ASAP: A LIGHTWEIGHT TOOL FOR AGILE PLANNING

Rasmus Rosenqvist Petersen
PLAN-IT
Vesterbro 116, 1 th, 5000 Odense C, Denmark
rasmusrosenqvistpetersen@gmail.com

Uffe Kock Wiil
The Maersk Mc-Kinney Moller Institute, University of Southern Denmark
Campusvej 55, 5230 Odense M, Denmark
ukwiil@mmmi.sdu.dk

Keywords: Agile Planning, Electronic Task Cards, Interactive Displays, Automatically Generated Views, Spatial Hypertext.

Abstract: Supporting planning tasks of agile teams is a challenge. In this paper, we present a lightweight planning tool. ASAP is inspired by concepts and principles from spatial hypertext, which support information analysis tasks. ASAP runs on a large interactive vertical display on which electronic task cards can be organized into iterations and releases using card hierarchies and separators (a novel visual concept). Several views of the evolving plan are automatically generated to assist the agile team with overviews of tasks, estimates, and assignments. Views are instantly updated to reflect changes to the plan.

1 INTRODUCTION

Project planning in agile teams is a collaborative process relying on face-to-face communication and shared information to succeed. A commonly used approach to planning involves teams using a large table to plan iterations using paper cards to represent tasks to be carried out. When a planning session is over, the plan is somehow recorded, and the cards are removed from the table. One downside to this is that cards' location on the table and their proximity to other cards may contain important information for the overall plan. When cards are removed from the table, their arrangement is often lost and with it so is the proximity and location information (Morgan et al. 2007).

Our overall goal is to develop a software tool to support agile planning, while preserving the benefits of physical card based planning. Card based planning is suggested by several agile development methods and is used by many agile software teams. During the development of ASAP, we have been working closely with local software companies to identify requirements for the planning tool, to get feedback on suggested features, and to test various versions of the planning tool. A number of important overall requirements have come out of this collaboration:

- A planning tool should support the work of the agile team in a manner that resembles the physical card based approach.
- A planning tool should be lightweight offering only the features that are necessary to solve the task at hand.
- A planning tool should visualize the consequences of the planners' actions allowing them to make informed decisions regarding the plan.

ASAP has been developed to fulfill these overall requirements. It runs on a large interactive vertical display on which electronic task cards can be moved around freely and organized into iterations and releases using card hierarchies and separators (a novel visual concept). Generated plans can be stored, printed, and retrieved again later for future planning sessions.

Once paper cards become electronic, several new opportunities arise. Different views can be automatically generated based on the layout and the attributes of the task cards. ASAP currently provides overviews of tasks, estimates, and assignments.

Views in ASAP are instantly updated to reflect the changes made by the planners. These views support the planners by visualizing the current status of the plan.

The remainder of the paper is structured as follows. Section 2 reviews some well-known agile software development methods to identify agile planning practices and techniques. Section 3 looks at existing software support for agile planning. Section 4 presents ASAP – including requirements, design concepts and features, current status and future plans, and evaluation. Section 5 summarizes the paper.

2 AGILE PLANNING

A number of agile software development practices and techniques focus on and involve planning. We will briefly look at some of these to investigate what types of activities ASAP should be able to support.

The Scrum agile software development method suggests several work practices where planning plays a role: pre-game planning and staging, sprint planning, and the daily meeting (Larman 2004). Pre-game planning and staging focuses on identifying desired features which are recorded in the Product Backlog and possibly one or more Release Backlogs. Iterations (sprints) start with two sprint planning meetings where the tasks for the upcoming iteration are planned and estimated. At the daily meeting the sprint plan is reviewed and tasks may end up in the Sprint Backlog.

Extreme Programming (XP) is a well-known agile method that includes 12 core practices – including the Planning Game (Larman 2004). The Planning Game is a meeting that occurs once per iteration. The Planning Game is divided into two parts. Release Planning Game focuses on determining what features are included in the next release, and when they should be delivered. Iteration Planning Game focuses on planning the activities and tasks of the developers in the upcoming iteration.

