Task Tracker

An Integrated Project Administration Tool

Emiliano Pérez, Sebastián Perez Escribano, Andrés Fortier†
LIFIA, Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina.
{emiliano.perez, sebastian.perez.escribano, andres}@lifia.info.unlp.edu.ar
†Also at DSIC, Universidad Politécnica de Valencia, Valencia, Espa˜na.
and CONICET

Keywords: Project Management, Peer Review, Agile Methodologies, Constant Integration, Seaside.

Smalltalk Dialect:VisualWorks 7.4.1 and 7.6 non-commercial

Introduction

Task Tracker can be described as a tool for managing and publishing code, but actually it's a little more than that. In a nutshell, the purpose of the tool is to provide dynamic workflow management with constant code integration. This process is based in agile methodologies and the Task Tracker is used as a tool to suppport this process. To weave the process management with the developer's enviroment we decided to embed the tool inside Smalltalk.

The main motivation for developing this tool is that the current support for code management package-oriented: source code is structured in code components using particular criteria, specifically by structure or by functionality. This mechanism usually is enough for a single developer but when a group of developers is working in the same project or projects there are others implications that have to be addressed. In particular we would like to:

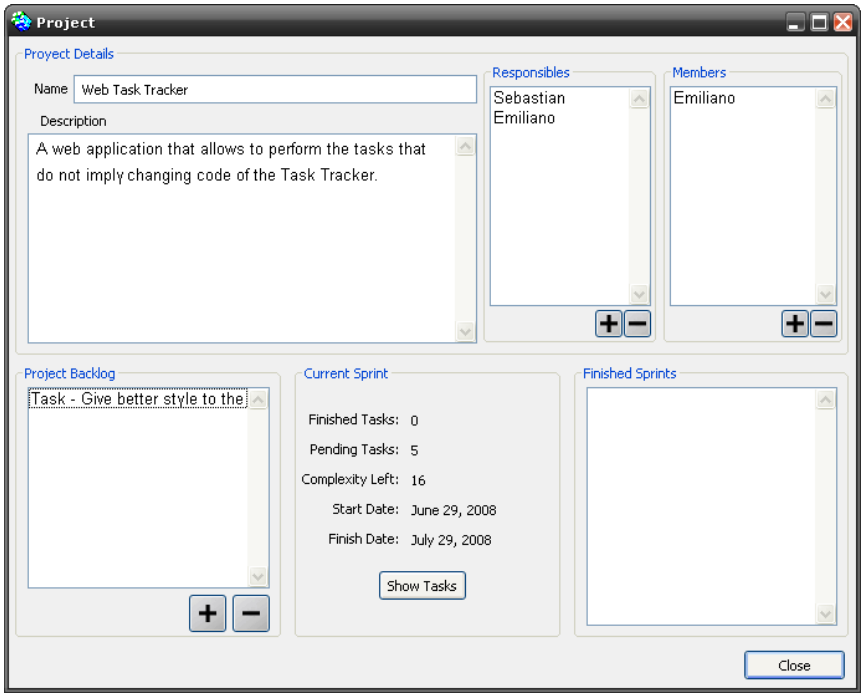
- Avoid external coordination when developers are working on the same code components. This also requires a versioning mechanism and complex branching and merging processes that have to be overseen by all the people involved.
- Automatically keep information about the development process. Besides the publication version, maturity level and developers comments, all the information about the context of the publication is lost. Working perdiods for the users that modified the code, what is the functionality being developed, wich project the code belongs to and peer reviews of the code, are just some examples of all the pieces of information that have to be documented by developers with external tools such as wikis or project management applications (and sometimes more than one is needed).

