Modeling Non-Functional Requirements of a Reactive System

Jorge A. Salcedo, Manuel H. Velarde, Alexander J. M. Montalvo, Claudia L. Rojas, Milagros Yarahuaman, Luis A. Araujo, Roxana L. Q. Portugal

Universidad Nacional de San Antonio Abad del Cusco, Cusco, 08001, Perú.

Abstract

An understanding of Non-functional Requirements (NFRs) is important for designing a software system, however, time and resource constraints, usually, lead to systems being developed mostly from a functional perspective. We explore the case of using programming libraries as a support to the design of software systems explicitly considering NFRs. We tackle the case of the React JS library, within the context of reengineering a Web based e-commerce software. This library operationalizes a set of NFRs needed for a system to be reactive. We abstracted these implementations as softgoals to derive an i* model with the NFRs made explicit. The resulting model, created collaboratively, is an example of using both functional and qualitative perspectives in designing a software system.

Keywords

Non-functional requirements, Reactive Systems, Front-End, E-Commerce

1. Introduction

Jagadeesan et al. propose that reactive systems are those that give a continuous response to stimuli from their environment, both the processing and the results obtained are driven precisely by the inputs given by their environment [1]. In the case of web systems, determining the stimuli that occur at a certain moment, for a user, becomes a challenge because the combination of software and hardware used cannot be defined. To develop a reactive web system, we may adopt a base architecture to implement them in two layers: Front-end, which has the user interface, and Back-end, which considers functionality and management on data [2]. At the Front-end level, various libraries and frameworks have been created, such as React JS [3], whose purpose is to deal with the stimuli of a user.

Reactivity appeared to us as an NFR (Non-functional Requirement) [4] during an ecommerce project [5]. In this project, we reused code from Yin's GitHub [6]. Through the analysis of Yin's architecture, we delve into the React paradigm, which led us to various sources of information such as the Reactive Manifesto [7], The Reactive Design Patterns [8], and React JS' own documentation [3]. The cited literature uses the NFRs: responsive, resilient, elastic and message-oriented, among others, to *satisfice*[2] [4] a Reactive System.

This work draws from these information sources to elicit Reactivity as an NFR, which are modeled for the e-commerce application (Figure 1). With the focus of linking the information sources on React with the Yin's code [6], we modeled, using a viewpoint strategy [9][10], the e-commerce project as to satisfice Reactivity.

Our article is structured as follows. Section 2 briefly cite related work on reactive approaches. Section 3 establishes the objectives of our investigation. Section 4 details the the development of the

Proceedings of the 14th International iStar Workshop, October 18-21, 2021, St. Johns (NL), Canada
EMAIL: 171572@unsaac.edu.pe (J. Salcedo); 171918@unsaac.edu.pe (M. Velarde); 171914@unsaac.edu.pe (A. Montalvo)
ORCID: 0000-0001-8467-2004 (J. Salcedo); 0000-0002-9903-1450 (M. Velarde); 0000-0001-5362-4317 (A. Montalvo)

[2] Term coined by Herbert Simon: Nobel Prize in Economics, and ACM Turing Award recipient. Source: Wikipedia

i* model, which was built using the iStar 2.0 tool, piStar[3]. Finally, in Section 5 we present our conclusions about the work done as well as the future research.

2. Related Work

To the best to our knowledge, reactivity is not being dealt from the NFR framework [4] point of view, which is building a software application with NFRs as first-class requirements. Existing work is focused on concurrent and distributed systems that uses a Rebeca modeling language for formal verification [11][12]. Other works focus is on safety-critical embedded systems which being safety-critical systems uses reactive modeling approaches such needs the aware contract language CoCoSpec [13] and ScenarioTools [14].

Regarding existing words on i*, we found that our work agrees with [10][15][16] in the use of collaboration as an important technique for better requirements engineering, particularly for a modeling that represents a common vision.

3. Research objectives

O1. React as an NFR can be operationalized by using React JS.

O2. Organize NFRs and Operationalizations existing in sources of information related to Reactivity using an intentional modeling.

O3. Create dependency relationships between the Reactive capabilities of React JS and the actor's responsibilities identified for the e-commerce application.

4. Work progress and contributions

According to Priola [17] when we are in a subjective research, we can perform an explanatory research to understand the relationships existing on diverse aspects of an issue. As our work uses viewpoints [9] as a mechanism to perform a collaborative model, then we are dealing with a subjective task which shows personal perspectives and interpretations, in our case, we based on examinations of documents [3][7][8]. Following we detail the process to achieve a consensus on the React JS library actor we modeled to meet the reactivity NFR and other NFRs that contributes with it, as well as operationalizations (tasks).

