

I n t e r n a t i o n a l T e l e c o m m u n i c a t i o n U n i o n

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.150

(02/2011)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE
ASPECTS FOR TELECOMMUNICATION SYSTEMS

Formal description techniques (FDT) – User Requirements
Notation (URN)

User Requirements Notation (URN) – Language requirements and framework

Recommendation ITU-T Z.150



International
Telecommunication
Union

Document Sharing Hub

ITU-T Z-SERIES RECOMMENDATIONS
LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

FORMAL DESCRIPTION TECHNIQUES (FDT)	
Specification and Description Language (SDL)	Z.100–Z.109
Application of formal description techniques	Z.110–Z.119
Message Sequence Chart (MSC)	Z.120–Z.129
User Requirements Notation (URN)	Z.150–Z.159
Testing and Test Control Notation (TTCN)	Z.160–Z.179
PROGRAMMING LANGUAGES	
CHILL: The ITU-T high level language	Z.200–Z.209
MAN-MACHINE LANGUAGE	
General principles	Z.300–Z.309
Basic syntax and dialogue procedures	Z.310–Z.319
Extended MML for visual display terminals	Z.320–Z.329
Specification of the man-machine interface	Z.330–Z.349
Data-oriented human-machine interfaces	Z.350–Z.359
Human-machine interfaces for the management of telecommunications networks	Z.360–Z.379
QUALITY	
Quality of telecommunication software	Z.400–Z.409
Quality aspects of protocol-related Recommendations	Z.450–Z.459
METHODS	
Methods for validation and testing	Z.500–Z.519
MIDDLEWARE	
Processing environment architectures	Z.600–Z.609

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Z.150

User Requirements Notation (URN) – Language requirements and framework

Summary

Recommendation ITU-T Z.150, together with other Recommendations in the ITU-T Z.150 series, defines URN (User Requirements Notation), which is intended for the elicitation, analysis, specification, and validation of requirements. URN combines modelling concepts and notations for goals (mainly for non-functional requirements and quality attributes) and scenarios (mainly for operational requirements, functional requirements, and performance and architectural reasoning). Such a notation is needed to capture user requirements prior to any design. This Recommendation focuses on language requirements for URN and on providing the context for a requirements engineering framework.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T Z.150	2003-02-13	17
2.0	ITU-T Z.150	2011-02-13	17

Keywords

Evaluation, formal specification, functional requirements, goal, graphical notation, hierarchical decomposition, non-functional requirements, requirements engineering activities, scenario, transformation.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2011

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

		Page
1	Scope	1
1.1	Motivation	1
1.2	Document organization	3
2	References.....	3
3	Definitions	3
4	Abbreviations and acronyms	5
5	Scope of URN.....	5
5.1	What is URN?.....	5
5.2	What is URN-NFR?	6
5.3	Why goal-oriented requirements engineering?	7
5.4	What is URN-FR?	8
5.5	Intended usage	8
6	Language requirements for URN-NFR	9
6.1	Expressing tentative, ill-defined and ambiguous requirements.....	9
6.2	Clarifying, exploring, and satisficing goals and requirements	9
6.3	Expressing and evaluating measurable goals and NFRs	10
6.4	Argumentation	10
6.5	Linking high-level business goals to system requirements	10
6.6	Multiple stakeholders, conflict resolution and negotiation support	10
6.7	Requirements prioritization	10
6.8	Requirements creep and churn and other evolutionary forces	10
6.9	Integrated treatment of functional and non-functional requirements	11
6.10	Multiple rounds of commitment.....	11
6.11	Life-cycle support.....	11
6.12	Traceability.....	11
6.13	Ease of use and precision	11
6.14	Modularity	12
6.15	Reusable requirements.....	12
6.16	Performance indicators	12
7	Language requirements for URN-FR	12
7.1	System trigger and termination conditions.....	12
7.2	System operations and responses	13
7.3	Complex and lengthy behaviour.....	13
7.4	Relationships among scenarios.....	14
7.5	Scenario cancellation and failure handling.....	14
7.6	Time-dependent behaviour	14

	Page
7.7 Component definition	15
7.8 Environment specification	15
8 Other language requirements for URN	15
8.1 Requirements traceability	15
8.2 Requirements test case specification	17
8.3 Evaluation of goal and NFR satisfaction	17
8.4 Performance analysis of requirements	18
8.5 Change management	18
8.6 Concrete representations	18
8.7 Usability	19
8.8 Extensibility	19
9 Language requirements summary	19
9.1 Requirements table format	19
9.2 URN requirements table	19
Appendix I – Requirements engineering activities	23
Appendix II – Guidelines for the maintenance of URN	26
II.1 Maintenance of URN	26
II.2 Rules for maintenance	26
II.3 Change request procedure	27
Bibliography	29

Introduction

- a) Coverage: This Recommendation focuses on language requirements for URN and on providing the context for a requirements engineering framework. Other Recommendations in the ITU-T Z.150 series define the notation for URN.
- b) Applications: URN is applicable within standards bodies and industry. URN helps to describe and communicate requirements, and to develop reasoning about them. The main application areas include telecommunication systems and standards, 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 system design and architecture.
- c) Status/Stability: This Recommendation provides the scope and requirements for URN. The main text is accompanied by the following:
 - Appendix I: Requirements engineering activities.
 - Appendix II: Guidelines for the maintenance of URN.
 - Bibliography.

