Tools for Teams: A Survey of Web-Based Software Project Portals

Jordi Cabot and Greg Wilson
Dept. of Computer Science
University of Toronto
{jcabot,gvwilson}@cs.utoronto.ca

August 31, 2009

Abstract

Web-based project portals are at the heart of modern software development, but have been studied much less than individual-oriented desktop tools like integrated development environments. In July–September 2008, we compared several popular portals and interviewed their developers in order to find out what needs they were intended to satisfy, how their feature sets had been chosen, and how they were likely both to evolve and to shape the evolution of distributed software development. Our key findings are that (1) most portals are strongly biased toward agile methods (and in fact may primarily exist in order to support agile development in geographically distributed teams), (2) the teams building these portals do not use agile methodologies themselves, but instead rely on informal collections of best practices, (3) as elsewhere, there is a clear trend toward hosted services, and (4) none of the portals studied provided any kind of support for modeling or user experience design, and only one directly supported test management.

1 Introduction

If you pick a software development project at random, the odds are good that its source code, bug reports, mailing list archives, and so on reside in a web-based portal. Whether it's a hosted service like SourceForge[1] or an installed system like Trac[2], in many ways that portal *is* the project: individual developers might come and go, but the portal's contents live on, and with them, the project.

Despite their widespread use, software project portals have been studied much less than individual-oriented tools such as integrated development environments (IDEs) [6]. To begin to rectify this, in July–September 2008 we examined the features of several representative portals and interviewed their

[1]http://sourceforge.net
[2]http://trac.edgewall.org

developers. Our research goals were to determine what needs those tools were intended to serve, how their feature sets had been chosen (e.g, how their developers had translated perceived needs into functionality) and how they were likely to evolve. To our knowledge, this is the first general study of web-based software project management tools, though their importance was pointed out in [1]; for comparisons of pre-web tools see [5].

We begin below by describing what a minimal portal might contain. We then explain how we selected and collected data about eleven specific portals. The remaining sections analyze our findings and speculate about the likely future evolution of software portals. It is important to note that for every system we included there were several others that we left out. Our choices should therefore not be taken as a judgment or recommendation.

2 What's (In) a Portal?

Just as programmers may use "editors" ranging from Notepad through Vi to Eclipse, so too do online project management tools range from shared to-do lists to multimedia collaborative environments. To be called a software project portal, however, we felt a system must have at least a few specific features. The first is some kind of task management, such as a to-do list, bug tracker, or full-blown workflow management system. For us, this is what makes project management tools distinct from other kinds of groupware.

The second core feature is a document repository. Systems aimed primarily at developers typically integrate a third-party version control system such as Subversion[3], while systems aimed at broader audiences usually include a simpler file upload mechanism. Both allow stakeholders to share and modify content, and to see who else has done so.

Conversational tools are third on the list. These include email, chat, wikis, blogs, bulletin boards, and other ways for stakeholders to communicate and coordinate. A growing number of systems present content in several forms, e.g., by providing RSS notification of updates to bulletin boards, or wiki transcripts of chat conversations.

Last but not least is search. One reason people use portals is to link the disparate pieces of information that comprise a project; one of the benefits of doing this is that a single query can then find email messages, version control check-in comments, and wiki pages all at once.

Other components tend to revolve around these four core capabilities. For example, some include a calendaring tool so that stakeholders can schedule work items (e.g., by grouping them into iterations, and then binding those iterations to target delivery dates). Others include time-tracking tools to help consultants with billing and schedule management; a tagging mechanism; a report generator; a web API for remote scripting and administration; customizable fine-grained access control; or a continuous integration back-end to automatically re-build and re-test code. As with document repositories, these may be more or less

[3]http://subversion.tigris.org

developer-centric, i.e., may require greater or lesser degrees of technical sophistication to use.

3 Gathering Data

Since dozens of portals exist[4], we had to decide which to include in our study. Our initial pool of candidates consisted of systems that one author had examined when selecting a portal to use in undergraduate software engineering courses. We enlarged the pool by conducting web searches for keywords prominent in those systems' descriptions of themselves, by including systems that previously-added systems compared themselves too, and through word-of-mouth referrals from colleagues and attendees at various conferences. The list was then filtered again according to the following criteria:

1. The system had an active developer community.
2. It was being used by people other than its developers.
3. Many or all of its users were software developers using it for software projects.
4. Someone central to its development was willing to be interviewed. We did not request contact with specific people but instead asked for interviewees playing a key role in the tool conception and development (founders, CIOs, chief developers or similar). To our pleasant surprise, only two of the groups we approached declined to take part [5] and thus were removed from the study.

