University of California, Los Angeles

Medical Imaging Informatics Group

BE M227 Project #1

Best Practices in Software Engineering for Medical Informatics

Author:
Nicholas J. Matiasz

Instructor:
Prof. Alex Bui

Submitted:
2013-10-31

1 Introduction

Research in medical informatics requires skills from various fields, such as computer science, medicine, bioengineering, biostatistics, and public health. Given its interdisciplinary context, software development in medical informatics can differ significantly from that of other pursuits, such as software engineering research and commercial software development [14]. Vetted software engineering methods are essential when making medical software because of the importance of maintaining patient safety, as well as the massive scale on which some medical systems operate. The recent struggles with the new Obamacare website demonstrate the consequences of a poorly executed software development plan. This paper discusses the ways in which software development is constrained by the unique context of medical informatics research; best practices are then presented for software engineering that applies specifically to work in medical informatics.

2 The complex context for medical software development

Software development in medical informatics occurs in a complex context, characterized by interdisciplinary collaboration and diverse stakeholder interests. As opposed to, for example, a single client hiring a software development firm, software development in medical informatics often involves multiple funding sources, end users, legal constraints, and governing bodies. It is difficult to discuss medical software development without acknowledging related factors that might normally fall outside of the scope of software engineering; the software development life-cycle (SDLC) must therefore address these challenges for the project to succeed [13]. A tension arises when researchers attempt to make software that can be not only integrated seamlessly into their institution's existing infrastructure but also adapted easily to other

clinical settings [14]. New medical software is valuable to the extent that it improves upon existing methods for delivering healthcare; however, even if new medical software proves helpful, its modification of healthcare delivery often results in multiple challenges, including resistance from physicians, IT restructuring, and legal barriers to its implementation.

2.1 System inertia in healthcare

To mitigate resistance to new technologies, a medical informatics software developer should not only deliver software that meets the needs of users but also design such software with the existing technological environment and clinical workflow in mind. For example, a developer who satisfies physicians with useful software but frustrates the IT staff responsible for deploying the software has failed to navigate the complex clinical environment successfully. This delicate balance that developers must maintain to ensure the adoption of their technology demonstrates how economic and political considerations are particularly relevant in medical informatics work. Coiera has suggested that stagnation in medical software systems commonly occurs due to a "system inertia" caused by the multitude of users, stakeholders, governing bodies, and laws in healthcare [3]. Thus, in light of national efforts to achieve EHR standardization and infrastructure interoperability, it may be instructive, albeit optimistic, to view each medical software development project as a mere subprocess that advances the state of the art toward these national goals. Existing standards for the creation of medical software that treat such systems as stand-alone might therefore be myopic [8, 9].

3 Recipe for effective software development methods

Although the details of a software development plan will vary widely depending on the requirements of the stakeholders and end users, the literature points to common strategies

that will usually help a project to succeed. The following sections describe these strategies and explain their specific utility for work in medical informatics.

3.1 Flexible and evolving development processes

Perhaps the most significant trend in the development of medical software is the growing need for adaptable software engineering methods. The complex context of these projects increases the probability that user requirements will change, administrative barriers will manifest, and technical challenges will arise unexpectedly. Arguing that health software and its development methods have become "too complex, noisy, and potentially inconsistent for the existing deterministic approaches of software engineering" [1], Clifton, et al. stress the need for flexible and evolving approaches that can accommodate chaos without derailing the entire development effort. These dynamics motivate the selection of software development methods with consistent end-user feedback, continuous integration of separate modules, and an avoidance of strictly sequential processes. Cunningham, et al. [4] discuss how their project benefitted from forgoing rigid software development methods, instead adopting an evolutionary approach with concepts borrowed from incremental process models. This project was still divided into traditional phases (e.g., requirements analysis, design, etc.), but the team's ability to update the development process regularly in response to user feedback was crucial to their success.

3.2 Security of patient data

Developers often prioritize software security during development, but it is particularly important for medical software due to both its high-stakes use cases and the privacy requirements for medical data. The Health Insurance Portability and Accountability Act (HIPAA) of 1996,

despite undergoing revisions to limit its restrictions, enforces numerous security measures regarding patient data. Nosowsky and Giordano argue that these restrictions impede both clinical research and the medical innovations that such work is intended to produce [11]. Developers are therefore challenged to make software that is sufficiently secure without being excessively limited in functionality. This charge is even more vital when a security breach could directly harm the patient [6]. Regardless of which software development strategy is selected, a project's security requirements must be identified early and articulated clearly if the software is to receive legal authorization and widespread adoption.

3.3 Emphasis on user experience design

Like software security, user experience design (UX) plays a critical role in medical software's efficacy. Of course, machines have not replaced physicians entirely; despite clinical decision support systems and new machine learning techniques, medical interpretation and diagnosis remain subjective endeavors. It follows that medical software is effective insofar as the physician maintains control over the clinical workflow [14]. For this reason, development strategies that allow for recurring user involvement and feedback may be optimal for software that relies heavily on user interaction [10] (e.g., data visualization tools). The user's ability to glean meaningful information from such interfaces will ultimately determine the success of the technology.

3.4 Scope of adoption

Medical informaticians who develop medical software tools and standards commonly seek widespread adoption of their technology. This goal is certainly valid because healthcare's vision of national interoperability depends largely on whether multiple institutions can agree

to use the same tools. When designing new software for a specific end user, developers are presented with yet another tension: the software should be not only customized for the context of its initial deployment but also general enough to encourage adoption by additional users [7]. In some cases, software is developed in-house for specific use cases in a single institution. Such software will likely need to be modified as additional customers implement it. Especially for medical software, a flexible development process is more likely to accommodate the variety of technological and administrative challenges that a growing user base will present [7, 12].

