Would You Like Better Visualization for Requirements Prioritization and Release Planning?

João Pimentel[1], Maria Lencastre[2]

[1] Universidade Federal Rural de Pernambuco (UFRPE), Brazil
[2] Universidade de Pernambuco (UPE), Brazil
joao.hcpimentel@ufrpe.br, mlpm@ecomp.poli.br

Abstract. On the one hand, requirements prioritization is a key requirements engineering activity that lacks proper visualization support. On the other hand, goal modelling is a visual approach for requirements engineering that enables expressing and reasoning about alternative solutions that best match the stakeholder needs, both for the early requirements and the late requirements phase. This paper proposes a set of strategies for visually representing prioritization and release planning information using goal-based models, aiming at minimizing the cognitive effort required in prioritization analysis. The proposal is supported by a tool that implements the visualization strategies here defined.

Keywords: Requirements Engineering, Requirements Prioritization, Goal Modelling.

1. Introduction

Requirements prioritization is a key activity of the requirements engineering process, enabling stakeholders to identify what are the most important features of the system-to-be [5][2]. Several studies have already provided significant contributions on this topic. For instance, many relevant prioritization criteria that can be adopted in order to inform the prioritization process have already been identified and classified – e.g., customer satisfaction, business value, risk and implementation effort [26]. Moreover, several requirements prioritization techniques can be used to guide such prioritization [6]. However, as evidenced by Cavalcanti et al. [8], there is a lack of visualization approaches supporting requirements prioritization.

Visualization comprises the transformation of raw data for easier assimilation through sight, increasing consciousness and insight on the data [13]. It is well known that visual artefacts facilitate thought [7]. Particularly, the beneficial role that visualization strategies can play in the context of Requirements Engineering has been extensively documented [10][1].

We advocate that proper visualization strategies can aid in the analysis of requirements' priorities, allowing to consider not only the priorities of individual requirements but also their relationships with other requirements. Furthermore, by combining prioritization visualization strategies with release planning visualization strategies, it is possible to identify mismatches between the results of both activities (e.g., low-priority requirements that have been unduly assigned to early releases).

In this paper we present an approach for the visualization of prioritization information – namely, the values assigned to each requirement regarding some prioritization criteria – and release planning information – specifically, the set of requirements that are assigned to future

releases of the system-to-be. In previous work, we have defined a prioritization process using visual artefacts, as well as an extension of *i** models in order to represent the resulting information [12][18]. The visualization strategies here presented are an alternative to the visualization adopted in our previous work, replacing tabular representation with quicker-to-grasp representations.

The choice of the *i** framework [28] as the baseline is due to its expressiveness in modelling and reasoning about early requirements and social networks. In particular, *i** models express not only system requirements but also stakeholders' goals and dependencies, as described in the next section.

Section 2 introduces the main concepts of *i**, which is the modelling framework upon which this proposal is built. Section 3 presents the visualization strategies that are proposed in this paper. Tool support for the proposal is described in Section 4. Related work is discussed in Section 5. Lastly, the paper is concluded in Section 6.

2. Baseline – *i** Framework

The *i** (i-Star) Framework [28] was chosen as the baseline for this proposal, since it includes a visual modelling language featuring some elements that can be used to support requirements prioritization. In particular, the iStar 2.0 standard [11], which aims at defining a consistent and clear set of core concepts of the *i** language, was adopted as the basis for this proposal.

To provide an overview of the *i** language, we adopted the example of an online store system (Fig. 1). The main goal of this system is to have the "Internet Shop Managed". In order to do so, it needs to handle item searching (goal) and internet orders (goal), as well as allowing to update the catalogue of products (task). Item searching can be handled by querying the database (task) or by consulting the catalogue of products (task). Internet orders can be handled using a Fill Secure Form Order (task) or a Fill Standard Form Order (task). Moreover, the system is expected to be attractive to new consumers, as well as providing security and availability.

Three concepts of the *i** language, mainly, make it a promising starting point for requirements prioritization and release planning:

- *Actors* – by including different actors and their dependency relationships with elements of the System-To-Be actor, it is possible to infer some priorities if one considers the weights of these actors. For instance, in Fig. 1 it can be seen that the Catalogue Consulting and Database Querying tasks are directly linked to customer needs. Nevertheless, this reasoning is partial, since not every stakeholder (such as developers and customers) take part in the system's strategic dependencies.
- *AND-Refinements* – since it is possible to visualize why the system includes each task, one can partially infer the priority of sub-elements. For instance, in Fig. 1, "Item Searching Handled" is an and-refinement of "Internet Shop Managed". Thus, if the latter is (for instance) high priority, the former will likely also be high priority.
- *OR-Refinements* – the information on alternative refinements can aid in release planning. For instance, if an alternative (e.g., Database Querying) has already been implemented, the other alternative (Catalogue Consulting) may be postponed to a later release, or perhaps even discarded entirely, since the first alternative is sufficient to satisfy the Item Searching Handled goal.
- *Contribution links* – the contribution towards qualities can aid the selection of alternative tasks. For instance, in Fig. 1 there are two options to have Internet Orders Handled: Fill Secure Form Order and Fill Standard Form Order. If we consider the Secure quality, the

best alternative would be the Fill Secure Form Order, which helps to satisfice the said quality.

