NEXT GENERATION SERVICE ENGINEERING

Daniel Amyot[1], Hanane Becha[1,2], Rolv Bræk[3] and Judith E. Y. Rossebø[3,4]

[1] SITE, University of Ottawa, Canada, damyot@site.uottawa.ca
[2] Nortel, Ottawa, Canada, hananebe@nortel.com
[3] NTNU, Department of Telematics, Trondheim, Norway, rolv.braek@item.ntnu.no
[4] Telenor R&I, Fornebu, Norway, judith.rossebo@telenor.com

ABSTRACT

Service engineering is the process of service development from domain analysis and requirements capture, through specification, design and implementation, to deployment and adaptation on service delivery platforms. Ideally one would like to specify and analyse services on a high level of abstraction, using modelling concepts close to the user and problem domain rather than at the platform and implementation domain, and then be able to derive design components and implementations from service models with a high degree of automation. It is argued in this paper that this conception is approaching reality and so is worth while pursuing to face the challenges of service engineering in a NGN context. The basis for this is new approaches to model services precisely, to analyse goals and tradeoffs concerning variability and context, and to transform service models into platform independent models from which implementations are automatically generated. Interestingly, the service models can provide information and mechanisms that help dynamic composition and adaptation at runtime. The approach is illustrated using a multimedia call service with access control requirements.

Keywords— Collaboration, Goals, NGN, Scenarios, Service Models, UML, User Requirements Notation

1. INTRODUCTION

With the advent of Next Generation Networks (NGN), we see two important trends:

- Unification of underlying network technologies and computing platforms enabling network and service convergence.
- Diversification of services as well as equipments at the network edges.
- Shifting the business focus from connectivity and traffic to services and content creates a need for rich and diverse *service offerings* that can be rapidly and cost-effectively developed.

The introduction of NGN represents an opportunity to look ahead and seriously consider what *Next Generation Service Engineering* (NGSE) could be and discuss a roadmap for getting there. We observe that service engineering in the past has been largely hampered by being too focused on the underlying network and computing technologies and too little focused on the properties and needs of the application domains. This has certainly been the case in the telecommunication domain, with (Advanced) Intelligent Networks and more recently with Parlay/OSA. Less obviously perhaps, it has also been the case in the business and information system domains with CORBA, Component Object Model (COM) and more recently Web Services and Service Oriented Architecture (SOA). Although they provide abstraction in terms of interfaces, the concepts are mainly drawn from the underlying computing and networking technologies. When these technologies evolve, the service definitions have to change, leading to high life cycle costs and poor investment protection.

Conceptual abstraction can help mitigate these issues. Its main idea is to replace concepts coming from the platform and realization domain with concepts closer to the user and problem domain. In the past, this has normally meant using quite informal approaches with little rigor, often postponing important decisions to later design and implementation stages, rendering service models less useful in the long run, and stimulating the "rush to code" syndrome. New developments are about to change this. We are getting closer to a situation where service models can be expressed in ways that combine readability with sufficient rigor to make them far more useful. For instance, it is now possible to specify service behaviour with high precision and completeness, to analyze properties, to perform tradeoffs and to ensure service quality in ways unimaginable at the implementation level. In addition, it is possible to develop tool-supported transformations from service models through design models to implementations ready to be deployed on a diversity of platforms. Interestingly, the service models also provide keys to composition, adaptation and monitoring throughout the ensuing life cycle steps.

Figure 1 illustrates our view for next generation service engineering.

- Service models are "first class citizens" used to precisely specify and analyze services in terms related to the problem domain rather than the implementation domain. Suitable languages are the Goal-oriented Requirement Language (GRL) and the Use Case Map notation (UCM), which are combined in ITU-T's stan-

Paper accepted for presentation at the ITU-T "Innovations in NGN" Kaleidoscope Conference, Geneva 12-13 May 2008, http://itu.int/ITU-T/uni/kaleidoscope.

dardization effort for defining a User Requirements Notation [1][9], as well as UML 2 Collaborations [16].

- Design models are used to precisely define systems and system components providing the services in a platform and implementation independent way. Suitable languages are GRL, SDL [7] and UML.
- Implementations are software entities ready for deployment on execution platforms, automatically generated from the design models using established technologies à la Model-Driven Architecture (MDA) [15]. Programming level languages such as C++, Java, BPEL and WSDL may be used.
- Execution platforms are the systems where software components execute processes to provide services. A wide range of architectural solutions exist ranging from typical telecommunication solutions to architectures coming from the business and information systems domains. There is no commonly agreed architecture, and one should expect diversity and evolution in this area.

