Introduction to the User Requirements Notation (URN)

Gunter Mussbacher
ECE, McGill University, Canada ◄► gunter.mussbacher@mcgill.ca
Motivation

As this tutorial builds on a long history of other tutorials, a special acknowledgement is due to Daniel Amyot (University of Ottawa, Canada)
Table of Contents

• A Simple Problem: What is the “best way” for you to commute?

• Overview of the User Requirements Notation (URN)
  • History of the User Requirements Notation (URN)

• Analysis of the Simple Problem

• Overview of Analysis with the User Requirements Notation (URN)

• Key Performance Indicators (KPIs)

• Conclusion and References
A Simple Problem:
What is the “best way” for you to commute?
Problem Description

• Need to commute to work 230 days of the year

• Need to spend considerable amounts of time and money on the commute

• Various solutions for commuting are available
  • Various types of public transport
  • Car (own car or colleague’s car)

• What is the “best” solution for you to choose and why?
Example: Commuting

- **home**: secure home
- **transport**: commute
- **elevator**: take elevator

What (Responsibility) Who (Component)

- ready to leave home
- in cubicle

**Simple Problem**

**Overview of URN**

**Analysis of Simple Problem**

**Analysis with URN**

**Key Performance Indicators**

**Conclusion**
Hierarchical Structuring (1/3)

Example: Commuting

- **home**: secure home
- **transport**: commute
- **elevator**: take elevator

**Two Out-Paths**
- Multiple Choices (Dynamic Stub)
- Single Choice (Static Stub)

ready to leave home

stay home

in cubicle

Simple Problem | Overview of URN | Analysis of Simple Problem | Analysis with URN | Key Performance Indicators | Conclusion
Example: Car

Hierarchical Structuring (2/3)

commute
transport
drive car

Plug-in Map (plugged into stub)
Hierarchical Structuring (3/3)

- commute

Example: Hitch a Ride

- transport

hitch a ride in car
Alternatives and Concurrency (1/2)

Example: Regular Bus

- Concurrency (AND-fork)
- Alternatives (OR-fork)
- Merge (OR-join)
- Synchronization (AND-join)

Commute

Deal with work email

Transport

Take #95

Take #96

Take #97
Example: Express Bus

commuter

deal with
work email
	
transport

take #100

Concurrency (AND-fork)

Synchronization (AND-join)
Waiting Place / Timer

Example: Take Elevator

take elevator

elevator

call elevator

elevator arrived

select floor

take stairs

timer (special waiting place)

Timeout Path
Multiple Results

Example: Secure Home

secure home

home

stay home

arm system

lock door

use alternative alarm system

in

out1

out2

Multiple Out-Paths
Example: Arm System

- **Condition:**
  - **home**
  - **alarm system**
    - **not armed**
      - [quit]
      - **code**
      - **accept**
      - **check**
      - [matched]
      - [not matched]
    - **armed**
      - [matched]
      - [not matched]
Continuations between Hierarchical Levels

Example: Secure Home

home

arm system

lock door

in

out1

out2

use alternative alarm system

Example: Arm System

home

not armed

armed

[quit]

alarm system

accept code

check code

[matched]

[not matched]

Plug-in Bindings!
Basic Modeling of Reasons

Example: Commuting

Why (Goal)

Qualitative Contribution Type
(various positive and negative types exist as well as Unknown)

Solution (Task)

Minimize time lost by commute

Minimize cost for commute

Bus

Car

Contribution

+ +

− −

16 October 2013. Introduction to the User Requirements Notation (URN). © 2013 Gunter Mussbacher
Example: Minimize time lost by commute (refinement 1)

Decomposition

- Minimize time lost by commute
  - Work during commute
    - Take public transport (OR)
      - Regular Bus
      - Express Bus
    - Take private transport (OR)
      - Take own car
      - Hitch a ride
  - Minimize travel time
    - Regular Bus
      - Express Bus
      - Take own car
      - Hitch a ride