3. Visualization strategies

This section presents the visualization strategies proposed in this paper in order to enhance requirements visualization for the purpose of prioritization (Section 3.1) and release planning (Section 3.2).

3.1. Visualization of prioritization attributes

Requirements engineering visualizations can be classified in five categories [10]: Tabular, Relational, Sequential, Hierarchical and Quantitative/Metaphorical. Regular *i** models are able to express relational and hierarchical information, i.e., sets of connected nodes and decomposition of elements, respectively. In this paper we propose to enrich *i** models in order to express prioritization attributes such as importance, cost and risk through quantitative/metaphorical visualizations, i.e., "visual clues such as color, shape, line thickness, and size to convey meaning at a glance" [10]. Moreover, Moody et al. [21] described eight visual variables that can encode information on visual models: horizontal position, vertical position, size, brightness, colour,

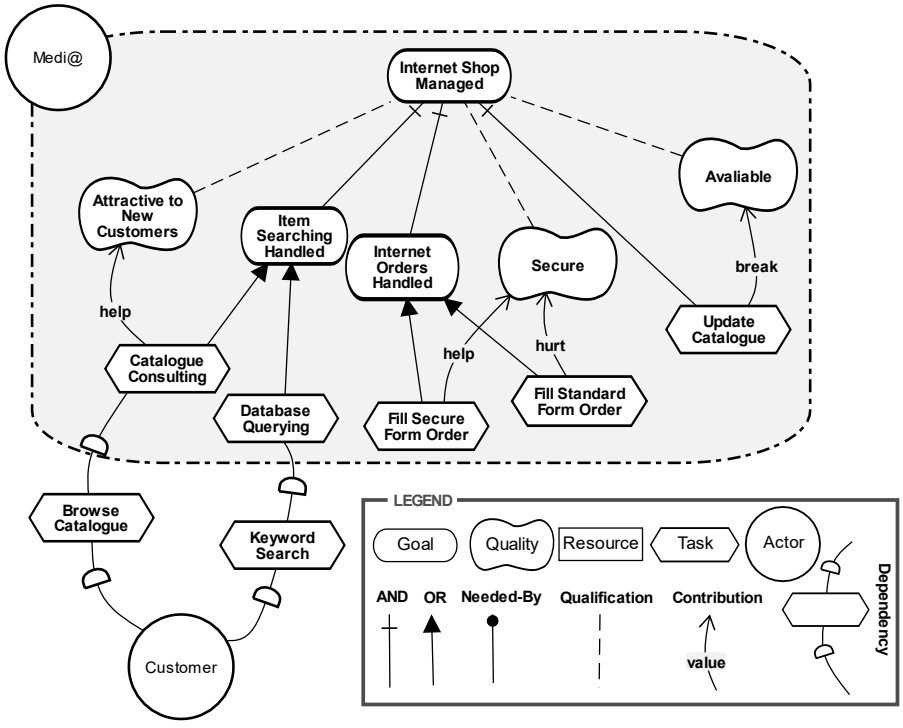


Fig. 1. Excerpt of the Medi@ Example

texture, shape and orientation. Whereas regular *i** models make use of horizontal and vertical position, as well as shape, our proposal makes use of colour, brightness and shape.

Specifically, a range from yellow to red is adopted in order to represent variation on the values of a prioritization attribute, from lowest to highest. The specific colours were carefully selected in a way to correspond to a brightness variation, where lower values have higher brightness and higher values have lower brightness. The benefits of this selection are twofold: as redundant coding [21], brightness acts as an additional signal that reinforces the meaning of the colour; moreover, it allows to differentiate (to a lesser degree) the attribute values even when the image is seen by people with visual colour deficiencies, even though further empirical evaluation is warranted in order to confirm the accessibility of this proposal.

Even though colour is one of the most effective visual variables, size is the one with higher *capacity* (i.e., number of perceptible steps), apart from shape [21]. Thus, we propose to also use a range of sizes in order to represent the value of prioritization attributes, ranging from small (lower values) to large (higher values).

Besides these three visual variables, a textual encoding was additionally adopted in order to express the values of a given prioritization attribute with precision. This encoding is represented by a numerical value contained within an icon that is attached to an *i** element.

In the remainder of this paper the terms *colouring* is used to refer to variations of colour and brightness together, *resize* to refer to variations of size, and *icons* to refer to the textual encoding. Combination of these variations are possible, both for a single attribute and for multiple attributes. For instance, colouring and resize can be used to represent the range of cost, thus reinforcing the signals regarding a single attribute. Alternatively, one could use colouring to represent cost while resize is used to represent benefit, thus representing multiple attributes (cost and benefit) in the same visualization. Regardless of the strategies chosen, consistency is key in order to prevent confusion within a team, project, or organization.

Fig. 2 shows different visualizations of prioritization values in an *i** model. Fig. 2-a shows a regular *i** model. Fig. 2-b shows the same model but now making use of the colouring strategy, where low to high values of importance are represented with colours from yellow to red, respectively. It can be seen, for instance, that Quality A has much higher importance than Task E. In black-and-white (shades of grey) versions of this image, the different values can still be observed, even though it is harder to make such distinction – brighter elements have lower importance, whereas darker elements have higher importance.