Our final list of portals was:

- Acunote[6]
- Assembla[7]
- BaseCamp[8]
- DotProject[9]
- Google Code[10]
- IBM Jazz[11]

[4]Or partially exist, or may have existed but are now moribund.
[5]Due to confidentiality issues we cannot identify these two tools nor provide more detailed information about the people we intervieweed
[6]http://www.acunote.com/promo
[7]http://www.assembla.com
[8]http://www.basecamphq.com/
[9]http://www.dotproject.net/
[10]http://code.google.com/hosting/
[11]http://www.jazz.net

- Mingle[12]
- Rally[13]
- SourceForge[14]
- Trac[15]
- VersionOne[16]

Having selected these systems, we itemized the features offered by each in the areas of authentication, collaboration tools, document management, work tracking, and time management, and where feasible gave it a test drive to validate and fill in gaps in their self-descriptions.

We then set up an interview with a key member of its development team. We originally intended to conduct hour-long structured interviews, but interviewees perceived their systems and markets in such different terms that no question script made sense for everyone. We did, however, cover all of the following areas:

- When and why did you start developing the portal?
- Why did you decide to create a new portal instead of adopting or extending an existing system?
- What user needs did you set out to satisfy?
- What was your initial business model?
- Does the portal favor any software development process?
- What software development process do you use yourselves?
- Who are your typical users, and how do you determine their requirements?
- How do you determine user requirements?
- How do you rank affordability, extensibility, flexibility, reliability, security, and usability as development drivers?
- Is it open source, closed source, or some mix of the two?
- Is it free to use or not? (This is orthogonal to the open/closed questions, since some groups provide a free service on top of closed-source software, while others offer a stripped-down version under an open license and a premium version with extra closed-source features for a fee.)

[12]http://studios.thoughtworks.com/mingle-project-intelligence
[13]http://rallydev.com
[14]http://www.sourceforge.net
[15]http://trac.edgewall.org
[16]http://www.versionone.com/

- Is the portal deployed as a hosted service, do users have to install it, or are both options available?
- Can third-party tools be plugged in by end users, and if so, how?
- How do you see the portal and the market evolving over the next couple of years?

4 Portal Evaluation

This sections presents the results of our portals evaluation. As we expected, a core group of portals fit the generic model described in Section 2 and share a basic feature set (see Section 4.1) but, at the same time, each tool has its own distinguishing characteristics, that basically reflect differences in their target markets (Section 4.2).

Therefore, as well as making sure it has the right feature set, every developer intending to use one of these portals must ponder the following questions: Is the portal biased toward a particular development process or can it (easily) be used to support many different ones? Is it open source, closed source, or some mix of the two? Is it free to use or not? Is the portal deployed as a hosted service? Does the portal manage user identities itself? Can third-party tools be plugged in by end users using a well-defined API or is some combination of reprogramming and screen scraping needed to create mashups?, to make sure the portal suits his/her needs. It is also worth to note that even if two tools share the same set of features, the way they enable or emphasize them may result in a completely different usage (and with a different purpose).

4.1 Common characteristics

All portals satisfy the two main conditions stated in section 2: they all offer some kind of ticketing system and a repository for managing the project tasks. Support for communication tools (third condition on the list) is also high: all portals offer email alerts and RSS feeds to keep team members informed about project progress. Search capabilities turn out not to be so widespread, basic (predefined) searchs within the portal data are common but only a few portals offer advanced search features (see the next section).

Besides this core features, other aspects were found to be shared by many portals. All portals offer a role-based access control system with portal-managed identities, external authentication (e.g. supporting LDAP as Mingle and IBM Jazz do) or both. All also allow users to group tasks into milestones (even if sometimes milestones are defined just as a specific type of task). Finally, most allow multiple projects to be hosted in one portal (the exception is Trac, which only supports one) and let managers to roll up information across multiple projects.