Decomposition (AND / OR / XOR)
Example: Minimize time lost by commute (refinement 2)
Example: Minimize cost for commute (refinement 1)
Stakeholders and Dependencies (1/2)

Example: Commuting

Stakeholder (Actor)

Dependency

Commuter

Provide public transport

Clear streets

City

Take private transport

OR

Take public transport

Colleague

Available to give a ride

Take own car

Hitch a ride
Example: Commuting

[Diagram showing stakeholders: Commuter, Colleague, City, with an arrow indicating a collapsed stakeholder (Actor).]
Example: Commuting

URN Link

Links

Simple Problem  Overview of URN  Analysis of Simple Problem  Analysis with URN  Key Performance Indicators  Conclusion
Overview of the User Requirements Notation (URN)
User Requirements Notation: History (1/2)

• In the 1990’s
  • Work on program slices, timethreads, and Use Case Maps
    • Buhr, Woodside, Vigder, Casselman, Amyot... (Carleton University/University of Ottawa)
  • Work on the Non-Functional Requirements (NFR) Framework
    • Chung, Mylopoulos, Nixon, Yu... (University of Toronto)
  • Work on the i* Framework
    • Yu, Mylopoulos... (University of Toronto)
  • Industrial research projects and standardization projects
    • Visser, Hodges, Monkewich... (Nortel)
    • Gray, Pinard, Mankovski... (Mitel)
• 1999
  • Nortel proposes to standardize Use Case Maps at ITU-T
  • Launching of UseCaseMaps.org
• 2000–2002
  • Mitel gets involved and suggests adding i* to UCMs
  • Canadian proposal for the User Requirements Notation
User Requirements Notation: History (2/2)

• 2000–2002 (cont’d)
  • Integration of subsets of i* and NFR into GRL
    • Yu and Liu (University of Toronto)
  • ITU-T Rapporteurs

• 2003: ITU-T Recommendation Z.150 (02/03)
  • User Requirements Notation (URN) – Language requirements and framework
    • Editor: D. Amyot

• 2005
  • First release of jUCMNav URN tool
  • Launching of the Wiki for the URN Virtual Library (www.usecasemaps.org/pub)

• 2008: ITU-T Recommendation Z.151 (11/08)
  • User Requirements Notation (URN) – Language definition
    • Co-editors: D. Amyot, G. Mussbacher

• 2012: ITU-T Corrigendum to Z.151 and Z.151 v2
  • Co-editors: D. Amyot, G. Mussbacher
**User Requirements Notation: Overview (1/2)**

- URN is a semi-formal, lightweight graphical language for modeling and analyzing requirements in the form of goals and scenarios.
- Formalizes and integrates two existing notations:
  - Goal-oriented Requirement Language (GRL)
  - Use Case Maps (UCMs)
- Support for the elicitation, analysis, specification, and validation of requirements.
- Allows systems/software/requirements engineers to discover and specify requirements for a proposed or an evolving system, and analyse such requirements for correctness and completeness.
- URN models can be used to specify and analyze various types of reactive systems, business processes and goals of organizations, and telecommunications standards.
A GRL / UCM model visually communicates business objectives and constraints / high-level functional requirements to all stakeholders.
URN Tool: jUCMNav – Juice Up Your Modeling!

- URN editor & analysis tool, Eclipse plugin, open source project
- GRL, Strategies, Evaluation
- UCMs, Scenario Definition & Execution, Test Suite
- Support for AoURN / CORE
- Support for BPM
- MSC Generation
- Export to DOORS / CSM

Pronounced: juicy – em – nav

http://jucmnav.softwareengineering.ca
ITU-T Z.151: URN – Language Definition

• URN is the first and currently only standard which explicitly addresses goals (non-functional requirements with GRL) in addition to scenarios (functional requirements with UCMs) in a graphical way in one unified language
  • International Telecommunication Union (ITU-T Z.150 series)
  • ITU-T Z.150 (02/03):
    User Requirements Notation (URN) - Language requirements and framework
  • ITU-T Z.151 (11/08):
    User requirements notation (URN) - Language definition

• Part of the ITU family of languages: SDL, MSC, TTCN-3, ASN.1…

• Definition of URN in Recommendation Z.151 (approved November 2008)
  • Modeling elements of notation and their meaning, analysis capabilities, interchange format

Why Use Case Maps?