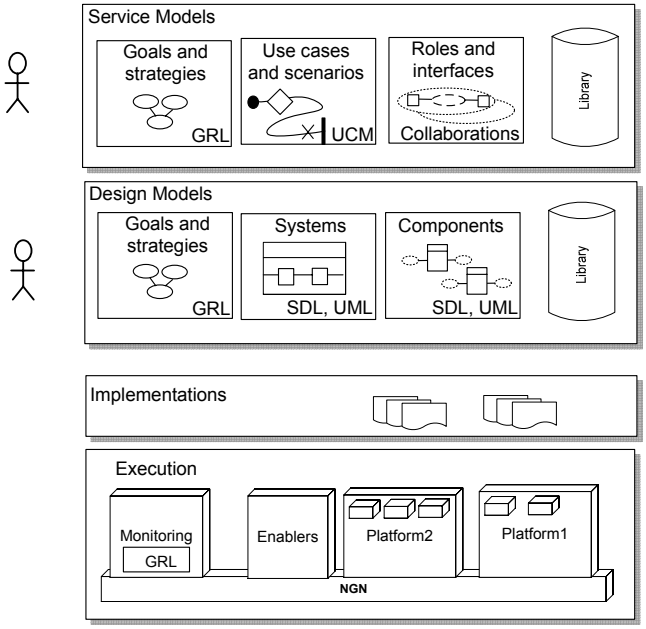

Figure 1. Overview of NGSE artefacts

This overall idea is not new. What is new is that service models may play a more prominent and pivotal role than in current model driven development. New developments enable: (1) increased precision, modularity and completeness in service models; (2) using GRL to analyze requirements and adaptations at the service level, and to implement strategies in the running system; (3) analyzing service models for consistency and realizability problems before deriving design models; (4) deriving correct design models automatically from service models; (5) using collaborations coming from the service models to support service composition at design time and run time using role binding and policies. Transformations from UML/SDL design models to implementations are well established for those willing to go that way [3][6] and hence can be built upon.

Note that a service here is considered as *an identified functionality provided by collaborating entities to establish some goals/effects.* This corresponds to the understanding of service one normally finds in requirements engineering in contrast to the understanding of a service as an interface to a component one finds in Web services and to some extent in SOA. One important aspect of services is that they normally are partial functionalities that cross-cut the component structure so that a service may involve several collaborating components while each component may participate in several services.

This paper will illustrate our view for next generation service engineering through a case study addressing availability concerns for a *multimedia over IP* call service (section 2). The roles and benefits of GRL, UCM, and UML collaborations for specifying and analysing service models will be described in section 3, followed by a brief discussion and conclusions.

2. A CASE STUDY ON SERVICE AVAILABILITY

2.1. Service availability and NGN

The NGN brings an increasing demand for new multimedia and networking services such as YouTube, Facebook and interactive IPTV services, and this in a multi-providers, hyperconnected environment. Service availability is an important concern in this enhanced, multimedia service environment.

The conceptual model for service availability presented in [16] is designed to meet the challenges of securing availability of NGN services and it can be used with the approach to NGN service engineering presented here.

In order to meet the demands of delivering services in the NGN environment, this paper defines service availability as a composite notion consisting of *exclusivity*, i.e. the property of being able to ensure access to authorized users only, and *accessibility*, i.e. the property of being on hand and useable when needed [18].

When this notion of service availability is applied to a *multimedia over IP* (MMoIP) call service, it means essentially that an authorized user expects to be able to place a multimedia call, and complete it without getting cut off in the middle, and be able to view and interact with the service at an acceptable quality.

2.2. An MMoIP case study

A multimedia over IP call service is considered for this case study with the focus on the aspect of ensuring access to authorized users only by providing suitable and adaptive access control. It is assumed that end users first must establish a session with a service provider before they may place or receive MMoIP calls. Access control policies are enforced by taking into account context information such as the physical location of a user, current security threat assessments and the system load levels. In some cases a user may be allowed to establish a session and place MMoIP calls without any explicit authentication and authorization (AA), while in other cases strict AA measures may be enforced.

This case study serves to illustrate that end-user services often are composed from other services (here MMoIP and AA services), that the composition may vary dynamically depending on context information, and that there are conflicting goals and tradeoffs to be made which cannot all be addressed at design time.

3. SERVICE MODELS

3.1. Goal, variability and strategy analysis using GRL

An important aspect of services and service engineering is determining what goals one wants to achieve, and what means are available to achieve them. In many cases, goals and means are conflicting and require tradeoffs.

As an example, consider the GRL graph in Figure 2. In GRL, *intentional elements* such as softgoals (clouds), measurable goals (rounded rectangles) and tasks (hexagons) can be used to capture non-functional requirements, functional requirements, and operational solutions, respectively. These intentional elements can be decomposed in an AND/OR graph, and be assigned to stakeholders called *actors* (dashed circles). Intentional elements can also influence each other through contributions (arrows) and side effects (dashed arrows), which can be positive or negative at various degrees (make, help, some positive, some negative, hurt, break).

Figure 2 focuses on the MMoIP Service Availability, an objective dear to the Service Provider. As explained in section 2.1, this can be decomposed into Accessibility and Exclusivity, which can be helped by several tasks. A closer look at the Provide Access Control task indicates that Authorization and Authentication are both required. However, there are several alternative ways of achieving both goals: User Pull or Server Pull can be used for Authorization (with different side effects on the Cost of the service), whereas a selection of four alternative AA patterns can be used for authentication (explained in [19]): Unilateral Two-Pass Authenticate (UTPA), Unilateral One-Pass Authenticate (UOPA), Mutual Two-Pass Authenticate (MTPA), or Mutual One-Pass Authenticate (MOPA).

