

The impact of domain knowledge on the effectiveness of requirements engineering activities

Ali Niknafs, Daniel Berry
David R. Cheriton School of Computer Science
University of Waterloo
{aniknafs,dberry}@uwaterloo.ca

1 Introduction

It is commonly believed that people involved in software engineering practice should possess knowledge of the problem domain at hand in order to be effective. No doubt, domain expertise helps practitioners achieve their goals faster. A domain expert will quickly catch up with development activities, since he/she understands other stakeholders' language and does not waste time learning about the domain.

The perception is that a person lacking domain expertise will slow down the pace of development, as he/she keeps learning about the domain and asking questions while others are busy trying to make progress.

This perception is mostly true, but what happens to creativity? Software engineering is considered a creative activity. While software engineering problems require creative solutions, people with prior experiences in the domain of the problem at hand tend to fixate on the solutions that they have seen before. Hence, knowledge of the problem domain may be a hindrance to the needed creativity.

In addition, each domain expert has tacit assumptions about the domain of the problem at hand that may be inconsistent with other stakeholders' assumptions. These inconsistencies may get coded into the system, and may be discovered late in the lifecycle or never. The later they are discovered, the more expensive they are to resolve and fix.

The question driving this research is “How can we reap the benefits of domain expertise while mitigating its weaknesses and not suppressing creativity?”

Domain ignorance might be a solution in knowledge-intensive activities such as business analysis, requirements elicitation, and inspection, all part of requirements engineering. A domain ignorant may explore a wider range of ideas than a domain expert to come up with a really creative solution. He/she might come up with ideas that a domain expert would never have dreamed of. Also, a domain ignorant is able to state ideas independent of any domain assumptions and ask revealing questions that could lead to exposing issues that domain experts have overlooked or to exposing inconsistencies among other stakeholders' assumptions. In fact, we hypothesize that for knowledge-intensive activities, a team with a mixture of domain expertise and domain ignorance outperforms both a team with only domain experts and a team with only domain ignorants. We want to do experiments to test this hypothesis in various ways.

2 Wanted from Industry

We would like to examine system requirements and other artifacts from past, current, and future projects. With a past project, perhaps we could determine after the fact the distribution of domain knowledge in the project and correlate that distribution with measures of the project's success.

With a current or future project, we would like the opportunity to observe the effects of different distributions of knowledge on the effectiveness of the project's team. With any project, we would use various data collection techniques, including interviews, questionnaires, and observations.

We would like also to be able to attend an occasional requirements engineering activity such as an elicitation or inspection session and to control the distribution of domain aware and domain ignorant people participating in the session and then to measure the effectiveness of the team.

3 Benefit to Industrial Participant

If the hypothesis is supported, then the participating organization will know it before any paper describing the results is published and will be able to use this knowledge to its benefit before others will know it.

References

1. Berry, D. M.: The Importance of Ignorance in Requirements Engineering. *Journal of Systems and Software* 28 (2), pp. 179-184, (1995)