Design document 0.4

SQUID

Helsinki 13th 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)
0.3	11.3.2005	Conventions added, Class diagrams improved (Esko Luontola)
		Subsystem sections (Samuli Kaipiainen
0.4	12.3.2005	Measurement controls (Samuli Kaipiainen)
		Class diagram modified (Esko Luontola)
		Class descriptions for Project data (Esko Luontola)

Contents

1 Introduction

1.1 Meaning and structure of the document

1.2 Glossary

2 Conventions

Everybody will follow the Code Conventions for the Java Programming Language set by Sun, with the following refinements.

- Line length will be set to 120 characters, because we prefer coding in high resolutions.
- If possible, set your IDE to use spaces instead of tabs (to avoid problems if somebody has set tab to 4 spaces, although it should be 8). Indentation is 4 spaces, as set by Sun.
- Every method and non-trivial field must have Javadoc comments. Every parameter, return value and exception of methods must be mentioned. For trivial getters and setters, only the @param or @return is necessary - generic comment is not required.
- Every if, for and while loop must use braces `{}`, even when there will be only one statement in the block, as set by Sun.
- The @author comment for every class should have the name of the person who wrote (and designed) the class. Then we will know who to ask, if there are some questions about the code.
- Every source file is subject to automatic code reformatting by a Java IDE, in which case the reformatter must follow these code conventions.
- TODO-comments should be set by the programmer, if there is some part that needs more work. The format is `"// TODO: comments"`

The Code Conventions are available at
`http://java.sun.com/docs/codeconv/`

This program will be written with Java 1.5. Every programmer should have a look at the new features that were introduced to the Java language. Especially noteworthy are Generics, Foreach-loop and Enums. The following article will explain them in a nutshell.
`http://java.sun.com/developer/technicalArticles/releases/j2se15/`

It is recommendable for everybody to have a quick glance at Design Patterns. Here are some useful links.
`http://sern.ucalgary.ca/courses/SENG/609.04/W98/notes/`
`http://www.dofactory.com/Patterns/Patterns.aspx`

3 Overview of the system

See Figure 1 and Figure 2.

4 Architecture description

5 Data classes and methods

5.1 Project data

5.1.1 Project

Package	ikayaki
Declaration	public class Project
Created by	ProjectExplorerPanel
Uses 1	MeasurementSequence (5.1.4)
Uses 2	MeasurementStep (5.1.5)
Uses 3	MeasurementResult (5.1.7)
Uses 4	MeasurementValue (5.1.9)
Uses 5	Squid (5.2.1)
Uses 6	RunQueue (5.5.1)
Uses 7	ProjectEvent (5.1.10)
Uses 8	MeasurementEvent (5.1.12)

Represents a measurement project file. Project is responsible for managing and storing the data that is recieved from the magnetometer measurements. Any changes made to the project will be written to file regularly (autosave).
Project is responsible for controlling the magnetometer through the SQUID API. Controlling the SQUID will be done in a private worker thread. Only one project at a time may access the SQUID.

Design patterns	Facade
Event A	*On property change* - Autosaving will be invoked and the project written to file after a short delay.
Event B	*On measurement started/ended/paused/aborted* - ProjectEvent will be fired to all project listeners.
Event C	*On measurement subphase started/completed* - MeasurementEvent will be fired to all measurement listeners.
Event D	*On declination/inclination/volume changed* - The updated transformation matrix will be applied to all measurements and a ProjectEvent will be fired to all project listeners.

