The Interactive Narrator Tool: Effective Requirements Exploration and Discussion through Visualization

Govert-Jan Slob[1], Fabiano Dalpiaz[1], Sjaak Brinkkemper[1], and Garm Lucassen[2]

[1] Utrecht University, Utrecht, the Netherlands
[2] SecFi Technologies BV, Amsterdam, the Netherlands
{g.j.slob, f.dalpiaz, s.brinkkemper}@uu.nl, garm@secfi.com

Abstract. Requirements visualization can contribute to requirements comprehension through the creation of conceptual models. However, these models can become hard to read and current tool support is minimal. Applying the right visualization mechanisms can help construct models that are more readable. To such extent, we present the Interactive Narrator tool: a web application that helps practitioners analyze software requirements at an abstract level. Interactive Narrator uses Natural Language Processing to derive conceptual models from user stories, which are then translated into an interactive network diagram with zooming and filtering capabilities. Interactive Narrator facilitates discussion and aims to accelerate the understanding of large sets of software requirements.

Keywords: Requirements Visualization, User Stories, Agile Development.

1 Introduction

Natural language is still the predominant software requirements notation in agile development [1, 2]. While natural language requirements are easy to read, as their quantity grows people experience more difficulty in constructing accurate mental models of the system under design.

In our previous work [3], we investigated the automatic generation of conceptual models as a possible solution to this issue, and we developed the Visual Narrator; a tool that automatically extracts conceptual models from sets of user story requirements. Our evaluation, however, showed that even such models quickly become too large to be effectively explored by analysts. The models' comprehensibility is hindered by the capacity of human's working memory, which restricts the comprehension of large models and causes a visual overhead to occur [4, 5].

The requirements engineering (RE) literature comes to help; the field of RE visualization (REV) is concerned with creating effective visualizations of RE artifacts. Inspiration for visualization can be drawn from the work of Moody [5] that provides guidelines for effective visualizations. A systematic review of the REV literature [6] concludes that more research is needed to support knowledge visualization for RE.

Cooper et al. [7] reviewed the papers that appeared in the REV workshop series, and distinguished different types of visualizations: tabular, relational, sequential, hierarchical, and metaphorical/quantitative. The outputs of our Visual Narrator tool falls in the relational category (hyper-graphs), but the less explored hierarchical aspect (decomposition into parts) is key for handling the large models comprehension problem.

While many approaches exist that suggest the use of models to represent requirements, the literature on visualizing existing natural language requirements as models is scarce. To date, RecVisu+ [8] is the most advanced tool in the field; it supports different visual exploration tasks and relies on clustering techniques and semantic similarity to reduce complexity. RecVisu+ determines similarity between requirements based on the frequency of co-occurrence in system documentation.

To address the problem of visual overload that affects our previous Visual Narrator tool, we propose a new tool, called the *Interactive Narrator*, that is inspired by RecVisu+ as well as by Shneiderman's visual information seeking mantra: "overview first, zoom/filter, details on demand" [9]. Interactive Narrator helps users inspect a model in an interactive fashion and enables them to construct mental models without being visually overloaded. Our tool puts forward a novel approach to requirements visualization; it is fully automated, considers relationships between entities, and is designed with touch screen usage in mind. Furthermore, unlike most work in conceptual modeling [10], our tool was conceived with high usability as a primary goal, and delivers a minimalistic and attractive design.

The rest of the paper introduces Interactive Narrator. After presenting its concept and main features in Section 2, we describe the contents of our demonstration in Section 3, and we discuss current and future work in Section 4.

2 The Interactive Narrator Tool

Interactive Narrator is a web application that relies on Natural Language Processing to derive conceptual models from text input. These models are subsequently translated into a network diagram that the user can interact with in different ways.

As input, the Interactive Narrator takes requirements in the form of user stories. This natural language notation has become popular with the rise of agile development methods and research showed that 90% of practitioners in agile development adopt user stories [11]. The most common template for a user story is: "As a {role}, I want {goal}, so that {benefit}". For example, "As a Visitor, I want choose an event, so that I can buy tickets for that event".

