Development of Telecommunications Standards and Services with the User Requirements Notation

Daniel Amyot and Gunter Mussbacher

SITE, University of Ottawa, 800 King Edward, Ottawa, ON, K1N 6N5, Canada
{damyot, gunterm}@site.uottawa.ca

Abstract. Requirements represent an important aspect of telecommunications standards, whether they apply to protocols, services, or architectures. This paper presents an overview of the proposed ITU-T Recommendation Z.151 – User Requirements Notation (URN), intended for the elicitation, analysis, specification, and validation of requirements. URN combines modelling concepts and notations for goals and intentions (mainly for non-functional requirements, quality attributes, and reasoning about alternatives) and scenarios (mainly for operational requirements, functional requirements, and performance and architectural reasoning). Basic concepts and notation elements are introduced, together with the main requirements analysis, transformation, and management techniques relevant to URN and supported by the jUCMNav Eclipse plug-in. Although URN is generally suitable for describing most types of reactive systems and information systems, we will illustrate its applicability to the development of telecommunications standards. Other applications of URN will briefly be enumerated, and an overview of future extensions to URN will be presented.

Keywords: Goal-oriented Requirement Language, jUCMNav, Use Case Maps, Scenarios, Standards, Telecommunication, User Requirements Notation.

1 Introduction

The *User Requirements Notation* (URN) aims to support the elicitation, analysis, specification, and validation of requirements. URN is the first standardization effort to address explicitly, in a graphical way and in one unified language, goals and scenarios, and the links between them. URN models can be used to specify and analyze various types of reactive systems as well as telecommunications standards [2].

The kind of modelling supported by URN is different from the detailed specification of "how" functionalities are to be supported, as described with languages such as SDL [23], Message Sequence Chart (MSC) [25], or UML [44]. Here the modeller is primarily concerned with exposing "why" certain choices for behaviour and/or structure were introduced, combined with an abstract view of "what" capabilities and architecture are required. The modeller is not yet interested in the operational details of internal component behaviour or component interactions.

Omitting these kinds of details during early development and standardization phases allows working at a higher level when modelling a current or future standard or software system and its embedding environment. Modelling and answering "why" questions leads us to consider the opportunities stakeholders seek out and vulnerabilities they try to avoid within their environment, whereas modelling and answering "what" questions helps identify capabilities, services, and architectures required to satisfy stakeholder goals.

In this paper, section 2 introduces URN's basic concepts and notation elements. Section 3 gives an overview of different URN-based analysis and transformation techniques. In section 4, we discuss the applicability of the notation and these techniques to the development of telecommunications standards. Section 5 explores other application domains of URN as well as potential enhancements, followed by our conclusions in section 6.

2 User Requirements Notation

Section 2.1 first gives an overview of the User Requirements Notation (URN) as proposed in the ITU-T Z.151 standard. URN consists of two complementary languages, the Goal-oriented Requirement Language (GRL) summarized in section 2.2 and the Use Case Map (UCM) notation summarized in section 2.3. Important URN concepts that are applicable to both, GRL and UCM, are discussed in section 2.4. The section closes with a description of the capabilities of jUCMNav, the most comprehensive URN tool currently available.

2.1 Overview of the Proposed Recommendation Z.151

URN is intended for the elicitation, analysis, specification, and validation of requirements. URN allows software and requirements engineers to discover and specify requirements for a proposed system or an evolving system, and analyse such requirements for correctness and completeness.

URN combines the *Goal-oriented Requirement Language* (GRL) for modelling goal-oriented and intentional concepts (mainly for non-functional requirements, quality attributes, and reasoning about alternatives) with the *Use Case Map* (UCM) notation for modelling scenario concepts (mainly for operational requirements, functional requirements, and performance and architectural reasoning). In particular, URN has concepts for the specification of stakeholders, goals, non-functional requirements, rationales, behaviour, scenarios, scenario participants, and high-level architectural structure.

The proposed Recommendation Z.151 [27] adheres to the guidelines of the proposed ITU-T Recommendation Z.111 [24] for metamodel-based definitions of ITU-T languages. The proposed Recommendation Z.151 specifies the abstract syntax of URN, a concrete graphical syntax for URN, an XML-based interchange format for URN, and a data language that is required for the formalization of conditions and expressions used by some features of URN. The data language is a subset of SDL's supporting Boolean, Integer, and Enumeration data types, with a concrete textual syntax that supports both SDL and Java/C expressions. The metamodel of the concrete syntax and the XML schema of the interchange format both specify layout information enabling the reconstruction of diagrams from the model.

The static semantics of URN is defined with the help of natural language descriptions and constraints on the abstract and concrete URN metamodel. The dynamic aspects of URN are defined by requirements and guidelines for (i) *propagation mechanisms* for GRL model evaluation, and for (ii) a *path traversal mechanism* for UCM scenario interpretation.

The GRL model evaluation allows for the comparison of alternatives and facilitates trade-offs among conflicting goals of various stakeholders. A general description of GRL model evaluation is provided along with three examples of evaluation/propagation algorithms. The language does not enforce a specific propagation mechanism as GRL can be used in different ways by different modellers, e.g., for qualitative evaluations or quantitative ones.

The dynamic semantics of the UCM notation, on the other hand, is precisely described with a list of requirements for a UCM path traversal mechanism. This mechanism is the basis for many advanced applications of UCMs, such as scenario highlighting and animation, the generation of MSCs, and the generation of test cases. Examples further clarify the usage and semantics of URN.

URN is applicable within standards bodies and industry. URN helps to describe and communicate requirements, and to develop reasoning about them. The main applications areas include telecommunications systems, services, and business processes, but URN is generally suitable for describing most types of reactive systems and information systems. The range of applications is from business goals and requirements description to high-level design.

URN is a notation that complies with Recommendation Z.150 [26]. It includes concepts and notations satisfying the language requirements of Z.150's URN-NFR (for non-functional requirements) and URN-FR (for functional requirements). URN integrates these concepts and notation into a single language. An assessment of conformity of the current URN representation with the language requirements for URN is also included in the standard, together with descriptions of various compliance levels for tools supporting the notation.

2.2 Goal-oriented Requirement Language

The subset of the URN language that addresses Z.150 URN-NFR language requirements is named *Goal-oriented Requirement Language* (GRL), which is a language for supporting goal-oriented modelling and reasoning about requirements, especially non-functional requirements and quality attributes. It provides constructs for expressing various types of concepts that appear during the requirement process. GRL captures stakeholders, alternatives that have to be considered, decisions that were made regarding these alternatives, and rationales that helped make these decisions.

GRL has its roots in two widespread goal-oriented modelling languages: i* [57] and the NFR Framework [13]. Major benefits of GRL over other popular notations include the integration of GRL with a scenario notation, the support for qualitative and quantitative attributes, and a clear separation of GRL model elements from their graphical representation, enabling a scalable and consistent representation of multiple views/diagrams of the same goal model.

The syntax of GRL (see Fig. 1) is based on the syntax of the i* language. There are three main categories of concepts in GRL: actors, intentional elements, and links. A GRL goal graph is a connected graph of intentional elements that optionally reside within an actor boundary. An *actor* represents a stakeholder of the system or another system. Actors are holders of intentions; they are the active entities in the system or its environment who want goals to be achieved, tasks to be performed, resources to be available and softgoals to be satisfied. A goal graph shows the high-level business goals and non-functional requirements of interest to a stakeholder and the alternatives for achieving these high-level elements. A goal graph also documents *beliefs* (rationales) important to the stakeholder.

