XPSuite: Tracking and Managing XP projects in the IDE

M. Angioni †
angioni@crs4.it

D. Carboni †
dcarboni@crs4.it

M. Melis ‡
marco.melis@diee.unica.it

S. Pinna ‡
pinnasandro@diee.unica.it

R. Sanna †
raffa@crs4.it

A. Soro †
asoro@crs4.it

† CRS4 - Center for Advanced Studies, Research and Development in Sardinia
Parco Scientifico e Tecnologico, POLARIS
09010 PULA (CA - Italy)

‡ Dipartimento di Ingegneria e Elettrica ed Elettronica
Università di Cagliari
Piazza D'Armi, 09123 Cagliari, Italy

ABSTRACT

In this paper we describe XPSuite, a tool comprising two parts: XPSwiki, a tool for managing XP projects and XP4IDE, a plug-in for integrating XPSwiki with the IDE IntelliJ-Idea. The reasons for this integration and the ability of XPSuite to collect process metrics are described. The system has a full object oriented implementation so it is possible to extract all data represented in the model that the system implements.

Categories and Subject Descriptors

D.2.1 [**Software**]: Software Engineering Requirements/Specifications [Tools]; D.2.6 [**Software**]: Software Engineering Programming Environments [Integrated environments]; D.2.8 [**Software**]: Software Engineering Metrics

General Terms

Measurement, Management

Keywords

Process data, software project management tool, Extreme Programming

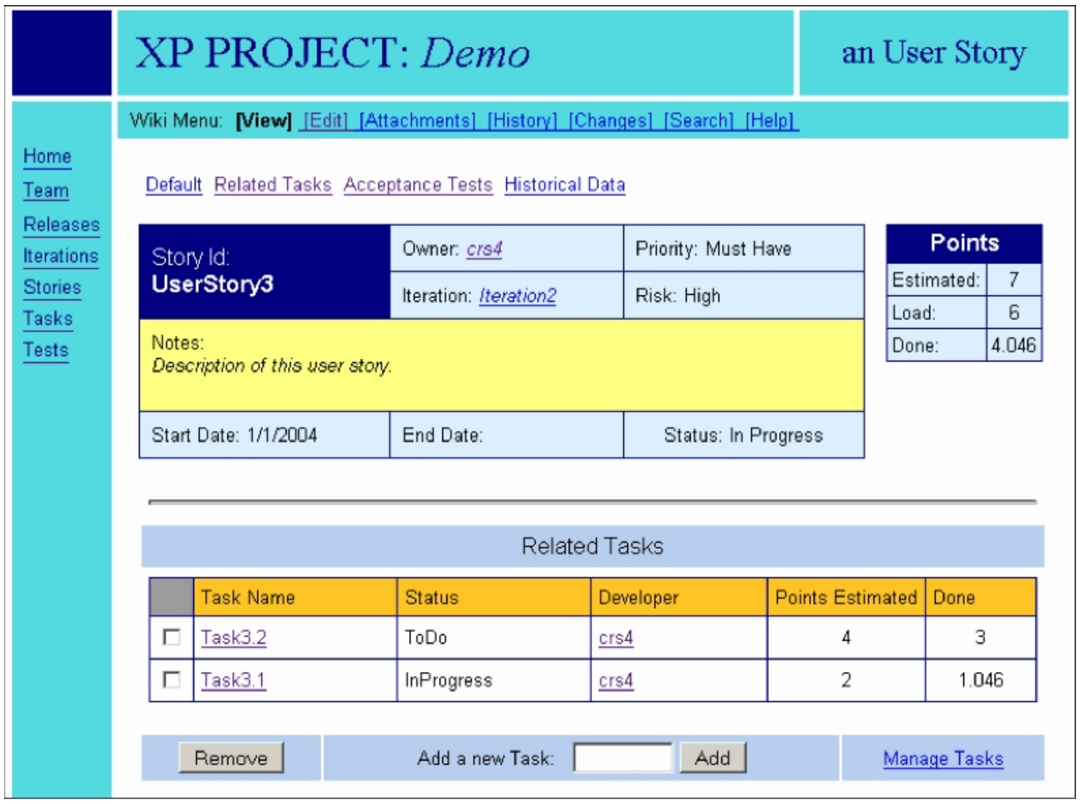
1. INTRODUCTION

Extreme Programming is a software development methodology which does not rely on any particular tool, but rather is based on the common understanding of fundamental values and on a disciplined application of best practices. In particular, projects developed with XP show that good results can be obtained using sheets of papers to collect user requirements, wall boards to show diagrams and other project-relevant information, and shared workspaces to maximize face-to-face communication. Nevertheless, there are a number of reasons that may lead XP teams to adopt software and Internet based tools to facilitate application of XP practices:

- process data can be automatically collected and analyzed afterwards. The process can be validated and team activities can be more effectively measured;
- managers and customers have the feeling that tools are something more "concrete" than pure methodologies;
- dispersed teams can work together with appropriate Internet connections.

In this paper we will describe XPSuite, a two-part tool incorporating XPSwiki, a tool for managing XP projects and XP4IDE, a plug-in for integrating XPSwiki with the IDE IntelliJ-Idea. The reasons for this integration and the ability of XPSuite to collect process metrics will be described. The system has a full object oriented implementation so it is possible to extract all data represented in the developed model.

In section §2 we briefly introduce the XP methodology and process. Section §3 contains a description of possible metrics for an XP process. In section §4 we present XPSuite, the suite of Internet tools we have developed for eliciting requirements, collection of user stories, process tracking and process data collection, while in section §5, we describe the underline conceptual model. Finally, section §6 describes the functionalities of this suite of tools while section §7 briefly discusses privacy issues.


XP PROJECT: *Demo*
an User Story
Wiki Menu: **[View]** [Edit] [Attachments] [History] [Changes] [Search] [Help]
Home
Team
Releases
Iterations
Stories
Tasks
Tests
Default Related Tasks Acceptance Tests Historical Data
Remove
Add a new Task:
Add
Manage Tasks

Story Id: **UserStory3**	Owner: *crs4*	Priority: Must Have
	Iteration: *Iteration2*	Risk: High
Notes: *Description of this user story.*		
Start Date: 1/1/2004	End Date:	Status: In Progress

Points	
Estimated:	7
Load:	6
Done:	4.046

Related Tasks

	Task Name	Status	Developer	Points Estimated	Done
☐	Task3.2	ToDo	crs4	4	3
☐	Task3.1	InProgress	crs4	2	1.046

Figure 1: The GUI of XPSwiki.

2. SHORT INTRODUCTION TO XP

Extreme Programming (XP) is an agile methodology for software development. "Agile" means that the primary goal is to continuously adapt activities to mutable requirements, mutable environmental factors and mutable market conditions. Simplicity, communication, feedback, and courage are the founding values of XP and all the XP best practices are coherent with these values (for a complete description of XP see [4]). XP is "extreme" because it carries to an extreme degree some of the best practices already used in traditional software development. For instance, if unit testing produces high quality code then it must be applied to its extremes writing test cases before the code to be tested. Similarly, if refactoring produces a code easier to maintain and to extend, then "refactor mercilessly". Also communication is "extreme": to avoid any misunderstanding and to develop exactly what is required, the customer (or an effective representative) is "embedded" as a member of the team and meetings are held on a daily basis to promptly identify problems and propose corrective actions.

The first phase of an XP project is the *Planning Game*, where developers and customers elicit requirements and define the functionalities to be developed during the project. Customers describe functionalities in informal descriptions called *User Stories*, which are assigned with a business value and a priority. Effort required for implementing a story is estimated by developers, who divide stories into micro-activities, called *Tasks*, to be performed in 2-3 ideal working days. Developers choose voluntarily which task to develop and take responsibility for their task being completed on time.

Each story generally requires an effort of 1-2 weeks. Stories are grouped into *Iterations* with a duration of 2-3 weeks. After a cycle of 2-3 iterations, the product is released to the customer. The customer is aware that the first release will provide only some functionalities and that many releases could be necessary before all functionalities are implemented.

