Extended Support for Visualizing Requirements

Filtering and Tracing Requirements in *ReBlock*

Deepti Savio
Research and Technology Centre
Siemens Technology and Services Pvt. Ltd.
Bangalore, India
deepti.savio@siemens.com

Ashok Pancily Poothiyot
Department of Computer Science and Engineering
Indian Institute of Technology, Indore
Indore, India
ashokpancily@iiti.ac.in

***Abstract*—The visualization of system requirements in a domain-independent, transparent and flexible manner with respect to the progress of the project for all project stakeholders is an area that still needs attention. There are several means to visualize requirements stored in a database, but few ways in which the status of a requirement – such as at what stage of realization it is at a given point in time, its associated dependencies with other requirements, multi-directional traces, and so on – can be discerned at a glance, while simultaneously keeping the overall big picture in view. The *ReBlock* plug-in was conceptualized and prototyped in order to address these limitations. Here, we discuss the implementation of several extensions to the initial prototype of the plug-in that make it more adaptable for visualizing a larger number of requirements.**

***Index Terms*—Requirements Visualization, ReBlock, Filtering, Cluster Tracing**

I. Introduction

System requirements for large-scale, distributed projects are often captured and stored in various Requirements Management (RM) tools. Normally, these requirements are represented in a combination of ways, such as natural language text, domain specific languages, diagrams and models, and so on. The concept of visualizing requirements as colored blocks within a three dimensional, expanding, top-down, wireframe pyramid was first described in [1]. *ReBlock* was designed and developed as an add-on plug in that sits on top of an underlying RM tool (Caliber RM, in this case).

The plug-in picks up requirements from the tool, converts these requirements into an intermediate XML format, and then displays each requirement as a colored cube, at various levels of hierarchy inside the pyramid. Each row of the pyramid denotes a specific level of refinement / requirement abstraction, and each face of the pyramid represents a particular stakeholder / stakeholder group's point of view, as shown in Fig. 1 [2]. This is similar to the concept of displaying continuous, real time information to stakeholders as discussed in [3]. If the number of requirement blocks for a particular face and row exceeds the geometric limitations of the pyramid area when stacked adjacent to each other, these requirements are grouped into a *cluster*, in order to avoid a spillover at each end of that face.

The additional implementations to the plug-in discussed in this paper pertain to the enhancement of its visual tracing mechanisms, as well as filter-based selection of requirement blocks, based on the end-user's input.

II. *ReBlock*: The First Prototype

Some of the key features implemented in the initial prototype of *ReBlock*, as described in [2], include the ability to:

- view requirements within an expanding, three dimensional pyramid, in a hierarchical manner, thereby providing users a means to visually discern the different levels of requirements breakdown.
- visualize requirement blocks from the perspectives of three to eight stakeholders, where each face of the pyramid displays blocks relevant for that particular stakeholder or stakeholder group.
- view requirement traces bi-directionally (also referred to as vertical tracing), from blocks in the lowest level of the pyramid back up to the root requirement block at the uppermost level.
- rotate the pyramid about numerous axes (apart from its central longitudinal axis) to facilitate easier views from various angles.

Cognitive amplification may be described as the use of means of visualization to help users visually sift through a large amount of data, in order to grasp meaningful information based on the patterns they see, as described in [4]. One of the main objectives of using *ReBlock* is to facilitate increased cognitive amplification of requirements, which are represented in a simple pictorial form, at various level of granularity, among collaborating stakeholders in a project.

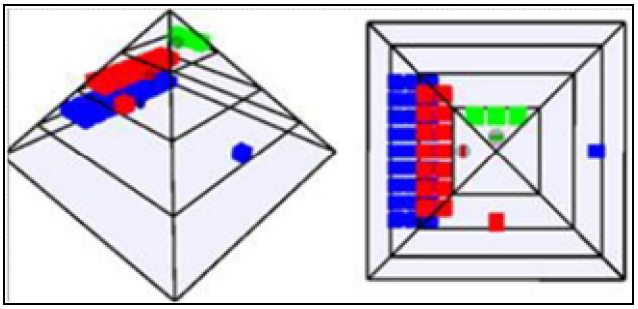

Fig. 1: Side and Top Views of the Pyramid [2]

To this end, it is also important for stakeholders to highlight requirements based on specific parameters, and to visually discern dependencies between requirements inside the same level. Extensions implemented to the second version of *ReBlock* along the lines of these two functionalities – filtering, bi-directional vertical tracing – are described below.

