Chapter 19
An Analysis of Process Characteristics for Developing Scientific Software

Diane Kelly
Royal Military College of Canada, Canada

ABSTRACT

The development of scientific software is usually carried out by a scientist who has little professional training as a software developer. Concerns exist that such development produces low-quality products, leading to low-quality science. These concerns have led to recommendations and the imposition of software engineering development processes and standards on the scientists. This paper utilizes different frameworks to investigate and map characteristics of the scientific software development environment to the assumptions made in plan-driven software development methods and agile software development methods. This mapping exposes a mismatch between the needs and goals of scientific software development and the assumptions and goals of well-known software engineering development processes.

INTRODUCTION

The methods for developing scientific software have recently garnered the attention of the software engineering community, resulting in several studies of scientists as "end-user developers". Segal (2004) differentiated scientists as "professional" end-user developers, pointing out their extensive knowledge in a technical domain and their high comfort level in writing code. There have been suggestions (Ackroyd et al., 2008; Pitt-Francis et al., 2008; Segal, 2005; Wood & Kleb, 2003) that agile software development methodologies (Highsmith & Cockburn, 2001) rather than "plan-

DOI: 10.4018/978-1-4666-2059-9.ch019

Copyright © 2013, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

driven" methodologies (Royce, 1970) are better suited to the development of scientific software.

This paper examines the characteristics of the environment in which scientists develop their own software and compares these to the assumptions and goals of plan-driven and agile software development methodologies. The conclusion is that there is a mismatch in both types of methodologies when applied to the development of scientific software.

We first define what we mean by scientific software and discuss characteristics of its development. In Section 3, we make initial observations about these characteristics and the assumptions inherent in plan-driven and agile software development methods. Section 4 uses a variety of models to further explore the characteristics of scientific software development as opposed to inherent assumptions that underlie plan-driven and agile development methods. Section 5 summarizes our findings and Section 6 concludes.

SCIENTIFIC SOFTWARE

Definition of Scientific Software Development

A project that we categorize as scientific software development has the following three characteristics. First, the software is written to answer a scientific question. The question can be general, such as, "Is it safe to operate this nuclear generating station?" or specific, such as, "Can I track satellites with this telescope?" Second, the writing of the software necessitates the close involvement of someone with deep domain knowledge in the application area related to answering the question. The person with the deep domain knowledge, the scientist, usually writes the software. This is Segal's professional end-user developer. We have observed cases where the scientist has not written the software, but the software developer has had to become enough of an expert in the domain to understand and implement what the scientist is describing. Third, the software provides output data to support the scientific initiative. A human is in the system loop to examine the data and make observations and ultimately, answer the scientific question. The expectation is that the data provided by the computer solution is correct and will not misguide the scientist answering the question.

The science application can be solving large systems of differential equations, analyzing immense amounts of data such as imagery or bioinformatics, and computing using analytical and empirical models. Our definition of scientific software, however, excludes the following: control software whose main functioning involves the interaction with other software and hardware; user interface software that may provide the input for and report of scientific calculations; and any generalized tool that scientists may use in support of developing and executing their software, but does not of itself answer a scientific question.

Scientific software may have an extensive graphical user interface or interact with other software or hardware to obtain data. It may run on complex multi-processors and require middleware support to make use of hardware capabilities. But in the following discussion, we are talking exclusively about the code that implements the science application whether it is designed as a separate module or inserted into a product that includes these other parts. We contend that this core scientific part of the product requires different considerations when it is being developed.

Characteristics of Scientific Software

Several studies have looked at the characteristics that differentiate the environment and activities of scientific software development from the development of other types of software (Basili et al., 2008; Boisvert & Tang, 2001; Matthews et al., 2008; Post & Kendall, 2004; Sanders & Kelly, 2008; Segal, 2003, 2004). Some characteristics are invariant whether the software is developed

15 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/analysis-process-characteristics-developing-scientific/69627

Related Content

Validating the End-User Computing Satisfaction Survey Instrument in Mexico
George E. Heilmanand Jorge Brusa (2008). *End-User Computing: Concepts, Methodologies, Tools, and Applications (pp. 1531-1541).*
www.irma-international.org/chapter/validating-end-user-computing-satisfaction/18268

What's in a Name?: Exploring the Metaphorical Implications of Data Warehousing in Concept and Practice
Elizabeth J. Davidson (1999). *Journal of End User Computing (pp. 22-32).*
www.irma-international.org/article/name-exploring-metaphorical-implications-data/55775

Logic Models as a Framework for Iterative User Research in Educational Technology: Illustrative Cases
Yvonne S. Kao, Bryan J. Matlen, Michelle Tiuand Linlin Li (2018). *End-User Considerations in Educational Technology Design (pp. 52-75).*
www.irma-international.org/chapter/logic-models-as-a-framework-for-iterative-user-research-in-educational-technology/183012

Design and Development of Intelligent Decision Support Prototype System for Social Media Competitive Analysis in Fashion Industry
Eric W.T. Ngai, S.S. Lam, J.K.L. Poon, Bin Shenand Karen K.L. Moon (2016). *Journal of Organizational and End User Computing (pp. 13-32).*
www.irma-international.org/article/design-and-development-of-intelligent-decision-support-prototype-system-for-social-media-competitive-analysis-in-fashion-industry/148144

Next Generation Cellular Network Planning: Transmission Issues and Proposals
S. Louvros (2008). *End-User Computing: Concepts, Methodologies, Tools, and Applications (pp. 2308-2328).*
www.irma-international.org/chapter/next-generation-cellular-network-planning/18298