One of the team members plays the role of *Tracker*. Tracking is the critical activity of measuring the team's progress. The essence of tracking is the face-to-face contact and a couple of questions per task. In fact, the tracker asks developers two questions about each task they have signed up for:

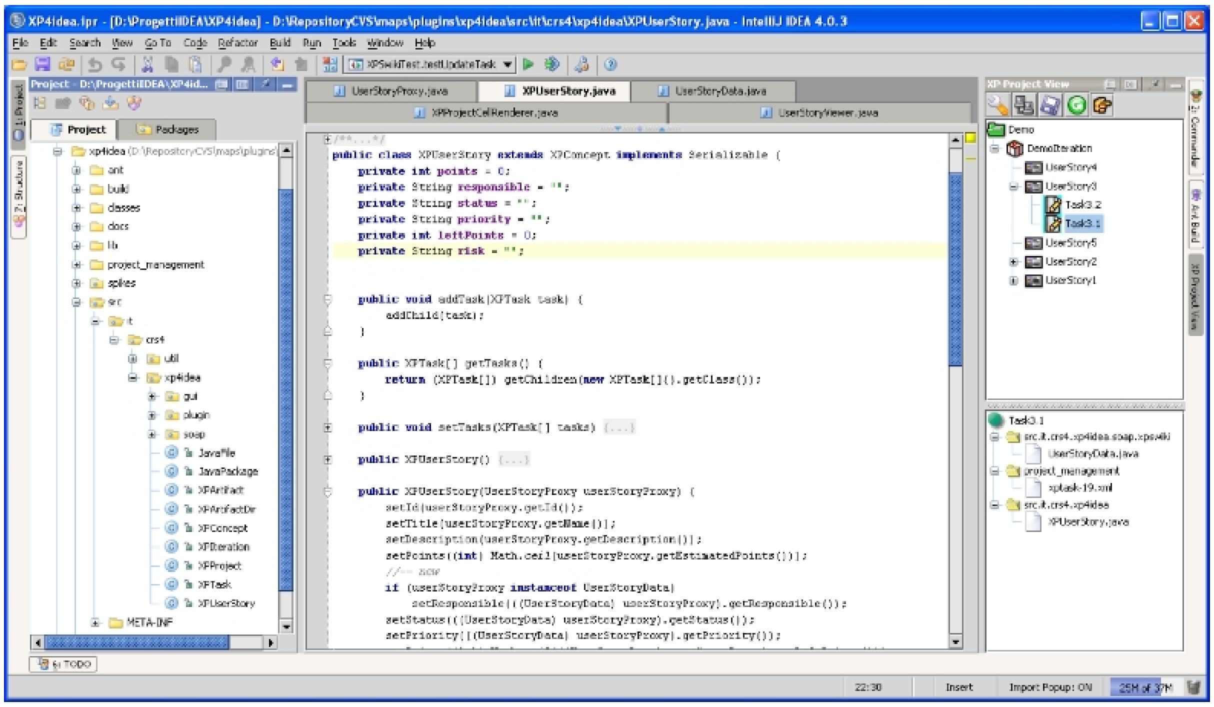
- how many ideal days have you worked on this?
- how many more ideal days do you need before you've done?

This information is critical to assess whether or not the effort estimated during the *Planning Game* is realistic.

3. MEASURING THE XP PROCESS

Measuring is a key factor in all engineering disciplines as *"You cannot control what you cannot measure" (Tom DeMarco)*. The objective of measuring something is to control, verify, estimate, and support decisions. What exactly is being measured can change depending on the point of view:

- human factors researchers may be interested in mechanisms and behaviors peculiar to an XP project [7];



```
/**...*/
public class XPUserStory extends XPConcept implements Serializable {
    private int points = 0;
    private String responsible = "";
    private String status = "";
    private String priority = "";
    private int leftPoints = 0;
    private String risk = "";


    public void addTask(XPTask task) {
        addChild(task);
    }

    public XPTask[] getTasks() {
        return (XPTask[]) getChildren(new XPTask[]{}.getClass());
    }

    public void setTasks(XPTask[] tasks) {...}

    public XPUserStory() {...}

    public XPUserStory(UserStoryProxy userStoryProxy) {
        setId(userStoryProxy.getId());
        setTitle(userStoryProxy.getName());
        setDescription(userStoryProxy.getDescription());
        setPoints((int) Math.ceil(userStoryProxy.getEstimatedPoints()));
        //== new
        if (userStoryProxy instanceof UserStoryData)
            setResponsible(((UserStoryData) userStoryProxy).getResponsible());
        setStatus(((UserStoryData) userStoryProxy).getStatus());
        setPriority(((UserStoryData) userStoryProxy).getPriority());
```

Figure 2: The GUI of XP4IDE embedded in IntelliJ IDEA.

