IGSTK: Development Process and Project Management Best Practices for an Open Source Software Toolkit for Image-Guided Surgery Applications

Kevin Gary[1], M. Brian Blake[2], Stephen Aylward[3], Julien Jomier[3], David Gobbi[4], Hee-su Kim[5], Rick Avila[6], Luis Ibanez[6], and Kevin Cleary[5]

[1] Division of Computing Studies, Arizona State University, Mesa, Arizona, 85212, USA
kgary@asu.edu
[2] Department of Computer Science, Georgetown University, Washington, DC, 20007, USA
blakeb@cs.georgetown.edu
[3] Computer-Aided Diagnosis and Display Laboratory, University of North Carolina, Chapel Hill, NC, 27599, USA
aylward@unc.edu, jjomier@cs.unc.edu,
[4] Atamai Inc., London, Ontario, N6B 2R4, Canada
dgobbi@atamai.com
[5] Imaging Science and Information Systems (ISIS) Center, Department of Radiology, Georgetown University Medical Center, Washington, DC, 20007, USA
hkim@isis.imac.georgetown.edu, cleary@georgetown.edu
[6] Kitware Inc., Clifton Park, NY, 12065, USA
rick.avila@kitware.com, luis.ibanez@kitware.com

Abstract. Open source technologies are increasing in popularity for software development. Many open source projects rely on skilled development teams whose members are distributed throughout the world. Often, agile development methods are employed by these teams, as the focus is on concurrent development and fast production over requirements management and quality assurance. The image-guided surgery toolkit (IGSTK) is an open source development project that relies on the collaboration of a skilled and distributed development team, yet addresses a domain that demands managing requirements as well as implementing a high degree of robustness and addressing safety concerns. Due to this unique cross-section of open source technology and the surgical domain, the IGSTK team has developed a set of best practices and requirements techniques to augment commonly applied agile methods. This paper presents the lessons we have learned as we have engaged in the software development process.

1 Introduction

The image-guided surgery toolkit (IGSTK) is an open source project aimed at developing robust software for medical applications. Image-guided surgery involves the use of pre-operative medical images to provide image overlay and instrument guidance during procedures. Image-guided surgery systems have been commercially

available for about ten years now, but this field of research is still active, and challenges still exist. These systems are software intensive and a lot of work is needed to develop them. Also, the software must be robust and well-tested since it will be used in a medical environment.

The IGSTK project was designed to address these issues. The toolkit contains the basic software components to construct an image-guided system, including a tracker and a four-quadrant view incorporating image overlay. Robustness and quality have been the highest priority in the design of the toolkit, which is based on the following existing opening source components: ITK for segmentation and registration, VTK for visualization, and FLTK for the user interface.

Managing an open source project with multiple geographically distant developers, complex application requirements, and a desire to produce a framework for extensible and reusable architecture components is a tremendous challenge. The IGSTK team has created its software processes to balance an agile development philosophy with an integrated requirements elicitation and management approach, and consequently has arrived at a methodology that is fast and flexible, yet meets the stringent needs of this application domain. In this paper we present this approach, describing the process implementation, supporting tools, and focus on requirements.

The IGSTK toolkit is Open Source software distributed under a BSD-like license equivalent to VTK and ITK licenses. The software is developed following the practices of Agile Programming and therefore is made available from the very early stages of development. Basic information related to IGSTK can be found at the web site http://www.igstk.org and the Wiki pages http://public.kitware.com/IGSTKWIKI. Instructions for configuring and building the toolkit are available at http://public.kitware.com/IGSTKWIKI/index.php/How_to_build_IGSTK. You are welcome to try the software, review the source code and send your comments to the IGSTK development team. The software is currently in beta form and evolving, but we expect a stable release by the end of 2005. Note that this software should only be use in clinical cases under IRB approval. You are allowed to use IGSTK for free in commercial applications but it is your responsibility to perform the tests and validations required by regulatory bodies such as the U.S. Food and Drug Administration (FDA).

2 Software Development Process

IGSTK development presents interesting challenges from a software development methodology perspective. These complexities derive from the nature of the requirements, the makeup of the team, the dependence on pre-existing software packages, and the need for high quality standards within this domain. We discuss these complexities in this section and suggest a working set of best practices that attempt to provide solutions.