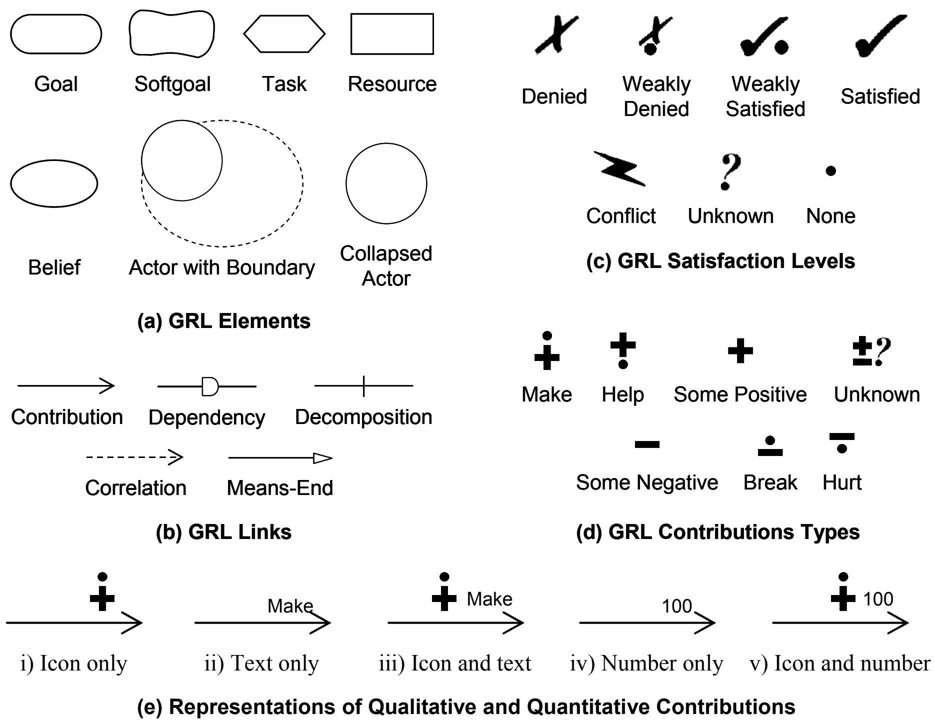


Fig. 1. Basic Elements of GRL Notation

In addition to beliefs, *intentional elements* can be softgoals, goals, tasks, and resources. *Softgoals* differentiate themselves from *goals* in that there is no clear, objective measure of satisfaction for a softgoal whereas a goal is quantifiable. In general, softgoals are related more to non-functional requirements, whereas goals are related more to functional requirements. *Tasks* represent solutions to (or operationalizations of) goals or softgoals. In order to be achieved or completed, softgoals, goals, and tasks may require *resources* to be available. Goals, softgoals, tasks, resources, and beliefs are intentional because they are used for models that allow answering questions such as why particular behaviours, informational and structural aspects were chosen to be included in the system requirements, what alternatives were considered, what criteria were used to deliberate among alternative options, and what the reasons were for choosing one alternative over the other.

Links (see Fig. 1.b) are used to connect isolated elements in the requirement model. Different types of links depict different structural and intentional relationships (including decompositions, contributions, and dependencies). *Decomposition links* allow an element to be decomposed into sub-elements. AND, IOR, as well as XOR decompositions are supported. XOR and IOR decomposition links may alternatively be displayed as *means-end* links. *Contribution links* indicate desired impacts of one element on another element. A contribution link can have a qualitative contribution type (see Fig. 1.d), or a quantitative contribution (integer value between -100 and 100, see Fig. 1.e). *Correlation links* are similar to contribution links, but describe side effects rather than desired impacts. Finally, *dependency links* model relationships between actors (one actor depending on another actor for something).

From the NFR framework, GRL borrows support for analysis of *strategies*, which help reach the most appropriate trade-offs among (often conflicting) goals of stakeholders. A strategy consists of a set of intentional elements that are given initial satisfaction values (see Fig. 1.c; e.g. a chosen intentional element is set to Satisfied whereas all other intentional elements are set to Denied). These satisfaction values, which can be qualitative or quantitative, capture contextual or future situations as well as choices among alternative means of reaching various goals. These values are then propagated to the other intentional elements through their links taking contribution types into account. This enables a global assessment of the strategy being studied.

GRL also takes into account that not all high-level goals and non-functional requirements are equally important to the stakeholder. Therefore, an importance attribute (again quantitative or qualitative) may be specified for intentional elements inside actors, which is used when evaluating strategies for the goal model. A good strategy provides rationale and documentation for decisions leading to requirements, providing better context for standards/system developers and implementers while avoiding unnecessary re-evaluations of worse alternative strategies.

As an example, Fig. 2 shows a simple GRL diagram (adapted from [2]) that describes the impact of the selection of the location of a new wireless service and its data in an existing network. For the service provider, keeping a low cost is important but providing high performance is even more important. The vendor is very much interested in a system that is highly evolvable, and keeping the load on a message switching center (a wireless switch) to a minimum would be helpful. This example shows that intentional elements can be decomposed, and that many local alternatives can have various impacts on different concerns of the stakeholders involved, with no obvious global solution that would satisfy everyone.

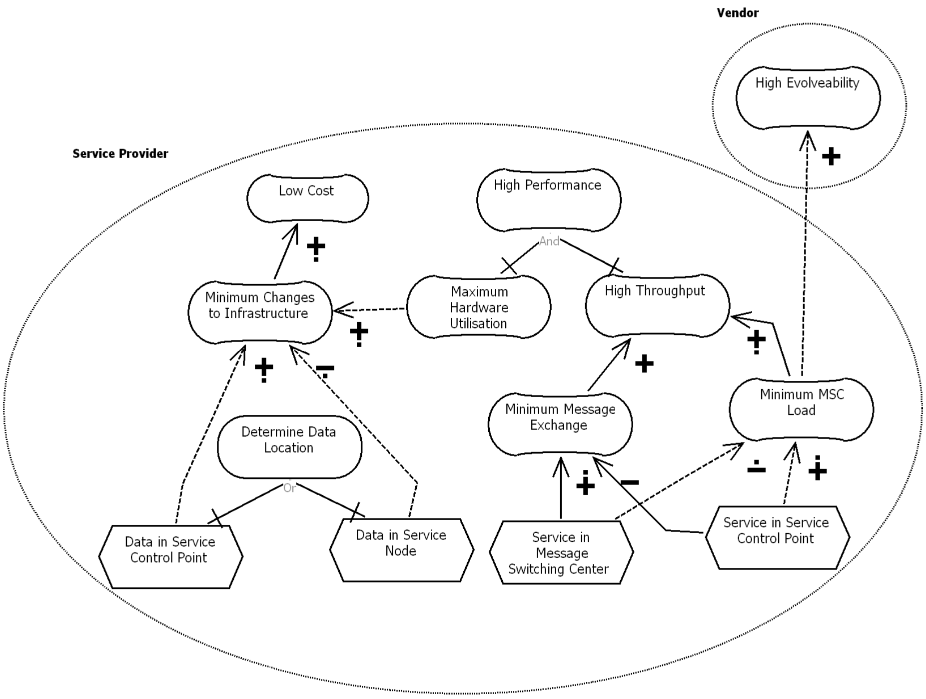


Fig. 2. GRL Model Example

2.3 Use Case Maps

The subset of the URN language that addresses Z.150 URN-FR language requirements is named Use Case Map (UCM) [10, 11]. UCM specifications employ scenario paths to illustrate causal relationships among *responsibilities*. Furthermore, UCMs provide an integrated view of behaviour and structure by allowing the superimposition of scenario paths on a structure of abstract *components* while abstracting from message and data details. The combination of behaviour and structure enables architectural reasoning after which UCM specifications may be refined into more detailed scenario models such as MSCs and UML sequence diagrams, or into state machines in SDL or UML statechart diagrams, and finally into concrete implementations. Validation, verification, performance analysis, interaction detection, and test generation can be performed at all stages. Thus, the UCM notation enables a seamless transition from the informal to the formal by bridging the modelling gap between goal models and natural language requirements (e.g. use cases) and design, in an explicit and visual way. The UCM notation allows modellers to delay the specification of component states and messages and even, if desired, of concrete components to later, more appropriate stages of the development process. The goal of the UCM notation is to provide the right degree of formality at the right time in the development process.

