Visualizing Evolving Requirements Models with TimedURN

Sahil Luthra
ECE, McGill University, Canada,
sahil.luthra@mail.mcgill.ca

Aprajita
ECE, McGill University, Canada,
aprajita.aprajita@mail.mcgill.ca

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

Abstract

Many modern systems are built to be in use for a long time, sometimes spanning several decades. Furthermore, considering the cost of replacing an existing system, existing systems are usually adapted to evolving requirements, some of which may be anticipated. Such systems need to be specified and analyzed in terms of whether the changes introduced into the system still address evolving requirements and continue to satisfy the needs of its stakeholders. Recently, the User Requirements Notation (URN) – a requirements engineering notation standardized by the International Telecommunication Union for the elicitation, specification, and analysis of integrated goal and scenario models – has been extended with support for the modeling and analysis of evolving requirements models by capturing a comprehensive set of changes to a URN model as first-class model concepts in URN. The extension is called TimedURN and this paper builds on TimedURN to analyze and visualize not just an individual time point in the past, present, or future but a time range consisting of a set of time points. Various visualization options are discussed, including their proof-of-concept implementation in the jUCMNav requirements engineering tool.

CCS Concept

• **Software and its engineering~Model-driven software engineering** • **Software and its engineering~Specification languages** • **Software and its engineering~Requirements analysis**

Keywords

goals, scenarios, evaluation mechanism, traversal mechanism, URN, User Requirements Notation, evolution, visualization

ACM Reference format:

Sahil Luthra, Aprajita, and Gunter Mussbacher. 2018. Visualizing Evolving Requirements Models with TimedURN. In *Proceedings of 10th Workshop on Modelling in Software Engineering (MiSE 2018). ACM, New York, NY, USA, 8 pages.* https://doi.org/10.1145/3193954.3193959

MiSE'18, May 27, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5735-7/18/05…$15.00
https://doi.org/10.1145/3193954.3193959

1 Introduction

Large systems are typically in use for several years or even decades. All systems evolve as new requirements arise or the needs of stakeholders, desired system qualities, and potential solutions change. Furthermore, new functionality may already be known but not yet implemented due to resource and scheduling constraints. Requirements modeling techniques, however, often focus on a single point in time, i.e., a snapshot, when specifying or analyzing a system. This is not sufficient for systems that require a problem to be analyzed over a longer period of time. For example, systems in the sustainability domain are long-lived, which increases the likelihood that the users' attitude towards the system may change and that new disruptive technology may appear leading to new applicable solutions.

The User Requirements Notation (URN) [1] is a requirements engineering standard published by the International Telecommunication Union (ITU). URN formalizes and integrates two complementary notations: the Goal-oriented Requirement Language (GRL) for goal modeling and Use Case Maps (UCMs) for scenario modeling [18] to elicit, specify, and analyze user and system requirements early in the development phase rather than later. URN goal models capture system qualities, requirements, stakeholder objectives, and relationships among those elements. URN goal model analysis (called evaluation) allows comparing alternative solutions to balance required system qualities and stakeholder needs. URN scenario models provide a more detailed description of these alternative solutions in terms of causal relationships among scenario steps, and combine scenarios into a high-level system view where behavior is superimposed over structural elements. URN scenarios analysis uses a path traversal mechanism. Upon model update, the traversal mechanism checks whether pre- and post-conditions of a scenario hold, hence establishing a test suite at the requirements level.

TimedURN [4][5] extends the User Requirements Notation (URN) with the ability to (i) specify changes to a model without having to duplicate the model, hence simplifying model management, and (ii) analyzing the model at any time point in the past, present, or future instead of just the current snapshot in time. TimedURN improves URN's modeling capabilities for evolving systems in combination with URN's existing capabilities to model dynamic systems with changing scenarios and structures at runtime [18] and to dynamically choose the most appropriate solution depending on contextual changes [23].

This paper builds on previous work on URN [1] and TimedURN [4][5] by analyzing and visualizing not just an

individual time point in the past, present, or future but a time range consisting of a set of time points. The evaluation of the whole system is introduced in addition to the existing evaluation of goals and stakeholders. Various visualization options for the results of goal model evaluation and scenario model traversal are discussed, including their proof-of-concept implementation in the jUCMNav requirements engineering tool [19].

The remainder of this paper gives an overview of URN and TimedURN and their analysis techniques in Section 2. Section 3 introduces URN system evaluation and describes various ways of analyzing a time range, and then discusses several visualization options, including those implemented in the jUCMNav tool as a proof of concept. Section 4 summarizes related work, while Section 5 concludes the paper and presents future work.

2 Overview of URN and TimedURN

The International Telecommunication Union publishes the User Requirements Notation (URN) [1] standard consisting of GRL (Goal-oriented Requirement Language) and UCM (Use Case Maps) in the Z.15x series [18]. The goal modeling notation GRL is based on i* [24] and the NFR Framework [8], while the scenario notation UCM is similar to UML activity diagrams.

A GRL *actor* (, e.g., "City" in Figure 1) represents a stakeholder of a system or the system. *Intentional elements* (softgoal, goal, task, resource) form a connected graph that may reside within an actor. *Softgoals* () and *goals* (, e.g., "Encourage usage of Renewable Energy") define high-level system objectives and qualities that are significant to a stakeholder. *Tasks* (, e.g., "Construction Phase") specify solutions considered to achieve objectives and qualities. *Resources* (, e.g., "Research Facility") may be required to achieve or complete softgoals, goals, and tasks.