The first challenge to IGSTK development derives from the nature of the requirements, which come in multiple flavors. Application-specific requirements exist for a set of applications that IGSTK is required to support upon completion. However, most requirements are framework-level requirements, which are difficult to

completely understand before development itself begins in earnest. Furthermore, IGSTK aims not only to support the fixed set of applications it is required to support contractually, but also to serve as a platform for further research and production-worthy software products. Because of these process requirements, waterfall-style development methodologies [8] that attempt to define requirements completely before development begins are not considered suitable. Additionally, Rational Unified Process oriented use-case driven analysis modeling [7] is only selectively applied, as we cannot assume that the non-functional requirements derived from the known set of applications today represent a complete set of such requirements for the future. Given the complex nature of our requirements process and our application domain, we will detail this issue in Section 3.

The second challenge to IGSTK development is the makeup of the team, which is comprised of academic and commercial partners collaborating in a widely distributed setting. Most if not all of the team members have other demands on their time. These factors create challenges for setting project deliverables and expectations over medium- and long-term horizons. Fortunately, most of the development team has worked with a common set of source code upon which IGSTK is based (VTK and ITK), and has great familiarity with common tools such as CMake and DART, which are further discussed below.

The requirements, team composition, and use of pre-existing software suggest that agile methods [4] should be applied to IGSTK. All team members have significant exposure to agile methods; some have even developed agile-ready tools that are employed on IGSTK [9]. However, the fourth challenge to IGSTK development – the high quality standards demanded the application domain – suggests that some agile practices need to be reinforced by best practices that address this issue. For example, FDA guidelines for approval of medical devices require traceability of requirements through implementation and testing. Agile methods, in general terms, tend to de-emphasize the value of requirements management processes. Therefore, some means of managing requirements while remaining open to change is needed. Given that requirements are also evolving as code is actively developed, code repository management needs to be in some way synchronized with requirements management.

To address these complexities, IGSTK has adopted an agile approach augmented by the following set of best practices:

Best Practice #1. Above all, recognize that people are the most important mechanism available for ensuring high quality software. This practice agrees with the philosophy espoused by the agile community [4]. The IGSTK team is comprised of developers with a high degree of training and experience with the application domain, supporting software, and tools. Their collective judgment is weighted over a high-level process mandate.

Best Practice #2. Facilitate constant communication. The evolution of a new framework and of supporting software upon which the framework is layered (particularly ITK, which is relatively young and still evolving), plus the distributed nature of the team, has been addressed by facilitating constant communication. IGSTK members participate in a weekly teleconference and

meet in person twice per year. IGSTK employs a mailing list and a Wiki for online collaboration.

Best Practice #3. Produce iterative releases. IGSTK's external release cycle includes twice-yearly releases that coincide with IGSTK advisory board meetings. Internally, six months was considered too long a horizon to manage development, so releases are broken down into approximately month-long "sprints" called iterations. At the end of an iteration, the team can stop, assess and review progress, and determine what code is considered stable enough to move to the main code repository. For one week at the end of an IGSTK iteration, developers perform code reviews of all new and modified code, and ensure high code coverage and passing unit tests across the entire code base.

Best Practice #4. Employ a Sandbox for evolving code. Providing a separate code line with a different check-in policy allows developers to share code that may not yet meet more stringent check-in policies on the main code base. Developers may check-in to the sandbox before corresponding requirements have been accepted, with a lower level of code coverage for their unit tests, and before the code has been reviewed by the rest of the team. Exploiting configuration management approaches even at this early informal sandbox phase helps to document and track project changes.

Best Practice #5. Emphasize continuous builds and testing. IGSTK uses the open source DART tool (http://public.kitware.com/Dart/HTML/Index.shtml) to produce a nightly dashboard of build and unit test results across all supported platforms. Developers are required to ensure that code coverage stays as close as possible to 100%, that their source code builds on all supported platforms, and that all unit tests pass. The dashboard is reviewed during the weekly teleconference.

Best Practice #6. Support the process with robust tools. Best Practice #5 describes the use of the DART tool for continuous testing. IGSTK also employs an open source cross-platform build solution called CMake (http://www.cmake.org/) and an open source documentation system called Doxygen (http://www.stack.nl/~dimitri/doxygen/). These tools are augmented with defined best practices for coding and documentation posted on the Wiki.