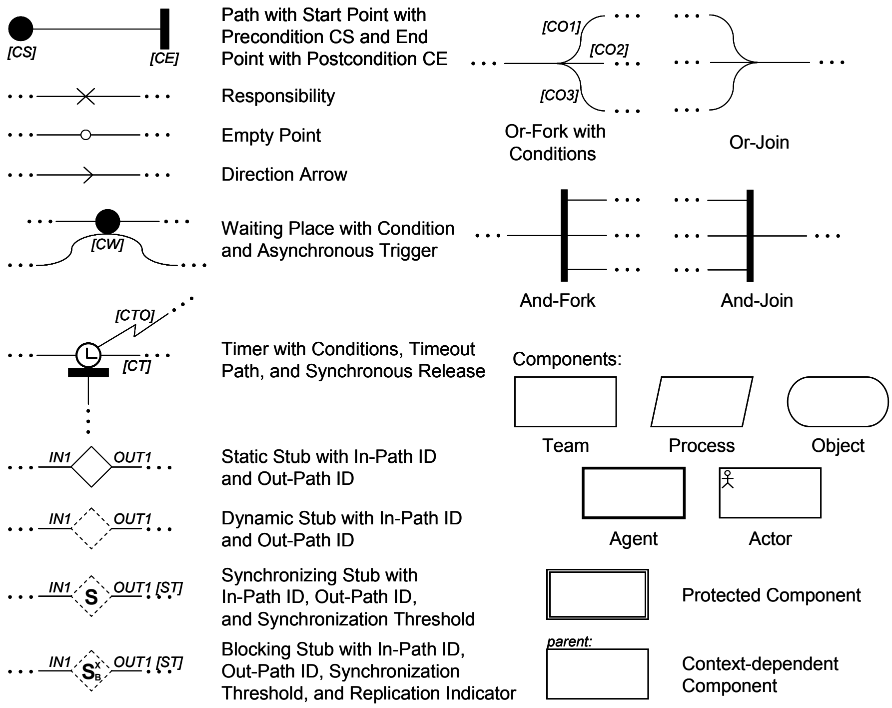


Fig. 3. Basic Elements of UCM Notation

The basic elements of the UCM notation are shown in Fig. 3. A *map* contains any number of paths and components. Paths express causal sequences and may contain several types of path nodes. *Responsibilities* describe required actions or steps to fulfill a scenario. *OR-forks* (possibly including guarding conditions) and *OR-joins* are used to show alternatives and path merging, while *AND-forks* and *AND-joins* depict concurrency and synchronization. Loops can be modelled implicitly with OR-joins and OR-forks. As the UCM notation does not impose any nesting constraints, joins and forks can be freely combined and a fork does not need to be followed by a join. *Waiting places* and *timers* denote locations on the path where the scenario stops until a condition is satisfied. If an endpoint is connected to a waiting place or a timer, the stopped scenario continues when this end point is reached (synchronous interaction). Asynchronous, in-passing triggering of waiting places and timers is also possible. A timer may also have a timeout path which is indicated by a zigzag line. End points and start points of paths can be connected to each other to indicate simple sequences of paths.

UCM models can be decomposed using *stubs* which contain sub-maps called *plug-ins*. Plug-in maps are reusable units of behaviour and structure. Plug-in bindings define the continuation of a path on a plug-in map by connecting in-paths and out-paths of a stub with start and end points of its plug-in maps, respectively. A stub may be *static,* which means that it can have at most one plug-in map, whereas a *dynamic*

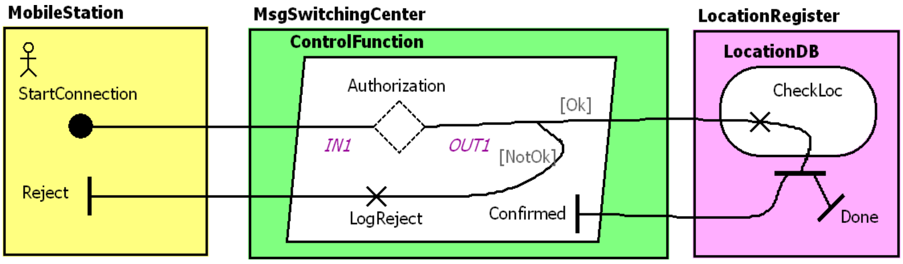
stub may have many plug-in maps which may be selected at runtime. A selection policy decides which plug-in maps of a dynamic stub to choose at runtime. A *synchronizing* stub is a dynamic stub that requires its plug-in maps to synchronize.
Components are used to specify the structural aspects of a system. Map elements which reside inside a component are said to be bound to the component. Components can contain sub-components and have various types and characteristics. For example, a protected component does not allow a second path to enter the component if one path is already executing inside the component and a component of kind object does not have its own thread of control while a component of kind process does. A component of kind actor represents someone or something interacting with the system under design. A context-dependent component is shown with the keyword "*parent:* " and indicates a component on a plug-in map that is linked to a component on the parent map through a component plug-in binding. The component on the parent map therefore defines the component on the plug-in map.

UCM specifications identify input sources and output sinks as well as describe the required inputs and outputs of a scenario. UCM specifications also integrate many scenarios or related use cases in a map-like diagram. Scenarios can be structured and integrated incrementally. This enables reasoning about and detection of potential undesirable interactions of scenarios and components. Furthermore, the dynamic (run-time) refinement capabilities of the UCM notation allow for the specification of (run-time) policies and for the specification of loosely coupled systems where functionality is decided at runtime through negotiation between components or compliance to high-level goals.

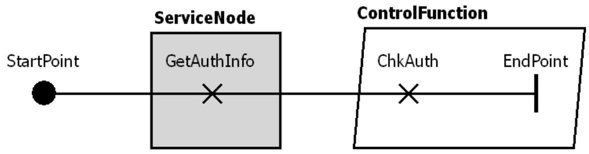
UCM scenarios can be integrated together, yet individual scenarios are tractable through *scenario definitions* based on a simple data model. UCMs treat scenario paths as first class model entities and therefore build the foundation to more formally facilitate reusability of scenarios and behavioural patterns across a wide range of architectures. Given the definition of a scenario or combination of scenarios, a UCM path traversal mechanism can highlight the scenario path or transform the scenario into more concrete design notations such as MSCs. The traversal mechanism effectively turns the scenario definitions into a test suite for the UCM model.

UCM also supports a standard set of annotations targeting performance modelling. In particular, scenario start points can have various kinds of workloads, components can be allocated to processing resources, resources can be used and external services invoked, and probabilities are attached to forks and to selection policies of stubs. Although no visual representation of these annotations is offered (tools usually provide access via property panels), their presence enables the reuse of requirements models for performance modelling and analysis.

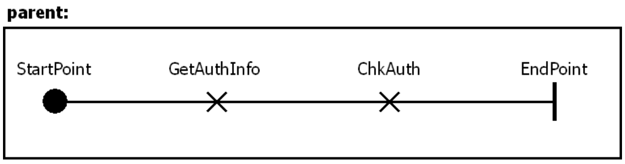
As an example, Fig. 4 shows a simple UCM model for a wireless connection use case composed of three diagrams. The top-level map (a) has an Authorization dynamic stub that contains two alternative plug-ins, whose selection depends on the states of global variables initialized in scenario definitions. In the first alternative (b), the authorization information comes from an external service node whereas in the second one (c) this information is already in the control function of the message switching center (referred to as the *parent* component here). Plug-ins can introduce new components, make reference to existing ones, or have parameterized (*parent*) components. The alternative plug-in maps capture two of the four potential combinations of choices discussed in the GRL diagram of Fig. 2.



a) Authenticated wireless connection use case: *Top-level* map



b) Service in MSC, data in external service node: *SvcInMSC_DataInSN* map



c) Service and data in MSC: *SvcInMSC* map

Fig. 4. UCM Model Example

UCM share many characteristics with UML activity diagrams, but UCM offer more flexibility in how sub-diagrams can be connected and how sub-components can be represented. UCM also integrate a simple data model, performance annotations, and a simple action language used for analysis. Activity diagrams, however, have better support for data flow modelling, object flows, cancellation and exception handling, and a better integration with the rest of UML. UCM, on the other hand, are better integrated with goal-oriented models, which are most useful in the early phases of development and standardization.

2.4 URN Links and Metadata

URN links, indicated by small triangles on model elements, can link any two URN model elements. In particular, links from GRL models to UCM models establish traceability between goal and scenario models in URN. Modelling both goals and scenarios is complementary and may aid in identifying further goals and additional scenarios (and scenario steps) important to stakeholders, or spotting spurious goals or scenarios, thus contributing to the completeness and accuracy of requirements.

Furthermore, *metadata* in the form of name/value pairs can be associated with any URN model element. This allows for domain-specific extensions to be added to URN and exploited by specialized tool support.

