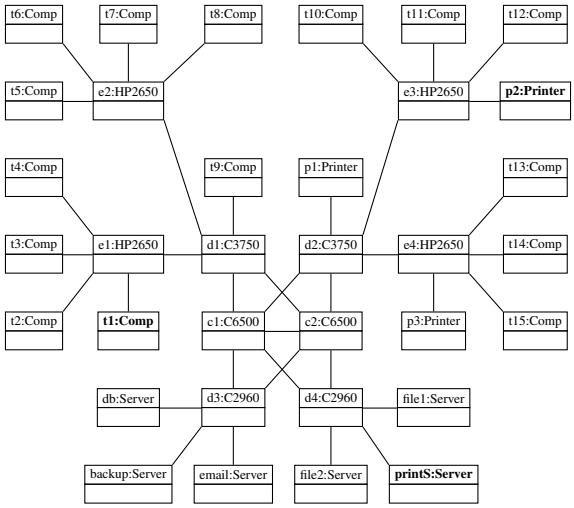

Figure 9. Network infrastructure presented in UML object diagram.

(Profiles, class diagram, object diagram and activity diagram). This step is completely automated.

F. Import service mapping pairs to VIATRA2 model space

As described in Section V-C, this step is accomplished with a custom-developed service mapping metamodel and importer plug-in. The XML file is parsed and the content tree traversed to identify elements. This step is completely automated.

G. Path discovery for service mapping pairs

In this step, the elements of each service mapping pair are matched to ICT components of the infrastructure (see Figure 5). The algorithm sees the infrastructure as a graph and iteratively extracts all possible paths between two vertices requester and provider. For instance, the first service mapping pair of Table I leads to the following paths:

t1→e1→d1→c1→d4→printS,
t1→e1→d1→c1→c2→d4→printS,
...

These paths are then added to VIATRA2 model space for further manipulation. The path discovery algorithm has been implemented using the VTCL language provided by VIATRA2. This step is completely automated.

H. Generate the UPSIM

The last step comprises matching the elements of the paths obtained in the previous step (Section VI-G) to the complete infrastructure given by the second step (Section VI-B). This step is completely automated and behaves like a filter on the complete topology, where only nodes which appear at least once in the discovered paths are preserved. Multiple occurrences are ignored. The output model is a UML object

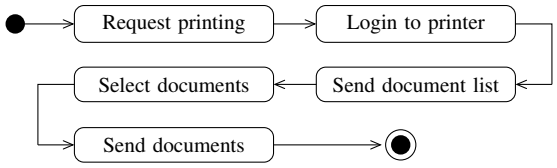

Figure 10. Printing service description in UML activity diagram.

diagram containing only that fragment of the infrastructure required for the execution of the printing service from client *t1* to printer *p2*, all redundant paths taken into account. This user-perceived service infrastructure model is shown in Figure 11. The types of *InstanceSpecifications* are adopted from the input infrastructure object diagram. Therefore, the service and network properties (MTBF, MTTR, etc.) are automatically inherited from the instantiated classes. Services and dependability properties are now correlated, facilitating extraction for further analysis.

To generate the UPSIM for a different perspective, say, the printing service from client *t15* to printer *p3* through the same printing server, we only have to make minor adjustments to the service mapping. Figure 12 shows the UPSIM from this user perspective.

VII. Outlook – Service dependability analysis

The generated UPSIM can be used to visualize the set of ICT components and their connections relevant for a particular pair requester and provider. This alone is very helpful in case of service problems, as it provides a quick overview on which ICT components can be the cause. More importantly, the UPSIM can be used to facilitate analysis of various user-perceived dependability properties (e.g.: availability, performability, responsiveness). For example, it can be used to assess user-perceived steady-state availability for a service, that is, the ratio of operational time over the livetime of a service, observed by a particular user. Such analysis can be performed by transforming the UPSIM to a *reliability block diagram* (RBD) or *fault-tree* (FT), in which entities correspond to components of the UPSIM. The availability for individual components can be calculated using the component attributes MTBF and MTTR, as seen in Formula 1. The service availability can then be computed using the RBD or FT. We present this complementary transformation to RBDs in [20].

