Blueprint for Software Engineering in Technology Enhanced Learning Projects

Michael Derntl, István Koren, Petru Nicolaescu,
Dominik Renzel and Ralf Klamma

RWTH Aachen University
Advanced Community Information Systems (ACIS)
Lehrstuhl Informatik 5, Ahornstr. 55, 52056 Aachen, Germany
lastname@dbis.rwth-aachen.de

Abstract. Many projects in Technology Enhanced Learning (TEL) aim to develop novel approaches, models, and systems by field-testing new ideas with software prototypes. A major challenge is that project consortia need to establish a distributed, typically understaffed developer community that has to align its development efforts with needs from application partners and input from research partners. Tackling this challenge, this paper provides a blueprint for software engineering process and infrastructure, which was distilled from successful practices in recent TEL projects. We present a composition of freely available instruments that support open, distributed software engineering practices using continuous integration processes. The blueprint considers the full development cycle including requirements engineering, software architecture, issue tracking, build management and social aspects of developer community building in TEL projects. Some lessons learned are provided, particularly related to open source commitment, innovation as a social process, and the essential role of time. We aim to make software development in TEL fit for Horizon 2020 with processes and instruments that can be readily adopted for planning and executing future projects.

1 Introduction and Challenges

In interdisciplinary projects the negotiation of the main outputs and work processes is key to success. This is particularly true for *Technology Enhanced Learning (TEL)* projects, since these projects face the challenge of bridging a potentially huge gap between learning theories and the enabling technologies [1]. Most R&D projects funded by the TEL unit in FP6 and FP7—most notably specific targeted research projects and large-scale integrated projects[1]—have aimed to develop novel TEL approaches by evaluating and field-testing new ideas with software prototypes. Since the TEL community is small and tightly connected [2], it is a natural consequence that the consortia often follow approaches that have succeeded in previous or similar projects. These might include practices like rapid prototyping, continuous integration, agile development, and others [3]. As

[1] For an overview see http://www.learningfrontiers.eu/?q=project_space

C. Rensing et al. (Eds.): EC-TEL 2014, LNCS 8719, pp. 404–409, 2014.

a complicating factor, collaborative research projects often face the challenge of an understaffed, distributed developer community that needs to align development efforts with needs from application partners and input from researchers. To resolve the forces in this area of tension, this paper presents a blueprint for software engineering in TEL research projects. The paper focuses on large-scale projects that require deployment and integration of scalable solutions and development processes based on available technologies, end-user involvement, and informed decision making.

The roots of the research presented in this paper lie in the technical leadership in two inherently different large-scale integrated projects in TEL, namely ROLE[2] and Layers[3]. While in ROLE the task was to develop and deploy a platform for responsive open learning environments supporting self-regulated learning (e.g., [4]), the challenge in Layers is to develop and deploy scalable, flexible and rapidly deployable infrastructures for informal learning [5] in two large SME clusters in the UK (health care) and Germany (construction). There is negligible overlap in functional requirements in these two projects, yet both exposed a considerable amount of common challenges and non-functional requirements related to the software architecture as well as the development and integration processes. Key architectural challenges are the need to make early architectural decisions, and to build the basis for flexible, customizable, traceable, and scalable solutions. Key development and integration challenges include the distributed developer community, the danger of a lack of stakeholder commitment, swift provision of software engineering infrastructure, and establishment of development practices.

Many of these challenges will be relevant to other large-scale R&D projects in TEL and beyond. It is understood that TEL innovation processes need alignment with research and practice that can build on previous findings [6]. In this spirit, to preserve and spread effective practices in TEL development we present in Section 2 successful solutions deployed first in ROLE and then adopted and refined in Layers in the form of a blueprint for software engineering in TEL to be reused in other projects that offer comparable scope and challenges. In the last section we provide lessons learned and conclude the paper.

2 Distilling the Blueprint

The blueprint tackling the challenges presented in the previous section integrates three core perspectives, as explained below and illustrated schematically in Fig. 1.

2.1 Stakeholder Perspective

Typical collaborative R&D projects involve goals and strategies of researchers, application partners, and developers. Put provocatively, researchers aim to publish high-impact papers; application partners want ready-to-use, custom tailored

[2] `http://role-project.org`
[3] `http://learning-layers.eu`

apps; and the developer force is distributed, understaffed, and members often pursue PhD research. It is obvious that this perspective will impose a divergent force upon most projects. To bundle the development capacity we propose to establish a *Developer Taskforce* that acts largely autonomously when negotiating and realizing short- to mid-term development objectives within the development roadmap framed by researchers, co-design partners, and project management. This development community may consist of project internal developer groups, involved student groups—e.g., in project-based learning (PBL) courses offered by project partners—and external groups like local OSS communities, established companies and emerging start-ups.