2.5 Tool Support with jUCMNav

The best tool supporting URN modelling, analysis and transformations is an open-source Eclipse plug-in named *jUCMNav* [28]. It supports analysis features for evaluating GRL models (strategies and propagation algorithms, see Fig. 5) [52] and executing UCM models (scenario definitions and traversal algorithms, see Fig. 6) [29]. A URN model can contain multiple interlinked GRL and UCM diagrams, and jUCMNav enables the sharing of definitions of actors, intentional elements, intentional links, responsibilities and components across diagrams.

jUCMNav prevents the creation of syntactically incorrect URN models. The tool also supports advanced functionalities such as scenario export to MSC (jUCMNav integrates an MSC viewer), export of performance models based on UCM performance annotations, integration with the Telelogic DOORS requirements management system, verification of user-defined static semantic rules written in OCL [43], import/export of GRL catalogues, report generation in PDF and HTML, and export of figures to various image formats.

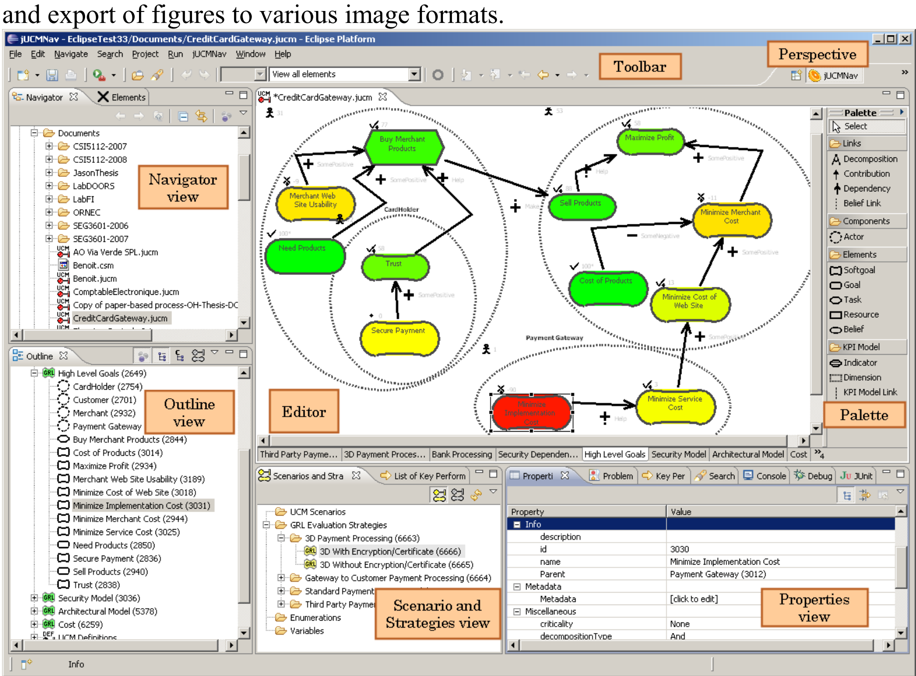


Fig. 5. jUCMNav Tool: GRL Editor with Strategy Evaluation

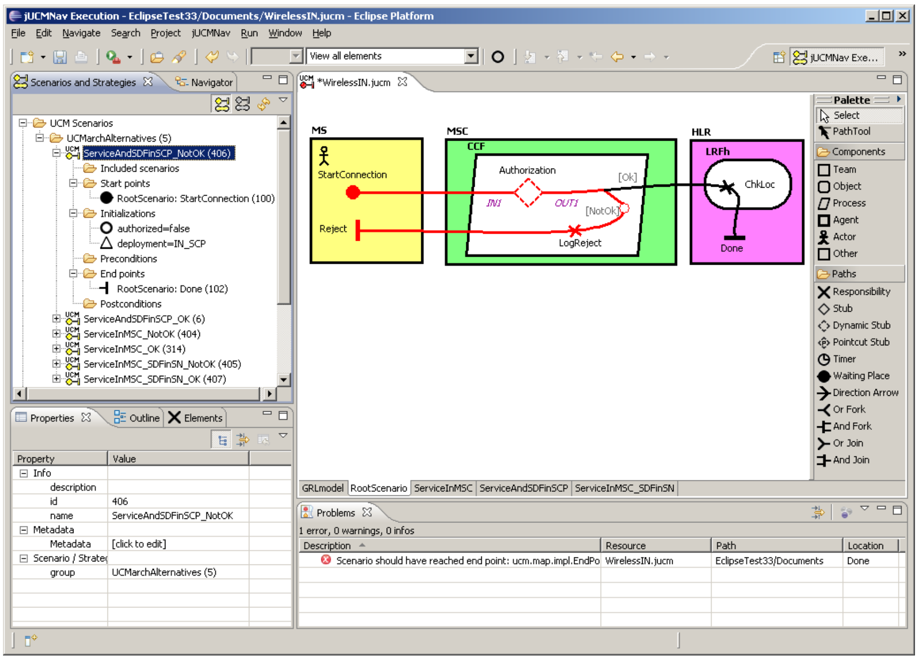


Fig. 6. jUCMNav Tool: URN Editor with Scenario Traversal

3 URN-Based Analysis, Transformations, and Management

The two key URN analysis techniques are GRL model evaluation with strategies and UCM model interpretation based on scenario definitions. These two approaches can also be combined for integrated analysis. Transformations of UCM models to MSCs and performance models also represent typical applications of URN models. Since URN models need to coexist with other types of requirements and design artifacts, they must be amenable to integration with requirements management systems. The next six subsections briefly illustrate each of these approaches.

3.1 Tradeoff Evaluation with GRL Strategies

As GRL models can be used for many different purposes, the goal-oriented modelling community has developed many analysis approaches. It is premature to standardize any of them, but three examples of evaluation algorithms based on strategies are formalized and illustrated in an appendix of proposed Recommendation Z.151. They are all based on the grammar metaclasses of GRL. Two of them are illustrated here for the GRL example introduced in section 2.2, where tradeoffs between stakeholders' objectives need to be analysed. These are bottom-up algorithms that compute the satisfaction level of link targets based on the satisfaction of source intentional elements and the nature of the links. Decomposition links are evaluated first, followed by contributions/correlations and finally dependencies.

Fig. 7 illustrates an example of *qualitative* analysis approach, where qualitative contributions, satisfaction levels, and importance levels are used. The strategy being evaluated here contains three initial satisfaction values (shown with stars *): the data and the service are located in a service control point, and the maximum hardware utilisation is weakly satisfied. These values are propagated to the other intentional elements through the various links. jUCMNav supports this analysis and also utilises a color palette to highlight what is satisfied (green), neutral (yellow), or denied (red) [52]. Globally, one can see the impact of all the decisions on the actors and their top-level goals.

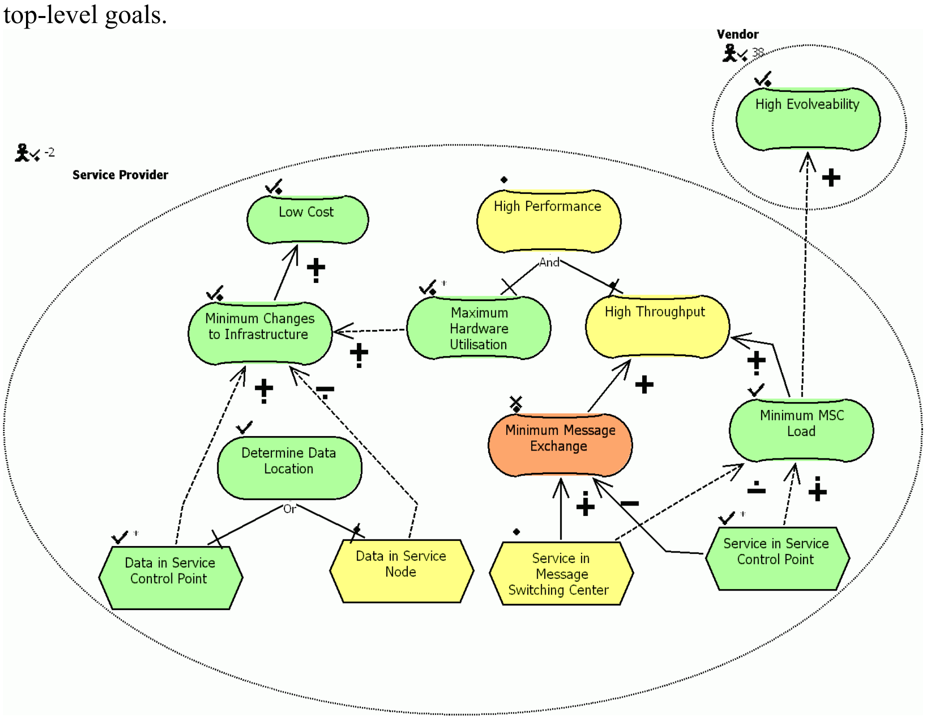


Fig. 7. GRL Model Evaluation: Qualitative Analysis Example

Fig. 8 shows an example of a *quantitative* analysis approach. This time, integer values ranging from -100 (denied) to 0 (none) to +100 (satisfied) are used for satisfaction levels. The same scale applies to negative and positive contribution links. In this example, the service is in a message switching center, the data is in a service node, and the maximum hardware utilisation is partially satisfied. The service provider has more or less the same level of satisfaction as for the previous strategy, but this time the vendor is quite dissatisfied. Hence, the first strategy seems better than this one (although usually they should be both compared based on the same propagation algorithm). Quantitative analysis provides more precise results than a qualitative approach but requires more precise input up front, something that is not always possible.

