Proposal: Separating Requirements from Goals and Design Decisions

RYAN DARRAS

Too often developers have a list of "requirements" dropped on them that isn't actually a list of requirements. Developers have to go through the process of eliciting the requirements themselves (whether through assumptions or pestering stakeholders/project managers) in order to being working on the project. This is due to elicited requirements often being goals or design decisions that don't dictate exactly what the project must do in order to be considered complete. By intentionally separating requirements from goals and design decisions, project managers and stakeholders will be able to more accurately create requirements that can be confidently passed to developers so that they can begin development.

ACM Reference Format:
Ryan Darras. 2018. Proposal: Separating Requirements from Goals and Design Decisions. 1, 1 (November 2018), 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the related literature, "requirements" has been defined in ways. *"requirements specifications state the desired functional and performance characteristics of some component independent of any actual realization"* [6] emphasizes that feature requirements don't actually exist yet. *"requirements are usually understood as stating what a system is supposed to do, as opposed to how it should do it"* [8] differentiates requirements from design decisions. *"a statement that identifies a capability or function that is needed by a system in order to satisfy its customer's needs"* [1] focuses on a single feature's functionality.

These definitions of "requirements" can often be considered synonymous with "goals" or "design", which can cause confusion when in the requirements elicitation processes. Author Paul Ralph describes the difference between requirements, *"a structural or behavioral property that a design object must possess"* [4] and goals/design decisions, *"goals describe the desired impacts of a design object on its environment"* [4] which we will use to dictate the structure of our proposed solution in order to obtain effective requirements.

Our tested environments will assume an overarching *project* for which stakeholders and project managers will develop *goals*. Similarly, for each goal the project managers will elicit a collection of *features* from the stakeholders. *Features* will be broken down into two categories; *required features* are features that must be implemented for the project to be considered complete, and *design features* are features that determine the design aspects of the project such as the components of UI and UX. For example, assume the client hired a team of developers to create a space simulation (the project) that will allow researchers to study the effects of extraterrestrial objects on theoretical man-made shuttles (the goal). The client states that this application must use accurate physics and allow for the user to simulate extraterrestrial objects of various characteristics so that they can simulate a broad range environments in which to research (the required features). The client also states that, while being astrophysicists and astronomers, the users of this program are not incredibly tech savvy and the program must be easy to learn and operate (design features).

Author's address: Ryan Darras.

XXXX-XXXX/2018/11-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 EVALUATION METRICS

2.1 Accuracy

This is ultimately the big-ticket metric that we want to strive for. When comparing our results, to the opinions of the developers/project managers assigned to the project we want them to agree with at least 90% of the modifications we made to the requirements.

2.2 Robustness

In the accuracy section, we stated a goal of 90% agreement among the developers and project managers. However, we don't want to have this goal be met for one specific set of requirements matching a single style of project. We want to meet that goal for every single set of requirements possible.

2.3 Accessible from all levels

By separating goals and design decisions from requirements we create a more clear development path to follow. We want this path to be clear to stakeholders, developers, and project managers alike so that at the end of the requirements elicitation phase everyone is on the same page. our updated requirements will be formatted as such that even a non-technical individual will be able to create a similar vision of the project that a developer would create.

3 EXPERIMENTS

We will start by creating a classification that will determine whether or not a stated requirement has characteristics of a goal or design decision. Given a generic set of requirements, we will determine how many need to be broken down into actual requirements, as they currently display goal or design decision traits. We will then evaluate and improve this initial goal reduction strategy over a couple of iterations to improve the accuracy on these generic sets.

After we have fine tuned the goal reduction strategy, we will put it up against more complex sets of requirements and test how it fairs with the opinion and development methodologies by real world software companies. If we run the same set of tests by 10 different companies, we can generate results that can be used to evaluate the success of our original goal reduction strategy.

4 RELATED WORK

There is a ton of related work in the area of requirements elicitation; documented below are a few that focus on requirements vs goals vs design decisions.

Gonzalo Génova and his team focus on quantifying requirements so that they can be measured in a fully automatic way in order to improve the quality of requirements as well as improve the writing skills for the requirements engineers [3]. They claim that trying to obtain all of these measurements by human hand would be impossible to do accurately, which is why they suggest using automated tools to collect this data. However, they mention that forcing these metrics on the requirements engineers isn't the best option as it would feel like a policing tool to prosecute or penalize the writers. To bypass this problem, the authors suggest a gradual implementation of automated tools to measure the quality of requirements such that the requirements engineers see the potential benefit of these tools as opposed to the oppressive penalization they might see.

Author Axel van Lamsweerde takes the approach of requirements elicitation with emphasis on goal-oriented requirements [7]. As the first sentence in his abstract he defines goals as, "Goals capture, at different levels of abstraction, the various objectives the system under consideration should achieve." Throughout the paper, the author describes how to create and understand goals that are effective in the requirements engineering process. The author summarizes the benefits to goal-oriented requirements engineering as follows:

- object models and requirements can be derived systematically from goals;
- goals provide the rationale for requirements;
- a goal graph provides vertical traceability from high-level strategic concerns to low-level technical details; it allows evolving versions of the system under consideration to be integrated as alternatives into one single framework;
- goal AND/OR graphs provide the right abstraction level at which decision makers can be involved for important decisions;
- the goal refinement structure provides a comprehensible structure for the requirements document;
- alternative goal refinements and agent assignments allow alternative system proposals to be explored;
- goal formalization allows refinements to be proved correct and complete.

