A Planning Poker Tool for Supporting Collaborative Estimation in Distributed Agile Development

Fabio Calefato
Dipartimento di Informatica
University of Bari
Bari, Italy
calefato@di.uniba.it

Filippo Lanubile
Dipartimento di Informatica
University of Bari
Bari, Italy
lanubile@di.uniba.it

***Abstract*— Estimating and planning are critical to the success of any software project, also in the case of distributed agile development. Previous research has acknowledged that conventional agile methods need to be adjusted when applied in distributed contexts. However, we argue that also new tools are needed for enabling effective distributed agile practices. Here, we present eConference3P, a tool for supporting distributed agile teams who applies the planning poker technique to perform collaborative user story estimation. The planning poker technique builds on the combination of multiple expert opinions, represented using the visual metaphor of poker cards, which results in quick but reliable estimates.**

Keywords- distributed; agile; estimation; planning

I. INTRODUCTION

Software estimation and planning activities aim to create meaningful cost and schedule estimates for a project. The ability to accurately estimate the time and cost for a project is a key factor to its successful conclusion. Hence, estimating and planning are critical activities also in the case of distributed agile development. Unfortunately, agile and distributed development practices are so different that, when blended together, the key characteristics of the former exacerbate the challenges intrinsic to the latter, creating a set of brand new challenges. In fact, as any agile method, agile planning is based upon intense interactions among individuals and thus, it emphasizes the need for frequent informal interaction and communication. On the contrary, in distributed software development communication and interaction are dramatically hindered due to the absence of collocation.

Collaborative software development across distances has become commonplace for a number of years [19]. However, there are still important problems to solve that are strictly related to the effects of distance among the members of a development team [7]. It is well known that a distributed approach to software development increases difficulties related to coordination, control, and communication mechanisms, which are fundamental for any software project. Quite the opposite, agile software development methodologies are based on strong collaboration and frequent informal communication among project members [13]. Among the underlying principles that underpin agile methodologies, personal relationships and direct communication among people are considered as the best resource in a project [4].

There is an increasing interest towards new experimental approaches that aim to combine the specific characteristics of agile methodologies with those of distributed software development [23]. Previous research has acknowledged that conventional agile methods need to be adjusted when applied in distributed contexts. However, we argue that also new tools are needed for enabling effective distributed agile practices. In particular, we argue that tools that provide better communication support are needed in order to cope effectively with the reduction of direct, synchronous interaction.

In this paper, we present eConference3P (eConference Planning Poker Plugin), a tool meant for supporting distributed agile teams who applies the planning poker technique to perform collaborative user-story estimation. The planning poker technique builds on the combination of multiple expert opinions, represented using the visual metaphor of poker cards, which results in quick but reliable estimates. Our tool has been developed as a plugin of the eConference system, a communication platform that connects to either Google Talk or Skype networks and thus, allows the organization of text- and audio-based conferences. Among the other features, eConference3P allows to visually edit user stories and import a backlog from many collaborative development environments such as Google Code, Assembla, Github, Trac, and Jira.

The remainder of this paper is structured as follows. Section 2 discusses in detail the planning poker estimation technique. Section 3 presents our agile planning prototype. Instead, related academic and industrial tools for agile estimation are illustrated in Section 4. Finally, we conclude in Section 5.

II. AGILE ESTIMATION & PLANNING POKER

Before starting a project, whatever agile methodology a team is applying, developers have to deal with iteration planning and, therefore, user story estimation. A user story is a brief description of functionality as viewed by a user or customer of a system. User stories are free-form and there is no mandatory syntax, although they are generally formulated according to the following template: *"As a <role>, I want <goal/desire> so that <benefit>"* [6]. In agile development,

 ISBN: 978-1-61208-165-6

user story estimates are not defined individually by just one developer. Instead, estimates are obtained collaboratively by (part of) the agile team, including those developers who will actually implement the user stories.