- industry managers are mainly interested in maximizing the value of products minimizing costs and time to market;
- engineers may be interested in measuring product quality to decide what has to be changed in the process to improve the quality.

The quantitative assessment of an on-going project is of primary concern for monitoring actual versus scheduled progress, so that any necessary corrective action can be taken. In an agile process, this assessment must be immediate and effective in order to cope with the impact of changes in requirements, technology, budget and staff. One of the principles in the Agile Manifesto [1] states:

> At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Consider, as an example of XP tuning, the effort estimated by developers during the *Planning Game*. More precisely, developers are expected to read a *User Story*, which describes a system functionality, and to evaluate how many working days they need to fully implement such a functionality. This estimate is influenced by non-quantitative factors such as: technical skills, team's past experience in similar features, whether leading programmers are inclined to be prudent or optimistic. The Goal Question Metrics approach proposed by Victor Basili [2, 3] may help to perform a fine tuning of *User Story* evaluation. First, we define the Goal:
`Goal: Reduction of the estimation error committed by the XP team for a User Story.`

Then we define the measuring context with the following questions:

`Q1: By how much does the estimated effort differ from the actual effort required?`
`Q2: What similarities does the User Story have with stories developed in the past?`

Then, we define for each question the appropriate metrics:
For Q1:
`M1: estimation error for a single completed User Story:`
`(actual effort - estimated effort)/estimated effort`
`M2: mean and standard deviation of the estimation errors committed by the team for all the completed User Stories`

For Q2:
`M1: estimation error for a single completed User Story:`
`(actual effort - estimated effort)/estimated effort`
`M3: mean and standard deviation of the estimation errors committed by the team for the completed User Stories similar to the one in question`

The measurements collected in this way can be presented in the form of tables and diagrams as a visual aid. In addition, a mathematical model could be defined and used to support the estimation process, although this approach is likely to be rejected by XP programmers. So far we have presented only those metrics associated with a specific goal. The XP Process is not metric centered. Ron Jeffries argues that the only valuable metric in XP is the number of running and tested features in time (RTF, see [6]). On the other hand, it may be useful to calculate other specific metrics in order to achieve a specific goal. In this context the suite of tools we have developed has a primary role in that it is able to gather process data that could prove useful for calculating the desired process metrics.

4. XPSUITE: TOOLS FOR THE XP PROCESS

In this section we provide an overview of two Internet based tools we have developed to support planning and tracking in an XP project: XPSwiki and XP4IDE.

4.1 XPSwiki

XPSwiki [8, 9, 11] (Figure 1) was developed to meet real needs for software businesses to collect requirements and schedules in an electronic format. Managers, customers and developers can access XPSwiki by means of a Web browser. XPSwiki is built on top of a Wiki [5] engine written in Smalltalk called Swiki [10].

Wiki is a very simple and effective content management system. Users can browse Wiki pages and, if they have the appropriate privileges, can also edit the pages directly from the Web using a small set of simple formatting rules.

Swiki also enables to add structure to pages, defining input forms for them. In this way, in a Swiki one can have pages constrained by a given structure, that are used to hold structured information, as well as free-format pages, freely added to existing pages, like Post-it notes.

The direct advantages of a Web based tool for the management of XP process data can be summed up as follows:

- the tracker and the developers can enter effort data directly from the Web without handling paper documents;
- dispersed teams can cooperate and view the overall project status;
- user stories, task assignments, and any relevant effort data is immediately available on the Web, and accessible from programmer pair's workstations.