An authentication pattern is always involved in the User Pull and in Server Pull authorization architectures. This is because authorization depends on authentication. This dependency could be shown in the GRL model but, for the purpose of this example, we simply separate the authentication part from the authorization. User Pull authorization encompasses activation and distribution of authorizations to the user by the Authorization Server (not shown), followed by the presentation of these authorizations to the Service Provider in order to access the service. In the case of Server Pull authorization, the Service Provider interacts directly with the Authorization Server in order to check the user's authorizations.

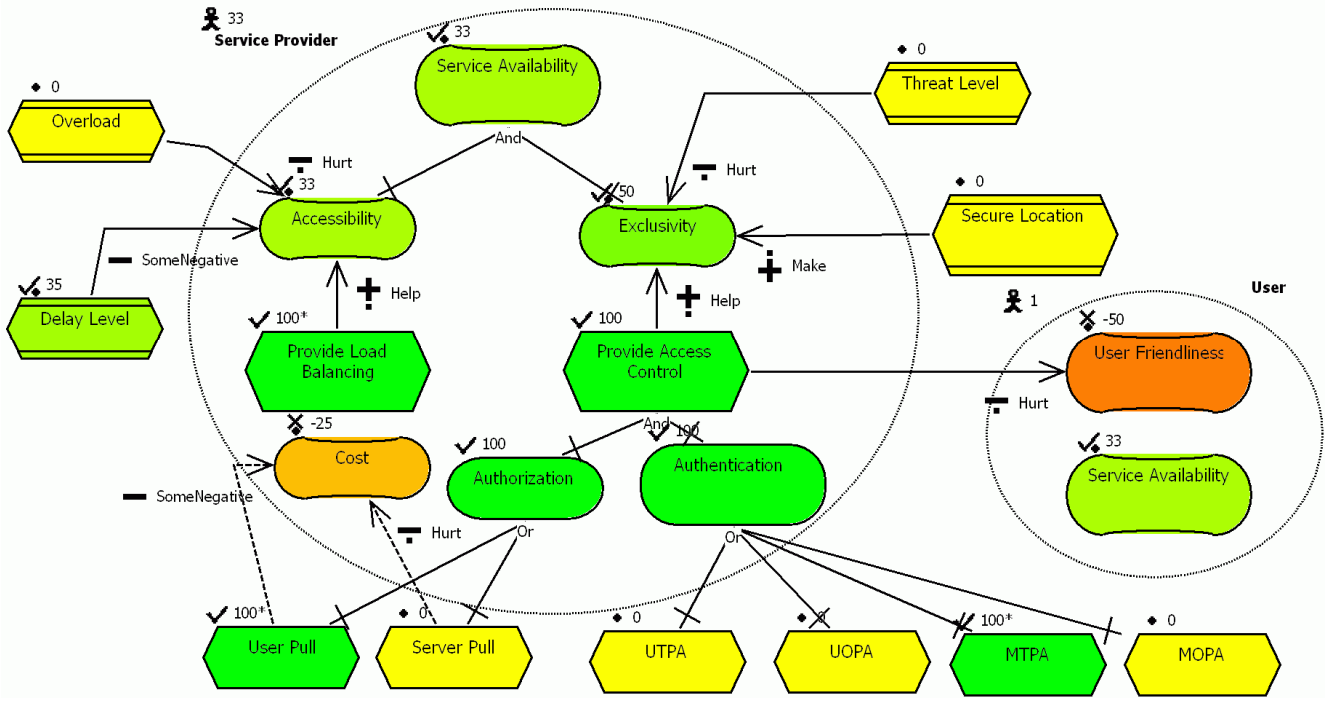

Figure 2. GRL Model for MMoIP Service with Emphasis on Availability

These alternatives have different strengths depending on whether one or both parties involved need to be authenticated and whether one pass (message) or more are necessary. Note however that each alternative could have side effects on Cost (not shown here), or on any other concern. For example, the User is interested in the Availability of the service, but also in its User Friendliness. The latter is negatively impacted by the provision of Access Control because additional interactions are then required to use the service, which slow down the access to the service.

Indicators (shown as double-lined hexagons), also provide additional contextual information that can influence the satisfaction of stakeholder objectives. Their satisfaction levels result from the monitoring of factors internal or ex-

ternal to the system [17]. For instance, the four AA patterns could introduce different Delay Levels at run-time, with some negative impact on the Accessibility. The current Threat Level could also have a negative contribution on the Exclusivity, which could in turn require stricter Access Control to compensate and result in good service Availability. On the other hand, being in a Secure Location would be sufficient to satisfy the Exclusivity goal, and Access Control would therefore not be required at all (while avoiding any negative impact on User Friendliness).