III. Filtering

Filtering refers to the feature of selecting and highlighting requirements based on available parameters that are specified in a drop-down list on the UI of the plug-in. The mechanism of filtering enables greater clarity about the status of various requirements, pertaining to any stage in the developmental process, hence allowing for informed analysis on which requirements of a project need greater focus.

A rudimentary, attribute-based filtering mechanism, which involved selecting and highlighting requirement blocks in the pyramid according to the 'status' filter parameter, selectable by the end user, was implemented in the initial prototype of *ReBlock* [2]. In the current version, filtering has been improved on the following fronts:

- degree of implementation:

the level of transparency of a particular block indicates the degree to which it has been implemented, and is visualized by the extent to which the color of the block has been made transparent. For example, a requirement that has been implemented only about quarter-way will be shaded to 25% of the original block color, as shown in Fig. 2. This mechanism enables the development team to communicate the status of requirement implementation easily to the other stakeholders in the project.

- status-wise filtering across all levels:

the user can determine and view requirements classified under various stages of project-specific development (for example, *accepted*, *deferred*, *implemented*, *tested*, and so on), as shown in Fig. 3.

- level-wise filtering:

requirements at any level of the pyramid can be highlighted.

- combined level / status filtering:

requirements of a particular status, at a particular level of the pyramid can be highlighted, as shown in Fig. 4.

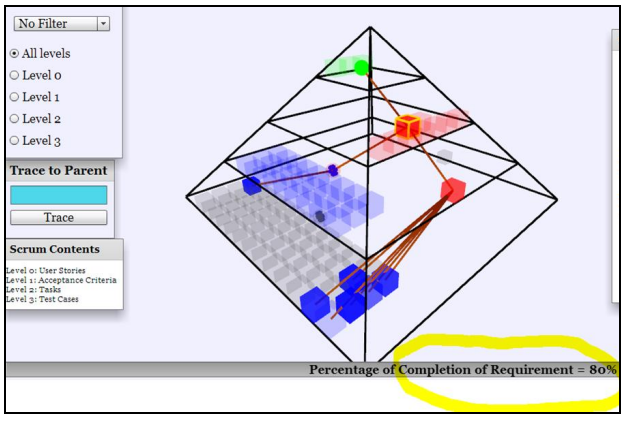

Fig. 2: Degree of Implementation of a Requirement

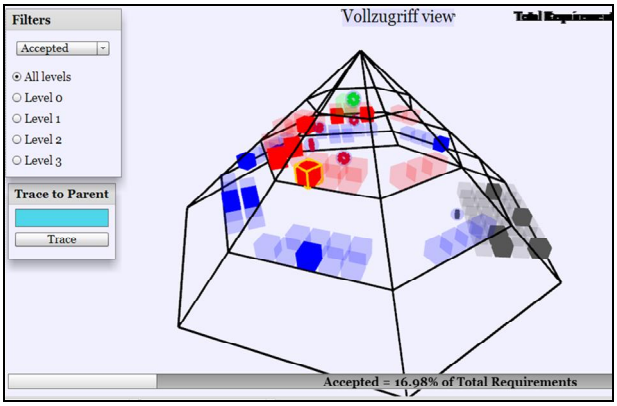

Fig. 3: All-level Filtering, for 'Accepted' Requirements

The improvements implemented for the filtering feature supplement the overall big picture view of the total set of requirements, in terms of discerning which requirements have and haven't been completed, and to what degree, which, in turn, allows for increased focus on development efforts on the lightly shaded blocks.

IV. Tracing

Tracing remains one of the main issues in managing requirements for many projects today [5]. Typically, when there are a large number of requirements in the pyramid, it becomes difficult to ascertain the logical coherence between requirements and the actual sequence of development.

In the current version of *ReBlock*, the tracing mechanism has been made more flexible, robust and scalable; and the plug-in attempts to provide visual clarity on the various types of linkages and traces between requirements in the following ways:

A. Vertical Tracing

Visual trace links from block to block in a vertical fashion in the pyramid complement the position of a particular block with respect to its hierarchy in the overall pyramid, facilitating an easily comprehensible relationship between levels of requirement blocks.