4.2 Portals' distinguishing features

Beyond the pricing and licensing, the major differences between them reflect differences in their target markets. For example, Rally and VersionOne have been built by and for developers. They are used by some organizations' marketing and customer support groups, but their typical users are more likely to use CruiseControl than PhotoShop or Excel. BaseCamp, on the other hand, primarily targets small organizations like print shops that are staffed by non-programmers working on short- or medium-length projects; software developers are a minority of its users (though there are still many of them). It therefore offers an easy-to-use file upload and synchronization service rather than a full-blown version control system. In what follows, we briefly describe each portal focusing on their main distinguishing features.

The two leading commercial portals, Rally and VersionOne, match each other almost feature by feature. Both target medium-to-large agile teams, and both companies offer extensive training and consulting as part of the sales and deployment process to, in their own words, help groups using more "traditional" development lifecycles "go agile". They support template-based project creation with customization so that teams can quickly start using local adaptations of different agile processes, and project and task hierarchies with roll-ups for reporting. Rally decomposes user stories (i.e., a very high-level specification of a system requirement) into tasks, tests, and defects, while VersionOne allows project managers to redefine the schema of work items (i.e., the fields to be recorded for each task) at will.

These core features are complemented by predefined connectors to legacy ticketing systems and build tools. In particular, both accept Subversion and Microsoft Team Foundation Server as source code management (SCM) tools and JIRA[17] and Bugzilla[18] (among others) as defect management systems. Both also offer web services-based APIs to create additional connections and mash-ups.

Mingle is younger than these two systems, but similar in most respects: it uses Subversion and Perforce for SCM, and programmers can manually link cards (i.e., work items) and source code by including the card ID number in the commit comments of the code. One distinguishing feature is that requirements, tasks, and bugs are represented as hierarchical cards whose types and attributes can be adapted to the needs of each project. Another is that report generation is based on an SQL-like query language, which makes it easy for users to set up filters so that they can track what they are most interested in.

Acunote and Assembla are aimed at small-to-medium teams. Acunote offers a simple task management system for SCRUM-based projects[19]. Unlike some systems, tasks can span several SCRUM sprints, which its creators feel reflects how projects unfold in real life. It emphasizes the use of time-tracking and burndown charts (type of chart that shows the remaining work in the current

[17]http://www.atlassian.com/software/jira/

[18]http://www.bugzilla.org

[19]Scrum [8] is an agile development process that emphasizes short incremental development cycles (called sprints)

sprint) for monitoring progress. Tasks can be hierarchical but their structure is fixed. Assembla is a hosted version of Trac and Subversion supplemented with custom permission, ticketing, and team management modules, though it also offers built-in milestone and ticketing systems. It also includes a job/recruitment section and a time tracking system, which is important to the independent contractors and job-specific teams it is meant to support. In particular, the executive at Assembla commented that Assembla is designed to support the whole development process, includig the payment process and that time tracking is one of their most popular tools since it is how managers decide how to pay people.

The granddaddy (and inspiration, as many interviewees acknowledged) of all portals is SourceForge[20], which offers a free hosting service for open source projects. Unlike the agile-oriented tools above, it is neutral with respect to development process—in fact, the interviewees reported that most of the projects it hosts don't really have a "process" per se. Despite including a bug tracker and other project management tools, it is primarily used for project releases, hosting version control repositories, and managing its mailing lists.

According to the interviewee from Google, Google Code was created in part to provide an insurance policy for the open source community in case SourceForge ever shut down. Its design is loosely based on Google's internal development tools and practices: for example, it is the only portal we examined that natively supports code review. Authentication is managed through GMail; like messages in that system, tasks have a predefined set of fields, but users can attach arbitrary labels to classify each task and use Google's free-text search technology to search this metadata. Projects can be connected to Subversion repositories but no other predefined integrations are available. A single issue tracking system manages software defects, change requests, technical-support requests, and development tasks.

Trac is a self-hosting portal that was originally designed as a replacement for CVSTrac. Each Trac installation can host a single project, though an extension allows a single project to span multiple version control repositories. Work is defined by tickets and milestones; the attributes of tickets can be redefined, but unlike other tools, this requires some programming effort. While it is very popular, its development has slowed in the past few years: each new release has largely been the effort of one dedicated volunteer, who usually then moves on to other interests (according to the interviewee from Trac: "We cannot really say that we plan ahead for some big features and they get done"). Like SourceForge, it is neutral with respect to the development process.