The Crystal family of agile methodologies (Cockburn 2005) includes the Blitz Planning technique. Blitz Planning is based on paper task cards laid out on a table surface. The overall idea is to gather the right people, discuss the project details, and end up with a project plan as a result of working through ten predefined steps. Blitz Planning is a variation of the XP Planning Game described above. The two differ in three ways (Cockburn 2005):

- The planning game cards list *user stories*, and the Blitz Planning cards list *tasks*.
- The planning game has the people assume there are no dependencies between stories, while Blitz Planning has people analyze the dependencies between tasks.
- The planning game assumes fixed-length iterations, while Blitz Planning does not assume anything about iteration length.

Another technique included in the Crystal family is the Daily Stand-up Meeting, which is quite similar to Scrum's daily meeting described above.

The above reviews show that planning is a central activity in agile software development. If we go one step deeper into the practices and techniques, they suggest the use of cards and larges surfaces (tables and whiteboards) to create and revise plans.

3 RELATED WORK

Over the last years, many commercial and open source tools have become available to support agile planning. Liu (2006) identified three categories of planning support systems: form-based, combined Wiki- and form-based, and board-based.

Form-based. The majority of tools that support agile planning belong to this category. Typical representatives of this category are browser based and provide forms to store a predefined set of information, e.g., effort estimates or priority rankings. Form-based systems provide basic functionality for creating and deleting as well as editing and prioritizing project planning artifacts and can derive supplementary information like total efforts for iterations or remaining work effort from existing data. Existing form-based tools comprise commercial products like Rally (2009), VersionOne (2009), and ScrumWorks (Danube 2009) as well as open source products like XPlanner (2009).

Wiki- and form-based. Tools like MASE (Maurer 2002) (University of Calgary) take the form-based approach one step further and combine it with another very popular way to share information between people. Users can attach Wiki-pages to stories that can be used to provide additional information related to a task.

Board-based. This category comprises tools like CardMeeting (2009), AgilePlanner (Liu 2006), Distributed AgilePlanner (Morgan 2008), and MasePlanner (Morgan and Maurer 2006). A commonality of board-based systems is that they are all mimicking card based planning to a certain extent. They provide ways to create, edit, and delete cards and visually group those cards to indicate relationships. CardMeeting attempts to bridge the gap between browser based systems and physical

card based planning. It displays electronic index cards in a web browser. It is primarily focused on the visual aspect of card based planning. It does not provide the iterations and progress tracking that other agile planning tools have. AgilePlanner and its successor Distributed Agile Planner (University of Calgary) are card based tools for collocated and distributed agile planning. Synchronous distributed planning meetings are supported by providing a shared workspace (displayed on vertical displays and/or digital tabletops) for creating, organizing, and editing electronic index cards. Changes made by one team member become visible immediately on connected clients all over the world. MasePlanner (also from University of Calgary) builds on features from MASE and AgilePlanner and is implemented as an Eclipse plug-in with web services for remote connectivity.

ASAP is a board-based agile planning tool supporting collocated agile teams. The ability to support well-known agile practices and techniques such as Crystal Clear's Blitz Planning and XP's Planning Game has played a major role in determining the features of the tool.

ASAP is developed to run on large interactive **vertical** displays. Many of University of Calgary's tools have special support for interactive horizontal displays (such as rotation of cards). According to the chosen lightweight strategy, it should be easy (and inexpensive) to run ASAP in existing meeting rooms. Very few meeting rooms have interactive horizontal displays (tabletops), while interactive vertical displays are more common. Commercial tools like the Tool-Tribe Connector (www.tool-tribe.dk) offer a simple, portable, and inexpensive solution that turns any whiteboard into an interactive surface. This allows ASAP to be used in any meeting room equipped with a whiteboard.

The development of ASAP is inspired by techniques from spatial hypertext (Shipman et al. 2001). This makes the features and interaction in ASAP different from existing board-based tools. Task cards are easy to create, manipulate, and organize. Besides the traditional spatial organization of task cards on a surface, ASAP provides two additional organization features – a hierarchical view of the organization of tasks and subtasks and a novel visual concept (the separator) used to separate and group task cards.

