

Improve the TIM system graphically and functionally



The European Organization for Nuclear Research

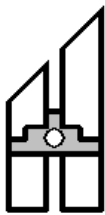
Bachelor project at Bergen University College,
Faculty of Engineering, Department of Computing,
Specialization: System and application development

2011

Author

VEGARD DEILA





BERGEN UNIVERSITY COLLEGE

Faculty of Engineering
Department of Computing

BACHELOR PROJECT TITLE PAGE

<i>Report title:</i> Improve the TIM system graphically and functionally	<i>Date:</i> 15.12.2011
	<i>Report number:</i> 21
<i>Author(s):</i> Vegard Deila	<i>Number of pages:</i> 46
	<i>Number of appendix pages:</i> 10
<i>Specialization:</i> System and application development	<i>Number of CDs/DVDs:</i> 1
<i>Bergen University College contact person:</i> Hege Austrheim Erdal	<i>Classification:</i> Open
<i>Remarks:</i>	

<i>External company/contact:</i> CERN / Peter Sollander	<i>External company reference</i>
<i>External contact person:</i> Peter Sollander	<i>Phone:</i> +41 22 767 80 81

Summary:

CERN, the European Organization for Nuclear Research, is one of the world's largest and most respected centers for scientific research. I have now been working at CERN for one year. Most of the time is used to improve a monitoring system used to monitor the technical infrastructure at CERN. This report will introduce the reader to the monitoring system and its usage. And most of all, it will focus on one of the projects I have been working on. This will include the problem itself, the initial planning, the requirements, problem analysis, the solution and how it was implemented and tested. At the end more of the management and planning for the project will be presented.

Keywords:

CERN	Technical Infrastructure Monitoring	Combined View
------	-------------------------------------	---------------

Høgskolen i Bergen, Avdeling for ingeniørutdanning

Postal Address: Postboks 7030, 5020 BERGEN

Phone 55 58 75 00

Fax 55 58 77 90

Visitor Address: Nygårdsgaten 112, Bergen

E-mail: post@hib.no

URL: <http://www.hib.no>

Preface

CERN, the European Organization for Nuclear Research, is one of the world's largest and most respected centers for scientific research. I sent my application for a technical student contract in August 2010. The contract was accepted October 2010 and I started 1.1.2011. My contract ends in the end of January 2012.

It was Peter Sollander, the leader of the TI (Technical Infrastructure) section, who gave me a position. Hege Erdal is my contact person from the Bergen University College, the university where I am taking my computer engineer degree.

I have been working at CERN on a monitoring system that monitors the technical infrastructure. In this document you will get a light introduction to CERN, and a heavier introduction to the monitoring system I have been working on for the last year. Most of this report will be about one of the tasks that I have completed at CERN.

My colleagues in the TIM team have been very friendly, and have been most helpful. Matthias Braeger is the project leader of TIM, and have revised my code and done complex testing on my implementations.

The project I will focus on is based on a request from Christ Wetton. He is an operator who has been most helpful in specifying the requirements and has provided lots of information that have helped me to understand what is requested.

By working at CERN I have gained much experience; the *Java* language, how to create graphical user interfaces, and systems for managing projects, applications and tasks.

Table of contents

1. Introduction	6
1.1 Project employer	6
1.1.1 The LHC	6
1.2 Problem description	7
1.2.1 The TIM system	7
1.2.2 The TIM system architecture	10
1.4 Solution idea	11
1.5 Project Method	11
1.6 Evaluation of tools and programming language	12
1.6.1 Software	12
1.6.2 Hardware	13
2. Task Definition	14
2.1 The Combined View	14
2.1.1 Toolbars	14
2.1.2 Links	15
2.1.3 Sketch	15
2.1.4 Requirements	15
3. Problem Analysis	17
3.1 Design solutions	17
3.1.1 Types of views in a Combined View	17
3.1.2 Creating a <i>Combined View</i>	18
3.1.3 The user interaction for creating a Combined View	18
3.1.4 Toolbars	20
3.1.5 Adding links	22
3.1.6 Changes from the initial requirements	22
4. Design and solution	23
4.1 Description	23
4.2 In pictures	24
4.2.1 Edit mode	24
4.2.2 The view in use	25
4.2.3 Generated XML	26
5. Implementation	27
5.1 Introduction to the classes	27