BaseCamp is an example of how a system aimed at a broader audience can be used in software development. It offers a file upload system instead of a version control system, to-do lists instead of tasks and tickets, and makes it easy to add external stakeholders with limited permissions to projects. On the other hand, it deliberately does not offer more sophisticated tools such as burndown charts,

[20]Though credit should also go to Richard Hipp's CVSTrac (http://www.cvstrac.org/), the first widely-used self-hosted web-based portal.

and instead of providing connectors to defect management and control quality tools, it focuses on integration with third-party invoicing, billing and accounting applications. Its developer explicitly aimed to support certain collaboration and project management practices, and to keep it so simple that, in his own words, "...learning to use the application was not a project in itself."

The self-hosting portal DotProject lies somewhere between BaseCamp and Trac: it has a fine-grained Access Control List (ACL) based permissioning system and hierarchical task management, but only limited support for releases or milestones (milestones are just a special kind of tasks), and does not attempt to manage code.

Finally, IBM's new Jazz project is more an extensible technology platform than a portal (e.g. the Rational Team Concert, the IBM collaborative software delivery environment, is built upon it). Its design draws heavily on IBM's experience with Eclipse, which is also a platform that happens to be very good for building IDEs. Unlike the other systems we studied, Jazz has a custom two-level version control system: each developer can commit changes to a local repository, which can then be dumped into the shared public one. It is also the only portal that allows multiple branches (called "development lines") for the same project. As fully-distributed version control systems such as Git and Bazaar become more popular, we expect to see other portals move to a similar model.

5 General Findings

Beyond the descriptive comparison of the previous section, our comparisons and interviews uncovered some unexpected patterns. The first was that portals mainly target agile teams: five of those we studied said so explicitly, and none explicitly encouraged more traditional development processes. We do not believe this was due to selection bias, though the co-emergence of SourceForge (which inspired several tools) and agile methods may be a factor.

The emphasis on agile methods had a strong influence on features offered, as many of the portals were built expressly to allow agile teams to scale up and (crucially) spread out geographically. We believe this explains why they emphasize asynchronous communication (e.g., bulletin boards) over synchronous (e.g., chat).

Three of these companies also offer consulting services to help customers who aren't agile make the transition. As the Rally executive said:

> You go back three years, we were doing a lot of explaining what agile is to our customers, and helping them understand why this is a good idea. There's a lot less need — now there's a lot more of the "how".

Those same companies acknowledge that they have included some features in their tools that aren't strictly "agile" in order to support big or cross-disciplinary teams. All said that simplicity was crucial, but felt that this pragmatism reflected the growing maturity of the agile community. For example, one interviewee said people had realized that methodologies like SCRUM [8] weren't really about developing software, they were about identifying and fixing problems in software development processes. It was therefore natural for different groups to evolve different "best" practices, and for tools to follow suit.

Another noteworthy point was that almost all of the groups we interviewed acknowledged that they didn't use any well-defined process themselves (not even the agile methods they preach). When asked, "Do you use XP [2]?" or "Do you use SCRUM?", they invariably replied that their developers used a mix of best practices that didn't strictly adhere to any published rulebook. None of the interviewees were defensive about this; all clearly believed that they had above-average developers who could be trusted to use pair programming, test-driven development, and whatever else was appropriate in context. This emphatically does *not* mean that their processes were chaotic: in all cases there was close and frequent coupling between development on one hand and requirements gathering and feature prioritization on the other. However, the day-to-day mechanics of actually producing high-quality code was trusted to developers and their consciences. It remains to be seen whether the users of the tools do follow specific agile methods or, as the tool developers', they just use their own mix of agile practices.

All of the people we interviewed also acknowledged that their own needs had been and were a major force in the tool's evolution. This is unsurprising for free open source tools like Trac, whose developers were also all users, but the vendors of commercial systems (both closed and open source) felt that their developers' experiences were "representative" enough to be mined for features. Mingle, Rally, and VersionOne also all felt that they were ahead of the curve in adopting the agile practices that their tools were intended to support, which also legitimized the recycling of their own experiences instead of following a more formal requirements elicitation process.