Best Practice #7. Emphasize requirements capture and management in lockstep with code management. As requirements evolve and the code matures, it is necessary to adopt flexible yet defined processes for managing requirements and code repositories. The organization and tracking of requirements is a complex process for a project such as IGSTK, and thus is detailed in Section 3.

Best Practice #8. Allow the process to evolve. Through constant communication, IGSTK members recognize when the complexities they face can be approached within the current process framework, when "tweaks" are required, or when entirely new practices should be adopted. Best Practices #1 and #2 above (people and communication) are emphasized here, and Best Practice #3 (iterative

development) provides a means to manage not only an evolving software product, but an evolving software process as well.

We believe that these best practices represent workable techniques for IGSTK and may work for other development teams facing similar complexities. However, as we have noted in our last best practice, we also recognize that these practices are themselves open to change and improvement, and we look forward to working with the emerging open source communities in such domains to continue developing these techniques.

3 Requirements Definition and Tracking of Requirements

The development of requirements in IGSTK consists of two phases. The initial phase is understanding the subject matter areas or taxonomy of all requirements relevant to the project. Although most of the developers have some familiarity with the medical domain, it is difficult for them to prioritize the needs of the solution applications without significant direction from a domain subject matter expert. We have investigated this initial phase in earlier work [1].

Here, we will describe the second phase of requirements definition: the discovery and management of requirements at development time. As previously discussed in Best Practice #7, requirements management and code management are significantly integrated. Although IGSTK development has not been a pure agile process, our project does see requirements for development as coming from the "bottom-up". Developers introduce new requirements for further capabilities as components are being developed. The IGSTK project has employed a new collaborative process for reviewing, implementing, validating, and archiving these requirements, and it is integrated with application development. This process is illustrated as a UML state diagram in Fig 1 and discussed next.

The two phases of requirements definition interact as follows.. In the initial requirements phase, for example, general requirements for tracking devices (localizers) were discovered. As the components for these initial requirements were developed, we discovered that additional requirements existed (i.e. perhaps specific validation requirements). Once a developer identifies new potential requirements (Conceptualized box in Figure 1), the developer will post a text description (Defined) on the shared web site (Wiki). At the same time, the initial code that fulfills the requirements is entered into the sandbox repository. The requirement would then undergo an iterative review sub-process where the team members would review, discuss, and potentially modify the requirement. Based on the team's decision, the requirements can be rejected/aborted or accepted. Rejected requirements are archived on the Wiki (Logged) so that they can be reopened later, if necessary. A unique approach in the IGSTK project is the use of an on-line open source bug tracker (PHP BugTracker) to store requirements. This approach is particularly effective as defect reports and resulting actions are also stored in the bug tracker. The accepted requirements are entered into the bug tracker and marked as "open." Once the supporting software is implemented and its functionality is confirmed, the

requirement is marked as 'verified." As the nightly builds takes place, all verified requirements are automatically extracted into Latex and PDF files, and are archived. Custom scripts were developed for this purpose.