Fig. 2-c exemplifies the resizing option, where smaller elements have lower importance and larger elements have higher importance. A potential issue here is: what to do with elements that have undefined values for a given attribute? We have chosen to make these elements smaller, so that they are not confused with high-valued elements. However, this may lead to a confusion between low-valued elements and elements with undefined values.

Lastly, Fig. 2-d shows the exact importance value of each element by means of a numerical value contained withing an icon attached to the element. In this particular example, these icons are also resized according to importance values – larger circles represent higher importance values.

In summary, Fig. 2-b, Fig. 2-c and Fig. 2-d represent values of the same attribute (importance) through different visualization options. In this particular example, it can be observed that: both Task D and Criterion Goal A have high importance, even though Task D *breaks* Criterion Goal A; Task E has low importance, even though Task E is a part of Task D, which has high importance. Based on these potential inconsistencies it may be decided that the current set of importance values are not satisfactory, hence further analysis is required.

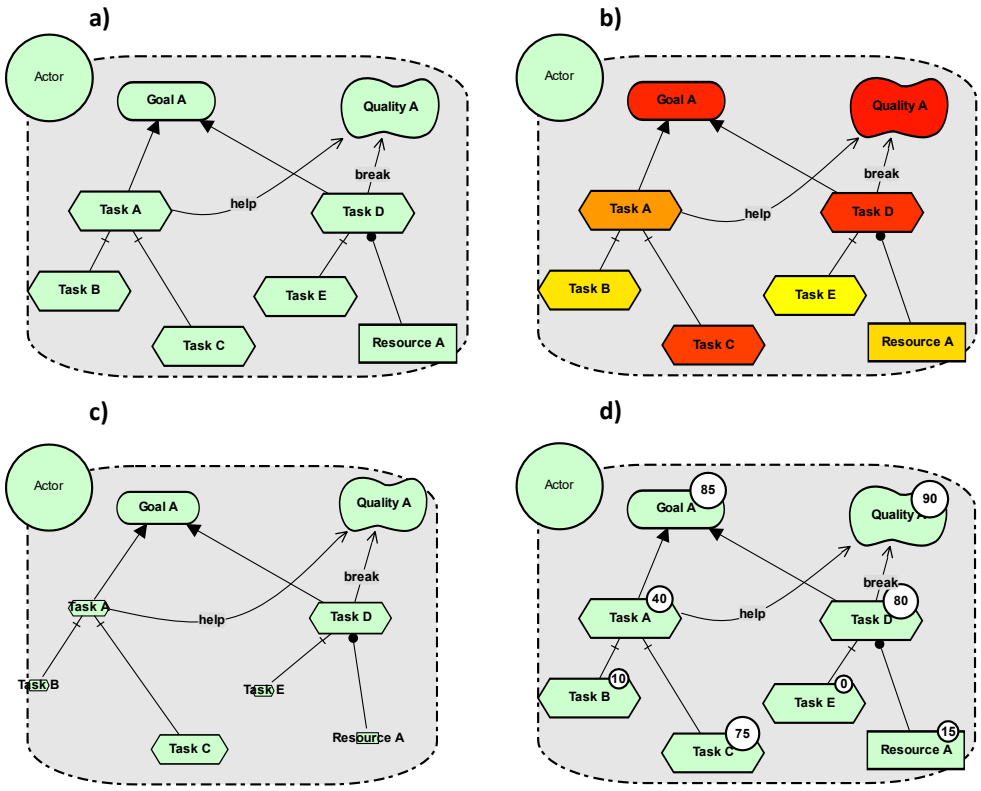


Fig. 2. Different visualizations of the same *i** model: a) original; b) colouring elements by importance; c) resizing elements by importance; d) importance value on icons attached to each element, with icons resized according to their value.

Multiple Attributes. While the attached icons shown on Fig. 2-d only present one attribute (namely, importance), many attributes can be displayed at once as well. Fig. 3 shows different options for visualizing the cost, benefit, and risk of each requirement (in this order). Proper tool support can enhance this visualization by providing interactive tooltips that display the name of the attributes being visualized.

On Fig. 3-a, each icon simply shows the value of each attribute (cost, benefit and risk, respectively). Fig. 3-b and Fig. 3-c show not only the value but also a visual reinforcement: size and colour (respectively). This reinforcement can make it easier to compare each attribute. Fig. 3-d shows a combination of the previous options, where the attribute values are also represented by both the colour and the size of each icon.

It is also possible to hide the value of each attribute, relying solely on visual cues to compare the attributes (Fig. 3-e, Fig. 3-f, and Fig. 3-g). In this case, the first letter of the name of each attribute is displayed (C - Cost; B – Benefit; and R – Risk). This option reduces the precision of the visualization but reduces the effort of remembering the names of the attributes that are being visualized.