4 Agile methods of software development

The medical informatics literature highlights the desirability of development strategies that evolve gracefully, incorporate end-user feedback, achieve continuous integration, and cater to the people who will use, deploy, and evaluate the software. A universal approach cannot exist, but all of these qualities are certainly found in agile software development methods. Agile methods are very people-centric in that they can adapt to the particular strengths of a team, as well as to the evolving understanding of the presented technological challenges [2]. As described above, software development in medical informatics occurs in a complex context, which does not always lend itself to traditional methods that strictly adhere to linear or sequential activities. By design, agile methods can accommodate shifting customer requirements, evolving perspectives on the problems to be solved, and the rapid pace of innovation—all of which are characteristic of the clinical environment. Adopting an agile philosophy on the level of individual projects may also facilitate the massive system integration efforts that lie ahead if national, or even international healthcare interoperability is to be achieved. For this long-term process to succeed, agile concepts such as frequent integration and customer collaboration must be incorporated into this arduous process.

As with any software development strategy, agile methods have their limitations. Despite the advantages of agile methods' flexibility, there are many projects that simply cannot afford to deviate from a predefined plan. Software that is intended to control medical devices, for example, must be developed under specific guidelines in order to receive various certifications, such as IEC 62304 [5]. It is reasonable to expect that agile methods might not be optimal for such cases. Therefore, the agile paradigm may be best suited to software development that occurs in a research setting, where the emphasis is on demonstrating a novel approach to a problem, as opposed to validating a technology for clinical use.

5 Conclusion

As might be expected, software development in medical informatics shares many guidelines with those of other domains; however, the complex, interdisciplinary context of this discipline invites extra emphasis on key aspects of the SDLC. The presence of multiple stakeholders, restrictive legislation, and the evolving state of medical knowledge and standards all influence medical software development and thus motivate its use of highly iterative development methods. Perhaps the most significant challenge when selecting a software development method is achieving a balance between agile methods that accommodate chaotic environments and systematic, sequential methods that yield secure, certifiable systems. A general prescription for this kind of software development may be impossible, but the literature certainly points to trends that new projects should acknowledge when designing a SDLC. It seems particularly fitting that agile methods—characterized by their emphasis on the needs and skills of individuals—should be favored by developers in healthcare, a field with a similar focus.

References

[1] David A Clifton, Jeremy Gibbons, Jim Davies, and Lionel Tarassenko. Machine learning and software engineering in health informatics. In *Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2012 First International Workshop on*, pages 37–41. IEEE, 2012.

[2] Alistair Cockburn and Jim Highsmith. Agile software development, the people factor. *Computer*, 34(11):131–133, 2001.

[3] Enrico Coiera. Why system inertia makes health reform so difficult. *BMJ*, 342, 6 2011.

[4] Alex P Cunningham, Antonis C Antoniou, and Douglas F Easton. Clinical software development for the web: lessons learned from the boadicea project. *BMC medical informatics and decision making*, 12(1):30, 2012.

[5] Michaela Huhn and Axel Zechner. Arguing for software quality in an iec 62304 compliant development process. In *Leveraging Applications of Formal Methods, Verification, and Validation*, pages 296–311. Springer, 2010.

[6] Raoul Jetley, Sithu Sudarsan, R Sampath, and Srini Ramaswamy. Medical software–issues and best practices. In *Distributed Computing and Internet Technology*, pages 69–91. Springer, 2013.

[7] Liv Karen Johannessen and Gunnar Ellingsen. Integration and generification—agile software development in the healthcare market. *Computer Supported Cooperative Work (CSCW)*, 18(5-6):607–634, 2009.

[8] Craig E Kuziemsky and John Knight. 5th international workshop on software engineering in health care (sehc 2013). In *Proceedings of the 2013 International Conference on*

Software Engineering, pages 1549–1550. IEEE Press, 2013.

[9] B. Larson, J. Hatcliff, S. Procter, and P. Chalin. Requirements specification for apps in medical application platforms. In *Software Engineering in Health Care (SEHC), 2012 4th International Workshop on*, pages 26–32, 2012.

[10] Francis Lau, Craig Kuziemsky, Morgan Price, and Jesse Gardner. A review on systematic reviews of health information system studies. *Journal of the American Medical Informatics Association*, 17(6):637–645, 2010.

[11] Rachel Nosowsky and Thomas J Giordano. The health insurance portability and accountability act of 1996 (hipaa) privacy rule: implications for clinical research. *Annu. Rev. Med.*, 57:575–590, 2006.

[12] Guido Porruvecchio, Giulio Concas, Daniele Palmas, and Roberta Quaresima. An agile approach for integration of an open source health information system. In *Agile Processes in Software Engineering and Extreme Programming*, pages 213–218. Springer, 2007.

[13] Xiping Song, Beatrice Hwong, Gilberto Matos, and Arnold Rudorfer. Understanding and classifying requirements for computer-aided healthcare workflows. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 1, pages 137–144. IEEE, 2007.

[14] Jens H Weber-Jahnke, Morgan Price, and James Williams. Software engineering in health care: Is it really different? and how to gain impact. In *Software Engineering in Health Care (SEHC), 2013 5th International Workshop on*, pages 1–4. IEEE, 2013.