4 THE ASAP APPROACH

This section presents ASAP including requirements, design concepts and features, current status and future plans, and evaluation.

4.1 Requirements

Based on interactions with local software companies three overall requirements for agile planning tools were identified:

- *A planning tool should support the work of the agile team in a manner that resembles the physical card based approach*. A computer tool like ASAP should not alter a workflow that works. It should simply support the existing workflow (in this case board-based planning) and, if possible, provide additional support for the individual steps in the workflow enabling the user to perform the tasks better and/or faster.

- *A planning tool should be lightweight offering only the features that are necessary to solve the task at hand*. There are many examples of tools that provide a lot more features than necessary to solve a given task. Let us consider Microsoft Word. Typical Word users only make use of a very limited subset of the features in Word (say 10 %) to solve most of their writing tasks (say 90 %). This can result in complex tools that are difficult to use. The entry barrier becomes higher for new users. The overhead of using the tool may outweigh the benefits of the tool. ASAP goes the other way and provides only the most essential features for agile planning. A lightweight planning tool that is intuitive and easy to use will result in a much lower entry barrier for new users and will be able to support most of their planning needs.

- *A planning tool should visualize the consequences of the planners' actions allowing them to make informed decisions regarding the plan*. The agile team is in charge of the planning process. The tool supports the team by using its computing power to instantly visualize the consequences of the individual steps in the planning process.

In particular, two local software companies have contributed to the generation of the above overall requirements. Mikro Værkstedet (www.mikrov.dk) is a small software development company (<50 employees) focusing on educational software. KMD (www.kmd.dk) is a large software development company (3000+ employees) focusing primarily on software for the public sector.

Formal interviews have been conducted with agile software development team managers at both places regarding requirements for an agile planning tool. Earlier versions of ASAP were discussed with the same team managers to focus the development of

ASAP on the most essential features. Finally, the team managers are currently using ASAP in ongoing software development projects using agile practices. First experiences from the use have been collected and ASAP is currently being evaluated (see Section 4.4).

A set of functional requirements for ASAP have been derived based on the overall requirements and the desire to support existing agile planning practices and techniques. These requirements are not surprisingly somewhat overlapping with the agile planning requirements presented by Liu (2006) and Morgan (2008).

1. **Supporting agile planning objects**. Creating, editing, and deleting task cards are core agile planning activities. We have adopted the Crystal Clear Blitz Planning notation of *tasks* instead of *user stories*.
2. **Organizing agile planning objects**. The ability to move task cards around freely and organize them into iterations and releases are core agile planning practices.
3. **Supporting multiple iterations**. Agile teams should be able to make both short term planning (next iteration) and long term planning (future iterations and releases).
4. **Supporting hierarchies of planning objects**. Breaking down tasks into subtasks is a well-known strategy for handling complexity (divide and conquer). Hierarchies also provide a way to handle large projects with many tasks (addressing the scalability issue).
5. **Visualizing consequences of planning actions**. Visualizing the consequences of the planners' actions allows them to make informed decisions regarding the plan.
6. **Supporting estimation and tracking**. Adding estimates to tasks cards allows the agile team to get an overview of the duration of iterations. Adding task status to Task Cards allows the agile team to track the status of single tasks as well as iterations.
7. **Managing team members and resources**. Assigning tasks to team members allows agile teams to plan their resources.
8. **Re-using experiences from past planning sessions**. Access to old planning sessions allows the team members to include past experiences in their current planning sessions.

We consider requirements 1 through 5 to be essential to support agile planning the ASAP way. Requirements 6 through 8 are also important, but according to the chosen strategy they will be provided in a lightweight manner.

4.2 Design Concepts and Features

The concepts used in ASAP are developed based on the three overall and eight functional requirements listed above. Figure 1 illustrates a planning session with physical cards laid out on a table.

Several visual features can be observed in the figure: The use of task cards, the large surface used to organize the cards, the organization of cards into hierarchies, the visual separation of cards, the location of cards, the proximity of cards, etc. These observations have influenced the design of concepts and features in ASAP.