The size of user stories can be estimated in story points or ideal days. *Story points* are a relative unit of measure, used to estimate the size of user story by combining the effort, the complexity, and the risk inherent in its development. *Ideal days*, instead, are used to evaluate the size of a story in terms of the amount of time it will take to be fully developed. Both story points and ideal days values are arranged in an estimation scale. Although any sequence might work, Cohn [6] suggests using nonlinear sequences (e.g., the Fibonacci sequence 0,1,2,3,5,8,13…). Because the gaps between values become appropriately larger as the numbers increase, such sequences better reflect the greater uncertainty associated with larger estimates.

To arrive at a shared estimate, agile teams rely on three main techniques: expert opinion, analogy, and disaggregation.

In the *expert opinion*-based approach, experts assign estimates to user stories relying on their intuition. Typically, multiple expert opinions are needed because implementing a system functionality described by a user story requires a number of multidisciplinary skills that normally belong to more than one developer. The expert opinion-based approach has been found to be more effective than others [17].

In the *analogy*-based approach, estimators compare the user stories to be estimated to one or two other stories already estimated before. This approach builds on the fact that humans find easier to estimate relative size than absolute size. Thus, in the typical scenario, if an estimator believes that user story A is twice the size of story B, which was estimated at 5 story points, then A is estimated at 10 points. The comparison can be of course generalized by comparing the size of user story A to a couple of stories already estimated. Obviously, this approach suffers from a cold start problem and, therefore, works better when at least a few user stories have been already estimated.

Finally, in the *disaggregation*-based approach, before estimating the expert splits a large user story into multiple smaller ones, easier to evaluate and compare. In fact, if user story A is much bigger than previously estimated user story B, it would be hard to say that A is fifty times as complex as B. Therefore, disaggregation works well with the analogy-based approach.

An effective way for combining the three estimation techniques is planning poker [6]. In planning poker, each estimator is given a deck of cards with a valid estimate shown on each. A feature is discussed and each estimator selects the card that represents the estimate. All cards are shown at the same time. Then, the estimates are discussed and the process repeated until agreement on the estimate is reached. Typically, a planning poker session is arranged at the beginning of a project, to estimate user stories so that the first iteration can begin. Then, further sessions may be arranged after each iteration to estimate new stories, if any.

Planning poker is an effective way to estimate user stories for at least a couple of reasons. First, it brings together a cross-functional, agile team of experts from different disciplines, whose averaged estimations tend to be more precise than individual scores [14]. Second, it fosters group discussion, as estimators need to justify their scores, which has also been found to lead to better results, especially in case of high amounts of uncertainty and missing information [18].

III. eCONFERENCE3P

eConference3P (see Figure 1) is a tool developed for supporting distributed agile teams who perform collaborative user story estimation by applying the planning poker technique. As shown in the figure, the eConference3P user interface has five main areas. The *message board* is the view that collects all the messages from the discussion that ensues upon any estimation. In particular, the message board is “threaded”, in the sense that messages get stored with respect to the user story that they are related to. We point out that our tool distinguishes the roles available in an agile team and, in particular, between project owner and developers. Relevant notes and decisions, taken through the meeting, are logged in the *decision place*, which can be only