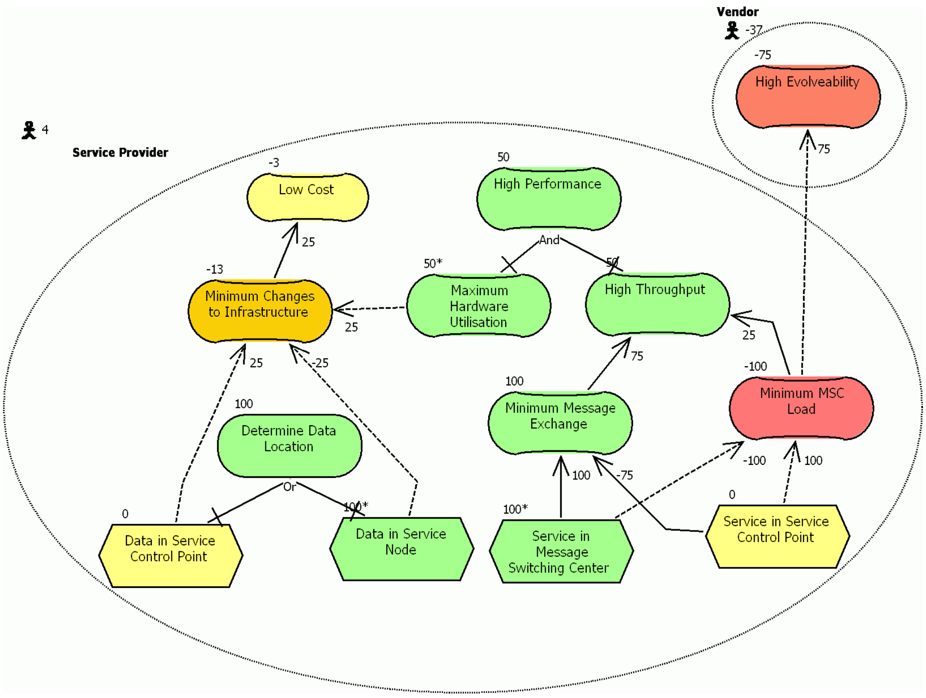


Fig. 8. GRL Model Evaluation: Quantitative Analysis Example

Such evaluations provide rationale and documentation for requirements-related decisions in context. URN also allows for quantitative values and qualitative values to be both used in a single propagation algorithm.

3.2 Scenario Analysis with UCM Scenario Definitions

The UCM path traversal mechanism is based on the abstract grammar metaclasses of the UCM notation. This mechanism traverses a UCM model by starting at the first start point as defined in a scenario definition by the modeller. A scenario definition contains start points, preconditions, expected end points, postconditions, and variable initializations. The actual path to be traversed is determined by the initial, user-defined values of global path variables and the changes to these values at responsibilities during the traversal. The URN data model contains a simple action language that supports variable assignments, expressions, and conditional statements. The path traversal mechanism moves from one path element to the next (forks, joins, timers, stubs, etc.) if *path continuation criteria* are met. If more than one next path element meet the continuation criteria, all of these path elements are visited in parallel. The traversal ends when the last end point is reached. Various errors and warnings are reported when progress is no longer possible or when non-deterministic choices are encountered.

UCM path elements have specific continuation criteria, which are defined among the 80 requirements that collectively capture the dynamic semantics of UCM models. These requirements allow implementers to develop their own traversal algorithm and optimize or extend various aspects of it according to their needs. jUCMNav implements such a traversal mechanism.

The pre- and postconditions of scenario definitions that must be met respectively at the beginning and at the end of the traversal enable the testing and validation of the model. Definitions can also be grouped and they can include other scenario definitions to simplify the maintenance and use of large collections of scenarios.

As an example, suppose two simple scenario definitions for the UCM model of section 2.3:

- ServiceInMSC_OK: The start point is StartConnection, the authorization variable is *true* ([Ok]), and the selected deployment plug-in map for the dynamic stub is SvcInMSC.
- ServiceInMSC_DataInSN_NotOK: The start point is StartConnection, the authorization variable is *false* ([NotOk]), and the selected deployment plug-in map for the dynamic stub is SvcInMSC_DataInSN.

jUCMNav uses such scenario definitions to highlight, in red, the paths traversed while running the scenario on the UCM model (see Fig. 6). Any error or warning is reported in the Eclipse Problems view. In addition, the scenario path resulting from the traversal can be exported as a separate URN model, so a flattened representation of the scenario can be visualized as a UCM diagram. For instance, the above scenario definitions resulted in the two diagrams in Fig. 9. Only the path elements and components traversed are represented. Also, stubs are flattened and alternatives are resolved thanks to the variable initializations in the scenario definition.

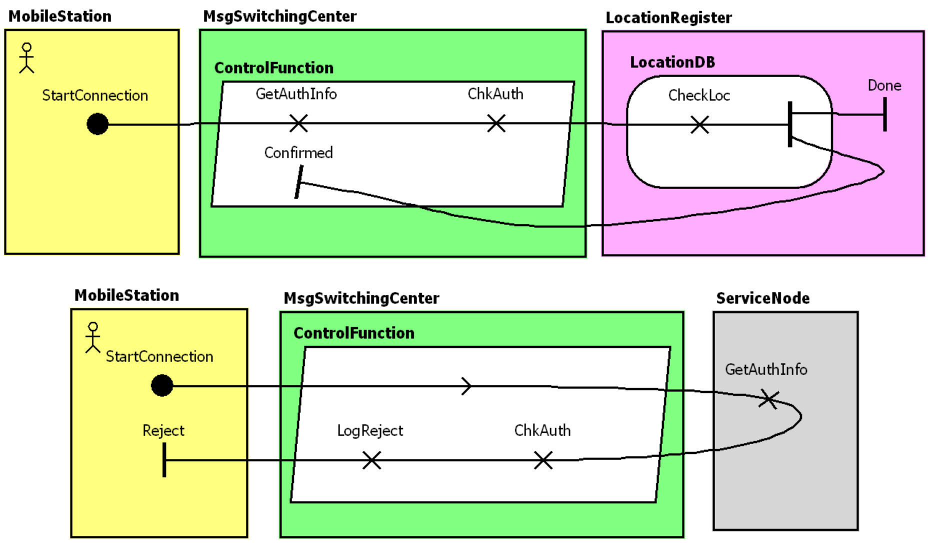


Fig. 9. UCM Models Resulting from the Two Scenarios

3.3 Combining Strategies with Scenario Definitions

GRL strategies and UCM scenario definitions can be used jointly, and their results can influence each other. The following example is not strictly part of the Z.151 proposed standard but it is not forbidden either. In [29], jUCMNav was extended to enable the automatic creation of UCM integer variables for each GRL intentional element. The current satisfaction level of an intentional element can hence be used in a UCM conditional expression, hence influencing the selection of paths at OR-forks and of plug-ins in dynamic stubs. Also, responsibilities can assign new values to these variables, hence influencing the propagation of satisfaction levels in a GRL graph.

This functionality was explored in [3] to support dynamic composition of services and runtime adaptation of changing context. The approach was illustrated using a multimedia call service with access control requirements.