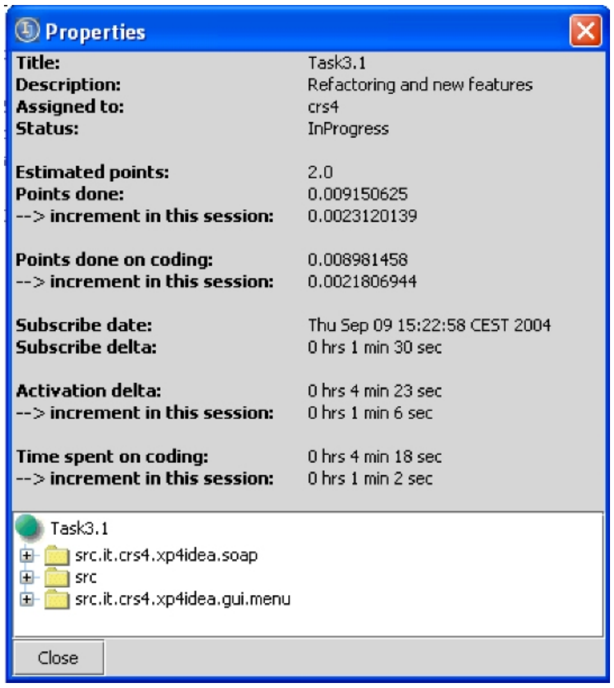
Although a Web based tool is effective during the *Planning Game*, it does lack one major feature: it is not integrated into the user's development environment. In the next section we describe XP4IDE, which integrates some of the XPSwiki functionalities directly into the IDE, providing two additional features:

- effort can be automatically recorded by the IDE;
- developers can view the XP process integrated with their development tools.

4.2 XP4IDE

XP4IDE is an Internet based tool which connects to a XPSwiki server and provides a view of the XP process data embedded in the GUI of the IDE (Figure 2).

The role of XP4IDE is twofold: on the one hand it enables the developer to view, directly integrated into the IDE, process data, *User Stories*, *Tasks*, acceptance tests and artifacts (classes, documentation files, Make files or Ant files, test cases, etc.), all managed and saved on the XPSwiki server. On the other hand, it measures the specific time spent on *User Stories*, *Tasks* and *Artifacts* and sends this information to the XPSwiki server, automating the tracking activity.



Title:	Task3.1
Description:	Refactoring and new features
Assigned to:	crs4
Status:	InProgress
Estimated points:	2.0
Points done:	0.009150625
--> increment in this session:	0.0023120139
Points done on coding:	0.008981458
--> increment in this session:	0.0021806944
Subscribe date:	Thu Sep 09 15:22:58 CEST 2004
Subscribe delta:	0 hrs 1 min 30 sec
Activation delta:	0 hrs 4 min 23 sec
--> increment in this session:	0 hrs 1 min 6 sec
Time spent on coding:	0 hrs 4 min 18 sec
--> increment in this session:	0 hrs 1 min 2 sec

```
Task3.1
  src.it.crs4.xp4idea.soap
  src
  src.it.crs4.xp4idea.gui.menu
```

Figure 3: Effort data and properties of a Task in the XP4IDE GUI.

The overall system architecture is quite simple: XPSwiki acts as backend in a client-server architecture where the application protocol is built on top of SOAP/TCP/IP. The integration between XPSwiki and XP4IDE provides the following advantages:

- each actor in the XP process accesses to process data with the right user interface for the right phase. Developers can use the IDE during development, while customers and managers can use XPSwiki by means of a simple web browser. Both of them can use the web interface during the *Planning Game*;
- it is possible to associate the *Artifacts* to the *Tasks* in which they are created and developed. This way, the tool collects data about the time spent on every *Artifact*, every *Task* and every *User Story*;
- measuring effectively the effort expended and presenting this information to developers (Figure 3), is one way of building a knowledge base useful for enhancing their ability to estimate stories in future planning activities (see the GQM example in section §3);
- beginners can immediately a view of the XP process. Concepts, roles and process data are presented in a structured way, minimizing the learning curve.

5. PROCESS DATA

This section describes the conceptual model underlying the XPSwiki and XP4IDE suite, showing the entities and their relations by means of UML class diagrams. This object-oriented structure can be navigated to extract process data for the computation of useful metrics.