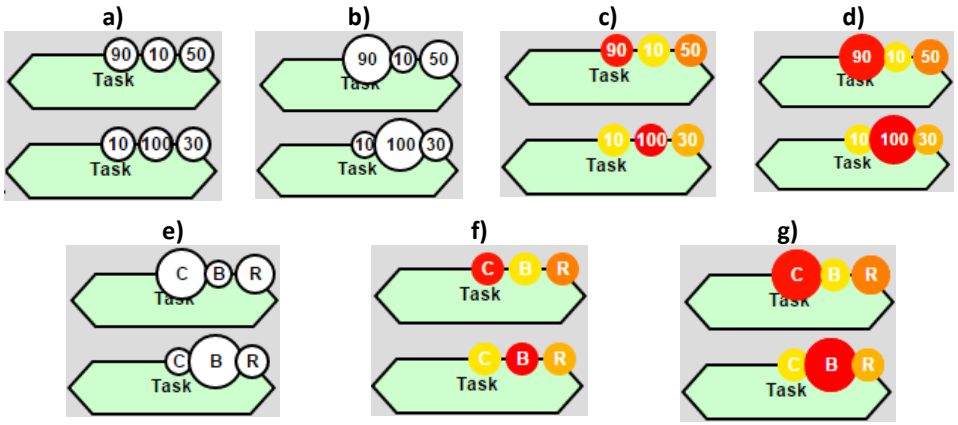


Fig. 3. Different options for icons showing, respectively, the Cost, Benefit, and Risk of each attribute. a) plain; b) resizing; c) colouring; d) colouring and resize; e) resizing, attribute first letter; e) colouring, attribute first letter; e) colouring and resizing, attribute first letter.

3.2. Visualization of release planning

Release planning is an activity strongly related to requirements prioritization [16]. We propose three strategies for visualizing the requirements that are selected to be included in a given release: line-decoration, resizing, and colouring. Here, resizing and colouring differ from the strategies described in the previous sub-section by considering only two values (namely, included in the release or not), instead of a range of values.

Fig. 4 shows different visualizations of the same *i** model, where Task D, Task E, and Resource A are scheduled for the first release. In Fig. 4-a, these elements are highlighted by means of thick, dashed lines. In Fig. 4-b, the elements that are *not* included in the first release were made smaller, thus allowing the viewers of the model to focus on the elements that are scheduled for the release under analysis. Lastly, Fig. 4-c highlights the elements scheduled for the first release by means of colouring.

These different visualization options can be combined, making the differentiation between releases even more distinct. Moreover, the visualization of releases can also be combined with the visualization of prioritization information previously described. By combining the visualization of prioritization information with the visualization of release planning, one can analyse the project planning by considering questions such as “Which highly important requirements are not included in a given release?” and “What is the cost, benefit, risk, etc. of each requirement in a given release?”

Nonetheless, not every combination of visualization strategy is advisable: the resizing strategies should not be applied simultaneously to prioritization information and to release planning information. This is the case since the viewer would not be able to know whether an element is small because it is not included in the release or because it has low priority, Similarly, colouring should not be applied, at the same time, to prioritization information and release information, in order to prevent confusion regarding the colour schemes.

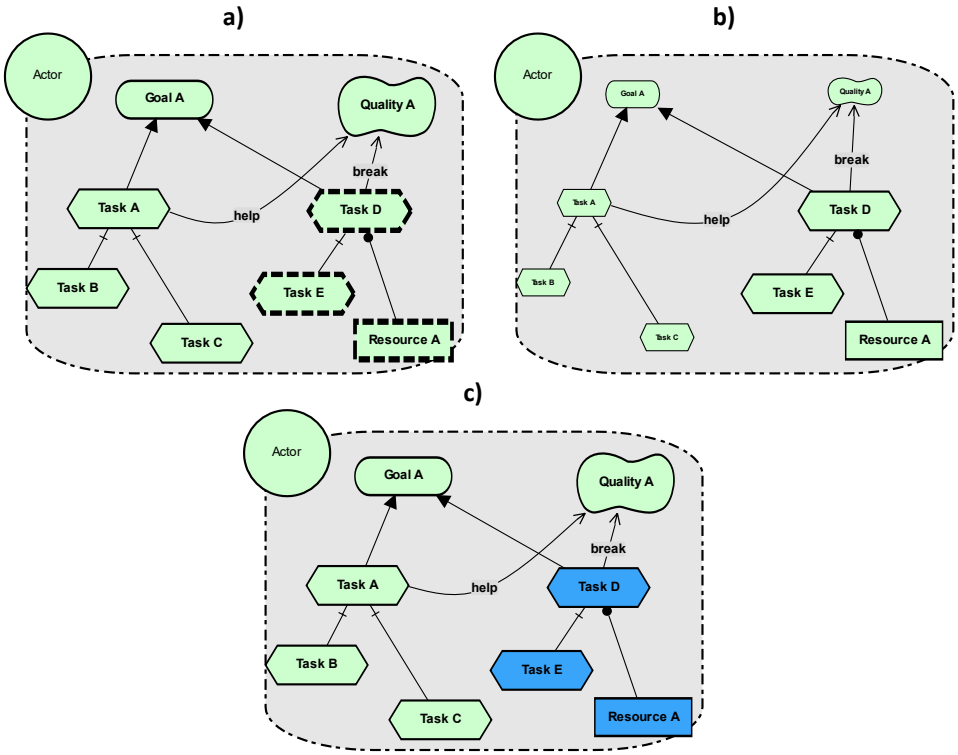


Fig. 4. Different options for visualizing release information: a) line decoration; b) resizing; c) colouring.