Both problems require external coordination and extra effort from the developers and the managers. To solve these issues, the Task Tracker takes a different aproach for project management and documentation. It adopts the concept of *task* (similar to the one proposed in Scrum [http://www.mountaingoatsoftware.com/scrum]) as the main funcional unit that represents a particular job to be done. Following the ideas presented in Scrum, a *project* is the top level concept, grouping users and tasks. Tasks are scheduled within iteratives and relatively short working periods called *sprints,*that also define the time period in which these tasks must be completed. At any given time there is one (an only one) active sprint in each project. Once a sprint has started, no tasks can be added or removed; this ensures that the requirements are stable during these small

cycles. Thus, new tasks are added to the project backlog and can only be included once the current sprint has finished and a new sprint is created.

Based on this methodology we implemented a tool that supports all the steps in the lifecylce of a project. In particular the tool supports:

- Project management in terms of projects, users, sprints and tasks.
- Code management using the task as the main functional unit. Code components (i.e. packages and bundles in VW) are still used to organize the project code. However, the developer is most of the time focused on solving tasks and not in code management.
- Project tracking by keeping logs of the development lifecycle:
  - Code changes are logged in a task basis, including the developer that did it and the time stamp of the change.
  - Activity periods are logged for each task.
- Peer code review over each completed task. When a developer finishes a task, all the changes are packed and sent to another member to be reviewed.
- Complexity and Priority information for each task, allowing to perform post-sprint and post-project analyis (e.g. time metrics for each kind of task).
- Semi-automatic publishing, documenting and integration support for each completed task.
- Different kinds of tasks: Requirements, Bugs, Enhancements, Peer Reviews, etc.
- Multi-User capabilities, allowing several developers to work within the same project without external sincronization.
- Using STORE as the underlying support for versioning and publishing. Each completed task reflects as a new partial version of a project.
- Full integration with the Refactoring Browser, allowing for a quick switch between the code development and the task administration.
- Web access to the proyect administration and peer review tasks (early version).

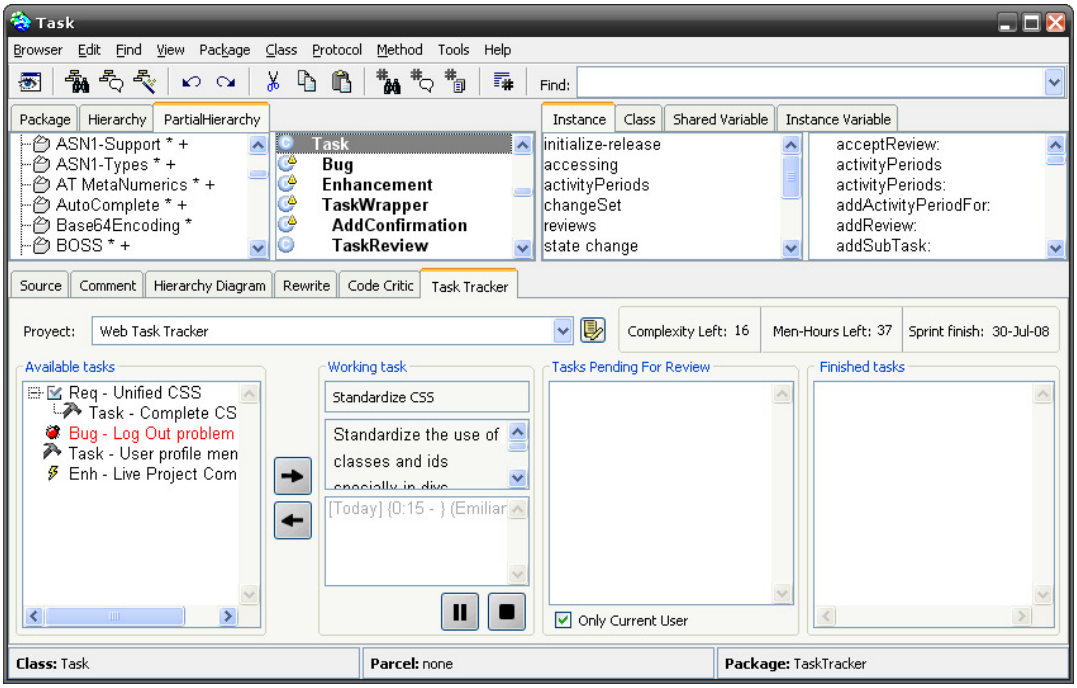


- *Figure 1. Project Editor.*

Task Tracker Overview

The first step when using the tool is to define a new project. This is done by means of the project editor (see Figure 1), where the basic information of the project is loaded (involved users, tasks in the backlog, current sprint, etc.).