Such GRL diagrams can be very helpful in describing stakeholders and their main objectives, alternative means of reaching these objectives (including variability in the service composition), and the various side-effects that need to be taken into consideration to reach and maintain, through adaptation, the best trade-off across conflicting objectives. The jUCMNav tool [21] supports the creation of GRL models but also their analysis with the help of *strategies*, where initial satisfaction levels (between -100 and +100) are associated to some of the intentional elements (those prefixed with a *, e.g. MTPA and User Pull in Figure 2) and then propagated to the other elements of the graph through the decomposition and contribution links. Satisfaction levels are also colour-coded (the greener, the more satisfied, the redder, the more dissatisfied). Figure 2 illustrates the results of such an evaluation for a MMoIP call where User Pull and MTPA are selected. Note the resulting low satisfaction of the various softgoals in the model.

3.2. Specifying and analyzing scenarios using UCM

A central problem in service engineering is to specify service behaviour as precisely and completely as possible without prematurely binding the design and implementation. The Use Case Map (UCM) scenario notation has proven to be helpful for specifying and analysing models in such a context. UCM models describe the sequences of activities, called *responsibilities* and shown as X's on a scenario path, which need to be performed in a given set of services. Modellers may start by first defining the responsibilities needed to be performed without considering distribution issues, and then introduce *components* to make distribution explicit, without needing to bind interaction detail or protocols. In this paper's example, distribution issues will be delegated to the UML collaboration level.

Figure 3 describes a UCM model for our example MMoIP service. Scenario start and end points are respectively shown with filled circles and bars. In this figure, two important phases are shown as *stubs* (diamonds), which are containers for sub-diagrams called *plug-ins*. The MMoIPServiceSession is particularly of interest here. This *dynamic stub* (dashed diamond) may contain many plug-ins, one of which is selected at run-time according to the current context and the stub's selection policy, both expressed with a simple data model. This stub contains two alternative plug-ins: MMoIPService and SecureMMoIPService (Figure 4). The former is similar to the latter but does not contain the Authenticate and Authorize stubs, and the session handler is not secured.

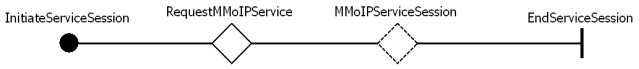

Figure 3. Top-level UCM model for a MMoIP service session

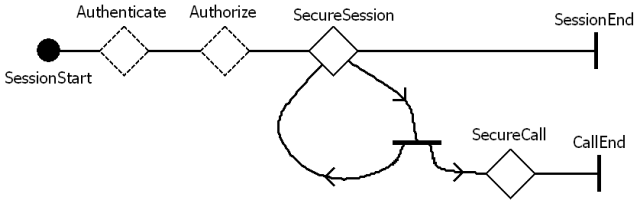

Figure 4. SecureMMoIPService plug-in UCM for stub MMoIPServiceSession

UCM models can be decomposed at any number of levels. A third level is shown in Figure 5 with the plug-in for the SecureCall stub of Figure 4. The plug-in's start and end points are connected to the parent stub's input and output segment through a binding relationship, hence ensuring the continuity of the scenario flow as we drill down and up the model. As illustrated in Figure 5, guarded alternatives and loops are easily expressed, and the notation also supports concurrent scenarios and inter-scenario relationships, including timers and (a)synchronous interactions. Plug-ins can be invoked in many stubs, therefore promoting the reuse of scenario patterns.

The UCM data model is composed of global variables of different types (Boolean, integer, enumeration), which can be used in guarding expressions (stub selection policies, paths forking as alternatives, etc.) and can be updated in responsibilities. Since UCMs are integrated with GRL in the User Requirements Notation, variables can also capture the satisfaction of GRL's intentional elements for a given strategy (as integers between -100 and 100). Therefore, GRL strategies can guide the selection of suitable behaviour at scenario variability points (dynamic stubs) or adapt it to a context where we want to optimize goal satisfaction. For instance, Authorize and Authenticate depicted in Figure 4 are dynamic stubs. There are two choices for the authorization architecture: one for User Pull and one for Server Pull. Additionally, for the authentication that must be executed prior to authorization, there are 4 sub-maps (one for each of the authentication patterns UTPA, UOPA, MTPA, MOPA). For example, for access to an online telephone catalogue, a username and password based unilateral two-pass authentication (UTPA) may be considered sufficient. However, for access to the MMoIP service, stronger authentication may be required, such as mutual two-pass authentication (MTPA) using a symmetric key-based protocol. The selection is guided by the task chosen in a GRL strategy. This analysis can be done at a very early stage, at the requirements level, and later used to govern dynamic adaptation in the running system. This enables dynamic choice of authentication pattern that best fits the security requirements according to a GRL model.

Tools can take advantage of such data model to guide the selection of UCM paths in a given context for analysing,

testing and transforming scenarios. In jUCMNav, modellers can declare *scenario definitions* where variables are initialized and selected start points triggered [11]. A traversal algorithm uses (and updates) the variables to highlight the paths traversed as the scenario progresses. Deadlocks, nondeterministic choices and other unexpected results can be detected by using this mechanism. In addition, the traversed path can be transformed into other, more linear representations such as ITU-T's Message Sequence Charts (MSC) [8]. For instance, 0 shows two scenarios extracted from the URN model described so far. The top one represents a successful regular call where no authentication or authorization is used, while the rightmost one is a successful secure call with UTPA-based authentication and user pull authorization. Conditions (expressed with waiting places on the paths) and concurrency are also preserved in this representation, and corresponding MSCs could be generated. Note that these scenarios do not include the details of the AA patterns and of the stubs and plug-ins used in the previous figures but not discussed. Finally, had we distributed the responsibilities to a structure of components, this would have been taken into consideration in the MSCs, and one MSC instance per component would have been generated, together with additional inter-component messages required to ensure the flow of causality across distributed entities. Such generated UCMs and MSCs can be used to visualize at a glance the combined results of GRL and UCM analyses.

