User Requirement Notation (URN)

Sakkina Banu Sikandar-gani
Graduate Student,
Department of Electrical
and Computer Engineering,
Mississippi State University,
MS, USA

Abstract – The International Telecommunication Union Study Group 10 (ITU-T, SG 10) has recently approved User Requirement Notation (URN) as a standard requirements notation for complex reactive, distributed and dynamic systems and applications. These include e-commerce applications, Web applications, wireless or IP-based telecommunications systems and other types of real-time embedded, distributed or reactive systems. Currently, URN proposed two languages: Use Case Maps (UCM) and Goal-oriented Requirements Language (GRL) to describe functional and non-functional requirements respectively. This paper attempts to describe what is URN, UCM and GRL, what does URN do, what problem does URN try to solve, how does the technology (URN) help in solving the problem, how does URN work, what are the advantages and disadvantages of URN, and what are the technical requirements to implement this technology.

Index Terms – User Requirements Notation (URN), Use Case Maps (UCM) and Goal-Oriented Requirement Language (GRL).

1. Introduction

Requirements engineering technique has been included as part of the development approaches in systems (reactive, distributed and dynamic systems) of various sizes and natures. Requirement engineering is a process that involves developing, eliciting, specifying and analyzing the stakeholder requirements that needs to be met by the systems [1].These systems are becoming increasingly complex in functionalities, architectures and protocols because they are becoming more dynamic in nature and evolve at run time [2]. For examples, users mobility and unpredictable policy-driven negotiations become new dimensions of complexity in wireless system and in agent-based communication technology (such as XML and IP), respectively [2]. In order to provide precise descriptions of the systems, a standardized notation and technique of requirement engineering has been approved by ITU-T, SG 10 called URN. URN is described in more details in section 2. URN was developed by researchers at Mitel Corporation, Carleton University (Ottawa, Canada) and University of Ottawa, Ontario, Canada [3]. URN technology is supported by telecommunication and network-based companies such as Mitel Corporation, Nortel, CITO and NSERC [3]. URN will provide a very useful requirement engineering technique to requirement engineers, systems architects, software engineers, Unified Modeling Language (UML) community
and test engineers [4]. UCM, which is a part of URN, can be used by requirement engineers to capture, elicit and validate user requirements. It can be used by systems architects in high-level design where alternative architectures are evaluated and architectural decisions are made at this level. It can also be used by software engineer in developing dynamic systems. It can be used by UML community to bridge the gap between requirements and design, and it can be used by test engineers to generate high-level test cases for the systems [4].

Figure 1. Relationship between URN, UCM and GRL

2. URN

URN combines two complementary languages to attain the important aspects of requirement engineering technique. Those two languages are GRL, which is used to capture non-functional requirements and UCM, which is used to capture functional requirements of the systems. URN attains the important aspects of requirement engineering technique by combining goals (from GRL) and scenarios (from UCM). With this combination, it has the ability to capture goals and decision rationale (through GRL), ability to move smoothly from analysis models to design models (through UCM) and ability to model dynamic systems whose behavior and structures change at run time (through UCM) [5]. The motivation for URN is to provide visual description and analysis of requirements of the systems. Diagram representation of motivation for URN is shown in Figure 2.

Figure 2. Motivation for URN [1].

In Stage 1, system is described in the user’s point of view in textual form, use cases, tables and informal diagrams. Stage 2 focuses on control flows between the entities and Stage 3 focuses on specifications of components [5]. URN is a higher level of modeling than UML or SDL since it does not require messages, components and components states. It captures user requirements with very little design detail. Since it focuses on performance analysis and design (with scenarios) at the early stages, undesirable interactions between features can be detected [5]. URN provides reusability of scenarios and components allocation to assist in alternatives architectural evaluation. It also provides dynamic refinement capabilities that enable design of unpredictable policy-driven negotiations systems. URN supports traceability and transformation to other languages such as UML. The main advantage of URN is
that it enables the designers to analyze the feature interactions and performance trade-off early in the design process so that appropriate decisions can be made. The disadvantage of URN is that its prime application domain is telecommunication services. URN usage in other domains has not yet been explored extensively.