 ISBN: 978-1-61208-165-6

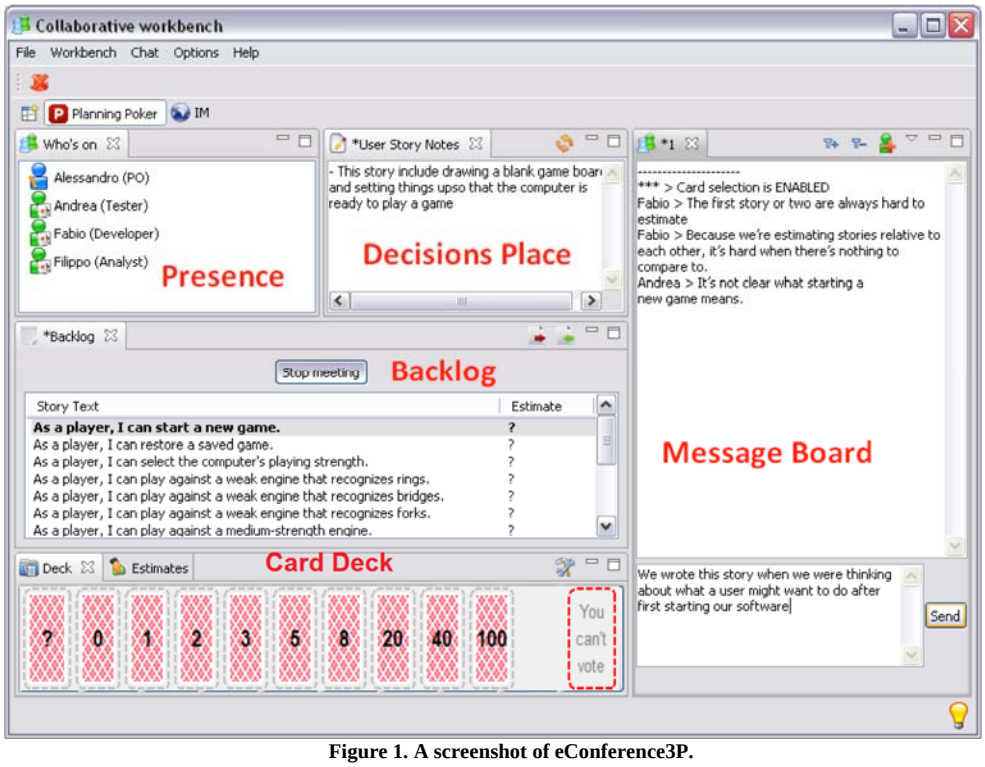

Figure 1. A screenshot of eConference3P.

edited by participants who receive from the project owner the right of acting as scribe. A project owner acts as the moderator of the planning session and, as such, the user interface of the tool enables specific actions to manage the meeting, load user stories, call for and accept the estimates, and grant/revoke rights from other participants. For the sake of space, through the rest of the paper we report the screenshots only from the perspective of the project owner. The *backlog* view allows starting and stopping the meeting, as well as importing and exporting the user stories, which are also listed together with the accepted estimates, once available. The *card deck* view shows the scale from which developers pick the score. Finally, the *presence panel* shows the team members that are participating in the planning meeting, along with their roles (e.g., project owner, developer) and their rights (i.e. to estimate, scribe, chat).

Through the rest of this section, we first describe the architecture of eConference3P, showing how its building components have been arranged together, and then, we discuss in more detail its features.

A. eConference3P Architecture

eConference3P is built around two main components, which are the results of two academic research projects named eConference and AgilePlanner. Both can be run as either standalone applications or Eclipse IDE plugins. In fact, such components could be seamlessly and almost effortlessly integrated because both are rich client applications, developed using the Eclipse Rich Client Platform (RCP) technology, a pure-plugin development platform that is fully extensible by architectural design [8].

eConference [5] is a distributed meeting system, previously developed by our research group at the University of Bari, Italy. Its primary functionality is a closed group chat, augmented with agenda, meeting minutes editing, and typing awareness capabilities. Around this basic functionality, other features have been built to help organizers control the discussion during distribute meetings. eConference can use either XMPP, an IETF standard protocol, or Skype. In the latter case, also VoIP communication is supported.

AgilePlanner [21] is a tool for synchronous, card-based agile planning meetings, developed at the University of Calgary, Canada. AgilePlanner mimics paper index cards as it simulates a whiteboard in a meeting room and utilizes electronic index cards (see Figure 2). AgilePlanner is a client/server application with its own communication protocol. The tool is specifically intended to support distributed agile teams (i.e. work with networked clients), rather than being an offline visual editor for planning artifacts.