5.1.2 Overview	27
5.1.1 Descriptions.....	28
5.2 Methods	29
5.2.1 The graphical user interface and the interactions.....	29
5.2.2 Input and output.....	29
5.3 Changes to the TIM system	30
6. Testing	32
6.1 Testing for the <i>Combined View</i> project.....	32
6.1.1 Testing of the model	32
6.1.2 Testing of the graphical user interface (GUI)	32
6.1.3 Showing it to an operator	32
6.2 Testing in general.....	32
7. Project management.....	33
7.1 My role at CERN	33
8. Summary, discussion, conclusions	34
8.1 Changes and limitations	34
8.2 Progress.....	34
8.3 Risks	34
8.4 Further development of the <i>Combined Views</i>	34
9. References and training	36
9.1 References.....	36
9.2 Training.....	36
Appendices.....	37
Appendix A: Dictionary	37
Appendix B: Progress plan	38
The initial progress plan.....	38
The final Gantt diagram	40
Appendix C: Risk list.....	42
Appendix D: User manual	43
How to create a new <i>Combined View</i>	43
How to use a Combined View	44
Appendix E: XML output	45

1. Introduction

1.1 Project employer

CERN, the European Organization for Nuclear Research, is one of the world's largest and most respected centers for scientific research. Their business is fundamental physics; to find out what the universe is made of and how it works. CERN was established in 1954 and is situated on the Swiss-French border near Geneva.

CERN employs about 2400 people who designs and builds the particle accelerator and ensures its operation. They also have about 10.000 visiting scientists from 608 universities and 113 nationalities that use CERN's facilities.

As this is an international organization, the funding comes mainly from 20 countries that are *member states*. In 2011 they contributed with about 1,1 billion CHF (About 6,8 billion NOK).^[10]

In addition to new advancements within physics, a lot of spin-offs have also seen the day of light during experiments at CERN, one of them is the very famous World-Wide Web, which began as a CERN project in 1989.

1.1.1 The LHC

The Large Hadron Collider (LHC) is a 27km long accelerator lying 175 meters under the surface. Here they are colliding particles at near the speed of light and with this physicists aim at getting a more fundamental understanding of the building blocks of nature.

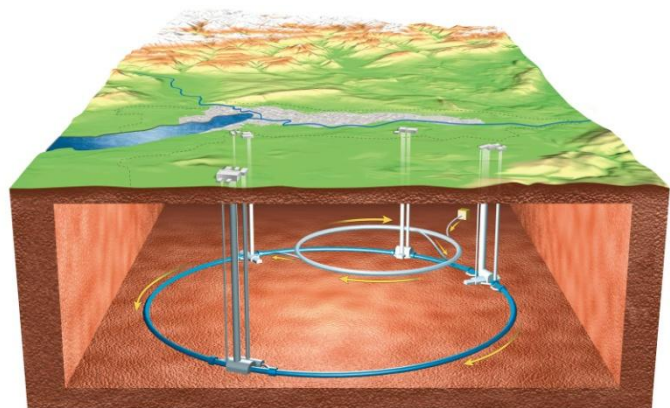


Figure 1 - The LHC

1.2 Problem description

There are a great number of things that can go wrong at CERN. To minimize this number they have many different kinds of operators who keep an eye on everything that is going on at CERN. I am in the *operation* group in the *technical infrastructure* section whose mission is to minimize the impact of technical breakdowns. To keep track of the whole technical infrastructure, the operators use several monitoring software systems. One of these systems is called the *Technical Infrastructure Monitoring system* (TIM), which is created and is being maintained by the SSE (*Safety Systems Engineering*) section. I am sent to do developing for this section to give more control to my section of what they should improve in this system.

