Exploring Issues in Software Systems Used and Developed by Domain Experts

Jette Henderson
Institute for Computational Engineering and Sciences
The University of Texas at Austin
jette@ices.utexas.edu

Dewayne E. Perry
Electrical and Computer Engineering
The University of Texas at Austin
perry@mail.utexas.edu

***Abstract*—Software engineering researchers have paid a good deal of attention to fault cause, discovery, and repair in software systems developed by software professionals. Yet, not all software is developed by software professionals, and consequently, not as much research has been conducted that explores fault cause, discovery, and repair in software systems developed by domain experts. In this exploratory paper, we outline research plans for studying the types of faults that domain experts encounter when developing software for their own research. To attain this goal we propose a multiple case study that will allow us to explore questions about domain expert software use, needs, and development.**

I. Introduction

In the past, software engineering empirical researchers have paid a good deal of attention to analyzing fault detection, repair, and causes in software engineered by software professionals. Our own studies in this area point to lack of domain knowledge as a major root cause of faults [1], and in conjunction with lack of domain knowledge, lack of software system knowledge also causes many faults [2] [3]. Yet, as researchers from many different disciplines have come to rely on software more and more to perform their research, software development has become not solely the domain of software engineers. Instead, domain experts have developed software tailored to the specific needs of their research, and with the nascency of domain-expert-developed software, we propose to study the type of faults that domain experts encounter.

Currently, domain experts in engineering disciplines seeking to apply computation to their research have two options when it comes to software. Either they can use software developed by a third party, or they can develop the software themselves. There are complementary strengths and weaknesses for each option. Using third-party-developed software might increase the risk of faults due to lack of domain knowledge, and developing software in-house may increase the risk of faults due to lack of software system building knowledge. In systems built by domain experts, we expect domain knowledge faults to be significantly less frequent, and instead, lack of software system knowledge to cause most faults.

In this paper we will lay the foundation for research concerning the issues domain experts encounter when building software systems for their research. We propose to lay this foundation by using multiple case studies to construct a picture of the domain experts' point of view when it comes to using software in their research. To begin our work we draw on our community, the Institute for Computational Engineering and Science (ICES) and Texas Advanced Computing Center (TACC) at The University of Texas at Austin.

A. Background

ICES is a research center that fosters the interdisciplinary collaboration between faculty members from different scientific and engineering fields on computational and engineering problems. ICES supports eleven research centers and alliances and six research groups drawn from five schools and colleges and eighteen academic departments. There are 93 faculty members, including 42 core faculty members. These ICES researchers strive to combine theory, experimentation, and computation to explore modern scientific and engineering problems. From biological research like simulating the cardiovascular system in order to aid in prevention and treatment of cardiovascular diseases to studying flow in social networks to simulating and modeling vehicle reentry into the atmosphere, ICES contains a wide spectrum of research questions and applications. It is one of the largest centers of its kind in the world. This fact, combined with the variety of the types of problems researchers consider, make it a very interesting place to study domain-expert-developed software.

TACC is an advanced computing resource at the University of Texas at Austin. TACC operates many of the high performance computers in the world, and among other things, they offer computational support, education, and software to help the researchers from many disciplines. Many members of ICES work closely with TACC to perform their computational research.

B. Overview

In this position paper, we formulate a research plan to explore software use at ICES with the initial aim of examining the types of faults that domain experts encounter when developing software. To this end, we propose an empircal approach that takes the form of multiple, individualized case studies focusing on the ICES core faculty. ICES is one of the few major, large interdisciplinary computational research centers in the world. The broad spectrum and variety of projects at ICES will help us gain a better understanding of domain expert software building methods, use, and issues. Our main hypothesis is that domain experts encounter fewer faults resulting from lack of domain knowledge and more faults from lack of software engineering domain knowledge. Our overarching goal is to gain a better understanding of software needs at ICES and how researchers meet these needs.