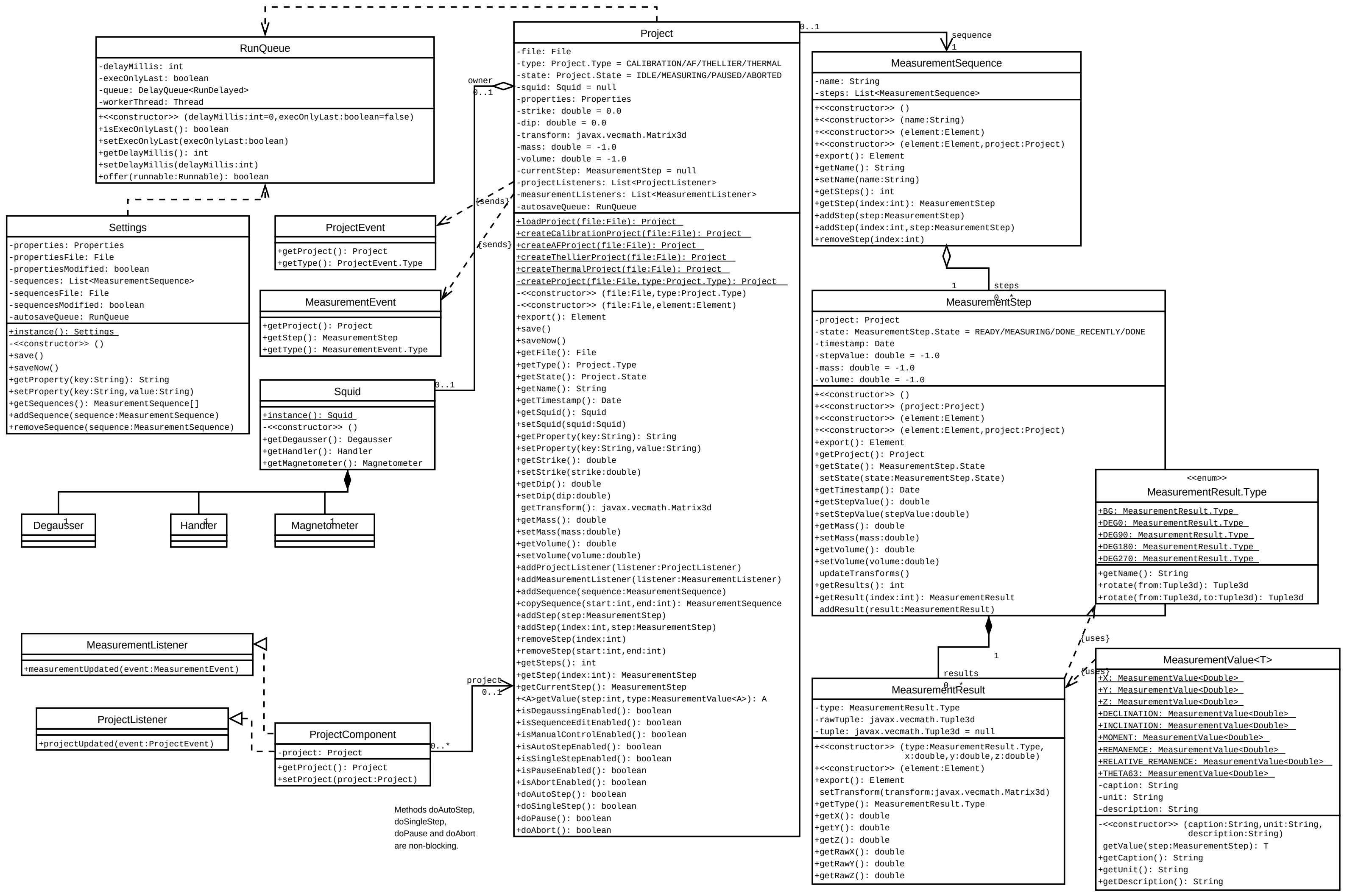
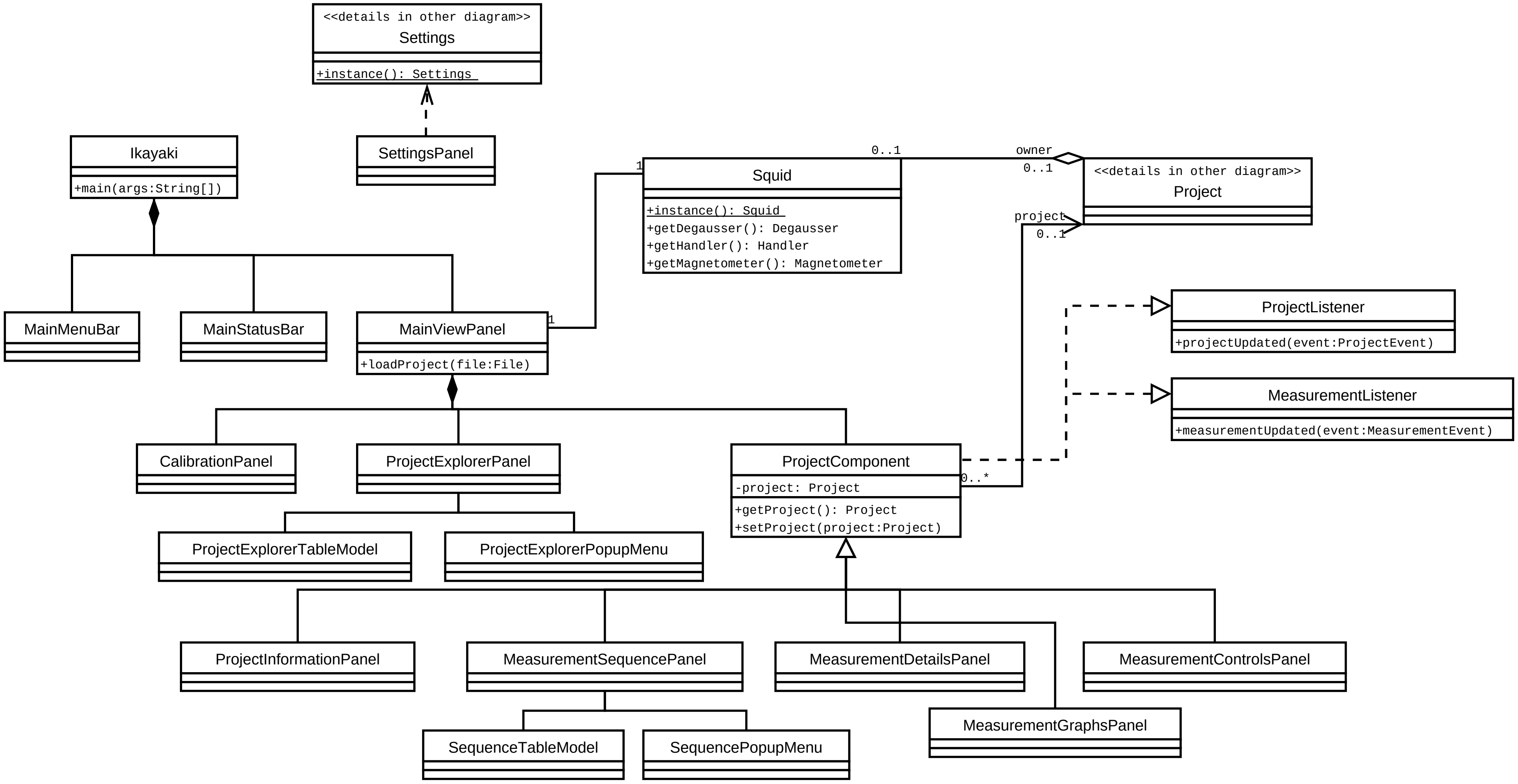