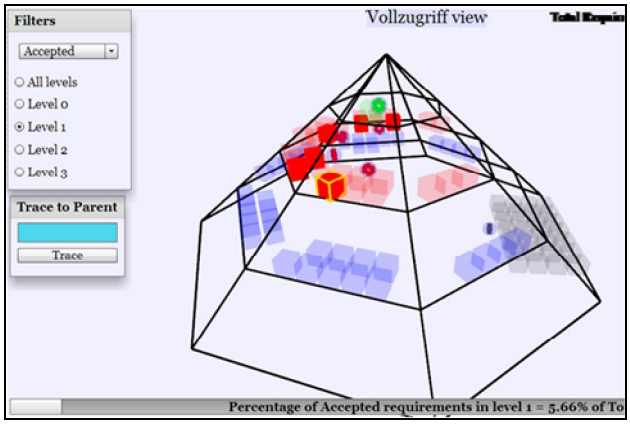

Fig. 4: Level-wise Filtering of 'Accepted' Requirements

In the initial prototype of *ReBlock*, vertical tracing was implemented in a rudimentary and static form, and only in the upward direction. In the present version, the end user is able to select a block or cluster, and then dynamically view the corresponding traces, represented in the form of thick, colored lines, up to the root requirement block / cluster at the tip of the pyramid and the lowest requirement block(s) at the bottom-most level (Fig. 5). These trace linkages are clearly visible, so this combination of top-down and bottom-up tracing allows the user to focus on a choice of possible refinements to existing parent-child relationships, should changes to requirements occur. Furthermore, the user can enter the unique ID of any requirement inside the 'TracetoParent' text box on the UI, to view its corresponding source requirement (Fig. 6).

B. Horizontal Tracing

Additionally, the mechanism to view dependencies between requirement blocks on the same level has been implemented, for immediate siblings of a selected block or cluster. Extending this to all requirement blocks within the same level depends on corresponding support provided by the underlying RM tool, as this information needs to be available beforehand for *ReBlock* to pick it up and render it in the intermediate XML form. This extension would then be realized using additional XML tags to capture the information that is to be visually represented as a series of horizontal connections within a particular level.

C. Cluster Tracing

Cluster tracing involves the capture and display of trace links between one or more clusters of requirements within the pyramid. Each cluster contains more than one requirement bundled together, and is an indication that the number of requirements in that level exceeds the structural limitations of the pyramid. The tracing function thus becomes scalable for a large number of requirements.

During implementation, the 'mesh' primitive was implemented as a super class of both the 'cluster' and 'cube/block' data types to handle both cluster and block tracing in order to facilitate code reuse with minimal changes to the existing codebase.

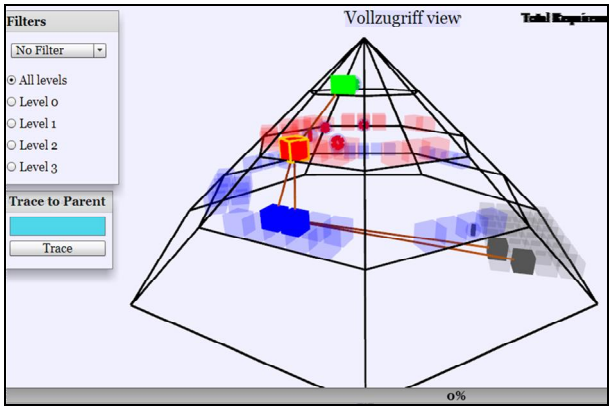

Fig. 5: Vertical Tracing

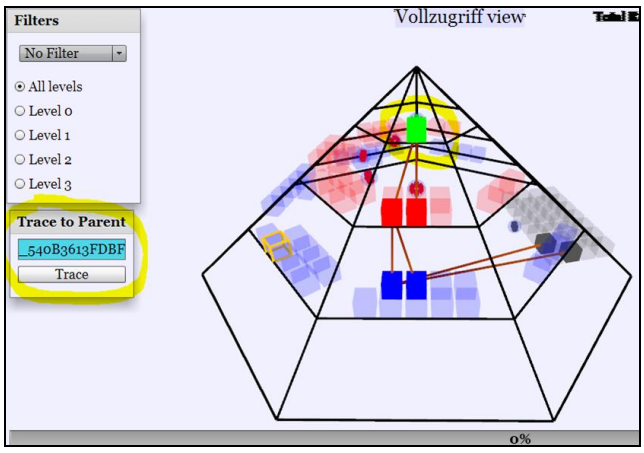

Fig. 6: *TracetoParent* Feature

Within this primitive, the value of the parameter 'BelongCluster' indicates whether a requirement is a single block or part of a cluster of requirements. In case of the latter instance, the entire cluster is highlighted, and the information of the relevant requirements that are part of the trace links within that cluster are displayed in a 'Cluster Trace Window' on the UI, in the order in which the clusters are visited as part of the trace function. This feature allows for scalable display of traces.