For the remainder of the paper, we will state and discuss our research questions, elaborate on the multiple case study and data analysis plans, and outline future work and research directions. This research will give us insight into the types of problems domain knowledge experts encounter and whether or not this group of domain experts tends to develop their own software or use third party software.

II. Research Questions

We have developed the following three categories of research questions:

A. *Questions about the State of Domain Expert Software Use*

These research questions are intended to build a clearer picture of the software needs and practices of domain experts. We would like to know what kinds of software needs these domain experts have, and how they go about fulfilling them. We would like to assess whether domain experts in ICES mostly develop their own software for their research or turn to third-party-developed software, and we would like to examine the motivation for such choices. We are specifically interested the following:

- What types of software do domain experts use, and why do they use it?
- What percentage of domain experts use third party developed software, and what percentage develop their own?
- In general, what kind of problems do domain experts making data-intensive computations encounter? What kind of problems do they have in getting their research done from a computational point of view? What are their software needs, and what is their motivation for picking third-party-developed software or developing their own?

B. *Questions about Domain Expert Software Development*

These questions are pointed toward domain experts developing their own software. The individualized case studies will allow us to study, from the perspective of the domain expert, the types of faults domain experts developing software encounter and how they resolve them. We would like to know what kind of conceptual issues they have when they develop software. In addition, we would like to know about their software development process, and what kind of design goes into the development process. Finally, we would like to know about the backgrounds of those developing the software, and what types of methods (informal or formal) they use to report and resolve faults.

C. *Questions about Third-Party-Developed Software*

While we expect third-party-developed software users to be in the minority, we are interested in the needs, motivations, and experiences of these users as well. First of all, we are interested in the source of the third-party software–does it come from a university lab, enterprise, etc? Secondly, we are interested in the kinds of problems domain experts encounter when using third party software–and from their perspective, where and how these problems arise. Once we know the source of the software, we can examine whether the software is also developed by domain experts. Thirdly, we are interested in the usability and documentation of the third-party software. Finally, we are interested in whether or not the domain experts integrate the third-party software with other software they use. To this end, we would like to know if they are integrating multiple platforms and tools, how hard these packages are to interface and integrate.

III. Proposed Interview Questionnaire

We have composed the following questionnaire that we will administer during interviews with ICES researchers.

1) Name of the project
2) Brief description of the project
3) What software do you primarily use during this project?
4) Is the software developed in-house, or do you obtain it from a third party?
5) Note to the interviewer: the following questions are for those who develop their own software
   a) What motivated you to develop the software in-house?
   b) Who is developing it? (e.g., graduate students, researchers, post docs, etc)
   c) When developing software, what languages does your team use?
   d) When developing software, what environments does your team use?
   e) What tools does your team use to test and debug?
   f) What version management system does your team use, if any?
   g) What kind of change management or bug tracking tool (eg, Bugzilla) does your team use, or are bugs handled using informal means (e.g., email)?
   h) Do you know what types of bugs are the most common?
   i) What are the underlying causes of these types of bugs?
   j) What do you think will enable you to detect or even eliminate these kinds of problems?
6) Note to the interviewer: the following questions are for those who obtain the software from a third party
   a) What is the source of the software?
   b) What motivated you to use software developed by a third party?
   c) Do you use multiple third party developed software packages, and if you do, have you had issues constructing interfaces between the software packages?
   d) What are the benefits to using this software?
   e) What are the software's main shortcomings?

f) Have you found faults in the software through its use?

7) Have you ever utilized TACC?
8) If you have utilized TACC, what facilities or tools have you used?
9) Are your programs designed to run in parallel?

IV. Case Study Design and Justification

As stated above, the multiple case studies will take the form of a questionnaire administered during interviews between one or two interviewers and one ICES core faculty member. We will send the questionnaire to the participants before the interviews take place. Our population consists of 42 ICES core faculty members, and we expect to interview a sizable subset of this population. Our goal is to limit each interview to at most one hour.

We propose two phases for the multiple case study. The first proposed phase is a pilot phase where we will interview three or four researchers. The second phase will integrate the insights gained from the pilot phase and consist of interviewing the rest of the participants using the refinements from the pilot phase.