The design of the user interface in ASAP is based on well-known interaction principles (such as direct manipulation, drag and drop, and cut and paste), interaction features (such as menus, toolbars, and shortcut icons), and interaction metaphors (Windows Explorer style hierarchies and Windows desktop style surface).

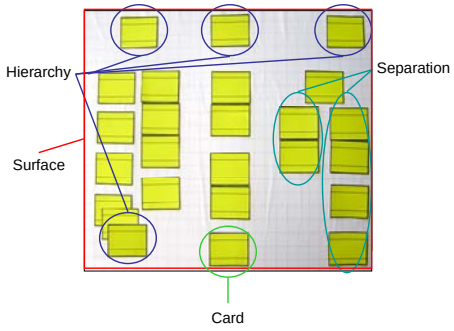

Figure 1: A planning session with physical cards on a table.

Figure 2 provides a screenshot of the main window in ASAP consisting of three parts – two views as well as the menus and toolbar at the top. The View to the left provides an overview of the task hierarchy using an interaction metaphor similar to the one used in Windows Explorer. The View to the right provides a large surface (Space) on which task cards can be organized freely using an interaction metaphor similar to the one used by the desktop in Windows. Each central design concept and tool feature of ASAP is explained below.

The **Space** (right hand View in Figure 2) is a well-known concept adopted from spatial hypertext (Shipman et al. 2001). A Space in ASAP is a large 2D surface used to organize electronic task cards. The **Hierarchy** (left hand View in Figure 2) provides a tree overview of the organization of tasks

and subtasks. The tree root reflects the name of the planning session, nodes in the tree are tasks containing subtasks, and leafs in the tree are tasks with no subtasks. The Hierarchy View and the Space View are synchronized in the sense that changes made in one View are instantly reflected in the other View. There are no limitations to the number of nested hierarchies. The two Views are separated by a divider that can be moved left or right to expand/minimize the Views depending on the users' preference.

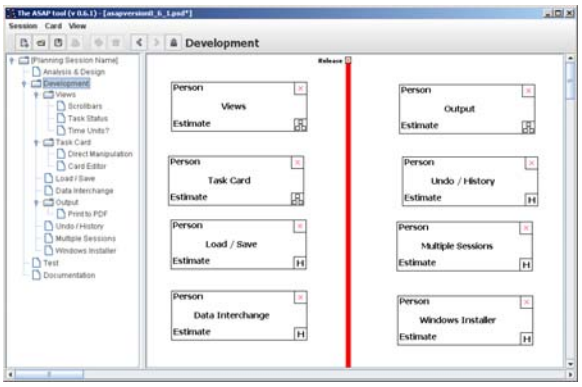

Figure 2: The main window in ASAP.

The **Task Card** is the basic planning object used in ASAP. It represents the paper equivalent. It is overlaid with a grid (3 by 3 cells) in which each cell is assigned specific meaning. The value of cells can be changed by the user. Task Cards are created by making a dragging gesture inside a Space that resembles the shape of a Task Card. A Task Card is deleted when pressing the "red cross" in the upper right hand corner. The icon in the lower right hand corner is used to create and traverse hierarchies. The icon can have two different forms. A card with no subtasks is depicted with an icon similar to an "H". If the user clicks the "H" icon, then a new subspace is created and opened. The user can now add subtasks in the subspace. A subspace offers exactly the same functionality as a Space. A card with subtasks is depicted with a different icon in the lower right hand corner (see Figure 2). If the user clicks this icon, then the subspace opens allowing the user to manipulate the subtasks. We use the term Space to cover both the top level Space and the subspaces.

Task Cards can be configured using a **Card Editor**. Since the optimal Task Card design may differ depending on the project team and planning task, a Card Editor allows for configuration (personalization) of the Task Card layout. By default the cell in the upper left hand corner is named Person and is of type [PERSON], the cell in the lower left hand corner is named Estimate and is of type [TIME], and the cell in the middle is named Task and is of type [TASK]. The Person and Estimate cells are used for estimates and tracking and for handling team members and resources.