RunQueue
-delayMillis: int
-execOnlyLast: boolean
-queue: DelayQueue<RunDelayed>
-workerThread: Thread
+<<constructor>> (delayMillis:int=0,execOnlyLast:boolean=false)
+isExecOnlyLast(): boolean
+setExecOnlyLast(execOnlyLast:boolean)
+getDelayMillis(): int
+setDelayMillis(delayMillis:int)
+offer(runnable:Runnable): boolean
Settings
-properties: Properties
-propertiesFile: File
-propertiesModified: boolean
-sequences: List<MeasurementSequence>
-sequencesFile: File
-sequencesModified: boolean
-autosaveQueue: RunQueue
+instance(): Settings
-<<constructor>> ()
+save()
+saveNow()
+getProperty(key:String): String
+setProperty(key:String,value:String)
+getSequences(): MeasurementSequence[]
+addSequence(sequence:MeasurementSequence)
+removeSequence(sequence:MeasurementSequence)
ProjectEvent
+getProject(): Project
+getType(): ProjectEvent.Type
MeasurementEvent
+getProject(): Project
+getStep(): MeasurementStep
+getType(): MeasurementEvent.Type
Squid
+instance(): Squid
-<<constructor>> ()
+getDegausser(): Degausser
+getHandler(): Handler
+getMagnetometer(): Magnetometer
Degausser
Handler
Magnetometer
1
1
1
0..1
owner
0..1
{sends}
{sends}
MeasurementListener
+measurementUpdated(event:MeasurementEvent)
ProjectListener
+projectUpdated(event:ProjectEvent)
ProjectComponent
-project: Project
+getProject(): Project
+setProject(project:Project)
0..*
project
0..1
Methods doAutoStep,
doSingleStep,
doPause and doAbort
are non-blocking.
Project
-file: File
-type: Project.Type = CALIBRATION/AF/THELLIER/THERMAL
-state: Project.State = IDLE/MEASURING/PAUSED/ABORTED
-squid: Squid = null
-properties: Properties
-strike: double = 0.0
-dip: double = 0.0
-transform: javax.vecmath.Matrix3d
-mass: double = -1.0
-volume: double = -1.0
-currentStep: MeasurementStep = null
-projectListeners: List<ProjectListener>
-measurementListeners: List<MeasurementListener>
-autosaveQueue: RunQueue
+loadProject(file:File): Project
+createCalibrationProject(file:File): Project
+createAFProject(file:File): Project
+createThellierProject(file:File): Project
+createThermalProject(file:File): Project
-createProject(file:File,type:Project.Type): Project
-<<constructor>> (file:File,type:Project.Type)
-<<constructor>> (file:File,element:Element)
+export(): Element
+save()
+saveNow()
+getFile(): File
+getType(): Project.Type
+getState(): Project.State
+getName(): String
+getTimestamp(): Date
+getSquid(): Squid
+setSquid(squid:Squid)
+getProperty(key:String): String
+setProperty(key:String,value:String)
+getStrike(): double
+setStrike(strike:double)
+getDip(): double
+setDip(dip:double)
getTransform(): javax.vecmath.Matrix3d
+getMass(): double
+setMass(mass:double)
+getVolume(): double
+setVolume(volume:double)
+addProjectListener(listener:ProjectListener)
+addMeasurementListener(listener:MeasurementListener)
+addSequence(sequence:MeasurementSequence)
+copySequence(start:int,end:int): MeasurementSequence
+addStep(step:MeasurementStep)
+addStep(index:int,step:MeasurementStep)
+removeStep(index:int)
+removeStep(start:int,end:int)
+getSteps(): int
+getStep(index:int): MeasurementStep
+getCurrentStep(): MeasurementStep
+<A>getValue(step:int,type:MeasurementValue<A>): A
+isDegaussingEnabled(): boolean
+isSequenceEditEnabled(): boolean
+isManualControlEnabled(): boolean
+isAutoStepEnabled(): boolean
+isSingleStepEnabled(): boolean
+isPauseEnabled(): boolean
+isAbortEnabled(): boolean
+doAutoStep(): boolean
+doSingleStep(): boolean
+doPause(): boolean
+doAbort(): boolean
0..1
sequence
1
MeasurementSequence
-name: String
-steps: List<MeasurementSequence>
+<<constructor>> ()
+<<constructor>> (name:String)
+<<constructor>> (element:Element)
+<<constructor>> (element:Element,project:Project)
+export(): Element
+getName(): String
+setName(name:String)
+getSteps(): int
+getStep(index:int): MeasurementStep
+addStep(step:MeasurementStep)
+addStep(index:int,step:MeasurementStep)
+removeStep(index:int)
1
steps
0..*
MeasurementStep
-project: Project
-state: MeasurementStep.State = READY/MEASURING/DONE_RECENTLY/DONE
-timestamp: Date
-stepValue: double = -1.0
-mass: double = -1.0
-volume: double = -1.0
+<<constructor>> ()
+<<constructor>> (project:Project)
+<<constructor>> (element:Element)
+<<constructor>> (element:Element,project:Project)
+export(): Element
+getProject(): Project
+getState(): MeasurementStep.State
setState(state:MeasurementStep.State)
+getTimestamp(): Date
+getStepValue(): double
+setStepValue(stepValue:double)
+getMass(): double
+setMass(mass:double)
+getVolume(): double
+setVolume(volume:double)
updateTransforms()
+getResults(): int
+getResult(index:int): MeasurementResult
addResult(result:MeasurementResult)
<<enum>>
MeasurementResult.Type
+BG: MeasurementResult.Type
+DEG0: MeasurementResult.Type
+DEG90: MeasurementResult.Type
+DEG180: MeasurementResult.Type
+DEG270: MeasurementResult.Type
+getName(): String
+rotate(from:Tuple3d): Tuple3d
+rotate(from:Tuple3d,to:Tuple3d): Tuple3d
{uses}
{uses}
1
results
0..*
MeasurementResult
-type: MeasurementResult.Type
-rawTuple: javax.vecmath.Tuple3d
-tuple: javax.vecmath.Tuple3d = null
+<<constructor>> (type:MeasurementResult.Type,
x:double,y:double,z:double)
+<<constructor>> (element:Element)
+export(): Element
setTransform(transform:javax.vecmath.Matrix3d)
+getType(): MeasurementResult.Type
+getX(): double
+getY(): double
+getZ(): double
+getRawX(): double
+getRawY(): double
+getRawZ(): double
MeasurementValue<T>
+X: MeasurementValue<Double>
+Y: MeasurementValue<Double>
+Z: MeasurementValue<Double>
+DECLINATION: MeasurementValue<Double>
+INCLINATION: MeasurementValue<Double>
+MOMENT: MeasurementValue<Double>
+REMANENCE: MeasurementValue<Double>
+RELATIVE_REMANENCE: MeasurementValue<Double>
+THETA63: MeasurementValue<Double>
-caption: String
-unit: String
-description: String
-<<constructor>> (caption:String,unit:String,
description:String)
getValue(step:MeasurementStep): T
+getCaption(): String
+getUnit(): String
+getDescription(): String