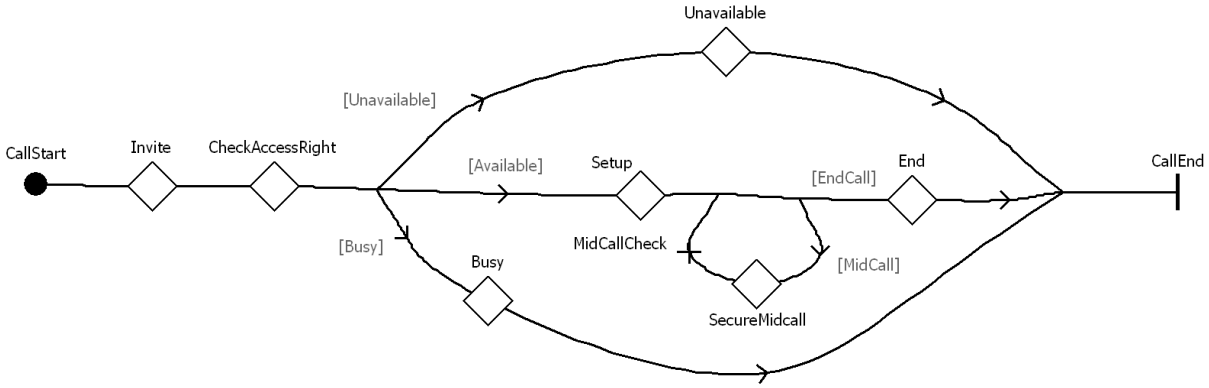

Figure 5. SecureCall plug-in UCM

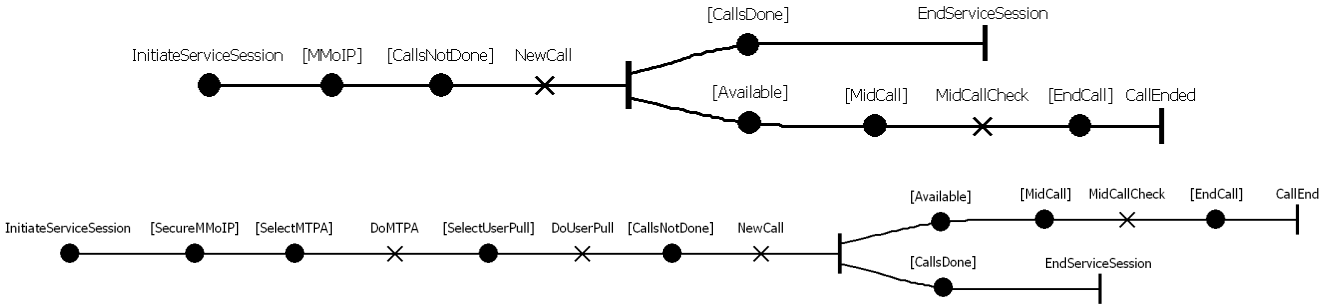

Figure 6. Two scenarios extracted from the URN model

3.3. Specifying roles structures using UML collaborations

An important aspect of workflow and service modelling is the identification of actors involved. At the early service modelling stage, it makes little sense to identify *implementation* level components, but it makes much sense to identify *domain entities* such as users and user groups with different responsibilities. It does not make much sense to identify *particular* components either. It is better to specify the properties that actors participating in a service shall have and specify these as *roles*. UML 2 Collaborations provide a way to define this information structurally and also to decompose and compose services by means of collaboration uses and to bind roles to classifiers (enclosing roles) defining service components [16][24]. Collaborations thereby provide a means for visually addressing the cross-cutting nature of services. The components represented in Use Case Maps will normally correspond to roles in the collaborations.

As shown by Figure 7, SecureMMoIP has the roles User, ServiceProvider and AccessControlServer, which interact with each other according to the collaboration uses Authentication, Distribute Authorizations, etc. Each collaboration use refers to a collaboration defined separately that describes the details of the interactions. When decomposing collaborations into sub-collaborations, one tends to identify sub-collaborations that involve just two roles. Such binary collaborations can be used to define interfaces and interface behaviour. This has some advantages: (1) the behaviours are relatively small, (2) they can be completely defined, (3) they are units of reuse and (4) they can be used to type interfaces of design and implementation level components as

illustrated in Figure 7 to support discovery and ensure consistency in dynamic links [23].