One clear trend in the portals we studied was providing a hosted service. SourceForge pioneered this, and only Mingle (among the recent portals), does not primarily use "software as a service" (SaaS) model[21]. DotProject and Trac did neither primarily use this option but we believe their choice of a more traditional customer-hosted model can be attributed to them being (comparatively) older systems, and because volunteer-authored systems lack the resources to provide scalable hosting. Mingle's authors, on the other hand, believe that enterprise clients would not want to put their data on someone else's servers, though competitors such as VersionOne and Rally have demonstrated that many will.

All of the vendors who focus on SaaS claimed that not having to install and configure the portal lowered their customers' costs, and allowed them to roll out new features in small, incremental steps. Some also pointed out that it made fixing bugs (particularly security bugs) simpler and more reliable, since in-house experts could do it for almost all customers in a single step.

[21] "Primarily", because several commercial vendors offer a customer-hosted option, though uptake is slight.

A corollary to being a hosted service is the use of a subscription model. Most portals charge by the month based on the size of projects [22], as measured by number of users and/or file upload and storage requirements. In most cases, month-by-month payments actually cost clients more than an annual subscription would, but as two interviewees independently pointed out, it's easier to get approval for a few dollars a month, with the illusion of being able to cancel, than it is to get the accounting department to sign off on a one-time expense of several hundred dollars.

We say "illusion" because none of the portals we examined makes it easy for users to export their projects for backup or use elsewhere. The contents and history of version control repositories can be relocated using third-party tools[23], but the tickets, wiki pages, and other content stored in portals can at best be dumped as a big blob of XML for parsing and interpretation. The high cost of switching means that customers are effectively locked in once they select a portal.

What the portals we studied *don't* include is equally important. None of those we studied supported modeling or user experience design, and only one (Rally) directly supported test management. Several advertised requirements management and traceability, but in practice this turned out to be nothing more than specially-labelled tickets manually linked to code or simple product backlogs. Finally, integration between portals and IDEs was weak (with IBM Jazz being a notable exception). This cannot be blamed on technology—the Mylyn plugin for Eclipse[24] (which integrates information from Bugzilla, Jira, Trac, and other systems) has been available for several years, and equivalent connectors for other tools would be straightforward to build. We suspect this is simply a matter of an idea not having reached its "tipping point", and predict that IDE-to-portal linkages will be commonplace within three or four years.

6 The Future

In the short term we expect project portals to focus on improving the functionality they already offer. As this paper was being written, for example, SourceForge announced that it will host virtualized versions of third-party open source applications so that teams can use them in a secure fashion without having to set them up. Other issues mentioned by several interviewees included scaling the portals to handle larger teams and heavier-weight development processes and better support for the needs of "non-geeks" (e.g., marketing and sales staff). We also expect to see much better support for peripheral awareness, such as the task context model provided by Mylyn [7].

In the longer term, interviewees speculated that project portals would merge or integrate with social networking tools such as LinkedIn[25] and personal life

[22]Many of them offer free accounts for open source or non-commercial projects.
[23]E.g., `svnadmin dump` and `svnadmin load` for Subversion.
[24]http://www.eclipse.org/mylyn/
[25]http://www.linkedin.com

management tools such as Google Calendar[26]. People already put much of the information project managers require, such as skills and availability, into these systems, and are unlikely to duplicate that information manually. Thanks to the open APIs offered by most of them, web-based portals are already becoming software buses for aggregating all kinds of project-relevant data; we predict that in the end, the system that does the best job of balancing ease of configuration with privacy protection will win.

Even before this happens, portals are adopting "Web 2.0" ideas as quickly as they can. For example, most tools support alerting of changes by means of RSS feeds, many offer some kind of wiki related to the project and some portals integrate a (limited) tagging mechanism and allow search across projects based on those tags' "folk" semantics. Others have expertise advertising and rating systems to help members of an organization locate useful people [4]. Use of social networking may facilitate team and improve the quality and quantity of communication channels between team members [3], resulting in higher-quality software [9].

Some portals are also considering the addition of modules for risk analysis, finance, and/or HR (e.g. the interviewee from Assembla pointed to the development of their recruiting system module as a distinguishing feature). Some of these extensions will likely be driven by new laws enforcing stricter policies for software development for government agencies (e.g., the stringent auditing requirements of the Sarbanes-Oxley Act in the United States). This will also drive portal vendors toward the software bus model, since organizations may need to satisfy very different jurisdictional requirements.

