Software Process for Multicomponent Multiphysics Codes

A. Dubey, Katherine Riley and Katie Antypas

October 29, 2015

Abstract

Multiphysics multicomponent scientific codes have increasingly been adopting software processes derived from outside the scientific domain. The driving force behind adoption is usually the realization that without using software engineering practices, the development, verification and maintenance of code becomes intractable. However, many software best practices need modification and/or customization. Sometimes the inherent physics of scientific applications require different software methodologies, while other times a premium is placed on performance rather than architecture. Still others are more sociological and due to the type of institutions that are the typical homes for such codes. The challenges for scientific applications range from their architecture to the process for their maintenance and growth. For example, sometimes modularity and encapsulation principles are challenged by the need to tightly couple physics solvers to data structures. Scientific codes are designed to explore phenomena that are not very well understood, so their verification strategies have to be customized. The institutional challenges in developing and maintaining scientific codes arise because the research institutions in which they are developed often have an unreliable funding model and a transient developer population of graduate students. Intellectual ownership, code distribution, attribution of credit and contribution policies can all become thorny issues because there is often a lack of communication and sometimes even distrust among the stakeholders. We elaborate on the above challenges and how they were addressed in FLASH, a code that has been extremely successful as a community code for several research communities. We outline specific solutions used by FLASH, and discuss their possible generalizations that are usable by other similar software efforts. In particular, we address the issues related to software architecture and modularization, design of a testing regime, unique documentation needs and challenges, use of versioning system for managing projects, and the tension between intellectual property management and open science.

1 Introduction

The computational science and engineering (CSE) communities have a mixed record of using software engineering and adopting good software practices. Majority of codes adopt software practices when the size and composition of the code makes it impossible to make progress without them. In rarer instances code projects start with an awareness of the importance of software process and build it into the DNA of the code. As more codes have crossed the threshold of being manageable without software engineering they have increasingly been adopting software processes derived from outside the scientific domain. The driving force

behind adoption is usually the realization that without using software engineering practices, the development, verification and maintenance of code becomes intractable. However, many software best practices are not well-suited for CSE codes without modification and/or customization. Sometimes the inherent physics of scientific applications require different software methodologies, while other times a premium is placed on performance rather than code architecture. Still others are more sociological and due to the type of institutions that are the typical homes for such codes. The challenges for scientific applications range from their architecture to the process for their maintenance and growth.

2 Domain Challenges

: this section will outline challenges that are unique to scientific codes because of the nature of problems they try to solve

3 Institutional Challenges

: this section will talk about challenges that come about because of the kind of institutions in which these codes are developed

4 Case Study: The FLASH Code

5 Generalization

: this section will discuss those aspects of FLASH solutions that are generalizable

6 Additional Future Considerations

: How the software, design and policies might need to change in Future.