Assuming that the behaviour of each collaboration use is completely specified in its defining collaboration, the overall behaviour of the SecureMMoIP collaboration can be defined by specifying a *choreography* of collaboration uses, i.e. how they are ordered. This can be done in alternative ways. Both UCM and UML activity diagrams may be used for this purpose. Indeed, some of the stubs and plug-ins in the UCM model of the previous section correspond to collaboration uses in Figure 7 and so partially specify the choreography. In [5], activity diagrams are used to define choreography using the composition operators weak and strong sequencing, alternative, interruption, and parallel. The same composition operators provide means to identify and resolve problems with direct realizability, i.e. that the resulting distributed behaviour is exactly the specified behaviour.

Not evident from Figure 7 is that one may want to dynamically select definitions for the collaboration uses depending on context and policies, analogous to selecting plug-ins for dynamic stubs in UCM, as explained in Section 3.2. This means that policy decisions are used to govern the overall choreography.

In [19][20], a framework was presented that classifies authentication and authorization (AA) patterns specified using UML collaborations. The behaviour associated with roles of the collaborations may be specified using semantic interfaces, defining the visible interface behaviour and the goals of the collaboration, as well as the goals and role-binding polices for each of the two roles. Behavioural information such as events, progress, and goals provided by the semantic interfaces may be used in the definition of composition policies for composing collaborations. This allows for flexibility in composing patterns with other service roles while enabling validation of behaviour using semantic interfaces [23]. The selection of patterns to be used in a service may be done similarly to the selection of plug-ins for dynamic stubs.

For each service that is specified as a composition of AA patterns and elementary binary service collaborations, a set of policy rules may be created for describing the ordering of the behaviour associated with the elementary collaborations and their associated roles that are to be played by the service entities. These rules define composition policies for ordering the elementary collaborations.

A UCM can be created for each collaboration use. The composition policies are then used to define how the UCMs are linked together, defining dependencies between the collaborations. Policy may dictate that one collaboration starts before another is finished, and UCMs can be used to depict this.

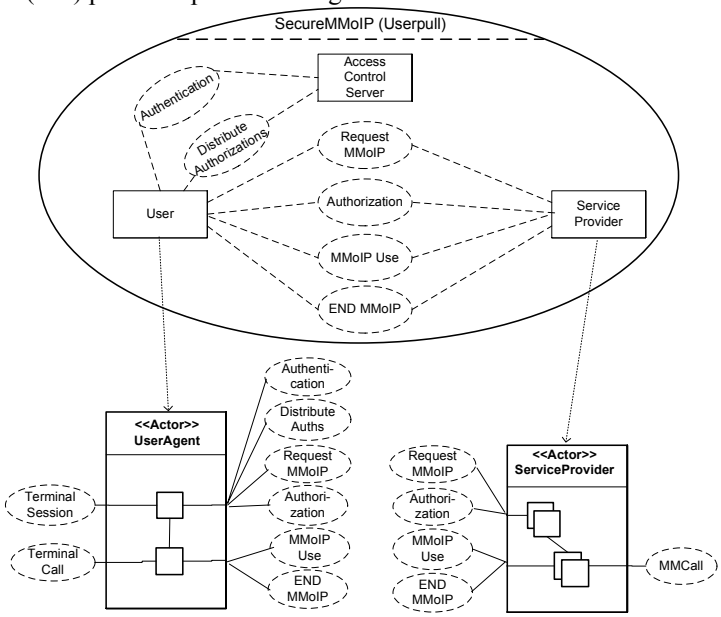

Figure 7. Illustration of the mapping from a collaboration model to components that implement the collaboration roles and are typed with semantic interfaces

3.4. Design composition

We assume here that the system design is defined in terms of communicating state machines. Communicating state machines enable precise and complete behaviour definitions. Extensive tool support is available for verification and validation, and it is possible to generate complete implementation code for a wide range of platforms. Being based on asynchronous communication they pose very few design restrictions on the services. We may use SDL agents or UML active objects to model the design components. There is much evidence indicating that state machines are key to efficient code generation from platform independent models, i.e. to make MDA practical [3][6].

It has been demonstrated that it is feasible to automatically generate design components in the form of state machines from service models [2][10][12][22]. More work is clearly needed in this area to realize NGSE industrially, but there seem to be no fundamental obstacles. Thus there is a potential that service models may become the main focus of ser-

vice engineers, with ensuing design and implementations derived with a high degree of automation.

As illustrated in Figure 7, the roles specified in a collaboration are mapped into inner role state machines of design components serving as the actors that execute the roles. The actors may be typed with collaborations (the semantic interfaces they define) to enable discovery and compatibility checks at design time and runtime. Before a collaboration can take place, it is necessary that its roles are bound to actor instances. In many cases this entails dynamic role binding where one actor requests another actor to play a role in a collaboration, addressing the problem of establishing dynamic links between design components participating in a service. The UserAgent may for instance request the ServiceProviderAgent to play MMoIP. The requested actor may now consult its policies, taking into account state information, user profiles and context information to decide whether is prepared to play the requested role or not. If not, it may reject the request, or propose an alternative, such as the SecureMMoIP. If the requesting actor agrees, then they can proceed playing roles according to the SecureMMoIP collaboration. Every dynamic role binding step provides a similar opportunity to negotiate roles, and in this way the policy decisions of the service models (initially expressed using GRL and UCM) can be mapped into corresponding decisions performed by the design and implementation level components.