 ISBN: 978-1-61208-165-6

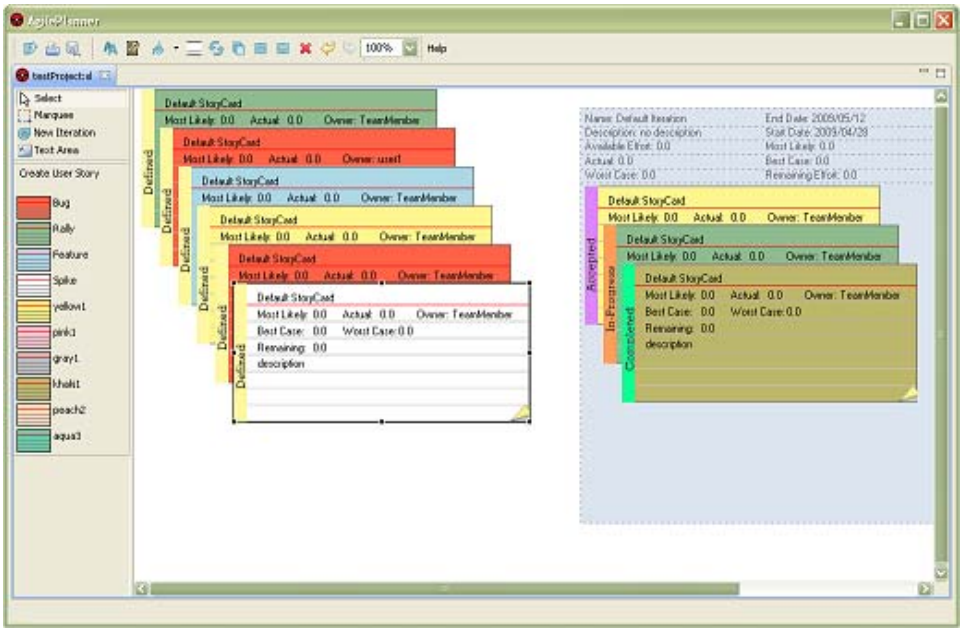

Figure 2. AgilePlanner with user story cards being assigned to an iteration.

B. eConference3P Features

In this section, we illustrate the features of our tool eConference3P against a few requirements acknowledged as critical in the field of distributed agile development [1] [20] [26].

1) Offline/online working switch.

eConference and, therefore, eConference3P too, work with XMPP-based Gmail accounts since the third release and with Skype accounts since the fourth. One of the benefits of our solution is that eConference products work without requiring any user or maintainer to install a server, thus minimizing the hassles coming from installations and configurations.

In developing our planning poker plugin for eConference, we selected AgilePlanner as the graphical editor of user stories and iterations. However, AgilePlanner required a connection to a server to work. Therefore, since we mostly needed AgilePlanner for its editing functionality, we patched it in order to support both online and offline mode [2]. In the offline mode, a user has the chance to store all the planning artifacts on a file and then load them back later. The online mode, instead, remained untouched. Because the transition from offline to online co-editing is not fluid (i.e. developers need to connect again to the Agile Planner server), at the end of the integration process, we felt that the presence of a proprietary server to install clashed with our intention of building an extensible, hassle-free planning poker tool.

2) Simultaneous interaction and manipulation of artifacts through telepointers

With respect to the supported planning activity, AgilePlanner is primarily focused on the interactive collaboration and meant for conducting real-time planning meetings, whereas it only has limited capabilities for progress tracking during the interaction.

The user interactions of AgilePlanner include the complete manipulation of planning cards (i.e., creating, moving, and deleting cards). Different colors are used to distinguish between the cards representing bugs, spikes, features, user stories, and finally iterations and backlog, to which they are assigned. To support distributed collaboration, AgilePlanner provides telepointers, which are a groupware technology that uses a remote mouse pointer to represent mouse movement happening on other connected computers so that remote collaborators can understand other team members' mouse movements, much like they would look at others' movements in a traditional co-located meeting.