4.1. Identification of Qualities and Goals

Using viewpoints [9][10] for the understanding of the Reactivity quality, we carried out nine different individual elicitation based on the mentioned sources [3][7][8], this individual work was given to allow each elicitor to create his/her model. Of the nine, we verified that only seven shared the main qualities in the React Manifesto [7]. Some of these i* models are shown in the project repository [18].

To make a consensus model, we collaboratively [10] elaborated two relationship maps, the first links elicited qualities to the main qualities of the Manifesto (Table 1). The second links project goals to the elicited qualities (Table 2).

In order to build a unique model, we established a threshold for a consensus, that is, there is a consensus if at least three authors share the same point of view. Then, the relations among qualities with help (1) in table 1, were not considered.

[3] https://www.cin.ufpe.br/~jhcp/pistar/tool/

Table 1. Consensus for Qualities

Qualities / Qualities	Reactivity	Resilience	Responsiveness	Elasticity
Responsiveness	help (7)			
Elasticity	help (7)		help (5)	
Resilience	help (7)			
Containment		help (7)		
Declarativeness			help (5)	
Replication		help (6)		
Scalability				help (1)
Message Oriented	help (7)	help (4)	help (3)	help (5)
Delegation		help (7)		
Isolation		help (7)		
Safety			help (1)	
Interactivity			help (1)	
Usability			help (1)	

Table 2. Consensus for Goals

Goals/ Qualities	Elasticity	Message Oriented	Delegation	Isolation	Containment	Replication	Declarativeness
That the components are rendered							help (3)
That the resources can be managed	help (3)						
That the states of the components be handled		help (3)	help (4)				
That the operation is asynchronous		help (5)		help (5)			
That component reuse is allowed					help (5)	help (4)	

4.2. Task Identification

For the identification of tasks, we use as a mechanism to set the goals as questions and then look for the tasks that give answers. Reactive Design Patterns [8] was chosen as a source of information, since the tasks should give us more concrete solutions, and we could relate them to the e-commerce project [19] code. The elicitation and modeling of tasks was performed in meetings where we, collaboratively [10], defined the tasks for the goals previously identified.

For example, for the goal called "Have the inputs of the services rendered", the tasks identified in the design patterns were "Get the inputs" and "Create a composition of inputs". This means, when our system receives inputs which will mean a change for it, through a composition, will perform an update of the active component, avoiding updating everything together, thus optimizing resources and time. The identification of the other tasks in Fig. 1 followed the same logic, but for better understanding, we summarized the concepts [20] needed for the reading of model in Fig. 1.

4.3. Resource Identification

As we identified the tasks, some of them required resources. In particular, we were able to link tasks found in the Design Patterns [8] to components reusing the React JS [3] code.

This is the case, for instance, for the task "Create a composition of inputs" that refines the goal "Have the inputs of the services rendered". The "Create a composition of inputs" is operationalized by the

component React DOM as per the React JS library [3]. This relation is modeled in the i* model (Fig. 1) as a Needed by relationship.

Other resources came from our project artifacts [5], e.g., a topic modeling.