4. Tool support

In order to enable the use of the visualization strategies described in the previous section, we have developed an extension of the piStar tool, a web-based general-purpose *i** modelling tool [22][23]. This extension, available online at https://www.cin.ufpe.br/~ler/supplement/wer2020/pistar/prioritization/ , comprises three new functionalities: prioritization setup, prioritization visualization, and release planning visualization.

The user of the tool (e.g., a requirements engineer), after creating an *i** model depicting the system requirements, may need to add some metadata to its requirements, such as: rationale, origin, cost, and benefit. Instead of documenting this metadata in a separate artefact, the user can include it in the model itself, by defining attributes for the model's elements. Since the piStar tool is flexible, any kind of metadata can be added. However, in order to facilitate its usage for our specific purposes, the extension provides a pre-defined set of meta-attributes that can be automatically added to all elements: benefit, cost, effort, penalty, risk, volatility, priority itself, and release. These options were selected based on the discussion in [4][5][27].

When analysing a requirements model, the user can apply the proposed visualization strategies in order to attain a clear comparison of the attributes pertaining to different elements. For instance,

one may want to identify the elements that have higher volatility, or to compare the cost-benefit-risk of different elements, through prioritization visualization. Additionally, the user can apply the release planning visualization functionality in order to quickly grasp the requirements that are scheduled for a particular release. By visually checking the metadata information, the user may identify adjustments to be performed, then proceeding to make such adjustments and re-applying the visualizations.

The user interface of the piStar tool has three sections: top, left, and right (Fig. 5). The left section is the *properties panel*, which displays the properties and contextual actions of the selected element of the model. This panel is used to manually add new properties/attributes (e.g., cost, risk, and priority) to individual elements, as well as to edit their values. The right section is the *modelling canvas*, on which the model itself is created and modified. On the top section of the screen there is the *main menu area*, with five options: File, Add, Options, Help and Prioritization plugin. The last menu provides access to the visualization functionalities:

- *Manage attributes* provides a straightforward way to add a pre-defined set of attributes for every element in the model: Benefit, Cost, Effort, Penalty, Priority, Release, Risk, and Volatility.
- *Visualize prioritization* presents the options to visualize prioritization-related information, as shown in Fig. 6-a: colouring ("Change colour based on…"), resize ("Change size based on…"), and icons ("Show attribute X"). For each visualization option the user chooses on which attribute it will be based. Up to three icons can be displayed, each with the options of presenting values, colouring and resize on the icon itself.
- *Visualize release:* allows highlighting the elements of a given release (Fig. 6-b). The user may choose how to highlight them: through line decoration, colouring, resize, or a combination thereof.

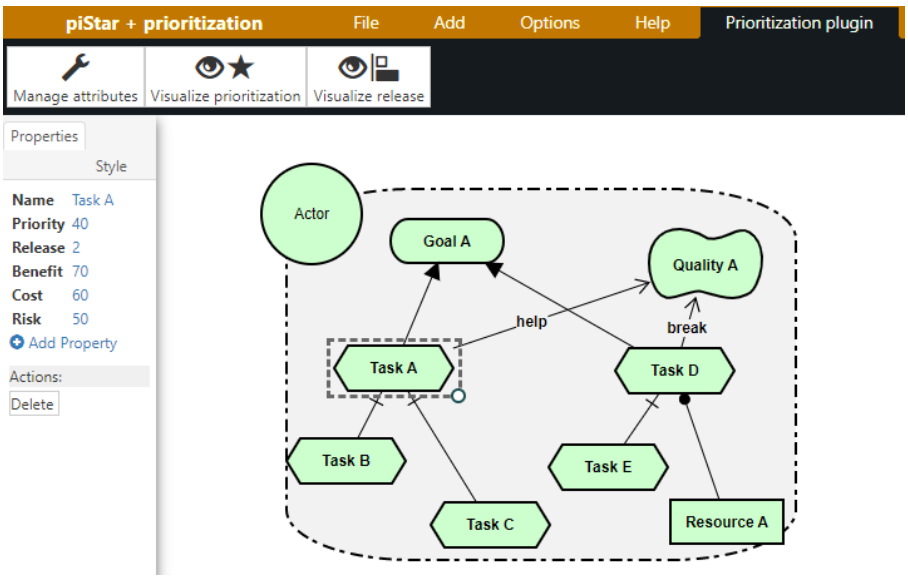


Fig. 5. Main screen of the piStar tool with the prioritization plugin. Top: Main menu; Left: Properties Panel; Right: Modelling canvas.