1.2.1 The TIM system

The TIM system is mostly used by the TI operators at the CCC, but is also used by others, like in the control room for some of the experiments. The number of people using it is expanding. TIM is a system that gathers data, states and measurements from all over CERN; energy consumption, heating, cooling, ventilation, air conditioning, cryogenics, alarms and much more. The TIM Viewer is not only a viewer, because if you have the access rights, you can give commands to equipments, like for example switch the lights on or off in the tunnels. The TIM system consists of several applications. Some of the applications are here presented to give a clearer picture of the TIM system:

TIM Viewer

The Tim Viewer is the main application, see Figure 2. The menu on the left shows folders, and the *views* that can be opened. One *view* can contain measurements and/or commands. The *view* has often animations dependent on some measurement value, for example how much water it is in the tunnel. Colors are often utilized so that the operators easily can see if a measurement is not as expected. If you are logged in and have the privileged, you can control the lights in the tunnel by opening the correct *view*, and by clicking on the commands to switch the lights on or off.

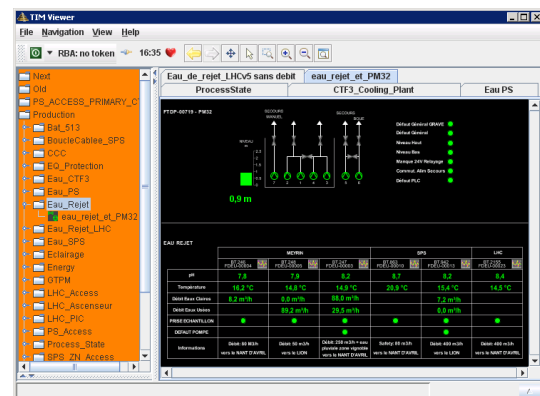


Figure 2 - TIM Viewer

With a few mouse clicks you can take one or several measurements from a *view* and put them into a chart for a given period of time. See Figure 3. The green curve shows how much water it is in the tunnel, the red curve shows when the pump is pumping out water from the tunnel. You can then see how the water decreases when the pump is on, and is pumping the water out of the tunnel.

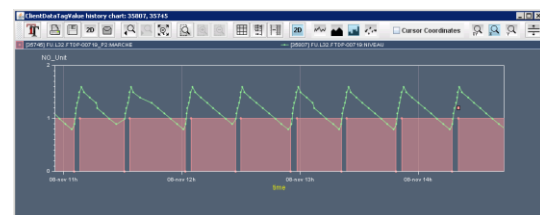


Figure 3 - A chart in TIM Viewer

TIM Web Viewer

The TIM Web Viewer is a limited version of the TIM Viewer. You can see most of the *views*, but you cannot send commands and you cannot create chartings.

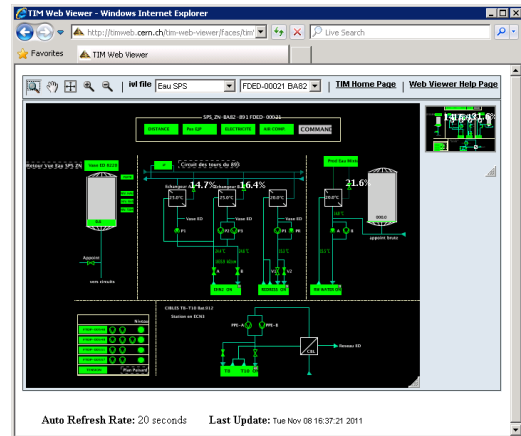


Figure 4 - TIM Web Viewer

TIM Video Viewer

The TIM Video Viewer can get live video streams from cameras that are set up by all entrances to the LHC. This is used by the operators to give access to the tunnel.



Figure 5 - TIM Video Viewer

TIM GTPM Editor

Figure 6 shows a *GTPM view*. This is a *view* that shows how different equipment depends on each other. When a *node* is red, that equipment or at least one of its children is having an error.

The TIM GTPM Editor is used to create such *views*.

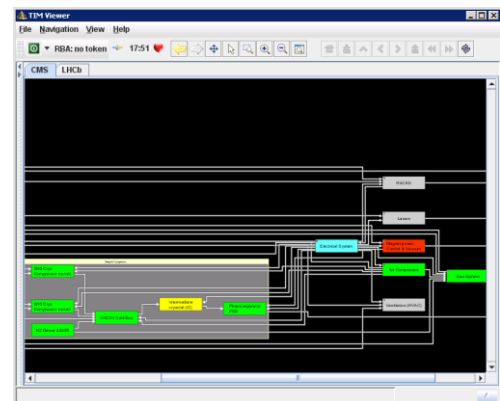


Figure 6 - A GTPM View

Symbol Editor

The symbol editor is made by IBM. With it you can create symbols that later can be used when creating *views* in the TIM Viewer. A symbol can for example be a drawing of a tank of liquid that is filling up as some measurement is increasing in value. See on the left side of Figure 4 for such a tank.