The user stories are analyzed by our Visual Narrator tool [3], which extracts relevant entities from the user stories and a set of relationships between those entities. For example, the user story example above would result in the entities Visitor, Event and Ticket, and in the relationships Choose(Visitor,Event) and Buy(Visitor,Ticket).

After the entities and relationships have been identified, the Interactive Narrator draws a network diagram that depicts the entities as nodes and the relationships as

edges between those entities. Nodes and relationships are labeled based on the noun and verb that identifies them in the user stories, respectively.

Two types of relationships exist: hierarchical relationships between entities depict a generalization relationship (e.g., is-a(Visitor,User)), and an actor-action relationship denotes an action that an actor performs on a concept (e.g., Buy(Visitor,Ticket)).

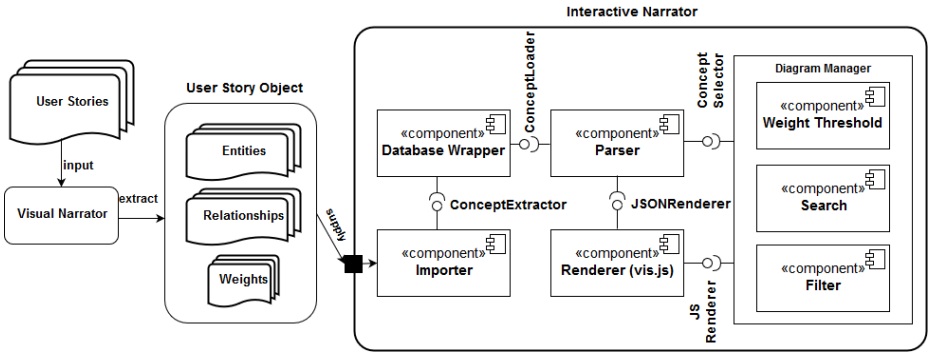

Fig. 1. Conceptual architecture of the Interactive Narrator

The conceptual architecture of Interactive Narrator is depicted in Fig. 1. The usage starts with the User Story objects provided by the Visual Narrator tool. From this point, the *Importer* extracts from the user story objects the relevant entities. These entities are stored by the *Database Wrapper* component that also provides services to supply entities to the *Parser* which not only retrieves the entities from the database but also translates them into JSON strings needed for drawing. Next, the *Renderer* is responsible for drawing the network diagram using the vis.js JavaScript library. Finally, when interacting with the diagram, the *Diagram Manager* communicates to the Renderer and the Parser which items to retrieve from the database and how to change the visualization subsequently. As soon as the visualization is rendered, the tool offers the following functionality:

- *Zooming and positioning*: Interactive Narrator allows users to zoom in and out on the diagram, allowing them to create an overview of a project or to focus on specific areas. It is also possible to reposition parts of the diagram.
- *Roles and sprints*: an important part of sustaining model readability is applying filtering techniques to customize the view to a more comprehensible format. Our tool allows users to show or hide entities based two criteria: sprint or role. It is indeed possible to upload user story sets per development sprint, and filtering by sprint greys out nodes that are not in the selected sprints. Filtering by role allows to only show the nodes that are connected to the roles in the user's selection. Optionally, all roles can be hidden to focus only on the entities.
- *Importance threshold*: by adjusting a slider mechanism, entities with a lower importance score than the setting are filtered out of the visualization. Node importance is based on the frequency of occurrence within the set of user stories, and this value is determined by our Visual Narrator tool [3].

- *Relationships*: by default, all relationships between entities are shown. To further increase model readability, users can hide the relationships between the nodes.
- *Search entities*: for a quick discovery of one of the entities in the diagram, a search function highlights the relevant node with a different color. This allows users to focus quickly on the area of interest, even in larger models.
- *User stories*: as each entity has been extracted from a user story, there are one or more stories behind each of the nodes. Clicking on a node will show a pop up listing all the user stories the concept is found in, enabling the user to understand the context of the visualization to full extent.