Figure 1: Squid class diagram: central classes



<<details in other diagram>>
Settings
+instance(): Settings
SettingsPanel
Ikayaki
+main(args:String[])
Squid
+instance(): Squid
+getDegausser(): Degausser
+getHandler(): Handler
+getMagnetometer(): Magnetometer
1
1
0..1
owner
0..1
<<details in other diagram>>
Project
project
0..1
MainMenuBar
MainStatusBar
MainViewPanel
+loadProject(file:File)
ProjectListener
+projectUpdated(event:ProjectEvent)
MeasurementListener
+measurementUpdated(event:MeasurementEvent)
CalibrationPanel
ProjectExplorerPanel
ProjectComponent
-project: Project
+getProject(): Project
+setProject(project:Project)
0..*
ProjectExplorerTableModel
ProjectExplorerPopupMenu
ProjectInformationPanel
MeasurementSequencePanel
MeasurementDetailsPanel
MeasurementControlsPanel
MeasurementGraphsPanel
SequenceTableModel
SequencePopupMenu

5.1.2 Project.Type

Package	ikayaki
Declaration	public enum Type

The type of the project. Options are CALIBRATION, AF, THELLIER and THERMAL.

5.1.3 Project.State

Package	ikayaki
Declaration	public enum State

The state of the project's measurements. Options are IDLE, MEASURING, PAUSED, ABORTED.

5.1.4 MeasurementSequence

Package	ikayaki
Declaration	public class MeasurementSequence
Created by	Project
Uses 1	MeasurementStep (5.1.5)

A list of measurement steps. Steps can be added or removed from the sequence.

5.1.5 MeasurementStep

Package	ikayaki
Declaration	public class MeasurementStep
Created by	Project, MeasurementSequencePanel
Uses 1	Project (5.1.1)
Uses 2	MeasurementResult (5.1.7)

A single step in a measurement sequence. Each step can include multiple measurements for improved measurement precision. A step can have a different volume and mass than the related project, but by default the volume and mass of the project will be used. Only the project may change the state and results of a measurement step.

5.1.6 MeasurementStep.State

Package	ikayaki
Declaration	public enum State

The state of a measurement step. Options are READY, MEASURING, DONE_RECENTLY and DONE.

5.1.7 MeasurementResult

Package	ikayaki
Declaration	public class MeasurementResult
Created by	Magnetometer

A set of X, Y and Z values measured by the magnetometer. The raw XYZ values will be rotated in 3D space by using a transformation matrix. The project will set and update the transformation whenever its parameters are changed.

5.1.8 MeasurementResult.Type

Package	ikayaki
Declaration	public enum Type

The orientation of the sample when it was measured. Options are BG, DEG0, DEG90, DEG180 and DEG270.

```
public String getName()
```

Returns	"BG", "0", "90", "180" or "270"

```
public Tuple3d rotate(Tuple3d from, Tuple3d to)
```

Rotates the raw XYZ values from the orientation of this object to that of DEG0. Rotating a BG or DEG0 will just copy the values directly.

Parameter 1	*from* - Old values that need to be rotated
Parameter 2	*to* - Where the new values will be saved
Returns	The same as parameter to, or a new object if to was null.

```
public Tuple3d rotate(Tuple3d from)
```

Rotates the raw XYZ values from the orientation of this object to that of DEG0. Rotating a BG or DEG0 will just copy the values directly.