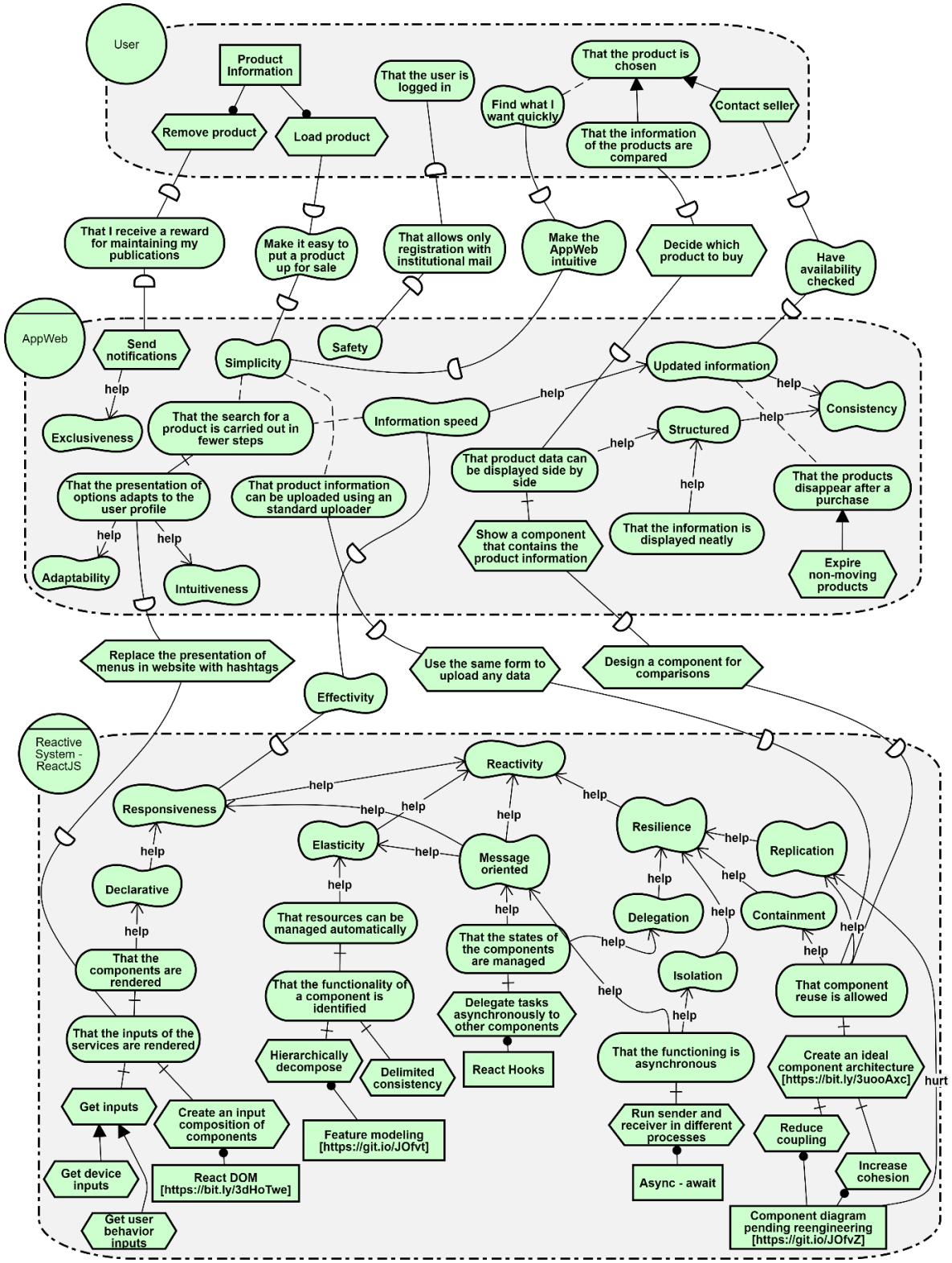

Figure 1. Reactive e-commerce i* model

4.4. Final model

After carrying out the previous steps, we elaborated an i* model of reactivity (Fig. 1), which we propose as the main contribution of this work. As can be seen, the collaborative work performed in section 4, was developed for the agent Reactive System JS. Later, we retake our work on reengineering the ecommerce application, as such, we modeled the actor User and the agent AppWeb with its goals. Finally, we tested the validity of the agent Reactive System JS by finding the dependences among the ecommerce actors and the operationalizations identified for the React JS library agent.

In this model most of the relationships are of the "help" type, which it was influenced on what is indicated in the Reactive Manifesto "we want systems that are Responsive, Resilient, Elastic and Message Driven. We call these Reactive Systems" [7]. As we worked in a consensus of models, we couldn't update relations, such as adding a hurt relation, or new qualities that came up in the meetings we had to identify operationalizations such as the tasks. It is important to note that some of the resources *Needed By* tasks inside the React JS boundary, are our own artifacts, we use them to verify if we meet the React JS NFRs.

5. Conclusions

Comparing with other works in reactivity, we can say that this work is novel in explicit the NFRs proposed by reactive libraries such as React JS [3], Angular [21], Vue JS [22]. For the objectives we set in this work, O1: "*React as an NFR can be operationalized by using React JS*." we found through our process in section 4 that different sources of information can be complemented to identify NFRs with its operationalizations (tasks). For the objective O2: *"Organize NFRs and Operationalizations existing in sources of information related to Reactivity using an intentional modeling."*, we found that i* helped us to model different tasks or resources as a means to achieve the React NFRs. Finally, for the objective O3: *"Create dependency relationships between the Reactive capabilities of React JS and the actor's responsibilities identified for the e-commerce application"*, we show in Figure 1, that we were able to create strategic dependencies (SD) between the actors' responsibilities of the e-commerce application we modeled and the React JS agent.

It is worth nothing that this model can be reused for the ones interested in understand the reactive paradigm of programming frameworks such as React JS, Angular, and Vue, and the ones dealing with the ecommerce domain and that they can also find the NFRs that can be meet.

The model produced complies with the notation rules specified for i* version 2.0 [23] and using the tool piStar [24].

Despite the positive results, we found our model is initial as it has mostly help-type contribution relationships, none of the seven points of view identified another type of contribution; we believe that this is due to the need for more time to analyze the sources of information or experience with the React Framework to determine negative impacts.