Trace-linked meshes could have been stored either as arrays or graphs. The CubeArray primitive is a data structure that specifies the collection of requirement cubes sorted on three criteria: the stakeholder to whom the requirement belongs, the corresponding level for each stakeholder, and the status within each level, in order to improve the efficiency of traversals and function operations. As an alternate approach to using the CubeArray primitive, graphs could have been implemented to store the trace-linked meshes [6], in order to facilitate easier bi-directional traversal. However, this approach would only be advantageous for densely populated graphs, with a large number of edges, or links. In the standard XML input samples that were used, the linkages were mainly sparse, and scattered throughout the levels of refinement. Using a large number of graphs to store information on each of these links, while leading to improved time complexity, would have nevertheless resulted in high overheads on memory and space.

V. Further Improvements to the Current Prototype

Envisioned improvements to the current version of *ReBlock* include the following:

A. Horizontal View Time-Lining

An application of the percentage-based shading of a block to indicate requirement implementation status would enable collaborating stakeholders to view snapshots of development efforts throughout the project life cycle, facilitating improved decision making on which requirements need attention. Frames of the requirement blocks may be visualized with a timeline bar below to indicate changing priorities and volatility levels at different phases and instances of project development. On

toggling the time-line bar, pyramids corresponding to different stages of development could be visualized for further analysis.

B. Extended Horizontal Tracing

For requirements on the same level of the pyramid, the indication of inter-linkages and dependencies may be pictorially represented in the form of horizontal lines, given that necessary support would be provided by the underlying RM Tool.

C. Using a Simultaneous 'Systems of Systems' Approach

Instead of using clusters to tackle requirements overflow in a level, sub-systems may be represented as small pyramid icons within the overall wireframe pyramid, symbolically indicating the overall system of systems view. On clicking the pyramid icon, a new, separate pyramid would visually bloat up within the same window, showing a more zoomed-in view of the requirements for that particular sub-system.

D. Requirements Interchange Format

Standardization of the common requirements interchange format may be done, to facilitate bi-directional communication between the *ReBlock* plug-in and standard Requirements visualization tools. The XML format utilized in *ReBlock* may also be modified for the same purpose.

VI. CONCLUSION

This paper describes the additional extensions made to the initial prototype of a plug-in for visualizing requirements that are stored in an underlying requirements management tool. The main objective of the plug-in is to alleviate ambiguity in requirements interpretation and communication by pictorially displaying requirements in a domain-independent form, at various levels of refinement, for collaboration among stakeholders in a project. Immediate extensions to the plug-in include the expansion of horizontal linkages between requirement blocks / clusters on the same level and the representation of requirement attribute information within each block. Eventually, we envision that *ReBlock* would be able to pick up requirements from popularly used RM databases and serve as a web-based, collaborative decision making aid for requirement-related resolutions that need to be made throughout the course of a project.

REFERENCES

[1] D. Savio, PC, Anitha and P.P. Iyer, "Visualizing Requirements: A Three Dimensional Pyramid Representation", Proceedings of the Fourth International IEEE Workshop on Multimedia and Enjoyable Requirements Engineering–beyond Mere Descriptions with More Fun and Games (MERE), 2011

[2] D. Savio, PC, Anitha, A. Patil and O. Creighton, "Visualizing Requirements in Distributed System Development", Proceedings of the Second IEEE Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems (RES4), 2012

[3] A. Sarma, A.V.D. Hoek, "Visualizing parallel workspace activities", No. UCI – ISR – 02 – 8, Institute for Software Research, University of California, Irvine, 2003

[4] T E. Lindquist, "Grappling with Complex Policy Challenges: Exploring the Potential of Visualization for Analysis, Advising and Engagement", Discussion paper for the HC Coombs Policy Forum, 2011

[5] J. Cleland-Huang, O. Gotel, A. Zisman, "Software and systems traceability," Vol. 2. No. 3, Springer, 2012

[6] E. R. Gansner, S.C. North, "An open graph visualization system and its applications to software engineering," Software Practice and Experience, Vol. 30, Issue 11, pp. 1203 – 1233, 2000