Links connect softgoals, goals, tasks, and resources with each other. AND, XOR, or IOR *decomposition links* (, e.g., between "Minimize Cost of Renewable Resources" and "Encourage usage of Renewable Energy") decompose an element into sub-elements. A *dependency link* (, e.g., between "Discover Renewable Controlled Nuclear Fusion Reactor" and "Research Facility") specifies how an element depends on another element. A *contribution link* (→, e.g., between "Build an Advanced Research Facility" and "Research Facility") captures the impact of one element on another element, either with qualitative labels (+ or −), quantitative integer value between −100 and 100, or relative contribution values [9][10][11].

GRL analyzes stakeholder objectives and system qualities with propagation-based *evaluation mechanisms* [2]. A GRL *evaluation strategy* describes a possible solution that can be compared with other strategies to find the most appropriate trade-offs among often conflicting stakeholder goals. Typically, a strategy initializes a set of intentional elements at the lowest level in the goal graph with qualitative or quantitative *satisfaction values* (e.g., 20(*) for "Increase Efficiency of Solar Power"; (*) and the dashed outline indicate that this value is an initial value), but any other intentional element may also be initialized. The evaluation mechanism then propagates these initial satisfaction values to higher-level goal model elements (e.g., 57 for "Encourage usage of Renewable Energy"). The satisfaction values are color-coded from Red (-100) to Yellow (0) and Green (100). The URN standard describes non-normative examples of evaluation algorithms [2], all of which have been implemented in the jUCMNav tool [19]. In addition, the jUCMNav tool supports the visualization of trends given a set of strategies [3]. GRL models have also been evaluated in a top-down fashion [10][22] and with relative contribution values, which requires normalization [9][10][11].

UCM specifies a scenario at a more abstract level above messages and data flow, focusing on the causal relationship among steps in a scenario, optionally superimposing steps over entities. A UCM *map* represents a scenario and consists of *components* and *paths*. Structural aspects of a system are specified using *components* (). Path elements inside a component are said to be bound to it. Someone or something interacting with the system is represented by a component of kind *actor* (, e.g., "Hiring Committee" in Figure 2).

Paths depict causal sequences and may contain several types of path nodes. *Paths* start at *start points* (, e.g., "Start to build") with optional pre-conditions and end at *end points* (, e.g., "Research Facility Ready") with optional post-conditions. A *responsibility* (, e.g., "Hire Staff") represents a step in a scenario. Alternatives are shown using *OR-forks* (), including guarding conditions, and *OR-joins* (), while concurrency is visualized through *AND-forks* () and *AND-joins* (). UCM allows *OR-joins* and *OR-forks* to be freely combined to model well-nested control-flow structures including loops as well as non-well-nested control-flow structures. *Waiting places* () and *timers* () indicate locations on the path where the scenario stops until a condition is satisfied. *Stubs* hierarchically decompose UCM models. They contain sub-maps called *plug-in maps*, which are reusable units of behavior and structure. Path continuation is ensured by defining *plug-in bindings* to connect in-paths and out-paths of a stub with start and end points of its *plug-in maps*, respectively. *Static stubs* () can have at most one *plug-in map*, while *dynamic stubs* (, e.g., "Secondary Evaluation") can have many parallel *plug-in maps*.

UCM analysis operates on the basis of a *scenario definition*, which defines a sequence of start points to be triggered, a set of end points to be reached, and pre/post-conditions to be checked. A set of scenario definitions defines a regression test suite for the scenario model. A single scenario definition describes a specific path through the scenario model where only one alternative at any choice point is taken, as specified by a *data model* that formalizes the conditions at each choice point (e.g., dynamic stubs and OR-forks). Responsibilities may specify code that modifies the values of the variables in this data model. A scenario definition initializes these variables. Given a scenario definition, a *path traversal mechanism* highlights the scenario path through the UCM model [20] (e.g., Figure 2 highlights traversed path elements in red whereas non-traversed path elements are shown in black). URN defines the operational semantics of the UCM language with the traversal mechanism [18], which is implemented in jUCMNav [19].