$$A_{comp} = 1 - \frac{MTTR}{MTBF} \tag{1}$$

In [8], a model is proposed to assess user-perceived steady-state availability and demonstrated with one exemplary service. The methodology in this paper now provides a partly automated implementation for this. The main advantage is that other service dependability properties, not exclusively steady-state availability, can be evaluated for different pairs requester and provider with only minor changes to

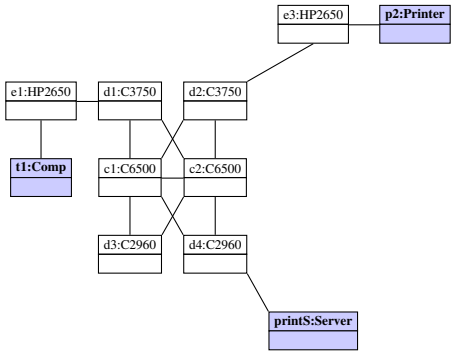

Figure 11. UPSIM for printing service requested by client T1 to printer P2, through server printS.

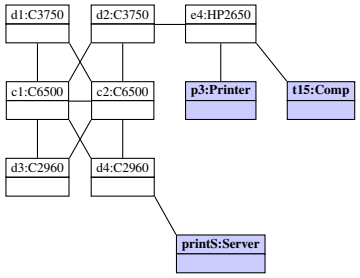

Figure 12. UPSIM for printing service requested by client T15 to printer P3, through server printS.

the mapping file. Our methodology then generates different UPSIMs with the specific user-perceived view for later analysis. Moreover, changes to intrinsic properties of network devices (MTBF, redundant components, manufacturer, etc.) can be performed directly in the class description and so reflect to all objects in the service infrastructure model. That way, the methodology provides a straight-forward way to enable different types of user-perceived dependability analysis, depending on the properties for each class of components.

VIII. Conclusion

Dependability assessment in modern service networks remains challenging. Ever more often, a system-view of dependability does not properly reflect the variable distribution of dependability within dynamic networks and is thus only of statistical relevance. As a potential remedy, this paper provides a methodology to facilitate the evaluation of *user-perceived* service dependability, that is, the dependability valid for a specific pair service requester and provider.

We achieve this by providing a set of models that describe (1) an ICT infrastructure, including non-functional properties for every entitity of the infrastructure, (2) an abstract service as a composition of atomic services, which are indivisible with respect to their functionality and (3) a mapping of atomic services to ICT components that enable

these services. The latter accounts for the fact that the dependability of a service depends highly on the dependability of the ICT infrastructure it is implemented on. The separation of the generation and usage of the individual proposed models allows the methodology to be well suited for dynamic environments.

Furthermore, we provide a methodology that uses those models and a specific pair service requester and provider to find all ICT components relevant for service provision of this pair, preserving their context within the network. We then generate a *user-perceived service infrastructure model* (UPSIM) that can be used in subsequent user-perceived dependability analysis.

To demonstrate the proposed methodology, we applied it to parts of the service network infrastructure at University of Lugano (USI), Switzerland. We showed how to generate the UPSIM for an exemplary printing service and how this UPSIM can be used, for example, to evaluate the user-perceived availability of the printing service for any given service requester and provider. It is also shown how the UPSIM can be used to evaluate other dependability properties. As a side effect, the methodology provides a practical way to automatically identify and visualize dependability-relevant ICT components, to give a quick overview on where the service problem might be caused.