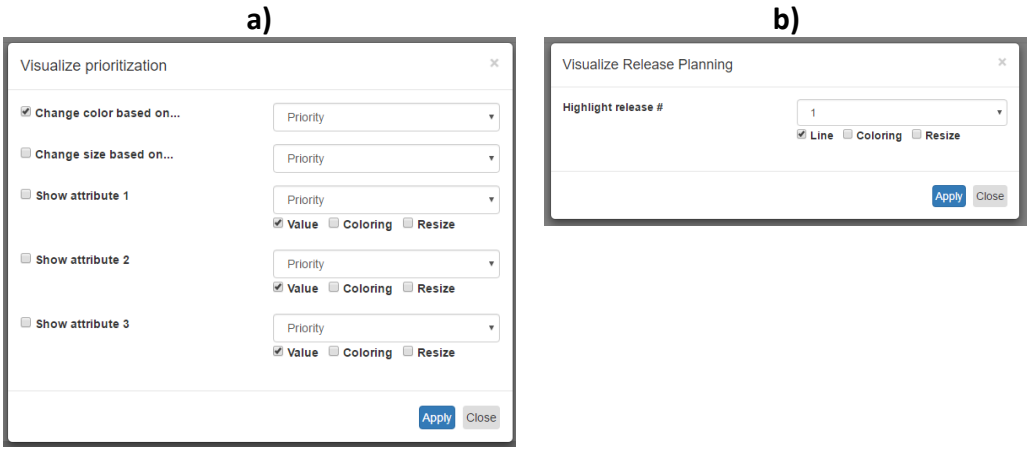


Fig. 6. Visualization options: a) prioritization visualization options; b) release planning visualization options.

4.1 Architecture and design

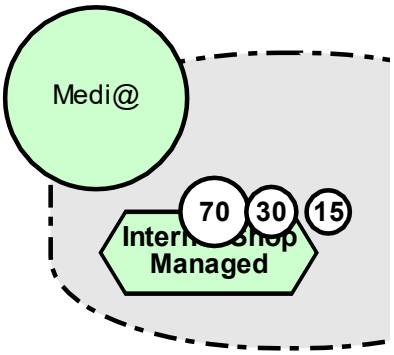
The tool adheres to the Model-View-Controller (MVC) architectural style. In particular, the *i** model has two views: the graphical view, created as an SVG (Scalable Vector Graphics) image; and a tabular view, to represent the attributes of each individual requirement. The user interface is also mostly developed as models and views, except for the overall layout of the tool and some static elements, which were written with plain HTML (HyperText Markup Language).

The *i** diagrams can be exported as images for use with other tools. Two options are available: export as SVG and export as PNG (Portable Network Graphics). The former is the highest fidelity option, as it is a vector file format. Nevertheless, since some tools do not support this file format (e.g., older versions of the Microsoft Office suite), we also developed the option to export as high-resolution PNG images. The model itself can be saved locally as JSON (JavaScript Object Notation) object. Fig. 7 shows an excerpt of a goal model with its corresponding JSON object, where an actor named "Medi@" has a task named "Internet Shop Managed". This task has cost 70, benefit 30, and risk 15.

The following libraries were used in this project: JointJS, to handle the edition of visual diagrams; jQuery and x-editable for the user interface; and Backbone for implementing the MVC style. The Bootstrap framework was used to provide a pleasant look-and-feel to the user interface.

4.2 Evaluation

The execution time for the application of a visualization strategy is a relevant acceptance factor for this proposal. This is the case since, if the person that wants to analyse the visualization needs to wait too long in order to actually see the visualization, he/she may get frustrated and give up on it. Thus, we performed an evaluation of the execution time for applying the most processor-intensive visualization mechanisms: colouring and resizing.




```
"actors": [
{
  "id": "0459ace2-8ece-4583-9df0-
a9473b3fee31",
  "text": "Medi@",
  "type": "istar.Actor",
  "x": 65,  "y": 29,
  "nodes": [
    {
      "id": "d1df4328-b44e-426c-aa42-
d8f8092d00fa",
      "text": "Internet Shop\nManaged",
      "type": "istar.Task",
      "x": 362, "y": 84,
      "customProperties": {
        "Cost": "70",
        "Benefit": "30",
        "Risk": "15"
      }
    }
...
```

Fig. 7. JSON object representing an excerpt of an *i** model

The largest model presented in a collection of goal modelling showcases [20] has approximately 493 elements. In a study that analysed the complexity of goal models, the larger example that was found contains approximately 200 elements [14]. Considering a safety margin of 100 percent, in order to perform this experiment we created random models with one thousand elements.

We developed a procedure to create a model with 1000 elements, where the kind of each element (goal, quality, task, and resource) is assigned randomly using JavaScript's default random function. Each element also has an associated benefit value, ranging from 1 to 100. Each experiment was repeated ten times on a fresh browser tab, in order to reduce interference from factors such as operating system tasks and browser's garbage collector. Additionally, a new model was created for each repetition of the experiment, in order to reduce the number of confounding factors.

The average execution time for colouring (i.e., changing the colour of every element according to the value of one of its attributes) was 7269.88 milliseconds, with a standard deviation of 292.84. This time includes not only the processing time (which decides which colour to apply), but also the time required for the browser to actually update the colour of each element.

Resizing (i.e., changing the size of each element according to the value of one of its attributes) presented an average execution time of 2466.72 milliseconds, with a standard deviation of 134.24.

