PRESENTATION TRANSCRIPT

INTRODUCTION

In this presentation we will preview how RE evolved and we will discuss an alternative perspective on the discipline of RE. We will also advocate the evolution of RE toward a discipline that is applicable to any domain. RE can be traced back to early origins to the techniques and methodologies of classic engineering. I am going to use, in the presentation, the term classic engineering as a synonym for what general public perceives as engineering. They are usually large-scale civil engineering projects. After the 1970s, Demand for large-scale software project management increased. This increment made many researchers turn their attention to applying classic engineering practices and techniques to managing software projects. So the first samples of software project management were full of failures, because of being used classic engineering practices to managing software projects. Over time, practitioners and researchers began to realize that the reason of many of the problems is the deficiencies in the requirements engineering. Because the requirements for the physical entities focused upon the structures, but software is something intangible, so you cannot touch it. So applying engineering practices and principles to software development was complex, challenging and it was a completely new experience.

REQUIREMENTS ENGINEERING

RE practices and researches were affected by the incredible diversity of the domains besides software development. But the vast majority of requirements engineering application remains focused on Software-Intensive-System (SIS), which is the systems that software plays a significant role. In this paper, we are going to argue that requirements engineering is not only for Software-Intensive-Systems.

WHAT IS REQUIREMENTS ENGINEERING?

One of the first engineering author-professors once made this definition in classroom for engineering: "The historical record demonstrates that we learned everything that we need to know about engineering a very long time ago; consider what it took to build the Great Wall of China or the pyramids in Egypt and in many locations in South America. Since then, all we have done is become better at every aspect." Actually the professor wanted the students to understand that we were standing on the shoulders of all the people before us. When we look at the history of engineering, the difference between then and now is refinement, not in substance. I mean, we still have to carefully consider what we are attempting, we still have to identify what we are going to do, we still have to determine what we need as the resources, we still have to manage the whole process etc. Okay, but what was the fundamental message that professor wanted to give to the classroom. The fundamental message is that critical thinking, a methodical approach and human resources management are fundamental elements of all successful engineering attempts or endeavors. As for requirements engineering, there is a definition oft-cited from a researcher called Zave: "Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families." This definition is made in 1997 and in the ensuing years, a group of researchers defined the core RE activities as elicitation, modeling and analysis, communication, agreeing (negotiation) and evolving (maintenance and management). Together with these core activities of requirements engineering, in 2007 Future of Software Engineering Conference, another few researchers built their research upon this definition to investigate research directions in RE. They note that RE is difficult "because requirements reside primarily in the problem space whereas other software artifacts reside primarily in the solution space… Stated another way, requirements engineering is about precisely defining the problem that the software is to solve (i.e., defining what the software is to do)…"

ANOTHER PERSPECTIVE ON REQUIREMENT ENGINEERING

Let's give another perspective on requirements engineering. As you know, textbook definitions of engineering are very useful for educational purposes,

but they do not reflect all aspects of the practice. Because engineering textbooks are designed to be general or generic. For example, the perspective of engineering textbooks does not contain the societal constrains upon any engineering practices. So we need a definition for engineering that more accurately represents the actual usage: "**Engineering** is a body of practice that, when applied to the creation of a solution to a problem, ensures that the resulting solution artifacts will function as intended, with high probability, while meeting the constraints imposed by the problem stakeholders and the society within which the solution is initially deployed." Following the same approach, we can put forward a new definition for requirements engineering practice: "**Requirements Engineering** is a body of practice that, when applied to the identification and definition of a problem, its stakeholders and their constraints, improves the probability that the results accurately represent the problem and the solution constraints and that the results are presented in a manner that facilitates communication between stakeholders." As you can see, these definitions make explicit the assertion/claim that requirements engineering practice increases the probability of success when properly applied. Yeah, we are talking about the requirements engineering, so you may ask that what happened to the word requirements. There is a new perspective for requirements as well in the paper: "**A requirement** is a stakeholder constraint upon the problem space and/or the solution space." This alternate perspective identifies a problem space, a solution space, and the constraints upon the solution which means requirements.

RE FOR SIS AND NON-SIS DOMAINS

So far, we have talked about engineering, requirements engineering and put forward the new perspectives for them. Now, in the light of these new perspectives, we are going to argue a more general model for requirements engineering practice to support it in software-intensive-systems and non-software-intensive-systems domains.

Figure 1. Generic RE Practice

This is our new generic requirements engineering practice model. As you can see, the requirements engineering process is depicted as a cyclic iterations here. The iterations proceeds until the goals are achieved. The goals here are the problem statement and requirement specification.

Stakeholders: is about the people affected by the problem.

Wants and needs: is about elicitation, exploration and about identifying the core needs, including the goals and objectives.

Problem definition and refinement: is about the practical aspects of wants and needs, analyzing the stakeholder expressions and leading to the later definition of functional and nonfunctional requirements.

Constraints (Requirements): is about the necessary constraints upon the problem space and upon the solution space. Constraints such as budget, personnel and task-ordering should be considered as well.

Management: is about making the requirements engineering process work as needed, ensuring that the result is as close as possible to what was intended at beginning of the requirement engineering process.

CROSSING DOMAINS

When software aspects began to dominate system development and maintenance costs, the need to study the sociological and human aspects of system development became dramatically important. These trends helped requirements engineering better understand how to support the early phases of system development. The phases that I am talking about are problem identification and definition, refinement, maintenance and system phase-out or decommissioning. It is obvious that these phases can be applied to any domain that faces complex problems. Actually there are some samples even in the requirements engineering conferences that are utilized the requirements engineering phases in non-SIS domains.

RELaw – Requirement Engineering and Law: The interactions of laws, policies and standards with software and system requirements to understand better ways to manage compliance, accountability,

and traceability. Main contribution is the application of requirements engineering process to assist legisators to improve the quality of their legistlation.

RE Interactive, Creative Collisions: This initiative focused on interactions wherein the participants were randomly paired with the other participants for a 15-minute working session. Here the participants are from different domains and in the working sessions they mixed their knowledge and expertise to produce some results for creating a research problem statement and research mission statement (these results are called as collisions). Then what happened? The results amazed everyone who was there. Because the results were really productive. The main contribution of this workshop is the gamification of requirements engineering brainstorming and demonstrating that enforced interaction with someone who you did not know can produce unexpected results.

NEW FRONTIERS FOR RE

Throughout this work, we advocated that RE should expand its frontiers beyond SIS and RE practice and research should embrace new horizons. We have discussed alternative perspectives on requirements engineering and on requirements. These alternative perspectives support the expansion of RE beyond SIS. Okay but, how could be done? It could be done by using as a fundamental concept a problem space, a solution space, and the constraints both the problem and the solution space. I mean, industry practitioners should report back their experience related to the requirements engineering to the requirements engineering community and academic researchers should present the results of their work not just to RE community but also to the fields that they are inspired by.

CONCLUSION

Expanding the frontiers of requirements engineering beyond the development of SIS can only make RE a more effective practice with the potential for delivering significant benefits to other domains. This paper argued that requirements engineering is a discipline that can be applied to any problem scenario. However, if the software industry does not widely embrace requirements engineering practices, how does the other domains embrace? I think this is the greatest threat to this work.