A **Separator** (red vertical bar) is used to group vertically dependent Task Cards visualizing the horizontal separation of dependencies (e.g., a time line). Separators are created by making a dragging gesture inside a Space that resembles the shape of a Separator. Separators can be assigned a name (description), a date, and a type. Currently, the following types of Separators exist: Separation, iteration, walking skeleton, release, and milestone. The Separator is a novel visual structuring concept that is introduced to support the planning task by visualizing dependencies and groupings among Task Cards.

The Space is immediately parsed by a **Spatial Parser** every time a change occurs. The algorithms parsing the Separator(s) sort them according to their position, and then apply methods to detect Task Cards to the left, to the right, and between two Separators. Spatial parsing is done along the y-axis clarifying vertical dependencies. The algorithms parse the Task Cards grouped together by Separators, according to their y-coordinates. The Card with the lowest y value is identified as the first task in the grouping (e.g., iteration) and so forth. The concept of a spatial parser is a well-known concept adopted from spatial hypertext (Shipman et al. 2001).

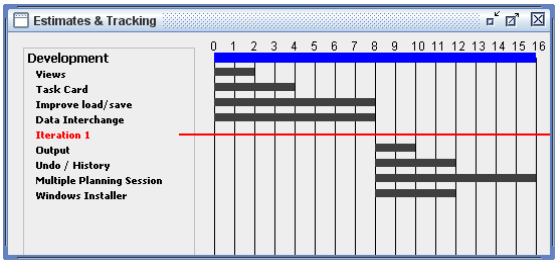

Figure 3: The auto-generated Estimates View.

The Spatial Parser automatically generates **Views** that are relevant to the planning task. The generated Views visualize the consequences of the planners' actions. The Spatial Parser can currently generate two types of Views. The Estimates View (Figure 3) is auto-generated based on the layout of Task Cards in the Space (and subspaces), the time estimate of Task Cards, and the position of Separators in the Space (and subspaces). When a Task Card or Separator is moved or when a time estimate is changed, the Estimates View is immediately updated to show the revised Plan. The time line at the top indicates the number of planning

units (hours, half days, or days) assigned to a task. The cell of type TIME in the Task Card holds the assigned time estimate of a task.

The Team Members View (Figure 4) is auto-generated based on the cells of type PERSON and TIME in the Task Card. The Team Members View allows the names of team members to be added. Once names are added, each task can be assigned to a team member by updating the cell of type PERSON. The View also shows the estimate of each task as well as the total estimate of the tasks assigned to a person. The Team Members View is instantly updated when a task has been assigned to a team member and when an estimate has been changed.

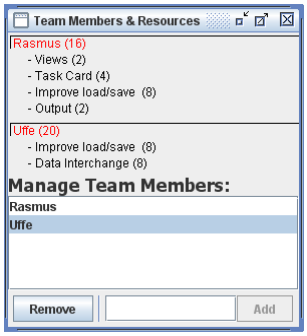

Figure 4: The auto-generated Team Members View.

A **Planning Session** is something that is not normally finished in one go; most often, there is a need to revise an existing plan. This implies the need for storing Planning Sessions and continuing (revising) them at a later time. ASAP allows Planning Sessions to be stored and opened again later. ASAP defines its own file type (*.psd – planning session data) for this purpose. This functionality is available both in the "File" menu and on the toolbar through shortcut icons.

Plans can also be **exported** to XML. Visual objects are serialized into an XML string which can be saved in a text file. The XML format contains information about Separators and Task Cards including a unique ID, cell information, card location, card size, font size, and other font information. Information about cells, their type and location is also stored. The use of XML as an interchange format makes it possible to transform the plan generated in ASAP into other formats that can be imported by other agile planning tools or third-party project management applications.

Finally, plans can be **printed** – either on paper or to a PDF file. It is possible to print a single View (i.e., current Space or Estimates View). A full print of the plan consists of the following parts:

- An overview of the Task Cards in each Space – the hierarchy (tree structure) is traversed and printed in a depth-first manner: top Space, all subspaces of the first Task Card, all subspaces of the second Task Card, etc.
- An overview of the Estimates (as seen in the Estimates View).
- An overview of the Team Members and their assignments (as seen in the Team Members View).
- A page for each team member with a list of assigned tasks.