Although the demonstration network is based only on parts of the service network at USI, the proposed methodology is scalable and applicable to complex, dynamic networks as well. We showed how different types of dynamicity affect only specific models so that in most scenarios, the majority of models remain unchanged. Also, the methodology implementation is automated whenever no human input or decision is necessary and it is thus possible to quickly update the UPSIMs in case of dynamic changes to the network or its services. More research is needed to demonstrate the applicability of the methodology to complex infrastructures such as cloud computing.

REFERENCES

[1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, 1st ed., ser. The Prentice Hall Service Technology Series from Thomas Erl. Prentice Hall, August 2005.

[2] N. Milanovic and B. Milic, "Automatic generation of service availability models," *IEEE Transactions on Services Computing*, vol. 4, no. 1, pp. 56–69, Jan-Mar 2011.

[3] B. Selic, "The pragmatics of model-driven development," *IEEE Software*, vol. 20, no. 5, pp. 19–25, Sep-Oct 2003.

[4] *Unified Modeling Language Infrastructure*, 2nd ed., Object Modeling Group, August 2011. [Online]. Available: http://www.omg.org/spec/UML/2.4.1

[5] W.-T. Tsai, "Service-oriented system engineering: a new paradigm," in *IEEE International Workshop on Service-Oriented System Engineering (SOSE)*. IEEE Computer Society, October 2005, pp. 3–6.

[6] I. Eusgeld, J. Happe, P. Limbourg, M. Rohr, and F. Salfner, "Performability," in *Dependability Metrics*, ser. Lecture Notes in Computer Science, I. Eusgeld, F. C. Freiling, and R. Reussner, Eds. Springer, 2008, vol. 4909, pp. 245–254.

[7] A. Dittrich and F. Salfner, "Experimental responsiveness evaluation of decentralized service discovery," in *International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*. IEEE Computer Society, April 2010, pp. 1–7.

[8] M. Malek, B. Milic, and N. Milanovic, "Analytical availability assessment of IT services," in *Service Availability*, ser. Lecture Notes in Computer Science, T. Nanya, F. Maruyama, A. Pataricza, and M. Malek, Eds. Springer, 2008, vol. 5017, pp. 207–224.

[9] N. Milanovic, B. Milic, and M. Malek, "Modeling business process availability," *IEEE Congress on Services - Part I*, pp. 315–321, July 2008.

[10] *Business Process Model and Notation (BPMN)*, 2nd ed., Object Modeling Group, March 2013. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/PDF

[11] *Application Interface Specification*, Service Availability Forum, March 2013. [Online]. Available: http://www.saforum.org

[12] S. Bernardi, J. Merseguer, and D. Petriu, "An UML profile for dependability analysis and modeling of software systems," University of Zaragoza, Tech. Rep. RR-08-05, May 2008.

[13] J. Bézivin and O. Gerbé, "Towards a precise definition of the OMG/MDA framework," in *16th Annual International Conference on Automated Software Engineering (ASE)*. IEEE Computer Society Press, November 2001, pp. 273–280. [Online]. Available: 10.1109/ASE.2001.989813

[14] K. Czarnecki and S. Helsen, "Feature-based survey of model transformation approaches," *IBM Systems Journal*, vol. 45, no. 3, pp. 621–645, 2006.

[15] T. E. Foundation. (2013, March) Eclipse development environment. [Online]. Available: http://www.eclipse.org

[16] ——. (2013, March) Papyrus UML modeling tool. [Online]. Available: http://www.eclipse.org/modeling/mdt/papyrus

[17] ——. (2013, March) VIATRA2, VIsual Automated model TRAnsformations. [Online]. Available: http://www.eclipse.org/gmt/VIATRA2/

[18] D. Varró and A. Balogh, "The model transformation language of the VIATRA2 framework," *Science of Computer Programming*, vol. 68, no. 3, pp. 187–207, October 2007.

[19] S. Even, *Graph Algorithms*, 2nd ed., G. Even, Ed. Cambridge University Press, November 2011.

[20] A. Dittrich and R. Rezende, "Model-driven evaluation of user-perceived service availability," February 2013, available on request.