Design document 0.25

SQUID

Helsinki 9th March 2005

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

Course

581260 Software Engineering Project (6 cr)

Project Group

Mikko Jormalainen
Samuli Kaipiainen
Aki Korpua
Esko Luontola
Aki Sysmäläinen

Client

Lauri J. Pesonen
Fabio Donadini
Tomas Kohout

Project Masters

Juha Taina
Jenni Valorinta

Homepage

`http://www.cs.helsinki.fi/group/squid/`

Change Log

Version	Date	Modifications
0.1	4.3.2005	First version with nothing in it (Samuli Kaipiainen)
0.2	8.3.2005	Some class descriptions (Aki Korpua, Samuli Kaipiainen)
0.25	9.3.2005	Macros for class/field/method documentation (Esko Luontola)
		RunQueue (Esko Luontola)

Contents

1 Introduction

1.1 Meaning and structure of the document

1.2 Glossary

2 Overview of the system

See Figure 1.

3 Architecture description

4 Classes and methods

4.1 Project Explorer

4.1.1 ProjectExplorerPanel

Package	ikayaki.gui
Declaration	public class ProjectExplorerPanel
Extends	JPanel
Created by	MainViewPanel
Uses 1	ProjectExplorerTableModel (4.1.2)
Uses 2	ProjectExplorerPopupMenu (4.1.3)

Creates a history/autocomplete field (browserField) for choosing the project directory, a listing of project files in that directory (explorerTable) and in that listing a line for creating new project, which has a textbox for project name, an AF/TH ComboBox and a "Create new" button for actuating the creation. Also has a right-click popup menu for exporting project files.

Event A	*On browserField change* - send browserField's text to RunQueue (singleton?) which schedules disk access and autocomplete. (This could be in a separate BrowserFieldComponent class)
Event B	*On explorerTable mouse right-click* - create a ProjectExplorerPopupMenu for right-clicked project file. (This could be in ProjectExplorerTableModel class)
Event C	*On createNewProject mouseclick* - create a new project, set it active and tell ProjectExplorerTableMode to update itself.
Event D	*On browseButton mouseclick* - open a FileChooser dialog for choosing the directory.