For making decisions with a potentially project-wide effect during the software engineering process, we propose to establish a governing body (in Layers the *Architecture Board*), in which relevant stakeholders are represented, and which makes binding decisions based on suggestions and input by project partners.

2.2 Continuous Integration Perspective

Integration shall impose a convergent force upon the project. It has several dimensions. First, there is the integration with the application partners, which is

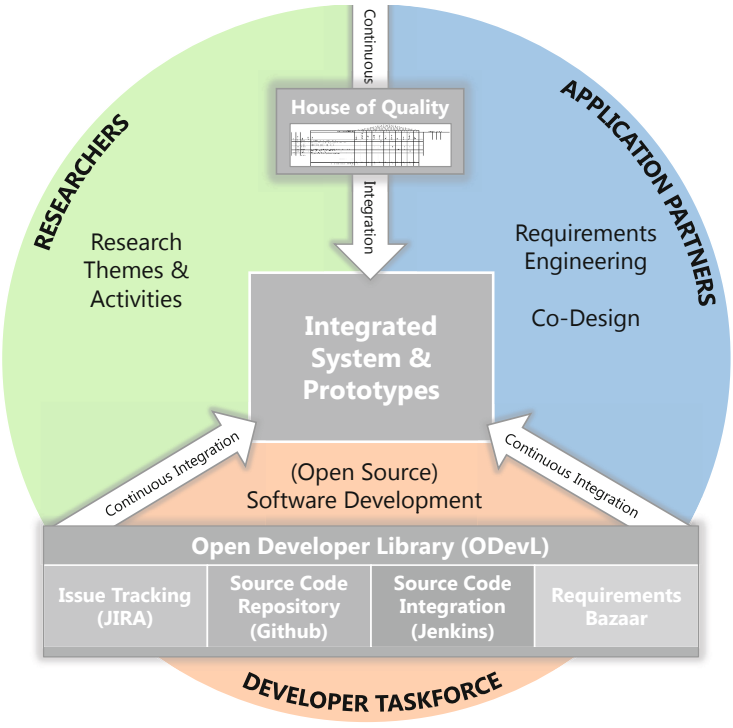

Fig. 1. Perspectives, activities and instruments in the software engineering blueprint

expressed in a systematic requirements engineering process. Especially in TEL projects, requirements engineering methodologies and tools must support an efficient communication and negotiation among the different stakeholder groups involved. Traditional requirements engineering techniques involving face-to-face interaction with end-users are resource-intensive [7], involving traveling costs and high preparation efforts. Such costs are justifiable in early project stages, when working processes are not yet established. However, in later project stages, such costs can be avoided by the use of effective and easy-to-use tools. As an open requirements platform we adopted Requirements Bazaar (see below). Second, there is the server-side integration, which facilitates the offering of unified services and tools developed in a distributed community. Third, there is the client-side integration that shall generate integrated end-user tools consuming common services. In addition there is continuous integration at the source code level (see below).

Integration efforts shall be driven mainly in *Co-Design Teams*, which subsume application partners, designers and developers as a permanent subgroup of the project working on a particular design challenge or design idea. They develop usage scenarios, wireframes, mockups, and first prototypes. Also they formulate demands for the further development of prototypes. Since there is often a wide variety of different design ideas in a project, it is of high importance to find the commonalities and key elements across these ideas. The assessment of alternative solutions shall be facilitated using the *House of Quality* (HoQ) approach [8], which pitches requirements against features offered by products to be adopted, developed and integrated. HoQ originated in Japan in 1972, and it was adopted during the 1980s by large U.S. companies such as Ford, Xerox and AT&T for their product development activities. It has proven to be a valuable instrument also in the Layers R&D context.

To make sure that co-design teamwork aligns with the overall project objectives, we propose to host project-wide *Integration Conferences* that focus on integrating the plethora of outputs from the co-design and development threads. This includes integration of theory, data, services, user interface, development processes and external tools.

2.3 Instrument Perspective

The key to let convergence win over divergence is the set of instruments deployed and offered to project partners to support continuous integration at all levels and in all areas where stakeholder interests meet. In particular we propose the *Open Developer Library (ODevL)* approach, which acts as a virtual hub offering tools and practices facilitating the Developer Taskforce:

- Source Code Repository: In a distributed software engineering process it is key for developers to have a reliable source code management in place. There are freely available solutions available. We suggest GitHub[4], since it is

[4] `http://github.com`

integrated with other tools and offers sophisticated support for a distributed development (e.g., full-fledged local repositories).
- Source Code Integration: Build-level code integration tools support and access source code management systems, e.g., to obtain code for regular integrated builds. The build process can be tailored to specific needs regarding the build triggers and the build process itself. In the blueprint we propose the open source solution Jenkins. Regular builds will offer timely notifications for those people in charge of integration.
- Issue Tracking: Issue trackers allow the management of software issues, including bugs, feature requests, etc. Most issue tracking software packages offer more than pure issue listings. The proposed system JIRA also offers project management tools (e.g., a Kanban board [9]), and integrated source code management tracking. The issue tracker is two-way integrated with the Requirements Bazaar.
- Requirements Bazaar: An open community toolkit that allows requirements and ideas generation and prioritization [10]. This is done in a bazaar-like metaphor, enabling negotiating around requirements using posting of artifacts, comments, and votes. This instrument provides a central requirements hub for all stakeholders in the project.