• **Bridge** the **modeling gap** between abstract and informal descriptions useful at early stages of system development and more formal and concrete descriptions useful at later stages of system development.

• Use Case Maps **integrate many scenarios** and allow reasoning about potential undesirable interactions.

• Provide **ability to model** dynamic systems where scenarios and structures may change at run-time:
  - E-business applications, Web services, business processes and workflows
  - Distributed systems based on agents, reactive systems

• **Effective learning tool** for people unfamiliar with the domain.

• May be **transformed** (e.g., into MSC/sequence diagrams, performance models, test cases).
Use Case Maps: Summary

• **Model scenario concepts**
  • Mainly for operational requirements, functional requirements, and business processes
  • For reasoning about scenario interactions, performance, and architecture

• **Use Case Maps provide …**
  • Visual description of behavior **superimposed** over entities (from stakeholders and users to software architecture to hardware)
  • Easy graphical manipulation of scenario descriptions
  • Single scenario view
  • **Combined system view**
  • Enhanced consistency and completeness
  • Connections to goal models
  • Smooth transition to design models (e.g., message sequence charts)
  • Connections to performance models and testing models
Why the Goal-oriented Requirement Language?

• Goals become an **important driver** for requirements elaboration. Yet, stakeholders goals and objectives are complex and will conflict…

• **GRL expresses and clarifies** tentative, ill-defined, and ambiguous requirements
  • Supports argumentation, negotiation, conflict detection & resolution, and in general decisions
  • Captures decision rationale and criteria (documentation!)

• **GRL identifies alternative** requirements and alternative system boundaries

• **GRL provides clear traceability** from strategic objectives to technical requirements

• **GRL allows reuse** of stable higher-level goals when the system evolves

• **Nothing like this in UML or BPMN or in other standard languages…**
The Goal-oriented Requirement Language is based on …
- i* (concepts / syntax)
- NFR Framework (evaluation mechanism)

Model goals and other intentional concepts – mainly for non-functional requirements, quality attributes, rationale documentation, and reasoning about alternatives and tradeoffs

The Goal-oriented Requirement Language is used to …
- Visually describe business goals, stakeholders’ priorities, alternative solutions, rationale, and decisions
- Decompose high-level goals into alternative solutions called tasks (this process is called operationalization)
- Model positive and negative influences of goals and tasks on each other
- Capture dependencies between actors (i.e., stakeholders)
Modeling Requirements with URN

• For requirements modeling, we need to answer the W5 questions
  • *Where, What, Who, When, and Why*

• **Goal-oriented Requirement Language (GRL)**
  • Business or system goals and rationales (*Why*)
  • Solutions/Tasks (*What*)
  • Stakeholders/Actors (*Who and Where*)

• **Use Case Maps (UCMs)**
  • Responsibilities (*What*)
  • Components (*Who and Where*)
  • Scenarios and causal sequences (*When*)

• **GRL & UCMs**
  • Link processes to business goals with URN links (▸), for traceability, completeness, alignment, compliance, what if scenarios, and evolution
Analysis of the Simple Problem
Scenario Definition “my own car, home armed, elevator”:
ready to leave home; matched; Car; elevator arrived → armed; in cubicle

Example: Commuting
**Scenario Definition** “regular bus #97, alternative alarm, stairs”:
ready to leave home; not matched; quit; alternative alarm; Regular Bus; #97; elevator does not arrive → not armed; in cubicle

**Example: Commuting**
Scenario Definition “stay home”:
ready to leave home; not matched; quit; stay home → not armed; stay home

Example: Commuting
Strategy “Regular Bus”:

Regular Bus = 100

Initial Satisfaction Level
(indicated by * and dashed outline)

Example: Commuting

Strategy Execution (1/7)
Strategy Execution (2/7)