3.4 Transformation to Message Sequence Charts

Scenarios resulting from UCM path traversals can be transformed to representations other than UCM. For many years, UCM scenarios have been converted to Message Sequence Charts in order to visualize scenarios in a scalable, linear form as well as to pave the way towards more detailed design activities where messages and component states need to be considered [33]. The main challenges here are to infer necessary messages ensuring that causal relationships between responsibilities in different components are correctly supported, and to handle the well-formedness rules of a linear scenario representation like MSCs, which are stricter than the general graph representation of UCMs.

jUCMNav supports the export of scenarios to MSCs [25]. For example, Fig. 10 and Fig. 11 present MSCs that correspond to the scenarios of section 3.2 and shown in Fig. 9. MSC instances are created for the UCM components involved, and each UCM responsibility becomes an MSC action. Start and end points are converted to self-messages. UCM conditions evaluated at OR-forks and in dynamic plug-ins are preserved as MSC conditions. Concurrent paths are converted to parallel inline statements. Note how messages, which do not exist in UCM, were synthesized during the transformation in order to ensure causality of actions in and across MSC instances. These abstract messages can be refined later on (with parameters, better names, or with more complex sequences of messages) as we progress towards protocol definitions.

3.5 Transformation to Performance Models

As discussed in section 2.3, modellers can supplement UCM model elements with performance annotations. These are not taken into consideration for the path traversal mechanism, but they can be used in transformations of UCM models to specialized performance models. This enables performance analysis from URN requirements models, before serious barriers to performance are frozen into the design and implementation.

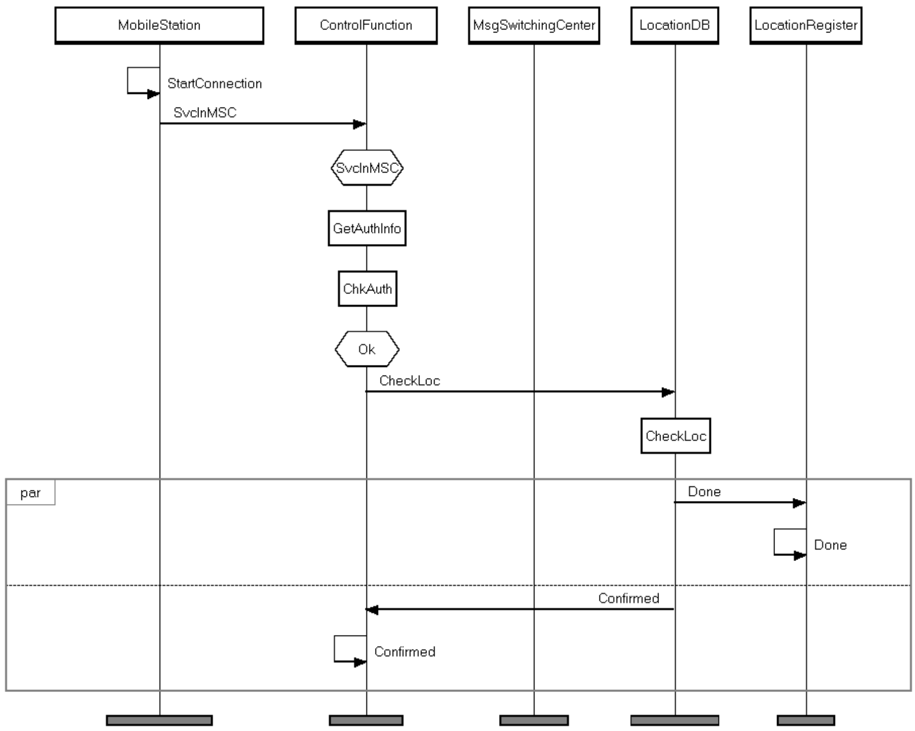


Fig. 10. Message Sequence Chart Resulting from Scenario ServiceInMSC_OK

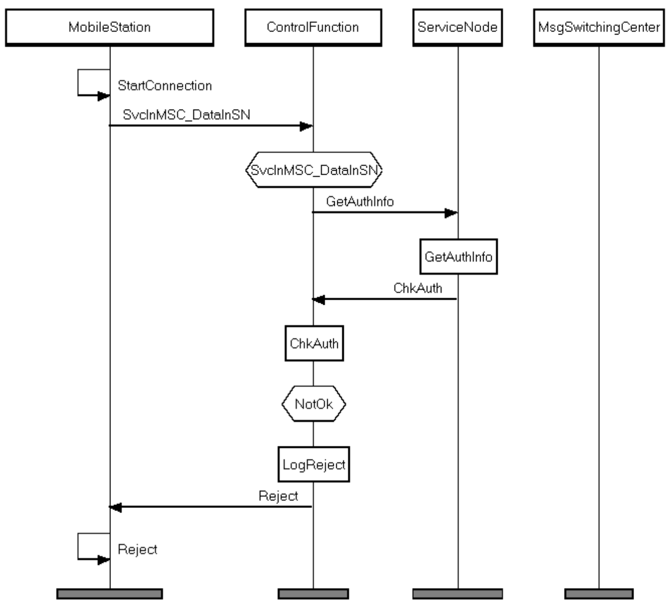


Fig. 11. Message Sequence Chart Resulting from Scenario ServiceInMSC_DataInSN_NotOK

The generation of *Layered Queueing Network* (LQN) performance models [14] directly from UCM models was explored and prototyped in [45]. LQN performance models can be used as a basis for exploring the performance solution space of a system. The kinds of analysis that can be performed include sensitivity analysis (how important are different values for parameters, especially estimated values), scalability analysis (how well does the system cope with more users or higher workload), concurrency analysis (how does the system respond to changes in the number of processing resources) and configuration analysis (how does the system respond to different deployments configurations in terms of bandwidth limitations, network delays, etc.).

More recently, the annotations have evolved to become more in line with the new UML profile for real time and embedded systems (MARTE). Also, instead of generating performance models directly, newer approaches now target the generation of an intermediate representation, such as the *Core Scenario Model* (CSM) representation [48]. CSM's purpose is to capture the essence of a range of scenario notations (e.g., from URN and UML) and enable simple transformations to various target formalisms (e.g., LQN, regular queueing networks, and stochastic Petri Nets), hence reducing the number and complexity of tools needed to analyze various aspects of a same system. The UCM annotations are very close to CSM's.

A transformation from UCM to CSM is defined in [58]. It has recently been adapted to a more recent version of the URN metamodel and implemented in jUCMNav. One of the main benefits of this approach is that the acquisition and release of resources is inferred implicitly from UCM models rather than requiring them to be defined explicitly as in profiled UML models. This simplifies substantially the creation and maintenance of models. Transformations from CSM models to LQN models and other types of performance models are discussed in [47] and are now supported by prototype tools.

3.6 Requirements Management

URN models capture only a fraction of the requirements of telecommunication standards and software products. Accordingly, such models need to be used in cooperation with complementary general requirements, and both views must be linked in a way that supports traceability, navigation, and analysis. The proposed URN standard ensures that model elements are uniquely identifiable inside a specification, which helps supporting such links. However, one of the main challenges that needs to be addressed here is the maintenance of these links as models and general requirements evolve.

In [30, 46], an approach is proposed to introduce scenario models into the Telelogic DOORS requirements management system and to maintain relationships as both views evolve over time. This is supported as an export filter for jUCMNav. The tool also provides a link auto-completion mechanism to minimize the possibly large number of links that have to be created manually by the DOORS user between external and UCM requirements. Automatically created links allow the DOORS user to become aware of and exploit links directly that would otherwise need to be discovered transitively via (potentially many) intermediate links. This tool was extended to support GRL in [52].

4 Standards Development with URN

Most of the URN notation elements and analysis, transformation, and management techniques covered in the previous two sections are directly applicable to the development of a wide range of telecommunications standards.

Pioneering ideas on how URN could complement standardization development processes and existing formal description techniques in the wireless domain were provided nearly a decade ago [20]. They were further explored for a group communication service [4] and wireless ATM mobile networks [8]. It was realized that goals and scenarios would fit particularly well the so-called "stage 1" requirements descriptions described in Recommendation I.130 [21]. These motivated in part the work that led to the proposed URN standard. UCMs were actually used in several proposals for new services for Wireless Intelligent Networks. UCM models were used to guide the development of an SDL specification for an IETF standard targeting a complex refreshment function for the Open Shortest Path First (OSPF) dynamic routing protocol in IP networks [34].