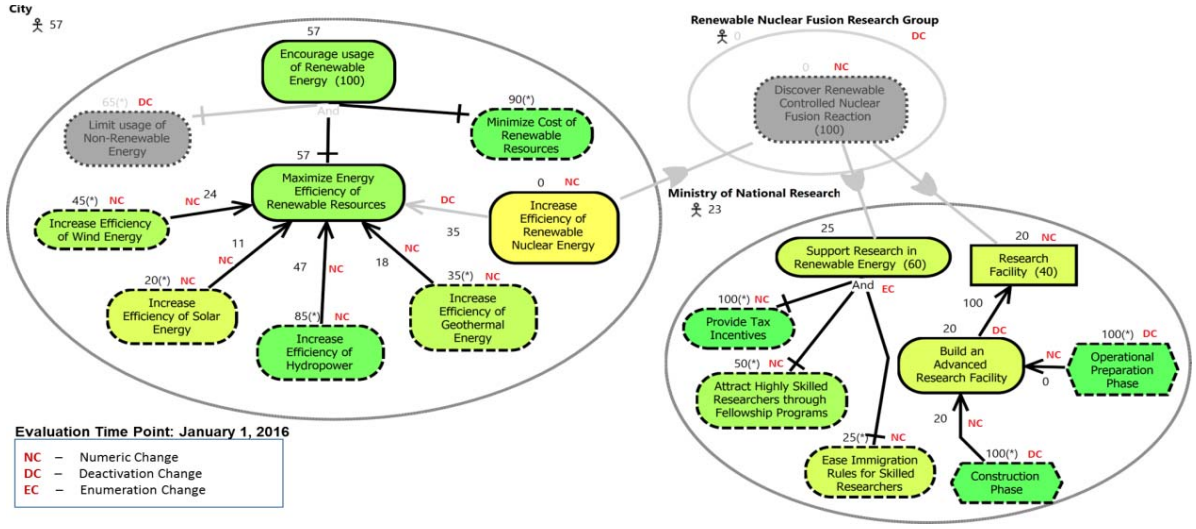

Figure 1: Evaluation of Goal Model for Energy Efficiency Goals of a City in 2016

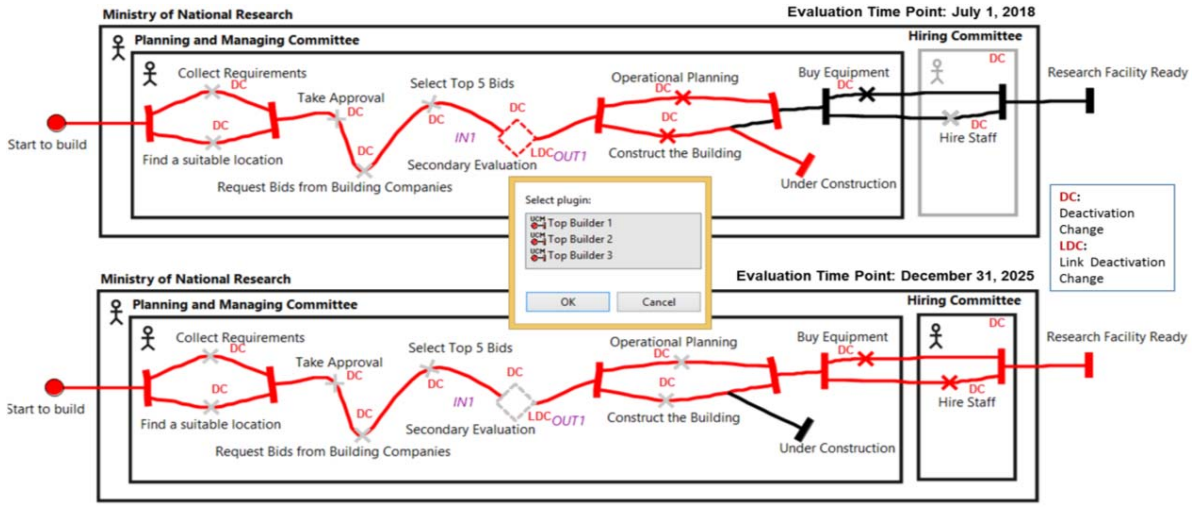

Figure 2: Traversal of UCM Model Related to "Build an Advanced Research Facility" Goal in 2018 and 2025

URN supports a full feedback loop, where satisfaction values of goal model elements may be used in conditions of UCM choice points (hence influencing scenario traversal through a UCM model) and where responsibilities may update satisfaction values (hence influencing goal model evaluation). Furthermore, *URN links* establish relationships among URN model elements, most notably between actors and components as well as tasks and path elements.

TimedURN [4][5] extends the URN standard by incorporating the concept of changes over time into the URN metamodel, providing an algorithm for the analysis of evolving goal and scenario models, and visualizing the changed URN model at a specific time point in the past, present, or future (all supported by the jUCMNav tool [19]). TimedURN captures all changes in one model. Hence, TimedURN avoids management and consistency difficulties related to the space-consuming, time-consuming, and error-prone tasks of maintaining several, slightly different model copies representing different time points.

Figures 1 and 2 depict the running URN example to illustrate various changes and their visualization. A city wants to encourage usage of renewable energy by maximizing its overall renewable energy efficiency considering the efficiencies of various energy sources while minimizing the cost of renewable energy and limiting usage of non-renewable energy. Figure 1 shows the evaluation of the goal model at the January 1, 2016, time point. The scenario model in Figure 2 provides further details for the goal "Build an Advanced Research Facility", showing the Planning and Managing Committee responsible for facilitating, planning, and overseeing the construction and

operational preparation of the Research Facility as well as the Hiring Committee responsible for hiring staff for the facility. The top and bottom of Figure 2 show the evaluations of the scenario model at the July 1, 2018, and December 31, 2025, time points, respectively.

A *Change* is the behavior expected in a URN model element over time. Each change has a *start date* and an *end date*, which specify the time period when that particular change is applicable to the element. A change may either affect a model element, its relationship with another element, or its property.

A *Deactivation Change* (marked as DC in Figure 1 and 2) is applied to the model element itself to define its actual existence timeline in the model. For example, the goal "Limit usage of Non-Renewable Energy" in Figure 1 is deactivated, because this is not something that the city wants to enforce now but in the future starting 2029. According to the estimates of the Ministry, the scenario elements in Figure 2 up to taking approval are deactivated in 2017 (when this milestone is expected to be reached), and the subsequent scenario elements up to the first AND-join are gradually deactivated until 2024. The Hiring Committee is initially deactivated and becomes active in 2025 to achieve the goal of having the facility ready for research by 2026. A deactivated element (and any contained elements, if applicable) is grayed out.

A *Link Deactivation Change* (marked as LDC in Figure 2) is similar to a *Deactivation Change* but operates on relationships among model elements. For example, the dynamic stub for the secondary evaluation of building companies in Figure 2 has such a change, which enables or disables its plug-in maps, each representing a building company (i.e., the relationship between the stub and a plug-in map is activated or deactivated). The change allows modeling which building companies are still considered for the project at various points in time.