Strategy “Express Bus”:
Express Bus = 100

Example: Commuting

- Minimize time lost by commute
- Minimize travel time
- Minimize cost for commute
- Minimize infrastructure cost
- Share ongoing cost

Strategy "Express Bus":
Express Bus = 100

OR

Regular Bus
Express Bus
Take own car
Hitch a ride

OR

Regular Bus
Express Bus
Take own car
Hitch a ride

Commuter

Simple Problem
Overview of URN
Analysis of Simple Problem
Analysis with URN
Key Performance Indicators
Conclusion

16 October 2013. Introduction to the User Requirements Notation (URN). © 2013 Gunter Mussbacher
Strategy Execution (3/7)

Strategy “Take own car”:
Take own car = 100

Example: Commuting
Strategy "Hitch a ride":
Hitch a ride = 100

Example: Commuting
Example: Commuting

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Importance</th>
<th>Minimize time lost by commute</th>
<th>Minimize cost for commute</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Regular Bus”</td>
<td>[0, 100]</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>“Express Bus”</td>
<td>[0, 100]</td>
<td>72</td>
<td>7</td>
</tr>
<tr>
<td>“Take own car”</td>
<td>[0, 100]</td>
<td>-62</td>
<td>-2</td>
</tr>
<tr>
<td>“Hitch a ride”</td>
<td>[0, 100]</td>
<td>-4</td>
<td>0</td>
</tr>
</tbody>
</table>

Rank 1: “Regular Bus”
Rank 2: “Express Bus”
Rank 3: “Hitch a ride”
Rank 4: “Take own car”
### Example: Commuting

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Minimize cost for commute (20)</th>
<th>Minimize time lost by commute (100)</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Regular Bus&quot;</td>
<td>20</td>
<td>80</td>
<td>1</td>
</tr>
<tr>
<td>&quot;Express Bus&quot;</td>
<td>7</td>
<td>72</td>
<td>4</td>
</tr>
<tr>
<td>&quot;Take own car&quot;</td>
<td>35</td>
<td>-62</td>
<td>3</td>
</tr>
<tr>
<td>&quot;Hitch a ride&quot;</td>
<td>35</td>
<td>-4</td>
<td>2</td>
</tr>
</tbody>
</table>

**Simple Problem** - Overview of URN - Analysis of Simple Problem - Analysis with URN - Key Performance Indicators - Conclusion
Example: Commuting

Minimize time lost by commute (15)
Minimize cost for commute (85)

Strategy “Regular Bus”: 20 80  
Strategy “Express Bus”: 7 72  
Strategy “Take own car”: 35 -62  
Strategy “Hitch a ride”: 35 -4
Integration of Goal and Scenario Models

Example: Commuting

- Choices in the goal model (i.e., the chosen strategy) may influence the scenario model
- Scenario model may also influence the evaluation of the goal model
- Full feedback loop
Overview of Analysis with the User Requirements Notation (URN)
Scenario Traversal Mechanism

• A scenario describes one path through the model (only one alternative at any choice point is taken)
  • Set of initial values for the variables used in conditions and responsibilities
  • Start points triggered, end points reached
  • Possibly pre/post conditions
• A traversal mechanism interprets the model given scenario description(s)
  • Requires the use of the data model in choice points (forks, dynamic stubs, timers, conditions) and responsibilities
• Extraction of individual scenarios (highlight, transformations)
  • Learning tool – allows focus on key scenarios
• Groups of scenarios can be run together (i.e., a test suite for regression testing)
• The goal model allows a particular configuration of intentional elements to be defined in a strategy (i.e., one possible solution)
  • Captures the initial, user-defined satisfaction levels for these elements
  • Strategies can be compared with each other for trade-off analyses
• In order to analyze the goal model and compare solutions with each other, a customizable evaluation mechanism executes the strategies
  • Propagating levels to the other elements and to stakeholders shows impact of proposed solution on high level goals for each stakeholder
  • Propagation starts at user-defined satisfaction levels of intentional elements (usually bottom-up)
  • Takes into consideration
    • Initial satisfaction levels of intentional elements
    • Links and contribution types
    • Importance defined for intentional elements
  • Qualitative or quantitative interpretation
• **Bottom-up** analysis
  • Typically propagates satisfaction values of low-level tasks (i.e., selected solutions) to those of high-level stakeholder goals