Parameter 1	*from* - Old values that need to be rotated
Returns	A new object with the rotated values.

5.1.9 MeasurementValue

Package	ikayaki
Declaration	public class MeasurementValue<T>
Uses 1	MeasurementStep (5.1.5)

Algorithms for calculating values from the measurements. A MeasurementValue object will be passed to the getValue method of a project to retrieve the desired value.

Design patterns	Strategy

5.1.10 ProjectEvent

Package	ikayaki
Declaration	public class ProjectEvent
Extends	EventObject
Created by	Project

ProjectEvent is used to notify others about the state change of a project.

5.1.11 ProjectListener

Package	ikayaki
Declaration	public interface ProjectListener
Extends	EventListener

Defines a listener for project events.

5.1.12 MeasurementEvent

Package	ikayaki
Declaration	public class MeasurementEvent
Extends	EventObject
Created by	Project

MeasurementEvent is used to notify listeners about the stages of an ongoing measurement.

5.1.13 MeasurementListener

Package	ikayaki
Declaration	public interface MeasurementListener
Extends	EventListener

Defines a listener for measurement events.

5.2 Squid interface

5.2.1 Squid

Package	ikayaki.squid
Declaration	
Extends	
Implements	
Created by	
Uses 1	(??)
Uses 2	(??)
Subclass 1	(??)
Subclass 2	(??)

Design patterns	
Event A	-
Event B	-

5.2.2 Degausser

Package	ikayaki.squid
Declaration	
Extends	
Implements	
Created by	
Uses 1	(??)
Uses 2	(??)
Subclass 1	(??)
Subclass 2	(??)

Design patterns	
Event A	-
Event B	-

5.2.3 Handler

Package	ikayaki.squid
Declaration	
Extends	
Implements	
Created by	
Uses 1	(??)
Uses 2	(??)
Subclass 1	(??)
Subclass 2	(??)

Design patterns	
Event A	-
Event B	-

5.2.4 Magnetometer

Package	ikayaki.squid
Declaration	
Extends	
Implements	
Created by	
Uses 1	(??)
Uses 2	(??)
Subclass 1	(??)
Subclass 2	(??)

Design patterns	
Event A	-
Event B	-

5.3 Squid emulator

5.3.1 SquidEmulator

Package	ikayaki.squid
Declaration	public class SquidEmulation
Extends	Thread
Created by	
Uses 1	SerialIO (**??**)

This class tries to emulate behavior of real squid-system. It generates random data values as results and generates random error situations to see that program using real squid system does survive those. Uses 2-3 COM ports. Usage SquidEmulator x z.. where x is 0 or 1 and indicates if Magnetometer and Demagnetizer are on same COM port. z... values are COM ports. Commands are at most five characters in length including a carriage return <CR>. The syntax is as follows: "<device><command><subcommand><data><CR>"

Event A	*On New IO Message* - reads message and puts it in Buffer

`private Stack messageBuffer`
buffer for incoming messages

`private bool online`
indicates if system have been started

`private int acceleration`
value between 0 and 127 default 5. Settings in the 20-50 range are usually employed.

`private int deceleration`
value between 0 and 127 default 10. Settings in the 20-50 range are usually employed.

`private int velocity`
value between 50 and 12 000. The decimal number issued is 10 times the actual pulse rate to the motor. Since the motor requires 200 pulses (full step) or 400 pulses (half step) per revolution, a speed setting of M10000 sets the motor to revolve at 5 revolutions per second in full step or 2.5 revolutions in half step. This rate is one-half the sample rate rotation due to the pulley ratios. The sample handler is set up at the factory for half stepping.

`private String handlerStatus`
5 end of move, previous G command complete, 7 hard limit stop, G motor is currently indexing

`private int commandedDistance`
value between 1 and 16,777,215

`private int currentPosition`
value between 1 and 16,777,215

`private int homePosition`
value between 1 and 16,777,215

`private int commandedRotation`
angles are between 0 (0) and 2000 (360)

`private int currentRotation`
angles are between 0 (0) and 2000 (360)

`private int degausserCoil`
(X, Y, Z) = (0,1,2) default axis Z

`private int degausserAmplitude`
0->3000 default amp 0

`private int degausserDelay`
1-9 seconds default delay 1 second

`private int degausserRamp`
(3, 5, 7, 9) default 3

`private char degausserRamp`
Z=Zero, T=Tracking, ?=Unknown

`private SerialIO[] messageReader`
starts Threads which reads messages from selected COM port

`public void getSequences()`
reads message and commits it.

`public void writeMessage(String message,int port))`
send message to SerialIO to be sented.

`public void run())`
runs sequence where read data from buffer and run cheduled actions (move, rotate, demag, measure) and send feedback to COM ports.

5.4 Serial communication

5.5 Utilities

5.5.1 RunQueue

Package	ikayaki.util
Declaration	public class RunQueue
Uses 1	RunQueue.RunQueueThread (5.5.2)
Uses 2	RunQueue.RunDelayed (5.5.3)

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.

`public 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

5.5.2 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()`

5.5.3 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.