In the telecommunications networks management domain, Recommendation M.3020 proposes the description of various types of requirements (functional, non-functional, administrative, etc.) with textual use case and UML use case diagrams [22]. Again, URN models fit nicely in such a process as they bring formality and executability to the use cases while enabling concrete support for goal models, which are useful to derive and analyse non-functional and administrative requirements.

More recently, the growing interest in Next-Generation Networks brought new needs for improved service description and engineering approaches. 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. This is essentially the abstraction level targeted by URN, as discussed in [3]. 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. The integration of GRL and UCM, as well as of strategies and scenario definitions as discussed in section 3.3, emphasizes the importance of enabling dynamic choices in the service modelling and design phases in order to take into account contextual information and differentiated service availability requirements in dynamic service composition, which are key aspects of NGN services.

5 Other Applications of URN

This section discusses a subset of the extensive body of research comprising over 200 publications and theses related to UCM, GRL, and URN available at the URN Virtual Library [54]. Many examples of URN models can be found in the publications referenced in this section.

Over the last decade, GRL and UCM have successfully been used for different types of service-oriented, concurrent, distributed, and reactive systems outside the telecommunications domain, including Web and e-commerce systems [6, 51], operating systems [9], and health information systems [42, 50].

Business process modelling is a growing area of application for URN, where its combination of goals and scenarios is a perfect fit for business objectives and processes/workflows [35, 37, 55]. In [49], GRL is extended with the concept of Key Performance Indicators (KPIs), and jUCMNav is enhanced to support new analysis features for managing business processes and for better aligning scenarios with goals. Strategies are also extended to access external sources of information (e.g. data warehouses or performance management tools) for online monitoring and runtime adaptation of business processes. The DOORS integration for requirements management discussed in section 3.6 is further extended to assess and maintain compliance of business processes and organizational policies with governmental policies, regulations, and legislation [15, 50].

In terms of other analysis and transformations approaches, it is worth noting the generation of test goals from UCM models [6, 7], the detection of undesirable interactions between (telecommunications) features and services [4, 31, 53, 56], the experimental synthesis of state machines in SDL [19] and UML [12] from UCM models, the specification and analysis of user interface requirements [1], and the formalization of patterns [40]. URN models can also be used in reverse-engineering and re-engineering contexts to describe existing systems and services [32], and both static and dynamic approaches to recovering UCM scenarios from code [5] and execution traces [16] have been proposed. Formal semantics and time extensions, which enable formal verification, are explored for UCMs in [17, 18].

Given the recent interest in aspect-oriented software development and the potentially high benefits of using such concepts to better encapsulate crosscutting concerns in requirements models, new extensions have been proposed for GRL and UCM in order to create a unified Aspect-oriented URN (AoURN) [36, 39, 41]. Partial tool support in jUCMNav is already available. The application of AoURN to the modelling of software product lines is explored in [38].

6 Conclusions

This paper presents an overview of the User Requirements Notation (URN) as defined in the proposed Recommendation Z.151. The two constituent languages of URN, the Goal-oriented Requirement Language (GRL) and the Use Case Map (UCM) notation are discussed as well as the analysis capabilities of URN, most notably GRL model evaluation with strategies and UCM model interpretation based on scenario definitions. Resting on the pillars of these two languages, URN complies with Recommendation Z.150, satisfying language requirements for both, non-functional and functional, requirements.

URN is a general purpose modelling language for the communication of and reasoning about requirements. URN can be applied to standards development as well as to telecommunications systems, services, business processes, and most types of reactive systems and information systems. URN spans the areas of business goals and requirements descriptions to high-level design activities. Examples for the applicability of URN for standards development are given. Tool support provided by the Eclipse-based jUCMNav plug-in is discussed and used to illustrate the potential benefits of using the notation.

The last part of the paper gives an overview of other types of applications as well as an outlook into the future of URN, illustrating possible future extensions of the proposed Recommendation Z.151. In particular, we can identify a better support for improved requirements for GRL evaluation algorithms, advanced workflow patterns [37], support for time in UCM [18], formal semantics (based on abstract state machines) [17], extensions for business process modelling with KPIs [49], and aspect-oriented extensions for URN [36].

Acknowledgments. This research was supported by the Natural Sciences and Engineering Research Council of Canada, through its programs of Discovery Grants and Postgraduate Scholarships.

References

1. Alsumait, A.: *User Interface Requirements Engineering: A Scenario-Based Framework*. Ph.D. thesis, Concordia University, Canada, August 2004.
2. Amyot, D.: Introduction to the User Requirements Notation: Learning by Example. *Computer Networks*, Vol. 42(3), 285-301, 21 June 2003.
3. Amyot, D., Becha, H., Bræk, R., and Rossebø, J.E.Y.: Next Generation Service Engineering. *ITU-T Innovations in NGN Kaleidoscope Conference*, Geneva, Switzerland, May 2008.
4. Amyot, D. and Logrippo, L.: Use Case Maps and LOTOS for the Prototyping and Validation of a Mobile Group Call System. *Computer Communication*, Vol. 23(12), 1135-1157, July 2000.
5. Amyot, D., Mussbacher, G., and Mansurov, N.: Understanding Existing Software with Use Case Map Scenarios. *3rd SDL and MSC Workshop (SAM02)*, Aberystwyth, U.K. LNCS 2599, Springer, 124-140, June 2002.
6. Amyot, D., Roy, J.-F., and Weiss. M.: UCM-Driven Testing of Web Applications. Prinz A., Reed R., and Reed J. (Eds) *SDL 2005: Model Driven*, LNCS 3530, Springer, 247-264, June 2005.
7. Amyot, D., Weiss, M., and Logrippo L.: UCM-Based Generation of Test Purposes. *Computer Networks*, 49(5), 643-660, December 2005.
8. Andrade, R.: Applying Use Case Maps and Formal Methods to the Development of Wireless Mobile ATM Networks. *Lfm2000: Fifth NASA Langley Formal Methods Workshop*, Williamsburg, Virginia, USA, 151-162, June 2000.

9. Billard, E.A.: Operating system scenarios as Use Case Maps. *ACM Workshop on Software and Performance*, 266-277, 2004.
10. Buhr, R.J.A. and Casselman, R.S.: *Use Case Maps for Object-Oriented Systems*. Prentice-Hall, 1996.
11. Buhr, R.J.A.: Use Case Maps as Architectural Entities for Complex Systems. *IEEE Transactions on Software Engineering*, Vol. 24(12), 1131-1155, December 1998.
12. Castejón Martínez, H.N.: Synthesizing State-Machine Behaviour from UML Collaborations and Use Case Maps. Prinz A., Reed R., and Reed J. (Eds) *SDL 2005: Model Driven*, LNCS 3530, Springer, 339-359, June 2005.
13. Chung, L., Nixon, B.A., Yu, E., and Mylopoulos, J.: *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, Dordrecht, USA, 2000.
14. Franks, G. Maly, P., Woodside, M., Petriu, D.C., and Hubbard, A.: *Layered Queueing Network Solver and Simulator User Manual*. Carleton University, Dec. 2005. http://www.sce.carleton.ca/rads/lqns/LQNSUserMan.pdf
15. Ghanavati, S., Amyot D., and Peyton, L.: Towards a Framework for Tracking Legal Compliance in Healthcare. *19th Int. Conf. on Advanced Information Systems Engineering (CAiSE'07)*, Trondheim, Norway. LNCS 4495, Springer, 218-232, June 2007.
16. Hamou-Lhadj, A., Braun, E., Amyot, D., and Lethbridge, T.: Recovering Behavioral Design Models from Execution Traces. *9th European Conference on Software Maintenance and Reengineering (CSMR)*, IEEE Computer Society, 112-121, March 2005.
17. Hassine, J., Rilling, J., and Dssouli, R.: An ASM Operational Semantics for Use Case Maps. *13th IEEE International Requirement Engineering Conference (RE05)*, IEEE CS Press, 467-468, September 2005.
18. Hassine, J., Rilling, J., and Dssouli, R.: Formal Verification of Use Case Maps with Real Time Extensions, *13th SDL Forum (SDL'07)*, Paris, France. LNCS 4745, Springer, 225-241, September 2007.
19. He, Y., Amyot, D., and Williams, A.W.: Synthesizing SDL from Use Case Maps: An Experiment. *11th SDL Forum (SDL'03)*, Stuttgart, Germany, July 2003. LNCS 2708, Springer, 117-136.
20. Hodges, J. and Visser, J.: Accelerating Wireless Intelligent Network Standards Through Formal Techniques. *IEEE 1999 Vehicular Technology Conference (VTC'99)*, Houston, Texas, USA, 1999.
21. ITU-T – International Telecommunications Union: *Recommendation I.130 (11/88), Method for the characterization of telecommunication services supported by an ISDN and network capabilities of an ISDN*. Geneva, Switzerland, November 1988.
22. ITU-T – International Telecommunications Union: *Recommendation M.3020 (07/07), Management interface specification methodology*. Geneva, Switzerland, July 2007.
23. ITU-T – International Telecommunications Union: *Recommendation Z.100 (11/07), Specification and Description Language*. Geneva, Switzerland, April 2008.
24. ITU-T – International Telecommunications Union: *Draft Recommendation Z.111, Notations to Define ITU-T Languages*. Geneva, Switzerland, September 2008.
25. ITU-T – International Telecommunications Union: *Recommendation Z.120 (04/04), Message Sequence Chart (MSC)*. Geneva, Switzerland, 2004.
26. ITU-T – International Telecommunications Union: *Recommendation Z.150 (02/03), User Requirements Notation (URN) – Language Requirements and Framework*. Geneva, Switzerland, February 2003.
27. ITU-T – International Telecommunications Union: *Draft Recommendation Z.151, User Requirements Notation (URN)*. Geneva, Switzerland, September 2008.
28. jUCMNav 3.2, University of Ottawa, September 2008. http://jucmnav.softwareengineering.ca/jucmnav/
29. Kealey, J. and Amyot, D.: Enhanced Use Case Map Traversal Semantics. *13th SDL Forum (SDL'07)*, Paris, France. LNCS 4745, Springer, 133-149, September 2007.