Recommendation ITU-T Z.150

User Requirements Notation (URN) – Language requirements and framework

1 Scope

This Recommendation provides motivation, scope and language requirements for the ITU-T User Requirements Notation. The specification of compliant notations belongs to other Recommendations.

The text of this clause is not normative.

1.1 Motivation

A notation is needed that can describe user requirements, goals and scenarios without any reference to specific inter-component communication facilities or system components and their states, but at the same time can capture the user requirements prior to design. The focus during the requirements specification stage is on behaviour and on quality attributes. The notation can also be used during the high-level design phase when activities or responsibilities specified in the scenarios are allocated to components. Scenario specification without sub-system component reference would facilitate reusability of scenarios across a wide range of architectures. The ability of the notation to straddle requirements specification and high-level design will facilitate negotiations between stakeholders and implementers.

Before URN was recommended, there was an increasing demand for non-static protocols with policy-driven negotiation using dynamic entities. Agent-based systems are examples of systems that require such policy-driven mechanisms. When specifying this kind of protocol, it is not possible to make an early commitment to messages and components at the requirements capture phase.

There is also the need for detection and avoidance of undesirable interactions between features or services. Older techniques require large investment in terms of messages and components that need to be checked for interactions. Using the notation specified in this Recommendation can provide insights at the requirements level and enable designers to reason about feature interactions early in the design process.

It is also important to deal with business objectives, goals, qualities, and non-functional requirements (NFRs) in a more systematic manner during requirements analysis and design. NFRs are requirements such as stringent performance constraints, systems operational costs, reliability, maintainability, portability, interoperability, robustness, and the like. In software development practice, many NFRs are stated only informally, making them difficult to analyse, specify and enforce during software development and to be validated by the user once the final system has been built. Goals and NFRs, however, do play a crucial role during system development, serving as selection criteria for choosing among alternatives during requirements analysis, for example, determining where the system boundaries should be and what functional requirements to include in the system.

Many of the alternative approaches to deal with NFRs originated from the technical work related to quality metrics. Such approaches attempt to quantify NFRs and then measure to what extent an existing system or parts of it meet the desired non-functional requirements. Useful metrics exist for NFRs such as performance, reliability, software complexity, and development process maturity. Other approaches, which recognize that many NFRs are often difficult, if not impossible, to quantify, use qualitative-oriented methods such as architectural change scenarios or combinations of both qualitative and quantitative methods to evaluate systems. These approaches, however, assume an already existing software system (or parts thereof) that is evaluated for its NFR properties. They do not assist in the specification of NFRs prior to building the system, nor do they provide support

during the analysis and design of systems. The notation proposed herein deals with NFRs and goals during the process of requirements analysis and system design; it allows for the expression of conflict between goals, of decisions that resolve conflicts and of the rationale for the trade-off decisions.

In addition, given that telecommunication systems are increasingly developed in the context of organization objectives and business processes, URN is also meant to be used to capture such objectives and processes. The unique combination of goals and scenarios found in URN enables not only to describe and analyse "what", "when", "who", and "where" aspects of business processes, but also "why" aspects and how they relate to business objectives.

The User Requirements Notation is defined to have the following capabilities:

- a) describe scenarios as first class entities without requiring reference to system subcomponents, specific inter-component communication facilities, or subcomponent states;
- b) capture user requirements when very little design detail is available;
- c) facilitate the transition from a requirements specification to a high level design involving the consideration of alternative architectures and the discovery of further requirements that must be vetted by the stakeholders;
- d) have dynamic refinement capability with the ability to allocate scenario responsibilities to architectural components;
- e) be applicable to the design of policy-driven negotiation protocols involving dynamic entities;
- f) facilitate detection and avoidance of undesirable interactions between features;
- g) provide insight at the requirements level that enables designers to reason about feature interactions and performance trade-offs early in the design process;
- h) provide facilities to express, analyse and deal with goals and non-functional requirements;
- i) provide facilities to express the relationship between goals and system requirements;
- j) provide facilities to capture reusable analysis and design knowledge related to know-how for addressing non-functional requirements;
- k) provide facilities to trace and transform requirements to other languages (especially ITU-T notations and UML);
- l) provide facilities to connect URN elements to external requirements objects;
- m) provide facilities to manage evolving requirements;
- n) provide facilities to measure the satisfaction of goals and qualities;
- o) provide facilities to interchange URN models;
- p) provide means of analysing goal and scenario models;
- q) provide lightweight mechanisms to extend or tailor the language to a specific domain.

The previous methods that used informal natural language for capturing requirements can leave too much open for interpretation and can contain invalid logic. Manual methods are used to validate these specifications with the result that defects are sometimes not caught until the implementation phase. Studies of software development have clearly shown that the earlier the defects are detected, the less costly they are to repair. Standardization of a notation supporting requirements engineering activities aims to make it easier to detect more defects at the requirements definition stage.