6 GUI classes and methods

6.1 Generic GUI components

6.1.1 ProjectComponent

Package	ikayaki.gui
Declaration	public class ProjectComponent
Extends	JPanel
Created by	MainViewPanel
Uses 1	Project (5.1.1)
Subclass 1	ProjectInformationPanel (**??**)
Subclass 2	MeasurementSequencePanel (**??**)
Subclass 3	MeasurementDetailsPanel (**??**)
Subclass 4	MeasurementControlsPanel (6.9.1)
Subclass 5	MeasurementGraphsPanel (**??**)
Subclass 6	ProjectExplorerPanel (6.4.1)
Subclass 7	CalibrationPanel (6.5.1)

Generic gui component which uses Project and listens MeasurementEvents and ProjectEvents.

`public ProjectComponent(Project project)`

Registers MeasurementListener and ProjectListener to project.

6.2 Main view and menu

6.2.1 MainViewPanel

Package	ikayaki.gui
Declaration	public class MainViewPanel
Extends	JPanel
Created by	Ikayaki
Uses 1	ProjectExplorerPanel (6.4.1)
Uses 2	CalibrationPanel (6.5.1)
Uses 3	Squid (5.2.1)
Uses 4	MainMenuBar (6.2.3)
Uses 5	MainStatusBar (6.10.2)
Uses 6	ProjectInformationPanel (**??**)
Uses 7	MeasurementSequencePanel (**??**)
Uses 8	MeasurementDetailsPanel (**??**)
Uses 9	MeasurementControlsPanel (6.9.1)
Uses 10	MeasurementGraphsPanel (**??**)

Creates the main view panels (split panels) and Squid and Project components. It also tells everybody if project is changed.

`private ProjectExplorerPanel projectExplorer`

`private CalibrationPanel calibration`

`private Squid squid`

`private ProjectComponent project`
currently active project

`private MainMenuBar menuBar`

`private MainStatusBar statusBar`

`private ProjectInformationPanel projectInformation`

`private MeasurementSequencePanel measurementSequence`

`private MeasurementControlsPanel measurementControls`

`private MeasurementDetailsPanel measurementDetails` `private MeasurementGraphsPanel measurementGraphs`

`public MainViewPanel()`

Loads default view and creates all components and panels. Splitpanel between Calibration,Explorer,Information and rest.

`public bool changeProject(Project project)`

Looks for file with filename, if not exist creates new other wise opens it. Then updates current project and tells Panels new project is opened.

6.2.2 MainViewPanel

Package	ikayaki.gui
Declaration	public class MainViewPanel
Extends	JPanel
Created by	Ikayaki
Uses 1	ProjectExplorerPanel (6.4.1)
Uses 2	CalibrationPanel (6.5.1)
Uses 3	Squid (5.2.1)
Uses 4	MainMenuBar (6.2.3)
Uses 5	MainStatusBar (6.10.2)
Uses 6	ProjectInformationPanel (**??**)
Uses 7	MeasurementSequencePanel (**??**)
Uses 8	MeasurementDetailsPanel (**??**)
Uses 9	MeasurementControlsPanel (6.9.1)
Uses 10	MeasurementGraphsPanel (**??**)

Creates the main view panels (split panels) and Squid and Project components. It also tells everybody if project file is changed.

`private ProjectExplorerPanel projectExplorer`

`private CalibrationPanel calibration`

`private Squid squid`

`private ProjectComponent project`

currently active project

`private MainMenuBar menuBar`

`private MainStatusBar statusBar`

`private ProjectInformationPanel projectInformation`

`private MeasurementSequencePanel measurementSequence`

`private MeasurementControlsPanel measurementControls`

`private MeasurementDetailsPanel measurementDetails`

`private MeasurementGraphsPanel measurementGraphs`

`public MainViewPanel()`

Loads default view and creates all components and panels. Splitpanel between Calibration,Explorer,Information and rest.

`public bool changeProject(String filename)`

Looks for file with filename, if not exist creates new other wise opens it. Then updates current project and tells Panels new project is opened.

6.2.3 MainMenuBar

Package	ikayaki.gui
Declaration	public class MainMenuBar
Extends	JMenuBar
Created by	MainViewPanel

Creates Menu items for Menubar and makes action listeners for them

Event A	*On newProject Clicked* - Opens File chooser and opens new file in selected folder
Event B	*On openProject Clicked* - Opens File chooser and opens selected file
Event C	*On exportToDAT Clicked* - Opens File chooser and tells Project to export in selected file
Event D	*On exportToDTDT Clicked* - Opens File chooser and tells Project to export in selected file
Event E	*On exportToSRM Clicked* - Opens File chooser and tells Project to export in selected file
Event F	*On configuration Clicked* - Opens SettingsPanel (frame)
Event G	*On helpItem Clicked* - Opens Help dialog (own frame?)
Event H	*On about Clicked* - Opens dialog with credits and version number
Event I	*On exit Clicked* - closes program

`private JMenu file`

```
private JMenu options

private JMenu help

private JMenuItem newProject

private JMenuItem openProject

private JMenu exportProject

private JMenuItem exportProjectToDAT

private JMenuItem exportProjectToDTD

private JMenuItem exportProjectToSRM

private JMenuItem exit

private JMenuItem configuration

private JMenuItem helpItem

private JMenuItem about

public MainMenuBar()
```

Creates all components and makes menu and sets ActionListeners.