30. Kealey, J., Kim, Y., Amyot, D., and Mussbacher, G.: Integrating an Eclipse-Based Scenario Modeling Environment with a Requirements Management System. *2006 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE06)*, Ottawa, Canada, 2432-2435, May 2006.
31. Leelaprute, P., Nakamura, M., Matsumoto, K., and Kikuno, T.: Derivation and Evaluation of Feature Interaction Prone Scenarios with Use Case Maps. *IEICE Transactions on Communications*, Vol. J88-B, No.7, 1237-1247, July 2005.
32. Medve, A.: Advanced steps with standardized languages in the re-engineering process. *Computer Standards & Interfaces*, 30 (5), 315-322, July 2008.
33. Miga, A., Amyot, D., Bordeleau, F., Cameron, C. and Woodside, M.: Deriving Message Sequence Charts from Use Case Maps Scenario Specifications. *Tenth SDL Forum (SDL'01)*, Copenhagen, Denmark, June 2001. LNCS 2078, Springer, 268-287.
34. Monkewich, O., Sales, I., and Probert, R.L.: OSPF Efficient LSA Refreshment Function in SDL. *Tenth SDL Forum (SDL'01)*, Copenhagen, Denmark. LNCS 2078, Springer, 300-315, June 2001.
35. Mussbacher, G.: Evolving Use Case Maps as a Scenario and Workflow Description Language. *10th Workshop on Requirements Engineering (WER'07)*, 56-67, Toronto, Canada, May 2007.
36. Mussbacher, G.: Aspect-Oriented User Requirements Notation: Aspects in Goal and Scenario Models. *Models in Software Engineering: Workshops and Symposia at MoDELS 2007*. Giese, H. (Ed.), LNCS 5002, Springer, 305-316, 2008.
37. Mussbacher, G. and Amyot, D.: Assessing the Applicability of Use Case Maps for Business Process and Workflow Description. *3rd Int. MCeTech Conference on eTechnologies*, Montréal, Canada. IEEE Computer Society, 219-222, January 2008.
38. Mussbacher, G., Amyot, D., Araújo, J., and Moreira, A.: Modeling Software Product Lines with AoURN. *Early Aspects Workshop @ AOSD08*, Brussels, Belgium, March 2008.
39. Mussbacher, G., Amyot, D., and Weiss, M.: Visualizing Early Aspects with Use Case Maps. *Transactions on Aspect-Oriented Software Development III*, Springer, 105-143, 2007.
40. Mussbacher, G., Amyot, D., and Weiss, M.: Formalizing Patterns with the User Requirements Notation. T. Taibi (Ed.), *Design Pattern Formalization Techniques*, IGI Global, 304-325, March 2007.
41. Mussbacher, G., Amyot, D., Whittle, J., and Weiss M.: Flexible and Expressive Composition Rules with Aspect-oriented Use Case Maps (AoUCM). *10th International Workshop on Early Aspects (EA 2007)*, Vancouver, Canada. LNCS 4765, 19-38, December 2007.
42. Ölvingson, C., Hallberg, N., Timpka, T., and Lindqvist, K.: Requirements Engineering for Inter-Organizational Health Information Systems with Functions for Spatial Analyses: Modeling a WHO Safe Community Applying Use Case Maps. *Methods of Information in Medicine*, Schattauer Gmb H, 299-304, 4/2002.
43. OMG – Object Management Group: *Object Constraint Language Specification, version 2.0*, May 2006.
44. OMG – Object Management Group: *Unified Modeling Language (OMG UML): superstructure version 2.1.2*, November 2007.
45. Petriu, D.B., Amyot, D., and Woodside, M.: Scenario-Based Performance Engineering with UCMNav. *11th SDL Forum (SDL'03)*, Stuttgart, Germany, July 2003. LNCS 2708, Springer, 18-35.
46. Petriu, D.B., Amyot, D., Woodside, M, and Jiang, B.: Traceability and Evaluation in Scenario Analysis by Use Case Maps. In: S. Leue and T. Systä (Eds.) *Scenarios: Models, Algorithms and Tools*, LNCS 3466, Springer, 134-151, 2005.
47. Petriu, D.B. and Woodside, M.: Software performance models from system scenarios. *Performance Evaluation*, 61(1), Elsevier, 65-89, June 2005.

48. Petriu, D.B. and Woodside, M.: An intermediate metamodel with scenarios and resources for generating performance models from UML designs. *Software and Systems Modeling*, 6(2), Springer, 163-184, June 2007.
49. 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. In *3rd Int. MCeTech Conference on eTechnologies*. Montréal, Canada. IEEE Computer Society, 3-15, January 2008.
50. Pourshahid, A., Peyton, L., Ghanavati, S., Amyot, D., Chen, P., and Weiss, M.: Model-Based Validation of Business Processes. V. Shankararaman, J.L. Zhao and K.K. Lee (Eds) *Business Process Management: Concepts, Technology, and Application*. Advances in Management Information Systems, M. E. Sharpe Inc., 2008 (to appear).
51. Pourshahid, A., and Tran, T.: Toward an Effective Trust Management System for E-Commerce: Modeling Trust Components and Processes Using URN. *Journal of Business and Technology (JBT)*, Atlantic Academic Press, 2008 (to appear).
52. Roy, J.-F., Kealey, J., and Amyot, D.: Towards Integrated Tool Support for the User Requirements Notation. *SAM 2006: Language Profiles - Fifth Workshop on System Analysis and Modelling*, Kaiserslautern, Germany. LNCS 4320, Springer, 183-197, May 2006.
53. Shiri, M., Hassine, J., and Rilling, J.: Feature Interaction Analysis: A Maintenance Perspective. *22nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 437-440, Atlanta, USA, November 2007.
54. User Requirements Notation Virtual Library, 2008, http://www.usecasemaps.org/pub/
55. Weiss, M. and Amyot, D.: Business Process Modeling with URN. *International Journal of E-Business Research*, Vol. 1(3), 63-90, July-September 2005.
56. Weiss, M., Esfandiari, B., and Luo ,Y.: Towards a Classification of Web Service Feature Interactions. *International Conference on Service-Oriented Computing (ICSOC)*, Amsterdam, Netherlands. LNCS 3826, Springer, 101-114, November 2005.
57. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. *3rd IEEE Int. Symp. on Requirements Engineering*, Washington, USA. IEEE CS, 226-235, 1997.
58. Zeng, X.Y.: *Transforming Use Case Maps to the Core Scenario Model Representation*. M.C.S. thesis, University of Ottawa, June 2005.