A. Terry Bahill researches methods to elicit requirements in effective ways [1]. This author explicitly compares requirements vs goals by stating that, "The Mission Statement, Concept of Operations, and Business Model contain goals, objectives, features, and capabilities. Formal requirements are contained in the Specific Requirements Sections of the Use Cases, Supplementary Requirements Specification, and Tradition Requirements Specification." Although requirements and goals are critical parts to a successful project, they must be treated as separate entities to get the job done effectively.

Author Paul Ralph focuses directly on preventing the creation of statements labeled requirements, when these statements are in fact goals or design decisions [5]. The author considers the idea that a developer or team member can't be effective when their requirements are stated as goals. Instead, the developers should be given explicit requirements that don't elicit additional questions. Similarly, in another paper the author proposes a formal definition for the design concept which incorporates seven elements: agent, object, environment, goals, primitives, requirements and constraints [4]. By linking these concepts together in an appropriate way, the authors proposed method should allow stakeholders and developers to classify design knowledge and classify design approaches.

Pierre Bourque authored a book that discusses, in detail, software engineering which encapsulates the processes of requirements elicitation [2]]. The author defines what a software requirement is and gives examples of software requirements. This book is a collection of useful information that can be used to review definitions and opinions of various software engineering areas.

Eric Yu focuses on the early stages of requirements elicitation as opposed to existing requirements techniques that are intended for the later phase of requirements engineering, "which focuses on completeness, consistency, and automated verification of requirements." Instead, the author sights in on the organizational context and rationals (the "Whys") that lead to the formulation of system requirements. By practicing and perfecting the early stage of requirements engineering, you can insure greater success in the overall outcome of the requirements.

5 BACKGROUND WORK

Bourque discusses many different types of requirements elicitation techniques and breaks them down into core principals [2]. These principles consist of Interviews, Scenarios, Prototypes, Facilitated meetings, Observation, and User stories but he also mentions that many other techniques exist. This paper will consider these six core principles to build a framework similar to that of Ralph's formal definition for the design concept [4]. By building a graph of Project -> Goals -> Features -> Required Features and Design Features -> Dependencies (other features) we can create a framework to utilize the six core principles to populate this map in such a way that allows for automated verification of requirements and insure a fully complete list of requirements that will take the project from start to finish.

6 RESEARCH PLAN

6.1 Challenges

6.1.1 Collecting Effective Sample Sets. The sample data used when segregating requirements from goals and design decisions is a huge factor on determining the success of this research. Sample sets should be based on projects that are convoluted enough to make this problem interesting so that we can generate data and results for said sample.

We also want a relatively broad range of problems included in our samples. We want sets of requirements that range from near perfect, to ultimately worthless so that we can test the effectiveness of our strategy and finding the right sample sets to use is going to be challenging.

6.1.2 Robustness. Our system should be able to handle any set of requirements and break them down equivalently. This means that for any given set of requirements, our proposed system will accurately detect goals and design decisions and flag them for reconsideration. Considering these requirements are man-made entities, the possibilities that we will need to evaluate are endless which is why our solution needs to be as robust as possible.

6.2 Timeline

6.2.1 Phase 1. will be all about devising a plan and discussing strategies on the topic. During so, research on related work will be conducted to find any similar applications that can be applied or combined to provide a useful solution. Very small scale tests will be performed on small sets of requirements (5-10) to calculate results.

6.2.2 Phase 2. will be all about real world implications. Testing our proposed solution on real sets of requirements and collecting data and feedback from developers and managers to determine any flaws in the system.

6.2.3 Phase 3. will be the refinement phase where we take what we have learned in Phase 2 to adjust our solution to produce better results.

7 CONCLUSIONS AND FUTURE WORK

Requirements elicitation is very hard. It takes up incredibly large amounts of time to generate them in the first place, but it also comes down to developers and project managers finding ambiguousness in the requirements that requires them to go to the stakeholders for more answers. While this proposed solution will not fix the entirety of the requirements elicitation field, we believe it will improve the satisfaction and performance of developers which will result in a product that more accurately fits the stakeholders vision. By the end of this research, we want to have full analytics and a guide on how other companies can implement these strategies into their own techniques.

We will develop one way to separate goals and design decisions from requirements, but the possibilities are endless to evolve this method to work in various different requirement elicitation techniques.

REFERENCES

[1] A. Terry Bahill, *Discovering system requirements*, In: Sage AP, Rouse WB (eds) Handbook of Systems engineering and Management. 2nd edn. (2009), 205–266.
[2] Pierre Bourque, *Guide to the software engineering body of knowledge (swebok)*, IEEE Computer Society Press (2004).
[3] Gonzalo Génova, José M. Fuentes, Juan Llorens, Omar Hurtado, and Valentín Moreno, *A framework to measure and improve the quality of textual requirements*, Requirements Engineering **18** (2013), no. 1, 25–41.
[4] Paul Ralph, *A proposal for a formal definition of the design concept*, In: Lyytinen K, Loucopoulos P, Mylopoulos J, Robinson W (eds) Design Requirements Engineering: A Ten-Year Perspective. Lecture Notes on Business Information processing **14** (2009), 103–136.
[5] Paul Ralph, *The illusion of requirements in software development*, Requirements Engineering **18** (2013), no. 3, 293–296.
[6] Gruia-Catalin Roman, *A taxonomy of current issues in requirements engineering*, Computer **18** (1985), 14–23.

[7] Axel van Lamsweerde, *Goal-oriented requirements engineering: A guided tour*, In: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering (2001), 249–262.
[8] Eric Yu, *Towards modelling and reasoning support for early-phase requirements engineering*, In: Proceedings of the Third IEEE International Symposium on Requirements Engineering (1997), 226–235.