The goal of the pilot phase is to make the interviews more consistent. After conducting these first interviews, we will evaluate the responses, the questionnaire, and our protocol for administering the questionnaire. We will not only examine the responses of the participants against one another, we will also examine the responses against our expectations of the interview and the protocol. We will then refine our protocol and possibly the questionnaire with the goal of making the interview process more consistent and ensure that we are executing the interviews in a timely fashion. The second phase will integrate the changes suggested from the pilot phase, and we will administer the refined questionnaire and protocol to the rest of the participants.

We chose to administer the questionnaire through an interview in order to make it more feasible for the interview participants. Since all of the participants are very busy, structured interviews are more manageable than each faculty member filling out the questionnaire on his or her own time. In addition, this format will also allow us, as interviewers, to obtain a level of detail and consistency that does not usually come from people filling out questionnaires by themselves. We will be able to ask the researchers to elaborate or clarify statements, so the body of responses is more consistent and detailed. Further, giving the interviewees the questionnaire in advance will allow them to prepare useful, detailed answers to those questions that may require more than just touching on the surface of those issues. Since this phase of our research is highly exploratory, we are seeking to gain as much software use information as possible in order to plan the next part of our research.

V. Data Analysis Plan

After conducting the interviews, we will perform appropriate quantitative and qualitative analyses of the questionnaire answers. Some of the answers to the questionnaire will lend themselves to quantitative analysis (e.g., the split of researchers who use third party software and those that develop software themselves), while other answers will require us to examine the body of responses and create categories that appropriately encapsulate the responses. The goal of the data analysis will be to explore and refine our research questions as well as hone in on the best direction in which to take our research.

VI. Research Plans Beyond the Questionnaire

Since this first phase of our research is exploratory, it is difficult to make exact plans for the steps following the execution of interviews and analysis, but our goal for the first phase is that the results from the individual case studies will lay a foundation for a more objective analysis of the issues that domain experts face when using and developing software.

One of the strengths of earlier fault studies [1] [2] [3] was that the researchers drew on preexisting fault documentation to make their analyses. This is a far more objective approach to the one we are proposing because we are asking the researchers themselves about what they perceive about their software use and development. In the future we would like to take a more objective approach to issues in domain expert software use. For example, while we expect that most of the core faculty members who develop their own software to use a more informal means of tracking and fixing faults, we propose working with a faculty member to implement a formal means of tracking and fixing faults, which will give us more objective data to analyze.

VII. Conclusion

The main goal of this paper is to outline a research plan that studies the types of software issues that domain experts encounter when utilizing software in their research and developing software for their research. Through individual interviews with the domain experts at the Institute for Computational Engineering and Science, we plan to gain an understanding of domain expert software landscape. The aim of our case studies is to explore to what degree these experts use software in their research, and if they do develop their own, for what purpose and what issues they encounter. The individualized case studies will just be snapshot of domain expert computing, but the variety of domain experts we will interview will give a good cursory view of domain expert computational needs and software use.

Acknowledgment

We would like to thank the ICES faculty for their cooperation in our research.

References

[1] D. E. Perry and C. Steig, "Software Faults in Evolving a Large, Real-Time System: a Case Study", 4th European Software Eng. Conf. – ESEC93, Garmisch, Sept 1993.
[2] M. Leszak, D. E. Perry and D. Stoll. "A Case in Root Cause Defect Analysis", Int. Conf. on Software Eng. 2000, Limerick, June 2000.

[3] M. Leszak, D. E. Perry and D. Stoll. “Classification and Evaluation of Defects in a Project Retrospective”, *J. of Syst. and Software*, vol. 61, pp. 173-187, April, 2002.

[4] R. Yin. *Case Study Research: Design and Method, 2*nd ed (Applied Social Research Methods Series, vol. 5). Thousand Oaks, CA: Sage, 1994.

[5] R. E. Stake. *The Art of Case Study Research*. Thousand Oaks, CA: Sage, 1995.