A threat to the validity of the model we had mitigated is that, for each author model, we asked in meetings to present evidence of the traces to the sources of information used.

We plan the development of a new project, in order to enrich the task and goals needed to meet the Reactivity qualities identified.

6. Acknowledgements

We thank our colleagues Luis Flores Aquino, Rosmel Deza Condori and Elizon Carcausto Mamani, for providing us with their material and their comments whenever we request. We also thank Professors Roxana Quintanilla Portugal and Julio Cesar Sampaio do Prado Leite for their guidance and their enthusiasm in carrying out this article.

7. References

[1] JAGADEESAN, L. J., Porter, A., PUCHOL, C., RAMMING, J. C., & VOTTA, L. G. (1997, May). Specification-based testing of reactive software: tools and experiments: experience report. In Proceedings of the 19th International Conference on Software Engineering (pp. 525-535).

[2] IBM Entorno de aplicaciones web. Available at: https://www.ibm.com/docs/es. Last Access: 05-04-2021

[3] REACT DOCS. Getting Started on React. Available at: https://reactjs.org/docs/getting-started.html. Last Access: 07-22-2021

[4] CHUNG, L., NIXON, B.A., YU, E., and MYLOPOULOS, J., 2000. Nonfunctional requirements in software engineering. Springer Science & Business Media New York

[5] UWUNSAAC. (2021). https://github.com/UwUnsaac/Ecommerce-Artifacts

[6] YIN, Z. Ecommerce-Reactjs. Available at: https://github.com/levelopers/Ecommerce-Reactjs

[7] JONAS B., ROLAND K., MARTIN T. The Reactive Manifesto 2014, Available at: https://www.reactivemanifesto.org/en

[8] KUHN R., HANAFEE B. ALLEN J. 2017. Reactive Design Patterns. Available at: http://manning-content.s3.amazonaws.com/download/f/79f0a30-f39b-4922-bf3b-67f8cd5f2be3/SampleCh02.pdf

[9] LEITE, J.C.S.P. Viewpoints on Viewpoints, In ISAW '96: Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints '96) on SIGSOFT '96 workshops, October 1996 Pages 285–288, https://doi.org/10.1145/243327.243682.

[10] PORTUGAL, Roxana LQ; DO PRADO LEITE, Julio Cesar Sampaio. Challenges in Modeling Non-Functional Requirements Collaboratively. In iStar@ ER. 2019.

[11] ALAVIZADEH, Fatemeh, et al. Using UML to Develop Verifiable Reactive Systems. In Software Engineering Research and Practice. 2006. p. 554-561.

[12] SIRJANI, Marjan, et al. Modeling and verification of reactive systems using Rebeca. Fundamenta Informaticae, 2004, vol. 63, no 4, p. 385-410.

[13] CHAMPION, A., GURFINKEL, A., KAHSAI, T., & TINELLI, C. 2016, July. CoCoSpec: A mode-aware contract language for reactive systems. In International Conference on Software Engineering and Formal Methods (pp. 347-366). Springer, Cham.

[14] GREENYER, J., GRITZNER, D., GUTJAHR, T., KÖNIG, F., GLADE, N., MARRON, A., & KATZ, G. 2017. ScenarioTools–A tool suite for the scenario-based modeling and analysis of reactive systems. Science of Computer Programming, 149, 15-27.

[15] MORALES RAMIREZ, Itzel, et al. Exploiting online discussions in collaborative distributed requirements engineering. En Eighth International i* Workshop, iStar 2015. 2015. p. 7-12.

[16] PANT, Vik; ERIC, S. K. Using i* to Analyze Trust-Building Strategies for Organizations under Coopetition. En iSTAR@ CAiSE. 2018.

[17] PRIOLA, Cinzia. Understanding Different Research Perspectives. 2016.

[18] UWUNSAAC (2021). https://git.io/Jlk32

[19] UWUNSAAC (2021). https://github.com/UwUnsaac

[20] UWUNSAAC (2021). https://git.io/JlN0W

[21] Introduction to the Angular Docs. Available at: https://angular.io/docs. Last Access: 08-20-2021

[22] Vue.js guide. Available at: https://vuejs.org/v2/guide/. Last Access: 08-20-2021

[23] DALPIAZ, FABIANO, FRANCH, XAVIER; HORKOFF, JENNIFER. istar 2.0 language guide. arXiv preprint arXiv:1605.07767, 2016.

[24] PIMENTEL, Joao; CASTRO, Jaelson. piStar tool–a pluggable online tool for goal modeling. In 2018 IEEE 26th International Requirements Engineering Conference (RE). IEEE, 2018. p. 498-499.