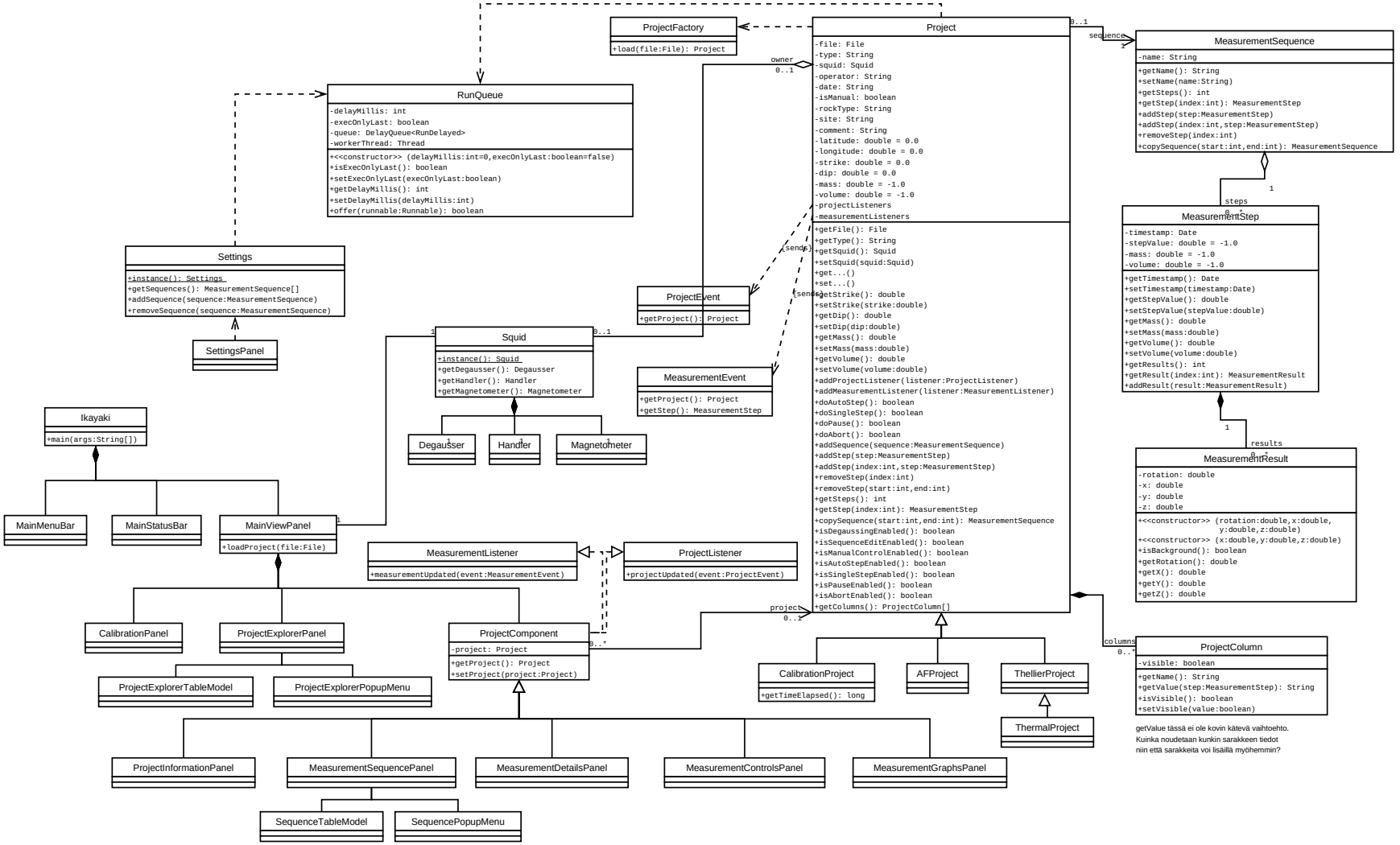

Figure 1: Squid class diagram

`private BrowserFieldComponent browserField`
(need a separate class for this?)

`private ProjectExplorerTableModel explorerTable`

`private JButton createNewProject`

`private JButton browseButton`

`private JTextField newProject`

`private JComboBox newProjectType`

`public ProjectExplorerPanel()`
Creates its components, places them and sets the last project folder as current project folder.

`public void createNewProject()`
Creates a new project, sets it active, sends it to MainViewPanel, tells explorerTable to reset newProject and newProjectType fields.

`public void doBrowse()`
Creates a FileChooser dialog and tells explorerTable and browserField to change to directory returned by it.

4.1.2 ProjectExplorerTableModel

Package	ikayaki.gui
Declaration	public class ProjectExplorerTableModel
Extends	AbstractTableModel
Created by	ProjectExplorerPanel
Uses 1	MainViewPanel (**??**)

Creates a list of project files in directory. Handles loading selected projects and executing export choice.

Event A	*On newDirectoryEvent* - updates list of project files on JTable
Event B	*On selectTable MouseEvent* - sets new Project active

`private File directory`
Default value null
Currently opened directory

`private Vector<File> files`
Default value new Vector<File>()
Project files in the current directory.

`public ProjectExplorerTableModel()`
Reads the contents of the default directory and initializes the file list.

`public ProjectExplorerTableModel(String directory)`
Reads the contents of the given directory and initializes the file list.
Parameter 1 *directory* - path to the directory to be opened

`public void loadDirectory(String directory)`
Update list to show given directory.
Parameter 1 *directory* - path to the directory to be opened

`public void loadProject()`
Gets the selected line from the file list and sends the file to MainViewPanel.

4.1.3 ProjectExplorerPopupMenu

Package	ikayaki.gui
Declaration	public class ProjectExplorerPopupMenu
Extends	JPopupMenu
Created by	ProjectExplorerPanel

Shows choices to export: AF, Thellier, Thermal and executes selected

Event A	*On selectItem mouseEvent* - tells selected Project to create selected file from it's self

`public ProjectExplorerTableModel()`
Builds the menu items

4.2 Utilities

4.2.1 RunQueue

Package	ikayaki.util
Declaration	public class RunQueue
Uses 1	RunQueue.RunQueueThread (4.2.2)
Uses 2	RunQueue.RunDelayed (4.2.3)

Executes Runnable objects in a private worker thread after a pre-defined delay. The worker thread will terminate automatically when there are no runnables to be executed. Optionally executes only the last inserted runnable. All operations are thread-safe.
This class can be used for example in connection with a "continuous search" invoked by a series of GUI events (such as a DocumentListener), but it is necessary to react to only the last event after a short period of user inactivity.
Design patterns Command

`private int delayMillis`

Default value 0

Defines how long is the delay in milliseconds, after which the events need to be run.