These same informal methods have also proven less than satisfactory for negotiating relative priorities of different business objectives and, in general, for managing trade-offs in the domain of non-functional requirements. The end result can be the delivery of a product to market that does not satisfy customers and does not meet business objectives. Standardization of a notation for requirements engineering activities aims to make it easier to define a product that balances stakeholder objectives and satisfies customer expectations.

Standardization of a formally defined notation used for capturing user requirements is a move to make the practice of this activity more rigorous and predictable and the results yielded by this activity clearer, more consistent, correct, and complete. These results should lead to a reduction of development costs, earlier delivery of product to market, and increased customer satisfaction.

1.2 Document organization

This Recommendation defines the language requirements for URN regarding both functional and non-functional aspects.

Clauses 2 to 4 cover background information on references, definitions, and abbreviations and acronyms.

Clause 5 details the scope of this Recommendation by providing an overview of the requirements engineering activity that uses URN, both from functional and non-functional perspectives.

Clauses 6, 7, and 8 provide detailed language requirements for URN-NFR, URN-FR, and other areas.

Clause 9 summarizes the list of language requirements that URN must fulfil.

Two appendices are defined:

- Appendix I introduces general activities in requirements engineering.
- Appendix II describes guidelines for the maintenance of URN.

A bibliography can be found at the end of this Recommendation.

2 References

None.

3 Definitions

This Recommendation defines the following terms:

3.1 additional information: The information stakeholders believe is important for designers and implementers to know but is not considered when validating a requirements specification.

3.2 behaviour: The sequence of actions with stimulus and responses aspects performed by a system that may change its state. See [b-ITU-T Z-Sup.1].

3.3 component: A generic and abstract entity that can represent software entities (e.g., objects, processes, databases, or servers) as well as non-software entities (e.g., actors or hardware).

3.4 dynamic refinement: A modelling mechanism that addresses issues related to structure and behaviour evolving at run time.

3.5 evaluation: The process used to determine the satisfiability (see clause 3.20) or satisficeability (see clause 3.21) of a solution with respect to its goal(s).

3.6 executability: An attribute of a model that can be interpreted or compiled and run. Executability applies mainly to functional models such as the one defined using URN-FR.

3.7 feature interaction: A desirable or undesirable interaction between two or more features, functionalities, services, policies, or scenarios. With respect to scenarios, a feature interaction is the set of conditions under which the execution of one scenario is affected by the execution of another. Undesirable feature interactions are also called conflicts.

3.8 functional requirement: A requirement (see clause 3.17) defining functions of the system under development.

3.9 goal: An objective or concern used to discover and evaluate functional and non-functional requirements.

3.10 high-level design: A design document describing system functionalities, the system architecture, and scenarios.

3.11 language requirement: A language requirement, as distinct from requirement (see clause 3.17), is a required characteristic of a language.

3.12 non-functional requirement: A requirement (see clause 3.17) characterizing a system property such as expected performance, robustness, usability, maintainability, etc. Non-functional requirements capture business goals/objectives and product quality attributes.

3.13 performance indicator: A performance indicator is a quantitative or qualitative value used to measure the performance or satisfaction of a quality attribute, a goal, or a system. A performance indicator relates a measured value with a satisfaction value.

3.14 postcondition: A postcondition expresses the condition following (successful) execution of a given operation or scenario. A postcondition normally expresses a relationship between the output variables in terms of the input variables. Where input variables may also be output variables, the relationship is defined in terms of initial system state and final system state.

3.15 precondition: A precondition expresses the conditions for which an operation or scenario is defined (i.e., if the conditions are not satisfied, then the result of the operation is not defined). A precondition normally expresses a relationship between input variables, or the system state prior to execution.

3.16 quality attribute: A non-functional requirement that relates to systems or products rather than to a business objective/goal.

3.17 requirement: A requirement, as distinct from a language requirement (see clause 3.11) and a user requirement (see clause 3.26), is an expression of ideas to be embodied in the system or application under development.

3.18 requirements engineering: The activity of development, elicitation, specification, and analysis of the stakeholder requirements, which are to be met by systems.

3.19 responsibility: A scenario activity representing something to be performed (operation, action, task, function, etc.). A responsibility can potentially be associated or allocated to a component.

3.20 satisfiability: The ability to determine whether a goal can be satisfied according to some strict criteria. For example, a performance goal may state that a system must generate a response to a certain event within a specified time, and it is possible to measure the time lapse between the occurrence of the event and the occurrence of the response.

3.21 satisficeability: The ability to determine whether a goal can be satisfied within acceptable limits. For "satisfied within acceptable limits" substitute "satisficed". For example, a security goal may state that a system cannot be accessed by unauthorized persons. A simple measure may ensure that most people will be unable to access the system. Ever more complex measures may ensure that a wider range of people will be unable to access the system. The law of diminishing returns applies here. It is impossible to guarantee that no unauthorized person can access the system.