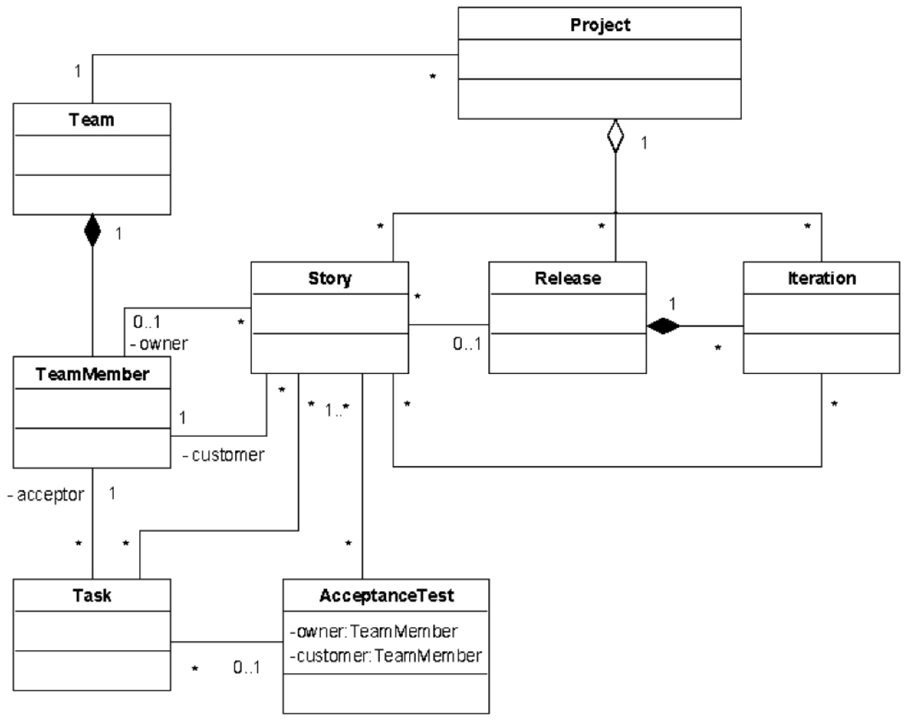


Figure 4: UML model of the XP process as supported by XPSwiki.

As previously mentioned in section §4, XPSwiki extends a wiki structure, inheriting in this way all its attractive features like versioning capability. This feature enables XPSwiki to keep track of all the versions of each single entity (*Task*, *User Story*, *Iteration*...).

Here we give a description of the UML model of the XPSwiki shown in Figure 4. Since this model represents the basis for managing an XP process previously described in section §2, only a brief description of the model is described here.
An XP *Project* is performed by a *Team*. A *Team* is composed of one or more *TeamMember*s. The *Project* is then divided into *Release*s and *Iteration*s of fixed length through the timeboxed approach of XP. A number of instances of *UserStory* must be implemented within an *Iteration*. Each *UserStory* is divided into one or more instances of *Task* and has one or more *Acceptance Test*s. The successful completion of all its *Acceptance Test*s proves the successful implementation of a *UserStory*.

To fully integrate the model with the development data collected by XP4IDE, we have extended the XPSwiki model to include other entities, such as *Artifacts*. As shown in Figure 5, *Tasks* and *Artifacts* are associated with a many-to-many relation (an *Artifact* could be created in a *Task* and modified in another one). An *Artifact* may be of different types such as *SourceCode* or a generic *Document*. A *Task* has its own *TrackTable* which stores information concerning the effort expended by developers for each specific working *Session*. In more detail, each *Session* contains the following fields:

- **subscriber**: the *TeamMember* who subscribed to the *Session* on that *Task*;
- **pairProgrammer**: the *TeamMember* who was programming with the *subscriber* in that *Session*;
- **startTime**: date and time *subscriber* started working on that *Session*;
- **subscribedDelta**: duration of the *Session*.

Moreover, each *Session* has its own *ArtifactsTrackTable* which keeps track of the effort spent on each *Artifact* modified or created during that *Session* (*ArtifactSession*). More in depth, each *ArtifactSession* stores the following data:

- **artifact**: *Artifact* involved in the *Session*;
- **startTime**: date and time *subscriber* started working on that *Artifact* in that *Session*;
- **activationDelta**: the period of time the *Artifact* window was active during that *Session*;
- **codingDelta**: the time spent on coding (typing) that *Artifact* during that *Session*.

6. WORKING WITH THE XPSUITE

In this section we describe the functionalities of the XPSuite from the developer point of view during a development session. To better harmonize with XP, the functionalities are described in the form of *User Stories*.