Finally, while this is still a young market, tools are already expanding their initial niche market (e.g. going from a specific agile process to a generic support for all kinds of agile-like processes and even to an initial support for non-agile teams) and competition for high-value niches is fierce, and we believe that it is likely that mergers (and failures) will soon leave only a small number of dominant players.

7 Threats to validity

Some threats to the validity of this study have been avoided by complementing the tools' evaluations with the set of interviews with key members in the tool conception and development. This has given us a better understanding of each tool and of its development and commercialization context.

Nevertheless, there are still some threats to the validity and generalizability of our conclusions that should be considered when reading this paper. First of all, the huge number of tools prevent us from studying all of them, and thus, our results are biased by the tools we finally selected. Second, there is also a bias in the opinion of all interviewees, always favourable to this kind of tools (since all of them were directly involved in their commercialization/development). An additional possible bias is the fact that we did not formally include in the study

[26]http://www.google.com/calendar

the opinions of the tools' users (beyond our own experience[27] and the informal feedback comments from colleagues we contacted in different kinds of companies and locations).

Despite these potential biases, we believe that our tool selection process and the key role of our interviewees in their companies makes our results representative enough to be useful for all software engineers thinking in adopting a web-based project management portal in their new development project.

8 Conclusions

Web-based project management portals are now the heart of many mature software development projects. In this paper we have presented the results of our study of these tools' origins, features, use, and likely evolution. The most noteworthy results are the fact that the creators of tools meant to support agile processes do not strictly adhere to those practices themselves, but instead trust their developers to use good practices when and as their own best judgment dictates, and the general lack of support for non-code-centric activities such as requirements management, modeling, user experience design, and test management.

Both observations raise questions about the focus and direction of much current software engineering research. In particular, we now wonder about the real importance of requirements elicitation and structured development process in the success of a development project (at least when all team members can be classified as expert software engineers). We also wonder if other niches of tools are developed in a similar way.

Other possible research directions derived from this work could focus on understanding how these portals are really used. For example, it would be interesting to analyze the ways in which ticketing systems are configured by their users (i.e. what information do they track, who (and when) enters that information, whether the information in the portal is enough to carry out the tasks or teams still rely on personal communications not electronically recorded in the portal, and so forth) in order to learn more about their actual development processes. This could influence how the next generation of portal tools is shaped to better satisfy the real users' needs.

9 Acknowledgements

We wish to thank everyone who agreed to be interviewed for this study, and apologize in advance for any over-simplifications in this paper. Jordi Cabot's work was supported by the Catalan Government (Grant 2007 BP-A 00128) and the Spanish Ministerio de Ciencia y Tecnologia (TIN2008-00444 project).

[27]Both authors are active users of this kind of tools and one author (Wilson) created a mini-portal suitable for classroom use based on Tract.

10 License

References

[1] M. Alshawi and B. Ingirige. Web-enabled project management: an emerging paradigm in construction. *Automation in Construction*, 12(4):349 – 364, 2003.

[2] K. Beck and C. Andres. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2004.

[3] K. Chang and K. Ehrlich. Out of sight but not out of mind?: Informal networks, communication and media use in global software teams. In K. A. Lyons and C. Couturier, editors, *CASCON*, pages 86–97. IBM, 2007.

[4] K. Ehrlich and N. S. Shami. Searching for expertise. In M. Czerwinski, A. M. Lund, and D. S. Tan, editors, *CHI*, pages 1093–1096. ACM, 2008.

[5] T. L. Fox and J. W. Spence. Tools of the trade: A survey of project management tools. *Project Management Journal*, 29(3):20–27, 1998.

[6] S. Helsen, A. Ryman, and D. Spinellis. Software development tools: Guest editors' introduction. *IEEE Software*, 25(5):18–21, 2008.

[7] M. Kersten and G. C. Murphy. Using task context to improve programmer productivity. In M. Young and P. T. Devanbu, editors, *SIGSOFT FSE*, pages 1–11. ACM, 2006.

[8] K. Schwaber. *Agile Project Management with Scrum*. Microsoft Press, 2004.

[9] G. Valetto, M. E. Helander, K. Ehrlich, S. Chulani, M. N. Wegman, and C. Williams. Using software repositories to investigate socio-technical congruence in development projects. In *MSR*, page 25. IEEE Computer Society, 2007.