The remaining type of change is a *Property Change* which is applied to the property of an element. Various subtypes exist depending on the type of the property. A *Numeric Change* (marked as NC in Figure 1) covers changes to numbers specified with arbitrary mathematical formula expressed in time (e.g., satisfaction and contribution values). An *Enumeration Change* (EC in Figure 1) specifies a change in an enumeration value (e.g., the decomposition type of "Support Research in Renewable Energy"). A *Text Change* may change the code of a responsibility or condition of an OR-fork, while a *Boolean Change* may change a static stub into a dynamic stub and vice versa.

The three key types of changes (*Deactivation Change*, *Link Deactivation Change*, and *Property Change*) enable the full manipulation of runtime instances of a URN model. Consider the modeling constructs of a UML object diagram used to visualize a particular set of runtime instances of a URN model. An object diagram consists of objects (can be added/removed with *Deactivation Change*), attribute values of objects (can be changed with *Property Change*), and links among objects (can be added/removed with *Link Deactivation Change*).

Last but not least, a *Dynamic Context* groups a GRL strategy, a UCM scenario definition, and a set of changes that are to be evaluated together. TimedURN builds on the already existing quantitative evaluation algorithm [2] and the existing path traversal mechanism [20] by adding a preprocessing phase that determines any applicable changes for a chosen *Time Point* and then updates the model accordingly. The resulting model from the preprocessing phase is analyzed by the existing evaluation algorithm and path traversal mechanism, which have been augmented with the capability to ignore deactivated elements in the goal model and scenario model. The existing visualization of the analysis result is reused but deactivated elements are grayed out. After analysis, the model is returned to its original state.

3 Visualization of TimedURN

This section first explains how the whole system described by the URN model is evaluated and introduces the analysis algorithm for a time range consisting of several time points, and then discusses the visualization of the analysis result of the time range.

3.1 System Analysis over Time Range

Based on initial satisfaction values defined by a strategy, the existing evaluation of a URN goal model determines satisfaction values for each intentional element as well as satisfaction values for each actor (e.g., 57 for City in Figure 1) based on the satisfaction values (e.g., 57 for "Encourage usage of Renewable Energy") and importance values of its high level goals (e.g., 100 shown in parentheses for the same goal). However, the concept of a comprehensive system satisfaction value (or system evaluation) does not exist in the URN standard. For this purpose, actor importance is introduced into the URN metamodel, allowing a modeler to also specify the relative importance of actors to the overall system. Similar to the evaluation of an actor, TimedURN computes the system evaluation using the following formula:

$$System\ Satisfaction = \frac{\sum_{actors}(actor\ satisfaction * actor\ importance)}{\sum_{actors}(actor\ importance)}$$

If importance is not specified for any actor in the URN model, system satisfaction is not computed, i.e., it is undefined. The importance of an actor is shown in parentheses next to the actor name (see 100 for City in Figure 3) similar to importance values of goals.

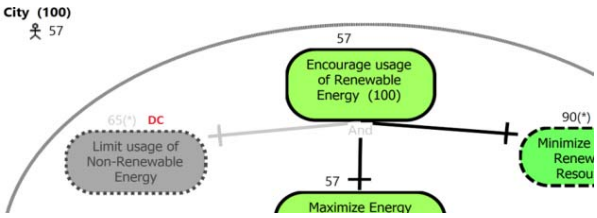

Figure 3: Actor Importance for System Evaluation

While evaluating a URN model at a single time point, the system evaluation is currently not visualized in the model. However, when a time range is evaluated, the system evaluation is visualized. The evaluation of a time range corresponds to the evaluation of a *Time Point Group*, which consists of a set of time

points. A time point group is evaluated either based on a user-defined frequency or based on explicit time points. The user-defined frequency defaults to one day (i.e., the URN model is evaluated for each day between the earliest time point and the latest time point in the time point group), but may be increased to any desired interval smaller than the whole time range to be evaluated. The explicit time points are the time points in a time point group (i.e., the URN model is evaluated for each time point in the time point group and no other time point).

In either case, each required time point is evaluated one after the other and the results stored for later processing. The results of a goal model evaluation at a specific time point are the satisfaction values of the intentional elements, actors, and the whole system. Intermediate results such as the importance values of intentional elements and actors, and the impact of contributions, decompositions, and dependencies on target elements are also captured for more detailed analysis. The results of a scenario model evaluation at a specific time point are the number of times a scenario element is traversed during the analysis (i.e., the *hit count*). The specifications of code in responsibilities as well as conditions may also be used for more detailed analysis.

The aforementioned results for each time point could easily be exported from jUCMNav to a csv file for statistical analysis by external tools. However, this is not the focus here. Similarly, the code of responsibilities and conditions of choice points could be easily exported to text files, which could then be analyzed by generic diff tools most appropriate for comparing textual specifications. Again, this is not the focus of this paper. Instead, the paper explores in the next subsection how to visualize the analysis results over a time range within the URN model as much as possible.

3.2 Visualization of Analysis Result over Time Range

The purpose of the visualization of the analysis result of a time range is to convey an overview of the result to the modeler. The result of each individual time point is already visualized by the existing implementation in the jUCMNav tool. Hence, a modeler can always explore the analysis result of each time point in greater detail.