6.1 Story: Login and Connection to a Project

To start a working session, the developer opens the IDE, selects the docked GUI called XPView, the view of the XP project, which prompts for login and password to connect to the XPSwiki server. Then the developer selects the pair programmer with whom he is going to work with and the

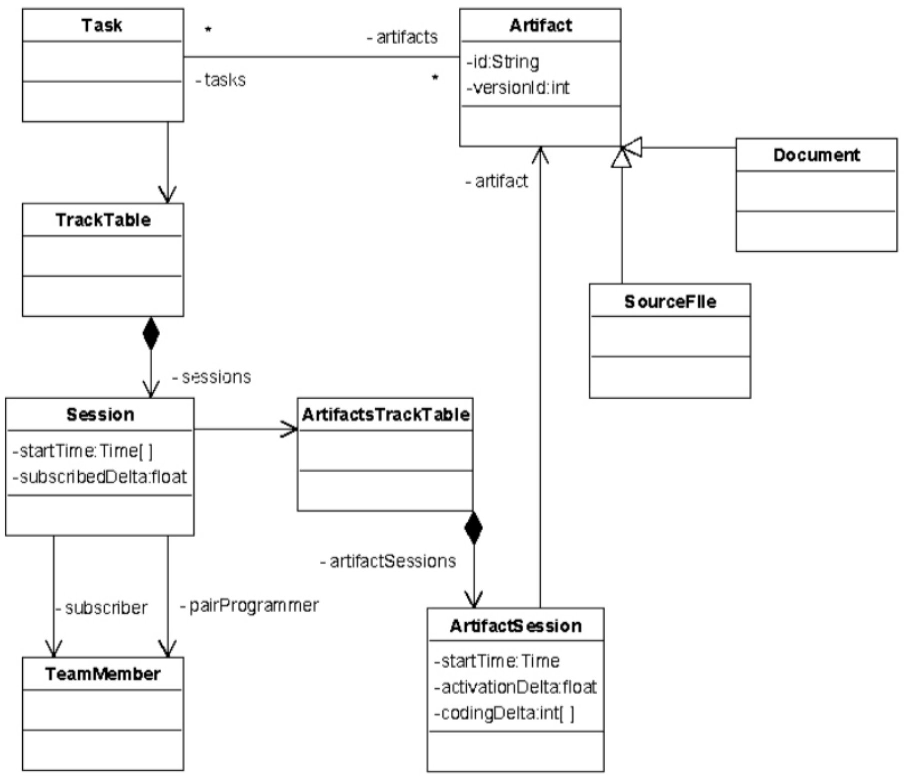


Figure 5: Extension of the XPSwiki model in order to support data collected by XP4IDE.

project among those available on the XPSwiki. Project data are loaded in the IDE and presented in the XPView in the form of a tree. The tree is composed of *Releases, Iterations, User Stories, Tasks* and *Artifacts*. *Tasks* already assigned to the developer are highlighted and sorted by priority. Every tree element has a property window which presents the attributes for that element (for instance, a story has a title, a description, a priority, a risk and an estimated effort).

6.2 Story: Subscribe to a Task

Once the developer has logged in, he can select a *Task* and subscribe to it. The status of the *Task* changes from "inactive" to "in progress". Developers subscribed to a *Task* can create and associate new *Artifacts* to that *Task* or modify existing ones even if associated with other *Tasks*.

6.3 Story: Automatic Measure of Effort Spent

During the working session, the XP4IDE measures:

- the *Artifact* activation delta (the period of time the *Artifact* window was active);
- the time spent on coding an *Artifact*;
- the *Task* activation delta (computed as the sum of the activation delta of all *Artifacts* developed in the *Task*);
- the time spent on coding within a *Task* (computed as the sum of the time spent on coding all *Artifacts* developed in the *Task*);
- the subscribe delta (total amount of time the developer is subscribed to a *Task*).

This measurement should be sent to the XPSwiki, choosing one or more of the following options:

- periodically;
- when the developer unsubscribes to a *Task*;
- when the *Task* is completed;
- when the IDE is closed.

6.4 Story: Open a Thread on a Story

A developer can post from the IDE user interface an issue or a question opening a new thread. The other members of the team can follow up the discussion and provide solutions or suggestions.