Once the project has been defined and a sprint started (from now on the *current sprint*) the developer can access the Task Tracker functionality from the Refactoring Browser (see Figure 2). In the next subsections we will summarize the main features of our tool



- *Figure 2: Main Task Tracker user interface.*

Change Registration

A task can be in four main states: not started, in progress, paused and finnished. At any given time only one task can be in progress, since in this state is assumed that the code modified in the image corresponds to that particular task. A new activity period is created when the task is activated and it is closed when the tasks is paused (deactivated) or stopped (finished).

Changes are registered using the standard VisualWorks mechanism for change logging: each task has a ChangeSet associated that receives all the changes made in the code of the current image while the task is active. Thus, any modification made to the environment is associated with a task.

Publish Process

When a user finishes a task, changes are checked running the smalllint rules with the Code Critic tool. If any errors are detected, the user is given the chance to go back and fix them. Once the code is checked, the tool starts a peer review for that task and lets the user choose a reviewer. The chosen reviewer will now see a new special kind of task (a *peer review*) in his available tasks list. During the peer review process the reviewer is able to see the sources and make comments to the author.

If the reviewer accepts the submission, the comments are sent back to the developer, which now gets a chance of doing the required modifications to his code and integrate his code to the base code. On the other hand, if the review was rejected, the developer has to fix the problems and send the task back for another review.

By using the associated change set, the tool can automatically select the code components to be published. In case the chosen components already have a version in the repository both versions are compared to detect inconsistencies. The tool automatically resolves all the version conflicts that are presented by using the Merge Tool, only requiring user intervention if the tool can´t decide which version to use. After this step the Pusblish Dialog is presented and autocompleted by the Task Tracker with information about the task that triggered the publishing process. Once the components are published the task is officially marked as reviewed and the code as integrated.

Multi-User environment

The Task Tracker tool uses a remote model by means of a client-server configuration, using Opentalk as a communication layer. Once an image is configured as server, many client images can share the same Task Tracker model.

Although this configuration allows for multiple users to be working in the same project at the same time, certain restrictions have to be made for the concurrent access. Particularly, a specific task can be only accessed by one user at the time. This constraint is achieved by locking a task when a developer starts to work on it (i.e. changing to the *in-progress* state). In case a user pauses a task and another user decides to keep working on it, all the changes made in the first image are automatically loaded in the latter image and the lock is given to the requesting user.

Task Tracker (Web Interface)

Even though some aspects of the tool are tightly integrated with the development environment, we can explode the model in the server image so that it can be also accessed from a browser. In particular project tracking information and the peer review processes can be of interest not only to project developers, but also to managers or external reviewers.

The web interface is being developed in parallel with the Task Tracker, using Seaside, Scriptaculous and Shore-Components. This feature is a work in progress so only the project information and Peer Reviews are handled by the web application, but we expect to be able to manage and audit most of a project development cycle.

In its current version, users can see their projects and the current tasks state and perform pending peer reviews from anywhere, without the need of having the Task Tracker installed or even be running a VisualWorks image. Also user information, such as his/her profilem can be edited within the web application.

About Software Versions

The Software was originally develped in VisualWorks 7.4.1 non-comercial but is now being ported to version 7.6. for Shore-Components compatibility. Some aspects of the Code Tool integration to the web browser have changed between these two versions, thus the client side of the Task Tracker is not working properly in 7.6.

We recommend using the 7.6. version of the Task Tracker as a server image and also for running the Web Task Tracker. The client version of the Task Tracker should run on a 7.4.1. image.