Because software development involves multiple stakeholders and is typically a team effort, requirements are often discussed in team meetings and during presentations. It is in such circumstances that a presentation experience can be created that fosters discussion about requirements, consequently improving the chances of lifting the requirements to a higher level. We therefore designed Interactive Narrator to work with large touch screen devices. During development, Interactive Narrator was tested on an 84-inch touch screen.

3 Demonstration Content

We illustrate Interactive Narrator using a set of thirty-four user stories that stem from a Content Management Software product for large enterprises. The presented screenshots provide a high-level overview of the demonstration we intend to deliver, but we intend to let the demo booth visitors interact directly with our tool, so to let them experience first-hand its capabilities.

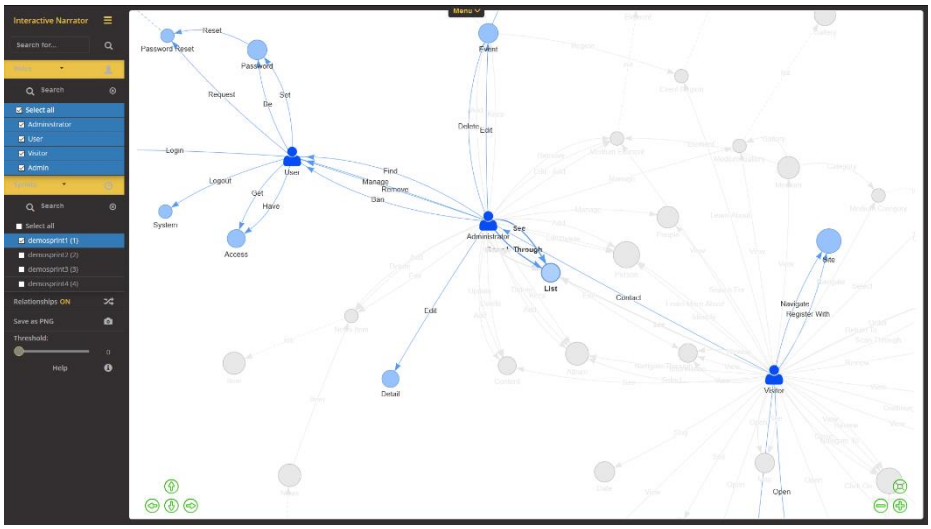

Fig. 2. Interactive Narrator, filtering out sprint 2,3 and 4

Fig. 2 shows the visualization, generated by four separate uploads of user stories, each upload represented as a sprint iteration unit from the SCRUM development method. Four roles and four sprints can be identified. The diagram has been adjusted by selecting all the roles present in the user stories, but only one of the four sprints. Consequently, the nodes from the deselected sprints are greyed out, drastically reducing the visual clutter in the image. The image shows three of the four roles: User, Administrator, and Visitor.