3) Real-time information sharing & estimation.

In eConference3P all the changes happening to the shared workspace are notified in real time, so that updated information is simultaneously available to each remote developer. In particular, eConference3P focuses on supporting synchronous interaction rather asynchronous interaction between distributed agile team members. When playing planning poker, near real-time interaction is fundamental to support the discussions and converge to a shared estimate, when individual scores differ.

With respect to estimation-specific features, the project owner can import, export, and edit user stories from the backlog view. In particular, as for the edit feature, selecting that menu entry will change the current perspective of the tool to that of Agile Planner, as shown in Figure 2, thus letting distant developers move user stories in or out of the backlog, as well as plan multiple interactions for long-term release planning. The project owner can also call for estimation. When an estimation procedure starts, the deck

 ISBN: 978-1-61208-165-6

becomes clickable, and each developer can pick the card with the desired score by dropping it in the drop zone on the right hand side of the deck view. The project owner can check who provided what estimate at any time. Instead, developers' estimates will not be visible to each other until all of them provide one. eConference3P also allows the project owner to select the estimate scale of choice before a planning meeting is started.

4) Integration with others development environments

Supporting integration with development environments increases the ease of access to the planning information for developers and makes it easier to track progresses. Therefore, in eConference3P we enabled the import of user stories from the most used, web-based collaborative development environments (CDEs). CDEs such as Google Code [11], Github [10], and Trac [24], offer issue-tracking features for storing items, such as bug descriptions, enhancements, and milestones. However, they are also used by developers of agile teams to store planning artifacts, such as the backlog of user stories and iterations. The import procedure locally stores the data retrieved in the same XML format supported by AgilePlanner, so that imported data can be graphically edited afterwards.

Table I shows the four CDEs supported by the import procedure. First, we notice that all the CDEs offered official APIs to programmatically query and retrieve the information from the project repository. The only exception was Trac, for which we had to develop a custom scraper that makes http requests and then parses the resulting html output to retrieve the information needed. Because this solution depends on the structure of web pages, using a scraper is considerably less stable than using an API, since even smallest changes to the graphical layout may end up breaking it. Second, we notice that Assembla [3] is the only CDE that specifically support user story entries for its repository, whereas Google Code and Trac allow customizing generic entries (called tickets) into user stories, and later retrieve them through custom search queries. Github, instead, does not offer any of the two solutions and, thus, proved to be the least effective CDE for hosting an agile project repository. Lately, we have also added support to Jira [16] and Fogbugz [9].

IV. RELATED WORK

Tools for supporting agile development have been some ten years in the making. To date, there are literally hundreds of agile project management tools, some more complete than others, which tend to focus only on a specific activity of the agile management process. These tools typically allow teams to manage agile projects following Scrum and XP agile methodologies. A list of the most used and well-known application can be found on UserStories.com [25]. Such tools, whether free or commercial, can be broadly divided into three main categories, according to their target platforms: web-based, standalone, and plugins.

Web-based is the category that accounts for the largest number of existing agile planning and management tools. This is because such applications only require a web browser to be executed on the client side. Besides, as for commercial tools, web application as are often sold in "hosted mode", which requires no installation by customers since companies sell seats to use the service running on their own servers. Among web applications, we can first identify general purpose Wikis, used for agile project management in general, and estimation as well, by letting developers create, edit, and publish story cards and other artifacts as web pages. As such, they do not offer any specific support to agile practices and, therefore, only meet a very minimal set of requirements for agile projects management. On the contrary, there are tens of tools designed for agile project management, both commercial (e.g., Mingle, VersionOne, and Rally) and free (e.g., XPlanner, Agilo for Scrum, Agilefant, and eXPlainPMT), which offer sophisticated features to represent and manipulate project data, but none of them support the planning oker technique. The only tool that supports the homonymous agile estimation technique is PlanningPoker.com [22], which we analyze in detail in the next section.