• **Top-down** analysis
  • Searches for the optimal result taking the structure of the goal model and the relationships between nodes in the goal model into account
  • Can be formulated as a planning problem
  • Is akin to a constraint solving approach
• **Quantitative Approach**
  • Contribution types: [-100, 100]
  • Importance: [0, 100]
  • Quantitative satisfaction levels: [-100, 100]

• **Qualitative Approach**
  • Contribution types: from Make to Break
  • Importance: High, Medium, Low, or None
  • Qualitative satisfaction levels

• **Mixed (Hybrid) Approach also possible**
  • Qualitative contribution types
  • Quantitative importance
  • Quantitative satisfaction levels
• Contributions can be shown for any element at any hierarchical level

• Using contributions at several levels has implications in terms of the evaluation mechanism
  • Not necessarily forbidden, but one has to be aware of the ramifications

• May be annoying while the goal model is being built
  • Start with high-level contributions
  • Refine → May need to add contributions at lower level and therefore may have to remove contributions at the higher level
  • Or vice versa
Key Performance Indicators (KPIs)
Goals are not enough... Need a way to measure functional and non-functional properties in terms of the domain units.
Evaluations Involving Key Performance Indicators

Key Performance Indicator
Key Performance Indicators

Example: Commuting

- Minimize time lost by commute
- Minimize cost for commute
- Minimize travel time
- Minimize infrastructure cost
- Share ongoing cost

- Average Work Time (in min)
- Monthly Infrastructure Cost (in $)
- Average Ongoing Cost (in $)

- Take public transport
- Take private transport
- Take own car
- Hitch a ride

- Commuter
- Work during commute
- Average Travel Time (in min)

Key Performance Indicator (KPI)
From Real World Values to Model Values (1/2)

Target Value (20), Threshold Value (40), Worst Value (80)

<table>
<thead>
<tr>
<th>Real World Value</th>
<th>Model Value (Satisfaction Value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Value</td>
<td>100</td>
</tr>
<tr>
<td>Threshold Value</td>
<td>0</td>
</tr>
<tr>
<td>Worst Value</td>
<td>-100 &lt; -100</td>
</tr>
</tbody>
</table>

Satisfaction Value

Real World Value (56)  Satisfaction Value (-40)
### From Real World Values to Model Values (1/2)

<table>
<thead>
<tr>
<th>Real World Value</th>
<th>Model Value (Satisfaction Value)</th>
<th>Average Work Time (in min)</th>
<th>Average Travel Time (in min)</th>
<th>Monthly Infrastructure Cost (in $)</th>
<th>Average Ongoing Cost (in $)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Value</td>
<td>100</td>
<td>49</td>
<td>56</td>
<td>10</td>
<td>68</td>
</tr>
<tr>
<td>Threshold Value</td>
<td>0</td>
<td>80</td>
<td>-40</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Worst Value</td>
<td>-100</td>
<td>-100</td>
<td>-30</td>
<td>-20</td>
<td>20</td>
</tr>
</tbody>
</table>

#### Real World Values
- Target Value: 60
- Threshold Value: 5
- Worst Value: 0

#### Model Values
- Target Value: 100
- Threshold Value: 5
- Worst Value: 0

#### Analysis
- Regular Bus: 49 (100), 29.75 (70), 4.5 (30), 4.5 (30)
- Express Bus: 56 (70), 52 (60), 24 (50), 24 (50)
- Take own car: 80 (60), 45 (50), -10 (40), -10 (40)
- Hitch a ride: 10 (40), 10 (30), 455 (300), 140 (300)
- Take own car: 68 (60), 76 (50), 120 (40), 92 (40)
- Hitch a ride: 80 (60), 60 (50), -20 (40), 20 (40)
Strategy “Regular Bus”:

