Knowing what to do and when is what we need to do to survive large software projects

Wnuk, Krzysztof

Published in:
[Publication information missing]

2012

Link to publication

Citation for published version (APA):
Wnuk, K. (2012). Knowing what to do and when is what we need to do to survive large software projects. [Publication information missing], 1-2.
Knowing what to do and when is what we need to do to survive large software projects

Krzysztof Wnuk, wnuk@cs.lth.se

Successful projects require two key elements to be set: the time frame for the project and the scope of the project. Accurately deciding these two attributes greatly increases the probability of success. If, for example we would like to build a house, we should start building it in spring so that we can lie the foundations of the house on a non-frozen ground. Moreover, we know that we should be ready with the roof before the winter comes so that we can continue working inside the house. Knowing the time frame of our house project, we need to decide the actual size and shape of a house and other important elements. In other words, we need to decide the ‘scope’ of our house project.

The situation is similar for software projects, they require a good time plan and a clearly set scope. Helping to set the scope, or in other words identify what should be implemented in a software project, is one of the functions of Requirements Engineering (RE). RE helps to identify, specify, negotiate and validate requirements for software projects. However, selecting an optimal scope for large projects for open markets is a complex task where many things may change and the number of possible decisions is very large.

This thesis constitutes two main parts: the first part investigated methods to decrease the number of requirements in large projects and keep the complexity of requirements engineering down, the second part investigates the process of scoping and proposes a method to visualize and analyze scoping decisions. Both parts focus on supporting managing requirements and decision making in large software projects.

The first part of this thesis explore identifying and managing obsolete software requirements as a way to reduce the complexity in large projects. Just like a periodic cleaning of a garage or a storage room, periodic reviews of ‘old’ requirements could help keeping the total number of requirements down and keeping only the most current requirements. Almost 80% from over 200 industry participants of a survey about obsolete software requirements indicated that these requirements can have a serious negative impact on software projects. At the same time, only about 10% of respondents admitted having any automated methods of handling obsolete software requirements.

Redundant and similar information can also increase the complexity of requirements databases. Similarly to many of daily life examples, handling duplicates of information is counterproductive in managing requirements. However, finding duplicate candidates among several hundreds or thousands of requirements is a time consuming activity. Automated methods of linguistically analyzing requirements could speed up the process of duplicate identification and thus decrease their number in requirements databases. This thesis presents results from an experiment designed
to assess if linguistic tool support can help to find more similar requirements. The results confirmed that linguistic methods could help miss fewer duplicates but not necessarily help to find more duplicate requirements faster.

The second main part of the thesis is dedicated to supporting requirements scoping and understanding the causes and consequences of overscoping, in other words 'biting more than you can chew'. The situation of setting a too large scope of a project is a known risk in project management literature and is widely mentioned as a strong failure contributor of software projects. According to the results of one of the included articles, overscoping could be caused by continuous inflow of new requirements, lack of early involvement of developers and lack of a clear product strategy. Moreover, overscoping may lead to a number of potentially serious and expensive consequences, including quality issues, delays and failure to meet customer expectations.

The thesis introduces and empirically evaluates two techniques to visualize scope changes in large projects. The techniques show the lifecycle of several hundreds of features in a single picture that is colored and sorted according to the total time the features were considered in the scope. The visualizations allow to spot decision moments and analyze decision patterns in large projects. Further, the visualizations give a quick assessment of the size of the fraction of an initial scope that actually was implemented in the projects.

The results from the research efforts presented in this thesis uncover an interesting picture of large-scale requirements engineering for open markets. In this picture, the complexity of requirements engineering and associated decision making is driven by the uncertainty associated with decisions to be made. This uncertainty of decisions is associated with inherited characteristics of open markets with no direct customers, fast changing market needs and fierce competition. The solutions presented in this thesis emphasize the need for considering requirements as temporal characteristics that may have a ‘shelf life’ as well as coming to the decision making problem with uncertainty in mind. Since the world around us is rapidly changing, so do the requirements we are about to implement in our systems. In that case, should we decide and lock ourselves in a commitment, or wait and see what the future brings?