The second category of agile project management applications is that of *standalone* tools, most of which run natively just on Windows with a very few alternatives for Linux and OS X built on Java. In this category, we identified no standalone tool supporting planning poker, other than eConference.

Finally, *Integrated Development Environments (IDE)*, such as Visual Studio and Eclipse, have also been extended through specific plugins in order to support, among the other things, agile practices and create an even more convenient development environment for closely managing and interconnecting code artifacts, such as test cases, and agile planning artifacts, such as story cards. In this category we identified WolfPoker [27], a planning poker plugin for Jazz, a commercial CDE developed by IBM that supports the customization and execution of any agile project management process of choice.

A. Comparing Planning Poker Tools

From the review in the previous section, we note that PlanningPoker.com (web-based) and WolfProject (Jazz plugin) are the only other existing tools that support the planning poker estimation technique as eConference3P (standalone), as shown in Table II. All the three tools support synchronous sessions (i.e., backlog editing and estimation), while PlanningPoker.com is the only one that also enables asynchronous estimation sessions.

Besides, we note that both PlanningPoker.com and WolfPoker support collocated groups of developers only in picking scores from a card deck and then visualize the estimates, while they completely lack any communication feature to support discussion. This is probably due to the fact that collocation and frequent direct communication are paramount for agile teams [4]. However, as distributed agile teams get more and more common [12], face to face communication cannot be given for granted any longer. Hence, distributed agile teams willing to adopt PlanningPoker.com or WolfPoker must also use such applications in combination with other communication tools.

 ISBN: 978-1-61208-165-6

TABLE I. THE CDES CURRENTLY SUPPORTED BY THE IMPORT PROCEDURE.

CDE	API	User story	Milestone	Custom ticket type	Custom search query
Assembla	X	X	X		
Github	X				
GoogleCode	X		X	X	X
Trac			X	X	X
Jira	X				
Fogbugz	X				

eConference3P, instead, integrates text-based and audio communication to support estimate synchronous discussions with no hassles. In addition, thanks to the AgilePlanner component, eConference3P allows collaborative editing of the backlog.

Finally, eConference3P is the only tool that can import a backlog from a number of CDEs, such as Google Code, Github, and Jira, whereas WolfPoker can only read file exported from MS Project.

V. CONCLUSIONS

In this paper, we presented eConference3P, a tool for enabling effective estimation meetings for distributed agile teams. The tool was built by integrating the AgilePlanner component, to enable iteration planning through a visual editor, and the eConference meeting system, to build a better communication tool and cope with the reduction of information exchanged in distributed settings. In fact, our review of existing tools for performing planning poker agile estimation revealed a lack of support for synchronous communication. Being based on the Eclipse RCP platform, specific plugins were then added to support the planning poker estimation technique and import user stories from web-based collaborative development environments.

TABLE II. A COMPARISON BETWEEN TOOLS SUPPORTING PLANNING POKER

Feature	eConference3P	PlanningPoker.com	WolfPoker
Category	Standalone	Web based	Plugin (Jazz)
Sync. sessions	Backlog editing, estimation	Backlog editing, estimation	Backlog editing, estimation
Async. sessions	Backlog editing	Backlog editing, estimation	Backlog editing
Comm. modes	Text, audio	None	None
Backlog editing	Yes (co-editing)	None	Yes
Integration w/ CDEs	Backlog import	None	Backlog import*

* only supports MS Project file format

REFERENCES

[1] Abrahansson, R., Salo, O., Ronkainen J., and Warsta J. (2002). *Agile Software Development Methods.* VTT Publications, vol. 112.

[2] AgilePlanner for eConference, http://code.google.com/p/agileplanner-for-econference (last accessed: Jul. 19, 2011).

[3] Assembla, http://www.assembla.com (last accessed: Jul. 18, 2011).