3 Conclusion and Lessons Learned

In this paper we have unfolded a blueprint for software development in large-scale R&D projects in TEL. We claimed that these projects often expose common challenges regarding the software architecture and the development process. We presented activities and instruments that allow approaching these challenges with proven tools and established practices in real-world developer communities.

The experiences about the software engineering process reported here are the synopsis of years of supporting TEL in creating and sustaining results within and beyond the scope of a funded research project. Some lessons learned include the insight that even with a good project plan, requirements and priorities of stakeholders change over time. These changes must be reflected in the process to avoid contracting the "not-invented-here" syndrome. Also it is advisable to make a strong commitment towards open source development, and to push the involvement of external developer communities to support the typically small project-internal developer force. A related issue which needs to be dealt with early is the OSS licensing models to be adopted. Also, time is an essential factor. It is paramount to start very early in providing the development infrastructure and grow continuously. In that sense, pre-configured development infrastructures like the blueprint presented in this paper, which can be rapidly deployed for a new project are a good choice. Last but not least, we emphasize the pivotal role of social factors in R&D projects; the success of deployed instruments will always depend on how well resistances can be resolved and how deeply stakeholders embrace the opportunity provided by such an infrastructure.

The actual resulting architecture was not explained in detail here since it will have limited general value outside of the Layers project. It is described in detail in [11]. With this paper we aimed to preserve previously and currently successful practice in a way that allows future TEL project consortia to plan and execute their software engineering processes, either as a dedicated work package, or by picking a subset of the instruments and activities that are tailored to their needs. We want to establish a culture of sharing and continued refinement of software engineering best practices in TEL across EU funded projects.

Acknowledgments. This research was co-funded by the European Commission through the FP7 Integrated Project "Learning Layers" (grant no. 318209).

References

1. Noss, R.: 21st Century Learning for 21st Century Skills: What Does It Mean, and How Do We Do It? In: Ravenscroft, A., Lindstaedt, S., Kloos, C.D., Hernández-Leo, D. (eds.) EC-TEL 2012. LNCS, vol. 7563, pp. 3–5. Springer, Heidelberg (2012)
2. Derntl, M., Klamma, R.: The European TEL Projects Community from a Social Network Analysis Perspective. In: Ravenscroft, A., Lindstaedt, S., Kloos, C.D., Hernández-Leo, D. (eds.) EC-TEL 2012. LNCS, vol. 7563, pp. 51–64. Springer, Heidelberg (2012)
3. Sommerville, I.: Software Engineering, 9th edn. Pearson, Boston (2011)
4. Carneiro, R., Lefrere, P., Steffens, K., Underwood, J. (eds.): Self-regulated Learning in Technology Enhanced Learning Environments: A European Perspective. SensePublishers, Rotterdam (2011)
5. Ley, T., et al.: Scaling informal learning: An integrative systems view on scaffolding at the workplace. In: Hernández-Leo, D., Ley, T., Klamma, R., Harrer, A. (eds.) EC-TEL 2013. LNCS, vol. 8095, pp. 484–489. Springer, Heidelberg (2013)
6. Beyond Prototypes consortium: Enabling innovation in technology-enhanced learning. Final project report (2013), `http://is.gd/bpinnovtel`
7. Law, E.L.-C., Chatterjee, A., Renzel, D., Klamma, R.: The Social Requirements Engineering (SRE) Approach to Developing a Large-Scale Personal Learning Environment Infrastructure. In: Ravenscroft, A., Lindstaedt, S., Kloos, C.D., Hernández-Leo, D. (eds.) EC-TEL 2012. LNCS, vol. 7563, pp. 194–207. Springer, Heidelberg (2012)
8. Hauser, J.R., Clausing, D.: The House of Quality. Harvard Business Review 66(3), 63–73 (1988)
9. Anderson, D.J.: Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press (2010)
10. Renzel, D., Behrendt, M., Klamma, R., Jarke, M.: Requirements Bazaar: Social Requirements Engineering for Community-Driven Innovation. In: 21st IEEE International Requirements Engineering Conference (RE), pp. 326–327. IEEE (2013)
11. Derntl, M., Klamma, R., Koren, I., Kravcik, M., Nicolaescu, P., Renzel, D., et al.: Initial Architecture for Small-Scale Deployment. Learning Layers Deliverable D6.1(2013), `http://is.gd/LayersD61`