6.5 Story: Managing a Task

A developer should be able to see the status of a *Task*, to launch its tests, to work on its *Artifacts*, to close the *Task* or to suspend the *Task* defining a prerequisite (for instance Task1 depends on Task2).

7. PRIVACY ISSUES

Concern may arise over privacy issues, since software tools such as XPSuite could clearly be used to spy the behavior of developers, instead of supporting them in their day to day activities, as they are supposed to do. Such objections, perfectly justified, should be seriously taken into account when developing and deploying systems like XPSuite. Even in those cases where it is possible to oblige developers to use automatic tracking tools like XPSuite, this should be seriously evaluated. In fact, Extreme Programming is claimed to be a people centered methodology, hinged on trust and reciprocal esteem between team members, and on making full use of human resources, instead of imposing control and authoritarian management.

In our opinion, XPSuite (and other similar tools) should be adopted, upon mutual agreement with team members, if necessary in a smooth, gradual way. On the other hand, the use of XPSuite has proven valuable in the experimentations conducted so far, both for the advantages that it offers when coordinating dispersed teams and for its ability to gather process data contextualized against the specific task and process phase. Being able to measure the exact time spent on any given artifact or task, and not rely simply on the estimates and reports provided by the developers, that may contain errors, is a clear advantage for the whole team.

These data can of course be of interest even to researchers, if they help to expose mechanisms and behaviors underlying the development process, spot critical phases and activities, prevent errors.

8. RELATED WORKS

There are a number of nascent or on-going projects similar to XPSuite suggesting that the integration of IDEs with server side XP management tools is recognized as a valid option. Among others we mention:

- **xplanner-idea**: a plug-in for the IntelliJ IDEA which connects to the XPlanner tool (http://sourceforge.net/projects/xplanner), using a SOAP API;
- **ecliXPweb**: a plug-in for the Eclipse IDE which allows to connect to a XPWeb database.

In our opinion, it may be of some advantage to define a common protocol for the communication between IDEs and server side Project Management tools, in order to facilitate interoperability between clients and third-party servers.

9. CONCLUSION

In this paper we have outlined the nuts and bolts of XPSuite, a set of tools aimed to provide support an assist metrics gathering in an XP project. The tools allow multichannel interaction with process data both via Web pages and by means of rich user interfaces directly embedded in the IDE. Although the tools have been used for internal development and have proven effective for supporting activities of dispersed teams, they are in an early-stage of development and we would like to have other teams use these tools to obtain valuable feedback. To facilitate the use of our system we intend implementing a plug-in for the Eclipse IDE.

10. ACKNOWLEDGEMENTS

This work was supported by MAPS (Agile Methodologies for Software Production) research project, contract/grant sponsor: FIRB research fund of MIUR, contract/grant number: RBNE01JRK8.

11. REFERENCES

[1] Manifesto for agile software development. http://agilemanifesto.org.

[2] V. Basili. Software modeling and measurement: The goal/question/metric paradigm, 1992.

[3] V. Basili and D. M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738, 1984.

[4] K. Beck. *Extreme Programming Explained: Embracing Change*. Addison-Wesley, 1999.

[5] W. Cunningham and B. Leuf. *The Wiki Way*. Addison-Wesley, 2001.

[6] R. Jeffries. A metric leading to agility, June 2004. http://www.xprogramming.com/xpmag/jatRtsMetric.htm.

[7] K. Mannaro, M. Melis, and M. Marchesi. Empirical analysis on the satisfaction of IT employees comparing XP practices with other software development methodologies. In *Proceedings of the 5th international conference XP2004*, pages 166–174. Springer, 2004.

[8] S. Pinna and al. Developing a Tool Supporting XP Process. In *Extreme Programming and Agile Methods, Lecture Notes in Computer Science*, pages 151–160. Springer, 2003.

[9] S. Pinna and al. XPSwiki: an Agile Tool Supporting the Planning Game. In *Proceedings of the 4th international conference XP2003*, pages 104–113. Springer, 2003.

[10] Swiki: the wiki engine written in Smalltalk. http://minnow.cc.gatech.edu/swiki.

[11] XPSwiki: an open source web tool for eXtreme programming teams. http://agile.diee.unica.it/xpswiki.