Fig. 8 shows the execution time for each repetition of the experiment. The experiment was run on a 64-bits machine, with an Intel i5 2.2GHz processor, running Microsoft Windows 10, using the Google Chrome web-browser.

Even though experimentation with real users is required in order to assess if these execution times are satisfactory, we believe that taking 7269.88 milliseconds for very large models will not compromise their workflow. Furthermore, subsequent colouring and resizing that would be

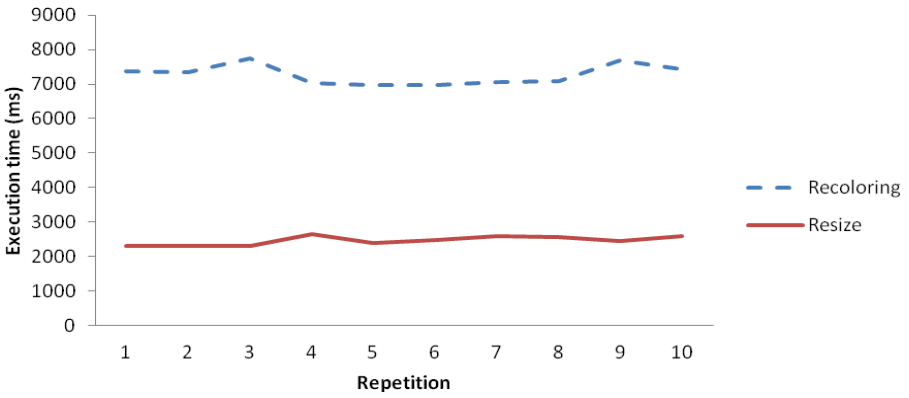


Fig. 8. Execution time for each repetition of the experiment

expected on iterative scenarios are expected to take less than one second, since the browser skips the re-colouring of any given element if the new colour is the same as the old colour.

5. Related Work

In this section we discuss related work, focusing on goal-based approaches for requirements prioritization or release planning. The work of Liaskos et al. [19] presents a proposal to use the Analytic hierarchy process (AHP) with *i** models, aiming to obtain quantitative indicators of softgoal satisfaction. Whereas Liaskos work adopts a prioritization technique to elicit new information, our proposal is different since it aims to visually represent prioritization information.

Horkoff and Yu [15] propose an approach to select alternatives on *i** models, based on desired levels of satisfaction of softgoals. This is achieved with an algorithm that, when unable to find an optimal solution, asks the user for additional information. Horkoff and Yu's work focuses on selection, more specifically on decision making, not on prioritization. Both [19] and [15] can be used together with our proposal, as follows. Liaskos et al. [19] enables the elicitation of precise values for contribution links, which can aid stakeholders when prioritizing the elements of the *i** model. On the other hand, stakeholders can use the visualization strategies here proposed in order to inform the decision-making of Horkoff and Yu [15].

The work of Kassab [17] demonstrates the application of a decision-making technique, AHP, along with the NFR Framework [9]. It aims at supporting trade-off analysis of the operationalizations of non-functional requirements, in the context of quantitative approaches. Such a technique can provide the raw data that will be visualized with our proposal.

The purpose of Regnell et al. [24] is to provide support for combinatorial optimization in the requirements engineering domain, adopting reqT [25] as the base requirements language. Using constraint satisfaction programming (CSP) and an automated solver, that proposal allows calculating values, to find solutions to constraints, and to perform queries on requirements models. It provides examples of requirements prioritization, release planning, and product line modelling. Whereas Regnell's approach focus on processing support for automatic evaluation, our proposal focuses on visual support for manual analysis.

Aydemir et al. [3] propose a goal-based approach for addressing the next release problem – that is, selecting which requirements to implement next, from a set of candidate requirements. They extend traditional goal trees with a set of inter-dependency constraints, such as REQUIRES (R1 requires R2 to function) and TEMPORAL (R1 needs to be implemented before R2). Such extension can be helpful for the analysis of release planning, representing a promising venue for future work.

6. Conclusion

In this paper we present a set of strategies to support the visualization of prioritization-related information and the use of said information in the context of release planning. Combining the expressiveness of *i** models with the proposed strategies for visualizing prioritization and release planning attributes, the proposal potentially provides rich capabilities for grasping, analysing, reviewing, and updating prioritization information, furthermore facilitating the decision-making process of release planning.

The approach is supported by a web-based tool, which has been evaluated for performance. The customization of the application of visualization strategies is a strong point of the tool, allowing for the use of different prioritization criteria, prioritization approaches, and visualization preferences. However, the large number of options may be daunting for some users. Thus, as future work we expect to perform empirical studies and identify the most effective combinations of visualization strategies, analysing their impact on the cognitive load of viewers of the model.

We also plan to analyse the use of additional inter-dependencies as discussed in [3], in order to provide even more information for the viewers of the models. Lastly, we plan to expand the visualization strategies beyond the scope of requirements prioritization and release planning – e.g., for tracking software development in terms of elements to do/doing/done.

.

Acknowledgements

This work was partially financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE).