3. GRL

GRL is a graphical notation for describing non-functional user requirements, business goals, alternatives and rationales. It supports goal and agent-oriented modeling and reasoning about requirements. These include usability, user interface, performance, operational, maintainability, security, social context – cultural and political, and legal requirements [6]. Initially the stakeholders will provide requirements in terms of objects and desired goals. By representing these goals in GRL instead of activities and entities, designers would be able to analyze various alternatives and trade-off. Then, the decision can be made wisely based on the early analysis. GRL has four main categories of concepts: intentional elements, intentional relations, actors and non-intentional elements. The intentional elements are goal, task, softgoal, resource and belief. These are called intentional because they are used to model answers to following questions:

- Why particular behaviors, information and structures were chosen for the system?
- What alternatives were considered?
- What criteria were used to deliberate among alternative options?
- What were the reasons for choosing one alternative over the other? [7]

Details of processes or systems requirements are omitted in GRL. Basic GRL notations are shown in Figure 3.

![Basic GRL Notation](image)

Figure 3. Basic GRL Notation [8].

Figure 4 shows concepts and notations in GRL.

- **Goal**: Quantifiable high-level (functional) requirement (illustrated as a rounded-cornered rectangle).
- **Softgoal**: Qualifiable but unquantifiable requirement, essentially non-functional (illustrated as a cloud).
- **Task**: Operationalized solution that achieves a goal, or that satisfies a softgoal which can never be fully achieved due to its fuzzy nature (illustrated as a hexagon).
- **Resource**: Entity whose importance is described in terms of its availability (illustrated as a rectangle).
- **Belief**: Rationale or argumentation associated to a contribution or a relation (illustrated as an ellipse).

There are also five categories of intentional relations, which connect elements:
- **Contribution**: Describes how softgoals, tasks, beliefs, and relations contribute to each other. Each contribution can be qualified by a degree: equal, break, hurt, some, undetermined, some, help, or none (see Fig. 1).
- **Correlation**: Contribution that indicates side-effects on other intentional elements (Jackson’s). An AND (“&”) link for tasks achieving goals. Different alternatives are allowed.
- **Decomposition**: Defines what is needed for a task to be performed (repetition), always AND.
- **Dependency**: Link between two actors depending on each other (half-circle).

Figure 4. Concepts and Notations in GRL [5].

4. UCM

UCM is a graphical scenario notation for capturing the functional requirements of reactive, distributed and dynamic systems. UCM uses causal relationships between responsibilities to describe
Causal relationships are when there is/are links between causes (such as preconditions and triggering events) to the effects (such as postconditions and resulting events) [9]. Scenarios are expressed at a higher level than message exchange between components level. Main advantages of UCM are that it can model dynamic refinement for various behaviors and structure, and it can integrate behavior and structural components in a single view [9]. Hence, bridging the gap between requirements and design. Initially the behavioral aspect of the system is described using UCMs. Then, the UCMs will be refined and component structures will be added to the map. From UCMs, more detailed UML models will be derived from other modeling languages. Disadvantages of UCM are that, stand-alone verification and validation is hard (because of its informality and looseness ) and it has to be translated to other tools before it can be done [3]. Also, the support for UCM is still weak because of limited number of prototyping editing tool.

Figure 5 shows UCM concepts and notations.

- **Start point**: Captures preconditions and triggering events (filled circle).
- **Responsibilities**: locators where computation (procedure, activity, function, etc.) is necessary (cross).
- **End point**: Represents resulting events and post-conditions (bar).
- **Pens**: Connects start points to end points and can link responsibilities in a causal way.

**Figure 5.** UCM concepts and notations [5].

### 5. References

http://www.UseCaseMaps.org/urn/


http://www.UseCaseMaps.org/pub