For a goal model, an overview should show the overall progression of satisfaction values in the model at a more detailed level than the existing trend indicators [3] with icons only for constant satisfaction values, steadily increasing or decreasing satisfaction values, or arbitrarily changing satisfaction values. Furthermore, the modeler is already used to the standard color coding of satisfaction values from Red (-100) to Yellow (0) and Green (100). Therefore, the analysis result for a time range including the system evaluation is shown in the form of a heatmap (see Figure 4). The rows of the heatmap represent the containment structure of the goal model, starting with the whole system at the top, followed by its actors and the intentional elements contained in each actor. Each column corresponds to the evaluation of one time point. In the example in Figure 4, a frequency-based evaluation of one day is chosen for a time point group with three time points stretching over one month. The three time points are highlighted at the top. For each date and model element, the satisfaction value of the model element is shown following standard color coding.

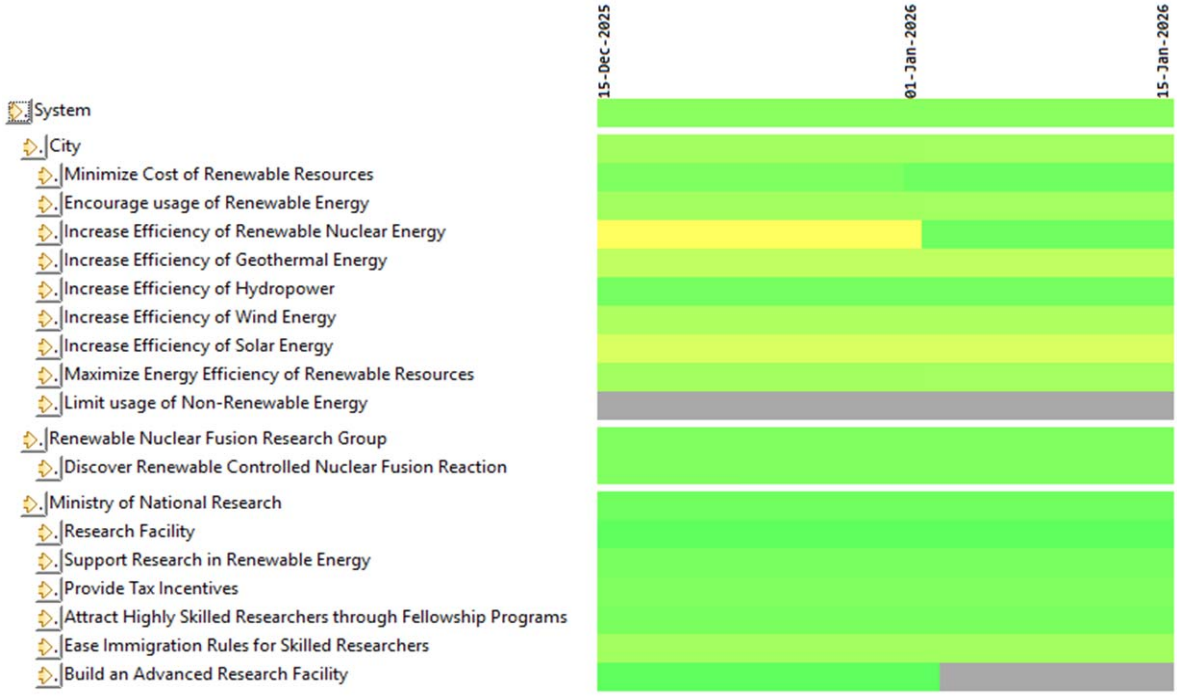

Figure 4: Heatmap for Goal Model Evaluation from 15th December, 2025, to 15th January, 2026

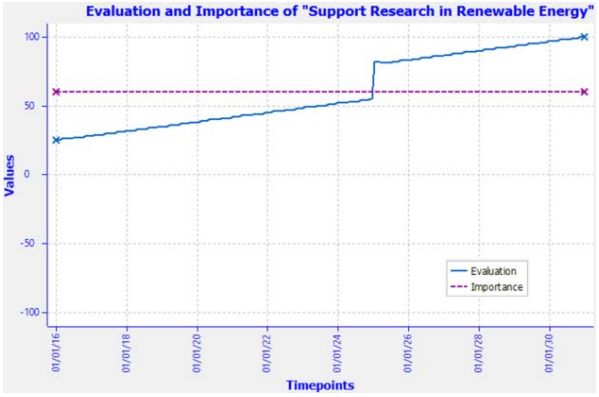

Figure 5: Variation Over Time in Evaluation and Importance of Goal "Support Research in Renewable Energy"

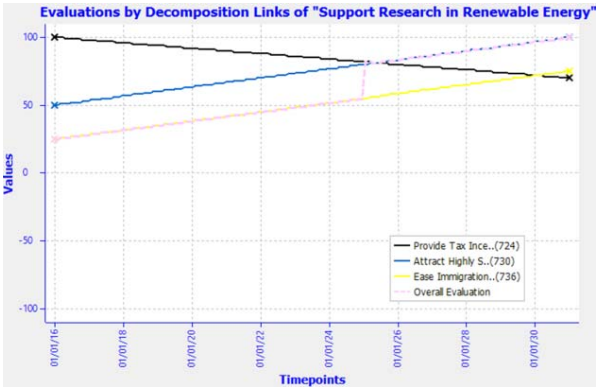

Figure 6: Variation Over Time in Impacts of Decomposition Links of Goal "Support Research in Renewable Energy"

A modeler can determine at a glance which elements are deactivated and when ("Limit usage of Non-Renewable Energy" throughout the whole time range and "Build an Advanced Research Facility" a little past half way through the time range). Furthermore, green and yellow color tones indicate when a model element is more or less satisfied. Negative satisfaction values in red color tones do not appear in this particular example.