References

1. Abad, Z. S. H., Noaeen, M., Ruhe, G. (2016). Requirements engineering visualization: a systematic literature review. In 2016 IEEE 24th International Requirements Engineering Conference (RE) (pp. 6-15). IEEE.
2. Achimugu, P., Selamat, A., Ibrahim, R., Mahrin, M. (2014). A systematic literature review of software requirements prioritization research. Information and software technology, 56(6), pp. 568-585.
3. Aydemir, F. B., Dalpiaz, F., Brinkkemper, S., Giorgini, P., Mylopoulos, J. (2018). The next release problem revisited: a new avenue for goal models. In 2018 IEEE 26th International Requirements Engineering Conference (RE) (pp. 5-16). IEEE.
4. Beck, K., Fowler, M. (2001). Planning extreme programming. Addison-Wesley Professional.

5. Berander, P., Andrews, A. (2005). Requirements Prioritization. Engineering and Managing Software Requirements.
6. Bukhsh, F. A., Bukhsh, Z. A., Daneva, M. (2019). A systematic literature review on requirement prioritization techniques and their empirical evaluation. Computer Standards & Interfaces.
7. Card, Stuart K., Mackinlay, J, Shneidermab, B.: (1999) "Readings in information visualization using vision to think". San Francisco.
8. Cavalcanti, C., Lencastre, M., Fagundes, R., Santos, T., Ferreira, D. (2018). Mechanisms to Support Requirements Prioritization: A Systematic Mapping Review. 21st Workshop on Requirements Engineering, WER 2018.
9. Chung, L., Nixon, B. A., Yu, E., Mylopoulos, J. Non-functional requirements in software engineering (vol. 5). Springer (2000).
10. Cooper Jr., J. R., Lee, S.-W., Gandhi, R. A., Gotel, O. (2009). Requirements Engineering Visualization: A Survey on the State-of-the-Art. Fourth International Workshop on Requirements Engineering Visualization (REV), p. 46–55.
11. Dalpiaz, F., Franch, X., Horkoff, J.: iStar 2.0 language guide. In: arXiv preprint arXiv:1605.07767 (2016).
12. Flório, C., Lencastre, M., Pimentel, J., Araujo, J. (2019). iStar-p: A Modelling Language for Requirements Prioritization. In International Conference on Conceptual Modeling (pp. 540-548). Springer.
13. Gotel, O. C., Marchese, F. T., Morris, S. J. (2007). On requirements visualization. In Second International Workshop on Requirements Engineering Visualization (REV 2007) (pp. 11-11). IEEE.
14. Gralha, C., Araújo, J., Goulão, M. (2015). Metrics for measuring complexity and completeness for social goal models. Information Systems, v. 53, p. 346–362.
15. Horkoff, J., Yu, E.: Finding solutions in goal models: an interactive backward reasoning approach. In International Conference on Conceptual Modeling (pp. 59-75). Springer, Berlin, Heidelberg (2010).
16. Jantunen, S., Lehtola, L., Gause, D. C., Dumdum, U. R., Barnes, R. J. (2011). The challenge of release planning. In 2011 Fifth International Workshop on Software Product Management (IWSPM) (pp. 36-45). IEEE.
17. Kassab, M.: An integrated approach of AHP and NFRs framework. In: Proceedings of the 7th International Conference on Research Challenges in Information Science. IEEE (2013).
18. Lencastre, M., Pimentel, J. (2019). A Metamodel for iStar-p: Requirements Prioritization with Goal Models. iStar Workshop 2019. Co-located with the International Conference on Conceptual Modeling.
19. Liaskos, S., Jalman, R., Aranda, J. (2012). On eliciting contribution measures in goal models. 20th IEEE International Requirements Engineering Conference, RE 2012 - Proceedings, p. 221–230.
20. Maiden, N., Yu, E., Franch, X., Mylopoulos, J. (2011). iStar Showcase' 11. iStar Showcase 2011.
21. Moody, D. L. (2009). The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering, v. 35, n. 6, p. 756–779.
22. Pimentel, J., Castro, J. (2018). piStar Tool – A Pluggable Online Tool for Goal Modeling. In: 26th IEEE International Requirements Engineering Conference (RE), Canada, 2018, pp. 498-499.
23. Pimentel, J., Castro, J., Ribeiro, M., Souza, A., Ramos, B. (2019). Creating Modelling Tools for i* Language Extensions. iStar Workshop 2019. Co-located with the International Conference on Conceptual Modeling.
24. Regnell, B., Kuchcinski, K.: A scala embedded DSL for combinatorial optimization in software requirements engineering. In: First Workshop on Domain Specific Languages in Combinatorial Optimization, pp. 19-34 (2013).
25. Regnell, B.: Reqt.org – towards a semi-formal, open and scalable requirements modeling tool. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 112-118. Springer, Berlin (2013).

26. Riegel, N., Doerr, J. (2015). A systematic literature review of requirements prioritization criteria. In: International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 300-317). Springer.
27. Wiegers, K., Beatty, J. (2013). Software Requirements. 3rd. ed. Pearson Education.
28. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J. (2011). Social modeling for requirements engineering. Mit Press.