Role-binding policies may be used to specify requirements/objectives specifically for the instance to be playing a role in a collaboration, such as constraints imposed on the agent by a role, as well as constraints an agent may impose on a role to be played. Role-binding policies govern the dynamic linking of components to be involved in a service collaboration. A particular role can be bound to the UserAgent, e.g. the UserAgent satisfies the policy rules for playing the role, and if the role is allowed by the UserAgent's policy. Composition policies, which are sets of policy rules defining dependencies between subordinate collaborations involved in a service, may be defined to govern the ordering of behaviour associated with sub-collaborations involved in a service collaboration, and declared along with the UCM defining the ordering of the collaborations. These policy rules may also be used to enable the ordering of roles to be composed within an actor. In this way it is possible to use the composition policies specified along with the UCMs as outlined above, to control what sub-roles of the composed role within an actor can be played and under what conditions, and in which order.

3.5. Implementation and run-time adaptation

Figure 1 illustrates that the execution environment provides service delivery platforms over NGN where the services can be deployed. Services can make use of each other in such environment, but they first need to locate each other. The information about the semantic interface types that has been used at design time to ensure the correctness of static associations is stored (e.g. in a repository) as part of a management system. At runtime, dynamic role-binding is performed using the component and semantic interfaces type information to ensure compatibility of dynamic links. By performing dynamic role-binding it is possible to ensure that the links satisfy the properties of the semantic interfaces.

Although GRL models can be used to specify many non-functional requirements that will guide design and implementation decisions, their utility is not limited to the design space. GRL models can also be used at runtime to monitor the NGN execution environment and enable dynamic adaptation of services (Figure 1). As discussed in section 3.1 and illustrated in Figure 2, indicators can convert metrics from the external world to satisfaction levels that can be propagated to the other nodes in a GRL graph. This in turn can influence the choice of suitable solutions/tasks dynamically. For instance, if the Threat Level suddenly increases, a stronger authentication pattern may have to be selected so the Service Availability remains unaffected. However, this is also constrained by the Delay Level and Overload introduced by a more restrictive authentication mechanism because they too affect Service Availability. This adaptive behaviour cannot be determined at design time and hence must be part of the runtime environment.

4. CONCLUSIONS

In this paper we have outlined a vision for NGN service engineering involving four phases of development from high level service models, to design models, and to implementation and service execution. The approach presented to support NGN service engineering combines GRL and UCM (part of ITU-T's proposed User Requirements Notation) modelling techniques with UML collaborations. GRL goal models offer a holistic view that integrates stakeholder goals, non-functional requirements, and alternative operational solutions for design time decisions, supplemented with indicators that enable adaptive behaviour at runtime. UCM offer scenarios that express variability points explicitly while offering much flexibility in ordering activities, which may be bound to components or not. UML collaborations can be used to introduce roles and refine the interaction patterns that need to be orchestrated (via semantic interfaces and dynamic role binding) to offer a complete service. UCM, GRL and UML collaborations also enable transformations to other modelling elements such as SDL and UML state machines as we move towards design and implementation. Various tool-supported analysis techniques exist for these three complementary views. This paper also explained that policies may be used to govern the ordering of separately specified behaviours. A case study addressing availability concerns for a multimedia over IP call service has been used to illustrate the possibilities that the different modelling techniques provide to the service engineering process while emphasizing the importance of enabling dynamic choices in the service modelling and design phases in order to take into account differentiated service availability requirements in dynamic service composition. These

contributions will fulfill important needs of next generation service engineering (NGSE).

Although such approach suggests that it is no longer necessary to "rush to code" to get tangible results and that a model-driven, service-oriented approach is becoming practically feasible, there are challenges ahead that remain to be faced. The approach needs to be validated on more concrete NGN services, from requirements to deployment, and tools need to be aligned to support NGSE processes in an integrated and usable way. The languages themselves are still evolving, but they do so in a way that could be beneficial to NGSE. For instance, more powerful workflow-like constructs are integrated to the UCM notation [13] and UCM and GRL are being extended to support aspect-oriented concepts [14], hence enabling other types of dynamic composition mechanisms. UML collaborations provide a framework for service modeling that enable service behaviour to be completely defined, analyzed for problems and automatically translated to executable design models, as explained in [4], [5] and [12]. They provide a new kind of reusable entity that can be composed statically at design time and used to govern dynamic composition at run time. While most current MDA practices focus on design models and automatic translation to executable code, our approach goes one step further by putting more emphasis on service modeling and providing automatic translation to design models. An important feature is that model elements forming the early service models remain useful during the ensuing design and execution steps by enabling novel mechanisms to support composition and adaptation. Future work will explore the above issues further in order to gather evidences and determine the options that provide real value for next generation service engineering.