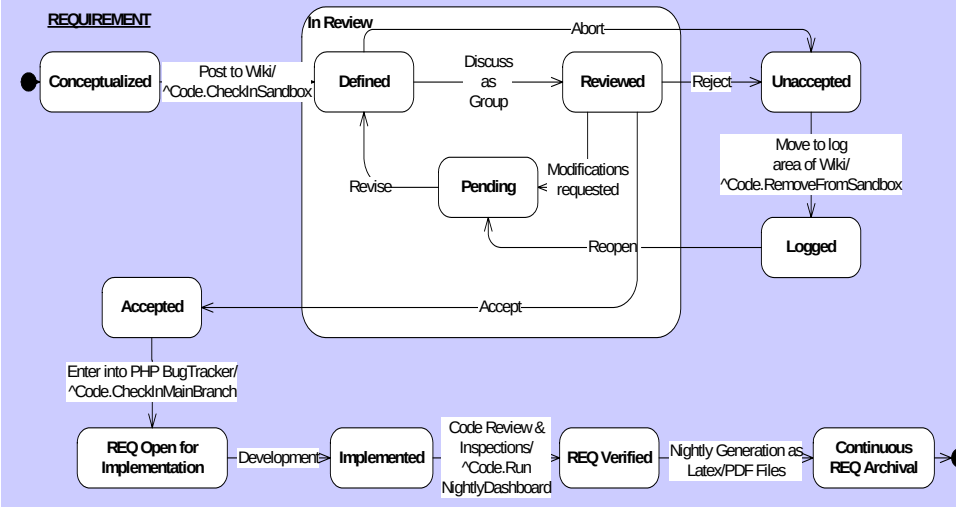

Fig. 1. Requirements Management for the IGSTK project as a UML state diagram.

4 Conclusions

The focus on requirements in the IGSTK project is novel, both within the medical image analysis community and among open source projects employing agile methodologies in general. Most open source projects originate from developers who are already intimately familiar with a particular application domain, and IGSTK is no different. However, there are important distinctions. The domain addressed by IGSTK mandates that requirements be documented, tracked as they are changed, and organized according to the type of information captured. IGSTK's goal is to serve as a framework for a pre-existing set of applications, as well as for applications anticipated by the community. Finally, because IGSTK addresses the surgical domain, its primary goal is to ensure program safety and functional quality. Such requirements and quality standards are unusual in open source collaborative development.

A significant part of IGSTK's response to the high quality demanded by the project is designed into the software itself, as it must be. However, IGSTK has created a set of best practices and a requirements process to address these unique needs. In this paper we have reported the lessons we have learned and the principles we believe are unique in this open source effort. Ultimately, a development approach that allows for evolution may demonstrate that open source technology can address critical domains.

5 Acknowledgements

This research is supported by the National Institute of Biomedical Imaging and Bioengineering (NIBIB) at the National Institute of Health (NIH) under grant R42EB000374 and by U.S. Army grant W81XWH-04-1-0078. The content of this manuscript does not necessarily reflect the position or policy of the U.S. Government. The authors would like to thank the IGSTK advisory board for their advice throughout the project: Will Schroeder of Kitware; Ivo Wolf of the University of Heidelberg; Peter Kazanzides and Anton DeGuet of Johns Hopkins University; and Ingmar Bitter, Matt McAuliffe, and Terry Yoo from the NIH.

References

1. Blake, M.B., Cleary, K., Ibanez, L., Ranjan, S.R., and Gary, K.: "Use Case-Driven Component Specification: A Medical Applications Perspective to Product Line Development," ACM Symposium on Applied Computing (SAC 2005), pp 1470 - 1477 Santa Fe, NM (2005).
2. Booch, G. Rumbaugh, J, and Jacobson, I.: "The Unified Modeling Language User Guide," Addison Wesley, Reading, MA (1999).
3. Cleary, K., Ibanez, L., Ranjan, S.R., and Blake, M.B.: "IGSTK: A Software Toolkit for Image-Guided Surgery Applications," Proceedings of the 18th International Conference on Computer-Assisted Radiology (CARS2004),pp 473-479, Chicago, IL, June (2004).
4. Cockburn, A.: Agile Software Development. Addison-Wesley (2002).
5. Cockburn, A.: "Characterizing People as Non-linear, First-order Components in Software Development." 4th International Multi-Conference on Systems, Cybernetics and Informatics, Orlando, Florida, (2000).
6. Ibanez, L., Schroeder, W., Ng, L., Cates, J., and the Insight Software Consortium: The ITK Software Guide, Kitware, Inc. Publishers, Clifton Park, NY (2003).
7. Kruchten, P.: The Rational Unified Process—An Introduction, Second Edition, Addison-Wesley (2000).
8. Royce, W.W.: "Managing the development of large software systems: concepts and techniques." IEEE WestCon, Los Angeles (1970).
9. Schroeder, W.J., Ibanez, L. Martin, K.M.: "Software Process: The Key to Developing Robust, Reusable and Maintainable Open-Source Software." Proceedings of the 2004 IEEE International Symposium on Biomedical Imaging: from Nano to Macro. pp 648-651 Arlington, VA (2004).
10. Sommerville, I., Sawyer, P.: Requirements Engineering. Wiley (1997).