6.3 Configuration window

6.3.1 SettingsPanel

Package	ikayaki.gui
Declaration	public class SettingsPanel
Extends	JFrame
Created by	MainStatusBar
Uses 1	Settings (**??**)

Creates its components and updats changes to Settings and saves them in Configuration file

Event A	*On Save Clicked* - saves current configuration to Settings-singleton and closes window
Event B	*On Cancel Clicked* - closes window (discarding changes)

`private JComboBox magnetometerPort`
COM port for magnetometer

`private JComboBox demagnetizerPort`
COM port for demagnetizer, can be sharing same port with magnetometer

`private JComboBox handlerPort`
COM port for sample handler

`private JTextField xAxisCalibration`
Calibration constants with polarization (factory set?)

`private JTextField yAxisCalibration`
Calibration constants with polarization (factory set?)

`private JTextField zAxisCalibration`
Calibration constants with polarization (factory set?)

`private JComboBox demagRamp`
how fast demagnetization goes

`private JComboBox demagDelay`
?

`private JTextField acceleration`
Handler acceleration

`private JTextField deceleration`
Handler deceleration

`private JTextField velocity`
Handler Max speed

`private JTextField measurementVelocity`
speed in measurement, should be small

`private JTextField transverseYPosition`

`private JTextField axialPosition`

`private JTextField sampleLoadPosition`

`private JTextField backgroundPosition`

`private JTextField measurementPosition`

`private JTextField rotation`

`private JComboBox handlerRightLimit`

`private JButton saveButton`

`private JButton cancelButton`

`public SettingsPanel()`
Creates all components and puts them in right places. Labels are created only here (no global fields). Creates ActionListeners for buttons.

`public void closeWindow()`
Closes window, no changes saved.

`public void saveSettings()`
Saves all settings to Settings-singleton and calls closeWindow().

6.3.2 SettingsPanel

Package	ikayaki.gui
Declaration	public class SettingsPanel
Extends	JFrame
Created by	MainStatusBar
Uses 1	Settings (**??**)

Creates its components and updats changes to Settings and saves them in Configuration file

Event A	*On Save Clicked* - saves current configuration to Settings-singleton and closes window
Event B	*On Cancel Clicked* - closes window (discarding changes)

`private JComboBox magnetometerPort`
COM port for magnetometer

`private JComboBox demagnetizerPort`
COM port for demagnetizer, can be sharing same port with magnetometer

`private JComboBox handlerPort`
COM port for sample handler

`private JTextField xAxisCalibration`
Calibration constants with polarization (factory set?)

`private JTextField yAxisCalibration`
Calibration constants with polarization (factory set?)

`private JTextField zAxisCalibration`
Calibration constants with polarization (factory set?)

`private JComboBox demagRamp`
how fast demagnetization goes

`private JComboBox demagDelay`
?

`private JTextField acceleration`
Handler acceleration

`private JTextField deceleration`
Handler deceleration

`private JTextField velocity`
Handler Max speed

`private JTextField measurementVelocity`
speed in measurement, should be small

`private JTextField transverseYPosition`

`private JTextField axialPosition`

`private JTextField sampleLoadPosition`

`private JTextField backgroundPosition`

`private JTextField measurementPosition`

`private JTextField rotation`

`private JComboBox handlerRightLimit`

`private JButton saveButton`

`private JButton cancelButton`

`public SettingsPanel()`
Creates all components and puts them in right places. Labels are created only here (no global fields). Creates ActionListeners for buttons.

`public void closeWindow()`
Closes window, no changes saved.

`public void saveSettings()`
Saves all settings to Settings-singleton and calls closeWindow().

6.4 Project Explorer

6.4.1 ProjectExplorerPanel

Package	ikayaki.gui
Declaration	public class ProjectExplorerPanel
Extends	ProjectComponent
Created by	MainViewPanel
Uses 1	MainViewPanel (6.2.2)
Uses 2	ProjectExplorerTableModel (6.4.2)
Uses 3	ProjectExplorerPopupMenu (6.4.3)

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 (createNewProjectButton) 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 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 createNewProjectButton click* - call Project.createXXXProject(File) with filename from newProjectField, send returned Project to MainViewPanel?, tell explorerTable to reset newProjectField and newProjectType.
Event D	*On browseButton click* - open a FileChooser dialog for choosing the directory.
Event E	*On MeasurementEvent* - hilight project whose measuring started, or unhilight one whose measuring ended.
Event F	*On ProjectEvent* - hilight selected project, or unhilight unselected project.

`private BrowserFieldComponent browserField`

(need a separate class for this?)

`private JButton browseButton`

`private ProjectExplorerTableModel explorerTable`

`private JTextField newProject`

`private JComboBox newProjectType`

Default value	AF/Thellier/Thermal

`private JButton createNewProjectButton`

`public ProjectExplorerPanel(Project project)`

Creates all its components, sets the last open project folder as current project folder.

`public void doBrowse()`

Creates a FileChooser dialog and tells explorerTable and browserField to change to directory returned by it.

6.4.2 ProjectExplorerTableModel

Package	ikayaki.gui
Declaration	public class ProjectExplorerTableModel
Extends	AbstractTableModel
Created by	ProjectExplorerPanel
Uses 1	MainViewPanel (6.2.2)

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.

6.4.3 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

6.5 Calibration

6.5.1 CalibrationPanel

Package	ikayaki.gui
Declaration	public class CalibrationPanel
Extends	ProjectComponent
Created by	MainViewPanel
Uses 1	MainViewPanel (6.2.2)
Uses 2	Project (5.1.1)

Holds predefined "Holder noise" and "Standard sample" projects for calibration; they are in a technically same table as Project explorer files. Also has a "Calibrate" button, which executes selected calibration project, similarly to clicking "Single step" in normal projects.