Regular Bus = 100

KPI “Regular Bus”:

Av. Work Time = 49 → 80
Av. Travel Time = 56 → -40
Mo. Infrast. Cost = 10 → 80
Av. Ongoing Cost = 68 → 80

Example: Commuting

Minimize time lost by commute (100)
Minimize cost for commute (50)
Minimize infrastructure cost
Share ongoing cost
Strategy Execution with KPIs (2/2)

Strategy “Hitch a ride”:
Hitch a ride = 100

Example: Commuting

**KPI “Hitch a ride”:**
- Av. Work Time = 4.5 → -10
- Av. Travel Time = 24 → 80
- Mo. Infrast. Cost = 140 → -20
- Av. Ongoing Cost = 92 → 20
Conclusion and References
Conclusion

- The User Requirements Notation (URN) is an ITU-T standard
- URN is a competitive notation for requirements engineering activities including business process modeling and analysis
- Modeling with the Goal-oriented Requirement Language (GRL)
  - Focuses on answering “why” questions
  - Intentions, business goals, functional / non-functional requirements, rationales
  - Key Performance Indicators (KPIs) are a must in a business environment to measure and monitor processes, compliances, and non-functional properties
- Modeling with Use Case Maps (UCMs)
  - Focuses on answering “what” and “when” questions
  - Scenarios, business processes, services, architectures
- Enables the elicitation/specification of systems, business processes and goals, standards, and products, as well as their analysis/validation from various angles
- Tool support
More Advanced URN Modeling Concepts

• Both GRL and UCMs have more advanced modeling concepts not covered in this course

• URN
  • Data model in support of GRL evaluations and UCM scenario definitions
  • Extensibility through profiling mechanism (metadata, URN links, and OCL)

• GRL
  • More types of intentional elements and links
  • More evaluation algorithms
  • KPI conversion and aggregation

• UCMs
  • More types of components
  • More advanced ways of binding elements on plug-in maps to stubs
  • More types of path elements that allow various workflow patterns to be modeled concisely
  • Extensions for failure and exception handling
References

**General**

URN Virtual Library (~350 entries), http://www.UseCaseMaps.org/pub/

URN tool: jUCMNav, University of Ottawa, http://jucmnav.softwareengineering.ca/jucmnav/


URN website: http://www.usecasemaps.org/urn

**Overview of URN**


Appendix: Notation Overview – GRL

(a) GRL Elements
- Goal
- Softgoal
- Task
- Resource
- Belief
- Actor with Boundary
- Collapsed Actor

(b) GRL Links
- Contribution
- Correlation
- Dependency
- Decomposition

(c) GRL Satisfaction Levels
- Denied
- Weakly Denied
- Weakly Satisfied
- Satisfied
- Conflict
- Unknown
- None

(d) GRL Contributions Types
- Make
- Some Positive
- Help
- Unknown
- Hurt
- Some Negative
- Break

(e) Representations of Qualitative and Quantitative Contributions
- i) Icon only
- ii) Text only
- iii) Icon and text
- iv) Number only
- v) Icon and number

(f) Performance Modeling
- KPI
- Dimension
Appendix: Notation Overview – UCMs (Behavior)

- Path with Start Point with Precondition CS and End Point with Postcondition CE
- Responsibility
- Empty Point
- Direction Arrow
- Waiting Place with Condition and Asynchronous Trigger
- Or-Fork with Conditions
- Or-Join
- And-Fork
- And-Join
- Timer with Timeout Path, Conditions, and Synchronous Release
- Synchronizing Stub with In-Path ID, Out-Path ID, and Synchronization Threshold
- Blocking Stub with In-Path ID, Out-Path ID, Synchronization Threshold, and Replication Indicator
Appendix: Notation Overview – UCMs (Structure)

Components:

- Team
- Process
- Object
- Agent
- Actor
- Protected Component
- Context-dependent Component

**parent:**