4.3 Current Status and Future Work

The current version of ASAP (March 2009) fulfills most of the identified functional requirements for an agile planning tool:

Requirements 1-5: Fully supported. All the desired features are supported.

Requirements 6-7: Partly supported. Many of the desired features are supported. ASAP supports estimation and management of team members.

Requirement 8: Partly supported. To some extent, this feature is already available in the current version. It is possible to load an old planning session into ASAP and reuse it as a starting point for a new planning session. In this way, experiences from an earlier project can be reused.

Our immediate future work plans aim to fulfill all the functional requirements:

- **Estimation and tracking**. We plan to add lightweight support for tracking (Cohn 2006).
- **Management of team members and resources**. We plan to add lightweight support for management of resources.
- **Sessions**. We plan to add an additional level of support allowing reuse from multiple past planning sessions.

We also plan to investigate the following issues in relation to ASAP:

- **Interchange**. We consider interfacing to existing project management tools (i.e., Microsoft Project) as well as existing commercial agile planning tools.
- **Collaboration**. Currently we only provide support for collocated meetings. We consider to provide support for distributed planning meetings also.

- **History**. We consider providing a history feature like the one available in VKB (Shipman et al. 2001). This would allow for unlimited undo of actions as well as scrolling back in history and possibly explore alternate planning steps – see VKB history toolbar below.

4.4 Evaluation

ASAP has been developed to support agile planning practices and techniques from well-known software development methods. We have previously shown (Petersen and Wiil 2008) that an earlier version of ASAP (January 2008) can support the activities of the Blitz Planning technique from the Crystal family (Cockburn 2005). The current version is developed to provide support for a broader set of planning tasks as specified in the software development methods reviewed in Section 2.

ASAP is developed as a lightweight planning tool providing only the most essential features for agile planning. Features are provided in an independent manner that allows the planners to only use the features that they need for the task at hand. Features in ASAP can be divided into two categories: basic features provided by the Space and Hierarchy Views and additional features provided by the Estimates and Team Members Views. Planners engaged in simple planning tasks may only need the basic features provided by the Space View. More complex planning tasks require the use of additional Views. Thus, planners only "pay" for what they use in terms of tool complexity. There is no additional overhead for features that are not being used.

ASAP is currently being used and evaluated in software projects conducted at local software companies.

Figure 5: Typical use of ASAP in a planning session.

Figure 5 depicts a typical set up of ASAP in planning sessions at the local software companies. The main window of ASAP (providing the Space and Hierarchy Views) is displayed on a large interactive surface (e.g., a SMART Board™) allowing the software team members to jointly plan and interact with the planning objects. The additional Views (Estimates and Team Members) allowing the team members to instantly see the consequences of their actions are displayed on a separate screen.

The purpose of the evaluations is to explore the borders of the applicability of our lightweight approach to agile planning. Projects are monitored with respect to project size and the required feature intensity. We wish to find out if there is a limit to the size of projects that ASAP can support (primarily in terms of numbers of tasks). We also wish to find out what features are being used in what types of projects. Finally, we wish to find out if the features provided by ASAP are sufficient for most agile planning purposes.

Evaluations are still ongoing. However, first results and feedback is positive and encouraging as indicated below by statements from end users:

- "It is a good idea to base the interaction in ASAP on well known concepts. The tool is intuitive and easy to use."
- "Plans are faster to make with ASAP than our previous method. ASAP supports an effective planning process."
- "We believe that the plans generated with ASAP are better than our usual plans. Plans are easy to store, revise, and distribute."
- "We always keep an overview of the plan visible in the project room. It helps the team members in their daily work."

Thus, we are confident that the chosen lightweight strategy turns out to be successful. We have also received suggestions for improving the tool:

- "We would like to track the progress of the individual tasks and see the estimates of the remaining time to complete the task."

  As mentioned in Section 4.3, this functionality is already under development and will be part of the next version.
- "It would be helpful to be able to create task cards in different colors."
- "A facility allowing the user to zoom in and out of the spaces would be useful."