REFERENCES

[1] Amyot, D., "Introduction to the User Requirements Notation: Learning by Example". *Computer Networks*, 42(3), 285–301, 21 June 2003

[2] Bochmann, G.v., and Gotzhein, R., "Deriving protocol specifications from service specifications". *Proc. ACM SIGCOMM Symposium*, 148-156, 1986

[3] Bræk, R. and Melby, G., "Model Driven Service Engineering". Invited chapter (XII) in *Model-driven Software Development. Volume II of Research and Practice in Software Engineering*. Beydeda, S. Book, M. Gruhn, V. (Eds), Springer, 464 pages, 2005

[4] Castejón, H.N and Bræk, R, "Formalizing Collaboration Goal Sequences for Service Choreography". *Proc. of the 26th IFIP WG 6.1 Intl. Conf. on Formal Methods for Networked and Distributed Systems (FORTE'06)*. LNCS 4229, 275–291, Springer, 2006

[5] Castejón, H.N, Bræk, R., and Bochmann, G. v., "Realizability of Collaboration-based Service Specifications". *Proc. of the 14th Asia-Pacific Software Engineering Conference (APSEC 2007)*, IEEE Computer Society, December 2007

[6] Dietz, P., Marth, K., Van Den Berg, A., Weigert, T., and Weil, F., "Practical Considerations in Automatic Code Generation". In J.-P. Tsai and D. Zhang (Eds), *Advances in Machine Learning Application in Software Engineering*, 346-409, IGI Global, Hershey 2007

[7] ITU-T, Recommendation Z.100 (08/02): Specification and description language (SDL), Geneva, Switzerland, 2002

[8] ITU-T, Recommendation Z.120 (04/04): Message sequence chart (MSC), Geneva, Switzerland, 2004

[9] ITU-T, Recommendation Z.150 (02/03): User Requirements Notation (URN) – Language requirements and framework, Geneva, Switzerland, 2003

[10] Kant, C., Higashino, T., Bochmann, G.v., "Deriving protocol specifications from service specifications written in LOTOS". *Distributed Computing*, Vol. 10, No. 1, 29–47, 1996

[11] Kealey, J., *Enhanced Use Case Map Analysis and Transformation Tooling*. M.Sc. thesis, University of Ottawa, Canada, October 2007

[12] Kraemer, F.A., Herrmann, P., Braek, R., "Synthesizing Components with Sessions from Collaboration-Oriented Service Specifications". *Proc. 13th Int. SDL Forum*, Paris, France. LNCS 4745, 166–185, Springer, 2007

[13] Mussbacher, G., "Evolving Use Case Maps as a Scenario and Workflow Description Language". *10th Workshop of Requirement Engineering (WER'07*), Toronto, Canada, 56–67, May 2007

[14] Mussbacher, G., Amyot, D., and Weiss, M., "Visualizing Early Aspects with Use Case Maps". To appear in: *LNCS Journal on Transactions on Aspect-Oriented Software Development*, Springer, 2007

[15] OMG, *MDA Guide V1.0.1*, omg/03-06-01, 2003.

[16] OMG, *Unified Modeling Language (UML)*, Version 2.1.1, February 2007

[17] Pourshahid, A., Chen, P., Amyot, D., Forster, A.J., Ghanavati, S., Peyton, L., and Weiss, M., "Toward an integrated User Requirements Notation framework and tool for Business Process Management". *3rd Int. MCeTech Conference on eTechnologies*, Montréal, Canada, 3–15, Jan. 2008

[18] Rossebø, J.E.Y., Lund, M.S., Husa, K.E., and Refsdal, A. "A conceptual model for service availability". *Quality of Protection: Security Measurements and Metrics (QoP'05)*, Volume 23 of Advances in Information Security, Springer, 107–118, 2006

[19] Rossebø, J.E.Y. and Bræk, R., "Towards a Framework of Authentication and Authorization Patterns for Ensuring Availability in Service Composition". *First Int. Conf. on Availability, Reliability and Security (ARES 2006)*, IEEE Computer Society, 206–215, April 2006

[20] Rossebø, J. E. Y., Bræk, R, "Towards a Framework of Authentication and Authorization Patterns for Ensuring Availability in Service Composition." Research Report 332, Institute for Informatics, University of Oslo, 2007

[21] Roy, J.-F., Kealey, J., and Amyot, D., "Towards Integrated Tool Support for the User Requirements Notation". *SAM 2006: Fifth Workshop on System Analysis and Modelling*, LNCS 4320, Springer, 198–215, 2006 http://jucmnav.softwareengineering.ca/

[22] Sanders, R.T., "Implementing from SDL". *Telektronikk*, vol. 96, no 4, 2000

[23] Sanders, R.T., Bræk, R., Bochmann, G.v., Amyot, D., "Service Discovery and Component Reuse with Semantic Interfaces". *Proc. 13th Int. SDL Forum*, Grimstad, Norway. LNCS 3530, 85–102, Springer, 2005

[24] Sanders, R.T., Castejón, H.N., Kraemer, F.A., Bræk, R. "Using UML 2.0 collaborations for compositional service specification". *8th Int. Conf. on Model Driven Engineering Languages and Systems (MoDELS 2005)*, LNCS 3713, 460–475, Springer, 2005