`private boolean execOnlyLast`

Default value false

Defines if only the last event should be executed. If false, then all of the events are executed in the order of appearance.

`private DelayQueue<RunDelayed> queue`

Default value new DelayQueue<RunDelayed>()

Prioritized FIFO queue for containing the RunDelayed items that have not expired. If execOnlyLast is true, then this queue should never contain more than one item.

`private Thread workerThread`

Default value null

The worker thread that will run the inserted runnables. If the thread has no more work to do, it will set workerThread to null and terminate itself.

`RunQueue()`

Creates an empty RunQueue with a delay of 0 and execOnlyLast set to false.

`public RunQueue(int delayMillis)`

Creates an empty RunQueue with execOnlyLast set to false.

Parameter 1 *delayMillis* - the length of execution delay in milliseconds; if less than 0, then 0 will be used.

`public RunQueue(boolean execOnlyLast)`

Creates an empty RunQueue with a delay of 0.

Parameter 1 *execOnlyLast* - if true, only the last event will be executed after the delay; otherwise all are executed in order of appearance.

`public RunQueue(int delayMillis, boolean execOnlyLast)`

Creates an empty RunQueue.

Parameter 1 *delayMillis* - the length of execution delay in milliseconds; if less than 0, then 0 will be used.

Parameter 2 *execOnlyLast* - if true, only the last event will be executed after the delay; otherwise all are executed in order of appearance.

`public synchronized boolean isExecOnlyLast()`

Returns true if only the last event will be executed after the delay; otherwise false.

`public synchronized void setExecOnlyLast(boolean execOnlyLast)`

Parameter 1 *execOnlyLast* - if true, only the last event will be executed after the delay; otherwise all are executed in order of appearance.

`public synchronized int getDelayMillis()`

Returns the delay in milliseconds

`public synchronized void setDelayMillis(int delayMillis)`

Parameter 1 *delayMillis* - delay in milliseconds; if less than 0, then the new value is ignored.

`public synchronized boolean offer(Runnable runnable)`

Inserts a Runnable object to the end of the queue. It will remain there until it is executed or another object replaces it. If execOnlyLast is set to true, the queue will be cleared before inserting this runnable to it. If there is no worker thread running, a new one will be spawned.

Parameter 1 *runnable* - the Runnable to be run after a pre-defined delay
Returns true
Throws *NullPointerException* - if runnable is null

4.2.2 RunQueue.RunQueueThread

Package ikayaki.util
Declaration private class RunQueueThread
Extends Thread
Created by RunQueue

Keeps on checking the RunQueue.queue to see if there are Runnables to be executed. If there is one, execute it and proceed to the next one. If an uncaught Throwable is thrown during the execution, prints an error message and stack trace to stderr. If the queue is empty, this thread will set RunDelayed.workerThread to null and terminate itself.

`public void run()`

4.2.3 RunQueue.RunDelayed

Package ikayaki.util
Declaration private class RunDelayed
Implements Delayed
Created by RunQueue

Wraps a Runnable object and sets the delay after which it should be executed by a worker thread.

`private long expires`

The point in time when this RunDelayed will expire.

`private Runnable runnable`

Contained Runnable object to be run after this RunDelayed has expired.

`public RunDelayed(Runnable runnable, int delayMillis)`

Creates a new RunDelayed item that contains runnable.

Parameter 1	*runnable* - the Runnable to be contained
Parameter 2	*delayMillis* - delay in milliseconds

`public long getDelay(TimeUnit unit)`

Returns the remaining delay associated with this object, always in milliseconds.

Parameter 1	*unit* - ignored; always assumed TimeUnit.MILLISECONDS
Returns	the remaining delay; zero or negative values indicate that the delay has already elapsed

`public Runnable getRunnable()`

Returns the contained Runnable.

Returns	the Runnable given as constructor parameter

`public int compareTo(Delayed delayed)`

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameter 1	*delayed* - the Delayed to be compared.
Returns	a negative integer, zero, or a positive integer as this delay is less than, equal to, or greater than the specified delay.

4.3 Some other specific part (subsystem) to this system...

5 Package structure

6 Bibliography