The latter two features (colored task cards and zooming) are already known from existing agile planning tools. We have already considered both in the past, but will do so again due to the comments from the end users. We are trying to find the right balance of features as part of the chosen lightweight strategy – what features need to be there and what can be omitted?

5 CONCLUSION

The ASAP approach to agile planning has been developed based on different types of analysis work:

- **Involving end users**. We have interacted with software companies that practice agile planning to get their input.
- **Exploring methods**. We have explored planning practices and techniques from several agile software development methods.
- **Studying related work**. We have found inspiration from existing agile planning tools.

Together, this resulted in three overall and eight functional requirements that have guided the development. Currently, most of the envisioned features are supported and the tool is being used and evaluated by local software companies.

ASAP is inspired by previous work on the use of spatial hypertext to support the knowledge management task known as information analysis (Shipman et al. 2001). ASAP is based on the Construct Space Tool (Wiil and Hicks 2001). The work has so far resulted in two main contributions:

- We have developed a board-based agile planning tool to help software teams plan their projects. The tool offers only the features that are necessary to solve the task at hand according to the chosen lightweight strategy.
- Agile planning is viewed as a complex knowledge management task. Specific features have been developed to support the planning requirements. The work has resulted in novel ideas such as the visual Separator concept, the Hierarchy View, and the use of the Spatial Parser to auto-generate Views that are relevant to the planning process.

We believe that the inspiration from several fields combined with the chosen lightweight approach has resulted in a tool that is well suited for agile planning.

REFERENCES

CardMeeting. 2009. CardMeeting (Online: http://www.cardmeeting.com).

Cockburn, A. 2005. Crystal Clear – A Human-Powered Methodology for Small Teams. Addison Wesley.

Cohn, M. 2006. Agile Estimation and Planning. Prentice Hall.

Danube. 2009. ScrumWorks (Online: http://www.danube.com/scrumworks/basic).

Larman, C. 2004. Agile & Iterative Development – A Manager's Guide. Addison Wesley.

Liu, L. 2006. An Environment for Collaborative Agile Planning. M.Sc. Thesis, Department of Computer Science, University of Calgary.

Maurer, F. 2002. Supporting Distributed Extreme Programming. Proceedings of XP/Agile Universe 2002, (Chicago, IL, August), pp. 13-22. LNCS 2418, Springer.

Morgan, R. 2008. Distributed AgilePlanner: A Card Based Planning Environment for Agile Teams. M.Sc. Thesis, Department of Computer Science, University of Calgary.

Morgan, R., and Maurer, F. 2006. MasePlanner: A Card-Based Distributed Planning Tool for Agile Teams. In Proceedings of ICGSE 2006, (Florianopolis, Brazil , October) pp. 132-138. IEEE Computer.

Morgan, R., Walny, J. Kolenda, H., Ginez, E., and Maurer F. 2007. Using Horizontal Displays for Distributed & Collocated Agile Planning. In Proceedings of XP 2007, (Como, Italy, June), pp. 38-45. LNCS 4536, Springer.

Petersen, R. R, and Wiil, U. K. 2008. ASAP: A Planning Tool for Agile Software Development. In Proceedings of Hypertext 2008, (Pittsburgh, PA, June), pp. 27-32. ACM Press.

Rally Software. 2009. Rally's Agile Lifecycle Management Solutions (Online: http://www.rallydev.com/products.jsp).

Shipman, F. M., Hsieh, H, Maloor, P., and Moore, J. M. 2001. The Visual Knowledge Builder: A Second Generation Spatial Hypertext, In Proceedings of Hypertext 2001, (Aarhus, Denmark, September), pp. 113-122. ACM Press.

VersionOne. 2009. Agile Project Management Tools (Online: http://www.versionone.com/products.asp).

Wiil, U. K., and Hicks, D. L. 2001. Tools and Services for Knowledge Discovery, Management and Structuring in Digital Libraries. In Proceedings of CE 2001, (Anaheim, CA, August), pp. 580-589.

XPlanner. 2009. XPlanner Overview (Online: http://www.xplanner.org/index.html).