For a more detailed view of a single model element over the whole time range, a modeler selects a model element to plot detailed graphs of its satisfaction value, its importance value, and the impacts of its contributions, decompositions, and dependencies as shown in Figures 5 and 6 for the goal "Support Research in Renewable Energy" over 15 years from 2016 to 2031. Figure 5 shows that the importance value of the goal is constant and that the satisfaction value increases continuously, and then jumps up sharply in 2025 before continuing to increase. The same progression of the satisfaction value is also shown with a dashed line in Figure 6, which captures the impacts of the three child elements of goal "Support Research in Renewable Energy" over the same 15 years. Most notably, the sharp increase of the satisfaction value in year 2025 can now be contributed to the change in decomposition type from AND (minimum of all three impacts) to OR (maximum of all three impacts). Due to the long time range of 15 years, the evaluation frequency used for these two charts is 30 days.

Graphs as shown in Figures 5 and 6 can be created for each numerical value in the URN model, i.e., they can also be created for the hit counts of the scenario traversal mechanism. In addition and similarly to the goal model, an overview of the analysis result of a time range is visualized directly in the scenario model. For this purpose, the existing visualization of the traversal mechanism is first improved. Currently, the traversal mechanism only highlights – for a single time point – each traversed element in red and all other elements in black while graying out deactivated path elements (see Figure 2). However, some path elements are traversed more often than others in a scenario, e.g., due to the presence of loops in the scenario model. To illustrate this, the scenario model in Figure 2 has been updated with three loops, representing individual requirements, individual building companies, and individual tasks in the operational planning phase (see Figure 7; note that a responsibility also replaces the stub from Figure 2).

The improved visualization of the scenario takes a temperature-based approach, where elements that are visited more often by the traversal mechanism get hotter and hence redder. Four hit counts for this scenario are shown in Figure 7. The requirements loop is traversed the most (91 times) and hence is the reddest, followed by the operational planning loop (71 times) in orange and the building company loop (41 times) in yellowish green. The rest of the scenario model is either green (traversed once), gray if deactivated, or black if not traversed at all. The maximum number of hit counts is configurable in the preference settings of jUCMNav.

The visualization of the analysis result for a time range builds on this improved visualization of a single time point. The overview of the analysis result of the time range conveys for how many evaluated time points the path element is active (i.e., the *traversal hit count*) and how often a path element is traversed on average (i.e., the *average hit count*: sum of all hit counts per evaluated time point divided by the traversal hit count of the element). To visualize this information, we considered three options: text size, color of path element, and width of path. We decided not to use text size, because jUCMNav allows text size to be customized (like any other Eclipse plugin). We again used a temperature-based approach to indicate the traversal hit count, i.e., the more active the path element, the hotter and redder it is. In addition, the width of each path segment is determined by its average hit count, i.e., active path segments that are traversed on average more often (e.g., because of a loop) are wider than paths that are traversed only once on average.

The example in Figure 8 shows the combined result of seven time points in the time range from 1st January, 2016 to 1st January 2024, i.e., the explicit time point approach is used. Recall from the earlier description of the scenario model that path elements are deactivated gradually from the start of the scenario

and the Hiring Committee is active only after 2024 (i.e., after the end of the evaluated time range). Therefore, the expectation is that the path elements closer to the start are colder (and hence greener) than those in the middle (which are more orange), and that the Hire Staff responsibility at the end of the scenario is not traversed in any time point and hence black. This can be observed in Figure 8. The coloring of the path segments shows that the main path at the center is red, because it is traversed all the time (the scenario starts at the start point for all time points), whereas a loop is only traversed if its corresponding responsibility is active. Hence, the first two loops are colder (and greener) than the third loop (which is orange). The traversal hit count is shown for six path segments and confirms the color coding (the higher the traversal hit count, the redder the path segment). The last part of the scenario including Buy Equipment is only traversed in one time point, but Hire Staff is still deactivated at that time point.

The average hit count is also shown for the same six path segments. Most of the path segments have a similar thickness, except for the three loops which are noticeably thicker (i.e., the average hit count is considerably higher than other average hit counts). For example, the "requirements" loop is the thickest, because the loop has a high hit count for the one and only evaluated time point for which it is active. The "operational planning" loop is thicker than other path segments, because it is active in six out of seven evaluated time points and has a relatively high hit count for each time point. Note that the path segments with the hit count pop-ups are thicker than other path segments to indicate the selected path segments for the hit counts.

All discussed visualizations of the analysis result of a time range, i.e., heatmaps, graphs for individual elements, and color-coded traversal results including average hit count and traversal hit count have been implemented in the jUCMNav tool as a proof-of-concept. All figures are based on screenshots, although some have been enhanced (e.g., to show several hit counts in one figure).