[4] Beck, K. et al. (2001). *Manifesto for Agile Software Development*. http://agilemanifesto.org (last accessed: Jul. 18, 2011).

[5] Calefato, F., and Lanubile, F. (2009). *Using Frameworks to Develop a Distributed Conferencing System: An Experience Report.* Software: Practice and Experience, vol. 39, no. 15, pp. 1293–1311

[6] Cohn, M. (2005). *Agile Estimating and Planning*. Prentice Hall

[7] Damian, D., Sengupta, B., Lanubile, F. (2008). *Global Software Development: Where Are We Headed?* Software Process Improvement and Practice. vol. 13, pp. 473-475.

[8] Eclipse RCP, http://www.eclipse.org/home/categories/rcp.php (last accessed: Jul. 19, 2011).

[9] Fogbugz, http://www.fogcreek.com/fogbugz (last accessed: Jul. 18, 2011).

[10] Github, https://github.com (last accessed: Jul. 18, 2011).

[11] Google Code, http://code.google.com (last accessed: Jul. 18, 2011).

[12] Herbsleb, J.D. (2007), *Global Software Engineering: The Future of Socio-technical Coordination*. Future of Software Engineering (FoSE'07), May 23-25, pp.188-198.

[13] Highsmith, J., Cockburn A. (2001) *Agile Software Development: The Business of Innovation*. Computer, vol. 34, no. 9, pp. 120-122.

[14] Hoest, M., and Wohlin, C. (1998). *An Experimental Study of Individual Subjective Effort Estimations and Combinations of the Estimates*. Proc. 20th Int'l Conf on Software Engineering (ICSE'98), Kyoto, Japan, Apr. 19-25, pp. 332-339.

[15] Jazz, https://jazz.net (last accessed: Jul. 18, 2011).

[16] Jira, http://www.atlassian.com/software/jira (last accessed: Jul. 18, 2011).

[17] Johnson, P., Moore, C.A., Dane, J.A., and. Brewer, R.S. (2000). *Empirically Guided Software Estimation*, IEEE Software, vol. 17, no. 6, pp. 51-56.

[18] Jorgensen, M., and Molokken, K. (2002). *Combination of Software Development Effort Prediction Intervals: Why, When and How*? Proc.14th Int'l Conf. Sw. Eng. and Knowledge Engineering (SEKE '02), vol. 27, Ischia, Italy, Jul. 15-19, pp. 425-428.

[19] Lanubile F., Damian D, Oppenheimer H. (2003). *Global Software Development: Technical, Organizational, and Social Challenges.* Software Engineering Notes. Vol. 28.

[20] Larman, C. (2004). *Agile & Iterative Development - a Managers's Guide*, Addison-Wesley.

[21] Morgan, R., and Maurer, F. (2008). *An Observational Study Of A Distributed Card Based Planning Environment.* Proc. 9th Int'l Conf. on Agile Processes and eXtreme Programming in Software Engineering (XP'08),. Limerick, Ireland, Jun. 10-14, pp. 53-62..

[22] PlanningPoker.com, www.planningpoker.com (last accessed: Jul. 18, 2011).

[23] Šmite, D., Wohlin, C., Gorschek, T., and Feldt, R. (2010). *Empirical Evidence In Global Software Engineering: A Systematic Review*. Empirical Software Engineering, vol. 15, pp. 91–118.

[24] Trac, http://trac.edgewall.org (last accessed: Jul. 18, 2011).

[25] UserStories.com, http://userstories.com/products, (last accessed: Jul. 18, 2011).

[26] Wang, X., Maurer, F., Maurer, R., and Oliveira, J. (2010). *Tools for Supporting Distributed Agile Project Planning*, in Agility Across Time and Space (Smite, Moe, Ågerfalk eds.), Springer, pp. 183-199.

[27] Wolfpoker, http://www.realsearchgroup.org/wolfpoker (last accessed: Jul. 18, 2011).

 ISBN: 978-1-61208-165-6