Event A	*On calibrateButton click* - call project.doSingleStep().
Event B	*On calibrationProjectTable click* - call Project.loadProject(File) with clicked project file, send returned Project to MainViewPanel?
Event C	*On MeasurementEvent* - hilight calibration project whose measuring started, or unhilight one whose measuring ended; enable/disable calibrateButton similarly.
Event D	*On ProjectEvent* - hilight selected calibration project, or unhilight unselected calibration project.

```
private JButton calibrateButton
```

```
private JTable calibrationProjectTable
```
Table for the two calibration projects; has "filename", "last modified" and "time" (time since last modification) columns.

6.6 Project information

6.7 Sequence and measurement data

6.8 Measurement details

6.9 Measurement controls

6.9.1 MeasurementControlsPanel

Package	ikayaki.gui
Declaration	public class MeasurementControlsPanel
Extends	ProjectComponent
Created by	MainViewPanel
Uses 1	Project (5.1.1)
Uses 2	MagnetometerStatusPanel (6.9.2)
Uses 3	ManualControlsPanel (6.9.3)

Has "Measure"/"Pause", "Single step" and "Stop now!" buttons for controlling measurements; "+z/-z" radiobuttons for changing sample orientation, help picture for inserting sample, picture of current magnetometer status, and, manual controls. Listens MeasurementEvents and ProjectEvents, and updates buttons and magnetometer status accordingly.

Event A	*On measureButton click* - call project.doAutoStep() or project.doPause(), depending on current button status.
Event B	*On singlestepButton click* - call project.doSingleStep().
Event C	*On stopButton click* - call project.doAbort().
Event D	*On MeasurementEvent* - call magnetometerStatusPanel.updateStatus(int, int).
Event E	*On ProjectEvent* - update buttons and manual controls according to project.isXXXEnabled().

```
private JButton measureButton
```

```
private JButton singlestepButton
```

```
private JButton stopButton
```

```
private JRadioButton zPlusRadioButton
```

```
private JRadioButton zMinusRadioButton
```

```
private JImage?  sampleInsertImage
private MagnetometerStatusPanel magnetometerStatusPanel
```

```
private ManualControlsPanel manualControlsPanel
```

6.9.2 MagnetometerStatusPanel

Package	ikayaki.gui
Declaration	public class MagnetometerStatusPanel
Extends	JPanel
Created by	MeasurementControlsPanel

Picture of current magnetometer status, including sample holder position and rotation.

```
public MagnetometerStatusPanel()
```

Sets magnetometer status to current position.

```
public updateStatus(int position, int rotation)
```

Updates magnetometer status picture; called by MeasurementControlsPanel.

Parameter 1	*position* - sample holder position, from a to b?
Parameter 2	*rotation* - sample holder rotation; 0, 90, 180 or 270.

6.9.3 ManualControlsPanel

Package	ikayaki.gui
Declaration	public class ManualControlsPanel
Extends	JPanel
Created by	MeasurementControlsPanel

Magnetometer manual control radiobuttons.

Uses 1	Project (5.1.1)
Event A	*On xxxN click* - call project.xxxN().

```
private JRadioButton demagX
```

```
private JRadioButton demagY
```

```
private JRadioButton demagZ
```

```
private JRadioButton measureX
```

```
private JRadioButton measureY
private JRadioButton measureZ
```

6.10 Status bar

6.10.1 MainStatusBar

Package	ikayaki.gui
Declaration	public class MainStatusBar
Extends	ProjectComponent
Created by	MainViewPanel

Creates its components and listens project events on status change and calculates estimated time for measurement.

Event A	*On Measurement Event* - recalculates progress and updates status for current measurement

```
private JLabel measurementStatus
```
text comment of current status(moving,measurement,demagnetization)

```
private JProgressBar measurementProgress
```
progress of sequence/measurement as per cent of whole process

```
private int[] currentSequence
```
current projects sequence

```
private int projectType
```
current projects type (we know if we are doing demagnetization or not)

```
public MainStatusBar()
```
Creates all components with default settings and sets Listener for MeasurementEvent.

```
          private void calculateStatus(String phase, int
sequenceStep, int currentStep)
```
Recalculates current progress and updates status.

```
      private void setMeasurement(int projectType, int[]
sequence)
```
Formats status and creates new measurement status values.

6.10.2 MainStatusBar

Package	ikayaki.gui
Declaration	public class MainStatusBar
Extends	ProjectComponent
Created by	MainViewPanel

Creates its components and listens project events on status change and calculates estimated time for measurement

Event A	*On Measurement Event* - recalculates progress and updates status for current measurement

```
private JLabel measurementStatus
```
text comment of current status(moving,measurement,demagnetization)

```
private JProgressBar measurementProgress
```
progress of sequence/measurement as per cent of whole process

```
private int[] currentSequence
```
current projects sequence

```
private int projectType
```
current projects type (we know if we are doing demagnetization or not)

```
public MainStatusBar()
```
Creates all components with default settings and sets Listener for MeasurementEvent.

```
          private void calculateStatus(String phase,int
sequenceStep,int currentStep)
```
Recalculates current progress and updates status.

```
      private void setMeasurement(int projectType,int[]
sequence)
```
Formats status and creates new measurement status values.

6.11 Graphs

7 Package structure

8 Bibliography