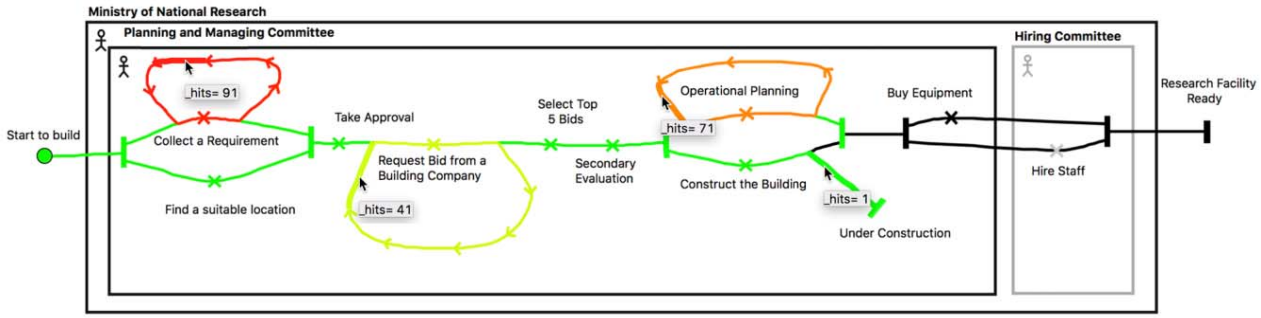

Figure 7: Improved Visualization of Traversal Result of a Single Time Point for 1st January, 2016

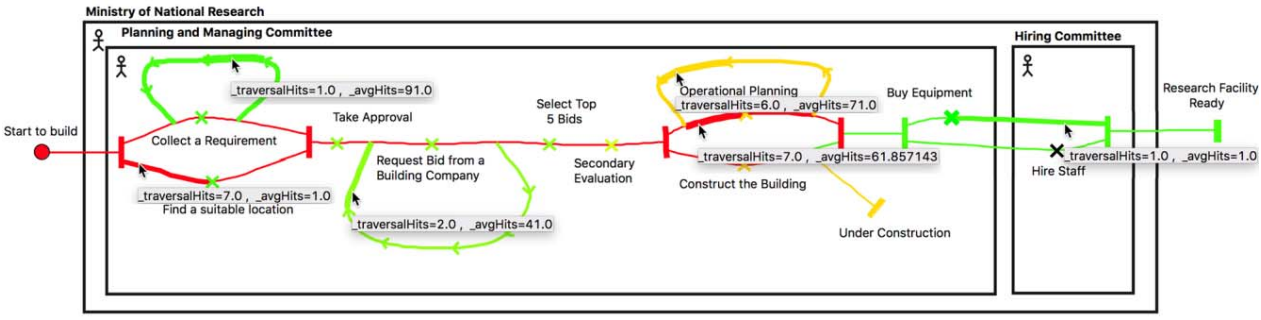

Figure 8: Overview of Traversal Result for Time Range from 1st January, 2016 to 1st January 2024 with 7 Time Points

4 Related Work

To the best of our knowledge, there has been no prior attempt to specify, analyze, and visualize the combined evolution of goal and scenario models, and certainly not in the context of the User Requirements Notation (URN) [1]. Many approaches for analyzing goal models exist [17], but few aim to capture, analyze, and visualize goal models over longer periods of time instead of a single snapshot in time and none do so together with scenario models.

The work by Grubb and Chechik [14][15] is most closely related to TimedURN, but focuses on capturing and analyzing the evolution of goal models. In contrast to TimedURN, their approach uses qualitative values and relative time instead of

quantitative values and concrete dates. Instead of using propagation-based evaluation and scenario traversal like URN, their goal model is translated into a constraint solving problem. In terms of visualization, only a specific (relative) point in time is visualized, but it is possible to explore a time range in a step-wise fashion.

Some goal models are defined with temporal logic. For example, Letier et al. [21] provide the ability to model timed properties of discrete-time event-based models as well as make model checking of timed properties possible. Other work looks at conditional non-monotonic temporal beliefs and goals [13]. Visualization, however, is not addressed.

Other work in requirements evolution by Ernst, Borgida, and Jureta [12] determines a new solution given an original requirements problem, its solutions, and a modified requirements problem by modifying/reusing solutions with minimal changes to the already existing solutions. However, visualization of changes over time is not considered at all.

More general work addresses the issues of model versioning (e.g., Hartmann et al. [16]), model differences (e.g., Cicchetti et al. [7]), and model management (e.g., Breu et al. [6]), but do not focus on the visualization of analysis results of a time range in an aggregated fashion.

5 Conclusion and Future Work

TimedURN [4][5] is an extension to the current User Requirements Notation (URN) standard to provide a comprehensive modeling environment for evolving systems expressed with goal and scenario models. This paper builds on previous work on TimedURN and focuses on (i) the analysis of a time range as opposed to a single time point and (ii) the visualization of the analysis result of the whole time range. In addition, the metamodel for URN is extended to support the evaluation of the whole system based on actor importance.

With the proposed visualizations of the analysis result of a time range, i.e., heatmaps, graphs for individual elements, and color-coded traversal results including average hit count and traversal hit count, a modeler can observe at a glance the overall progression of satisfaction values over time as well as the participation of path elements in scenarios over time.

TimedURN aims at the modeling of evolving systems that are typically in use for several years or even decades such as systems in the sustainability domain. To establish whether the proposed visualizations are intuitive for modelers, empirical user studies have to be performed in future work. In addition, better integration with standard diff tools for the analysis of the evolving code of responsibilities as well as conditions needs to be investigated for the jUCMNav tool. One option would be to make use of the textual syntax of URN, which is currently being standardized by ITU in an upcoming version of the URN standard.

Acknowledgment

Supported by FRQNT Grants.

References

[1] Amyot, D. and Mussbacher, G., "User Requirements Notation: The First Ten Years, The Next Ten Years", *Journal of Software (JSW)*, 6(5):747–768, 2011. DOI: 10.4304/jsw.6.5.747-768.

[2] Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., and Yu, E., "Evaluating Goal Models within the Goal-oriented Requirement Language", *Intl. J. of Intelligent Systems (IJIS)*, Wiley, 25(8):841–877, 2010. DOI: 10.1002/int.20433.