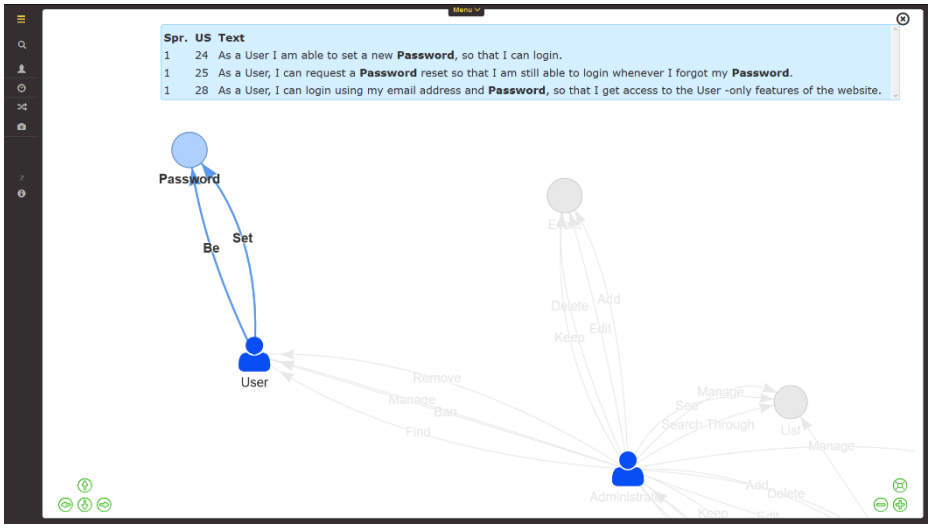

Fig. 3. Interactive Narrator, filtering out roles 1, 3 and 4 and inspecting user stories

The screenshot in Fig. 3 depicts Interactive Narrator when applying the filters for roles and zooming in. Of the four roles only the User role has been selected, again greying out all entities from user stories not associated with this role. Furthermore, the threshold has been set to only display entities and their relationships with a score of 2 and higher. Finally, because the Password node has been selected, a pop-up containing all user stories about this entity is shown.

4 Work in Progress and Future Work

Our work with the Interactive Narrator is still ongoing and there is room for improvement. The main direction of current development is to improve existing features such as speeding up the rendering performance, adding simple clustering techniques and export functionality. A second avenue of work is to report on the case study research we conducted in which we probed the feasibility of the Interactive Narrator.

Future work will involve the development of features such as clustering based on semantic relatedness, adding dependency, redundancy and inconsistency detection and support for themes and epics. Initial steps toward theses directions were made through our algorithms for identifying missing requirements and conflicting terminol-

ogy that are implemented in our REVV tool [12]. Interactive Narrator is available at https://interactivenarrator.science.uu.nl

References

1. Kassab, M.: An Empirical Study on the Requirements Engineering Practices for Agile Software Development. In: Proc. of the EUROMICRO Conference on Software Engineering and Advanced Applications. pp. 254–261 (2014).
2. Lucassen, G., Dalpiaz, F., Van Der Werf, J.M.E.M., Brinkkemper, S.: Visualizing user story requirements at multiple granularity levels via semantic relatedness. In: Proc. of the International Conference on Conceptual Modeling (2016).
3. Lucassen, G., Robeer, M., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: Extracting conceptual models from user stories with Visual Narrator. Requirements Engineering. 22, (2017).
4. Aranda, J., Ernst, N., Horkoff, J., Easterbrook, S.: A Framework for Empirical Evaluation of Model Comprehensibility. In: Proc. of the International Workshop on Modeling in Software Engineering (2007).
5. Moody, D.: The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on Software Engineering. 35, 756–779 (2009).
6. Abad, Z.S.H., Noaeen, M., Ruhe, G.: Requirements Engineering Visualization: A Systematic Literature Review. In: Proc. of the IEEE International Requirements Engineering Conference. pp. 6–15 (2016).
7. Cooper, J.R., Lee, S.W., Gandhi, R.A., Gotel, O.: Requirements engineering visualization: A survey on the state-of-the-art. In: Proc. of the International Workshop on Requirements Engineering Visualization. pp. 46–55 (2009).
8. Reddivari, S., Rad, S., Bhowmik, T., Cain, N., Niu, N.: Visual requirements analytics: A framework and case study. Requirements Engineering. 19, 257–279 (2014).
9. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: Proc. of the IEEE Symposium on Visual Languages. pp. 336–343 (1996).
10. Gulden, J., Reijers, H.A.: Toward advanced visualization techniques for conceptual modeling. In: Proc. of CAiSE Forum. pp. 33–40 (2015).
11. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: The use and effectiveness of user stories in practice. In: Proc. of the International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 205–222 (2016).
12. Dalpiaz, F., van der Schalk, I., Lucassen, G.: Pinpointing Ambiguity and Incompleteness in RE via Information Visualization and NLP. In: Proc. of the International Working Conference on Requirements Engineering: Foundation for Software Quality (2018).