[3] Amyot, D., Rashidi-Tabrizi, R., Mussbacher, G., Kealey, J., Tremblay, E., and Horkoff, J., "Improved GRL Modeling and Analysis with jUCMNav 5", *6th International i* Workshop (iStar 2013)*, 2013. CEUR-WS 978:137–139.

[4] Aprajita and Mussbacher, G., "TimedGRL: Specifying Goal Models Over Time", *6th International Model-Driven Requirements Engineering Workshop (MoDRE 2016)*, IEEE CS, 125–134, 2016. DOI: 10.1109/REW.2016.035.

[5] Aprajita, Luthra, S., and Mussbacher, G., "Specifying Evolving Requirements Models with TimedURN", *9th Workshop on Modelling in Software Engineering (MiSE 2017)*, IEEE CS, 26–32, 2017. DOI: 10.1109/MiSE.2017.10.

[6] Breu, R., Agreiter, B., Farwick, M., Felderer, M., Hafner, M., and Innerhofer-Oberperfler, F., "Living Models - Ten Principles for Change-Driven Software Engineering", *Int. J. Software and Informatics*, 5(1-2):267-290, 2011.

[7] Cicchetti, A., Ruscio, D.D., and Pierantonio, A., "A Metamodel Independent Approach to Difference Representation", *Journal of Object Technology*, 6(9):165–185, 2007. DOI: 10.5381/jot.2007.6.9.a9.

[8] Chung, L., Nixon, B.A., Yu, E., and Mylopoulos, J., "*Non-Functional Requirements in Software Engineering*", Kluwer Academic Publishers, 2000.

[9] Duran, M.B. and Mussbacher, G., "Investigation of Feature Run-Time Conflicts on Goal Model-Based Reuse", *Information Systems Frontiers (ISF)*, Springer, 18(5):855–875, 2016. DOI: 10.1007/s10796-016-9657-7.

[10] Duran, M.B. and Mussbacher, G., "Top-down Evaluation of Reusable Goal Models", *17th International Conference on Software Reuse (ICSR 2018)*, to be published.

[11] Duran, M.B., Mussbacher, G., Thimmegowda, N., and Kienzle, J., "On the Reuse of Goal Models", *17th International System Design Languages Forum (SDL 2015)*, Springer, LNCS 9369:141–158, 2015. DOI: 10.1007/978-3-319-24912-4_11.

[12] Ernst, N.A., Borgida, A., and Jureta, I., "Finding incremental solutions for evolving requirements", 19th International Requirements Engineering Conference (RE 2011), IEEE, 15–24, 2011. DOI: 10.1109/RE.2011.6051656.

[13] Gonçalves, R., Knorr, M., Leite, J., and Slota, M., "Non-monotonic Temporal Goals", *12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013)*, Springer, LNCS 8148:374–386, 2013. DOI: 10.1007/978-3-642-40564-8_37.

[14] Grubb, A.M., "Adding Temporal Intention Dynamics to Goal Modeling: A Position Paper", *7th Intl. Workshop on Modeling in Software Engineering (MiSE 2015)*, IEEE, 66–71, 2015. DOI: 10.1109/MiSE.2015.19.

[15] Grubb, A.M. and Chechik, M., "Looking into the Crystal Ball: Requirements Evolution over Time", *24th IEEE Intl. Requirements Eng. Conf. (RE'16)*, IEEE, 86–95, 2016. DOI: 10.1109/RE.2016.45.

[16] Hartmann, T., Fouquet, F., Nain, G., Morin, B., Klein, J., Barais, O., and Traon, Y.L., "A native versioning concept to support historized models at runtime", *17th International Conference on Model-Driven Engineering Languages and Systems (MODELS 2014)*, Springer, 252–268, 2014. DOI: 10.1007/978-3-319-11653-2_16.

[17] Horkoff, J. and Yu, E., "Comparison and Evaluation of Goal-Oriented Satisfaction Analysis Techniques," *Requirements Engineering Journal (REJ)*, Springer, 18(3):199–222, 2013. DOI: 10.1007/s00766-011-0143-y

[18] International Telecommunication Union, "*Recommendation Z.151 (10/12), User Requirements Notation (URN) – Language definition*", 2012. http://www.itu.int/rec/T-REC-Z.151/en

[19] jUCMNav website, version 7.0, University of Ottawa. http://jucmnav.softwareengineering.ca/jucmnav

[20] Kealey, J., *"Enhanced Use Case Map Analysis and Transformation Tooling"*, M.Sc. thesis, SITE, University of Ottawa, Canada, September 2007.

[21] Letier, E., Kramer, J., Magee, J., and Uchitel, S., "Fluent Temporal Logic for Discrete-time Event-based Models", *SIGSOFT Software Engineering Notes*, ACM, 30(5):70–79, 2005. DOI: 10.1145/1095430.1081719.

[22] Luo, H. and Amyot, D.: "Towards a declarative, constraint-oriented semantics with a generic evaluation algorithm for GRL", *5th International i* Workshop*, pp. 26–31, 2011.

[23] Vrbaski, M., Mussbacher, G., Petriu, D., and Amyot, D., "Goal Models as Run-time Entities in Context-Aware Systems", *7th International Workshop on Models@run.time (MRT 2012)*, ACM, 3–8, 2012. DOI: 10.1145/2422518.2422520.

[24] Yu, E., "*Modeling Strategic Relationships for Process Reengineering*", Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995.