TIM Dashboard Editor

With the TIM Dashboard Editor it is possible to put together many symbols to create a new *view*. The symbols can either be custom made with the Symbol Editor, or from a set of predefined symbols. Then all the symbols are linked to different measurement points, as for example how much water it is in a tank, or the temperature in a server. When the *view* is saved, it can be opened directly from the TIM Viewer.

The intention of having all these tools is that the operators can create the *views* that they want themselves, thus minimizing the maintenance needs for the developers, and the developers can continue to improve the system instead of using all the time on creating *views* for the operators. See Figure 7 to get a simple overview of how an operator can create *views*. As there are several types of *views*, they have several tools for creating them. Each type of view has some specific strengths and weaknesses. The *GTPM view* in Figure 6 is the type of *view* that is the easiest to distinguish from other types of *views*.

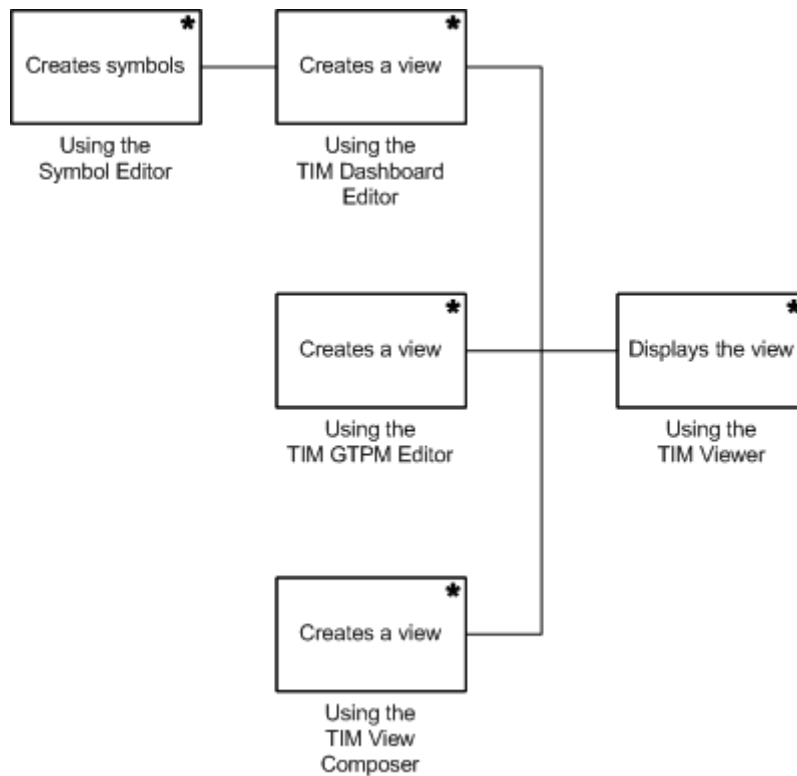


Figure 7 - What the operators can do to create a new view

1.2.2 The TIM system architecture

All the TIM applications are made with Java and use the *Java Message Service (JMS)* with *ActiveMQ* for communication between the server and clients. A client can subscribe to one or several *tags* on the server. A *tag* has a value indicating a measurement, or a set of measurements. So in the end, all *tags* are connected to some equipment, or a set of equipments. When a new measurement value arrives from some equipment, it is sent to all the clients that subscribe to this *tag*. See Figure 8 below to get an overview of how the parts communicate with each other.

The *DAQs* at the bottom of the figure is the generic layer that gets status updates from the equipments and communicates it to the server through the Java Messaging Service.

C²MON means *CERN Control and Monitoring Platform*, and is developed by the same people who developed the TIM system. *C²MON* is a generic platform to be reused in other monitoring scenarios.

The *Short Term Archive* is a database with the records of all the updates from all equipments the last 30 days and can be accessed through the server to create charts.

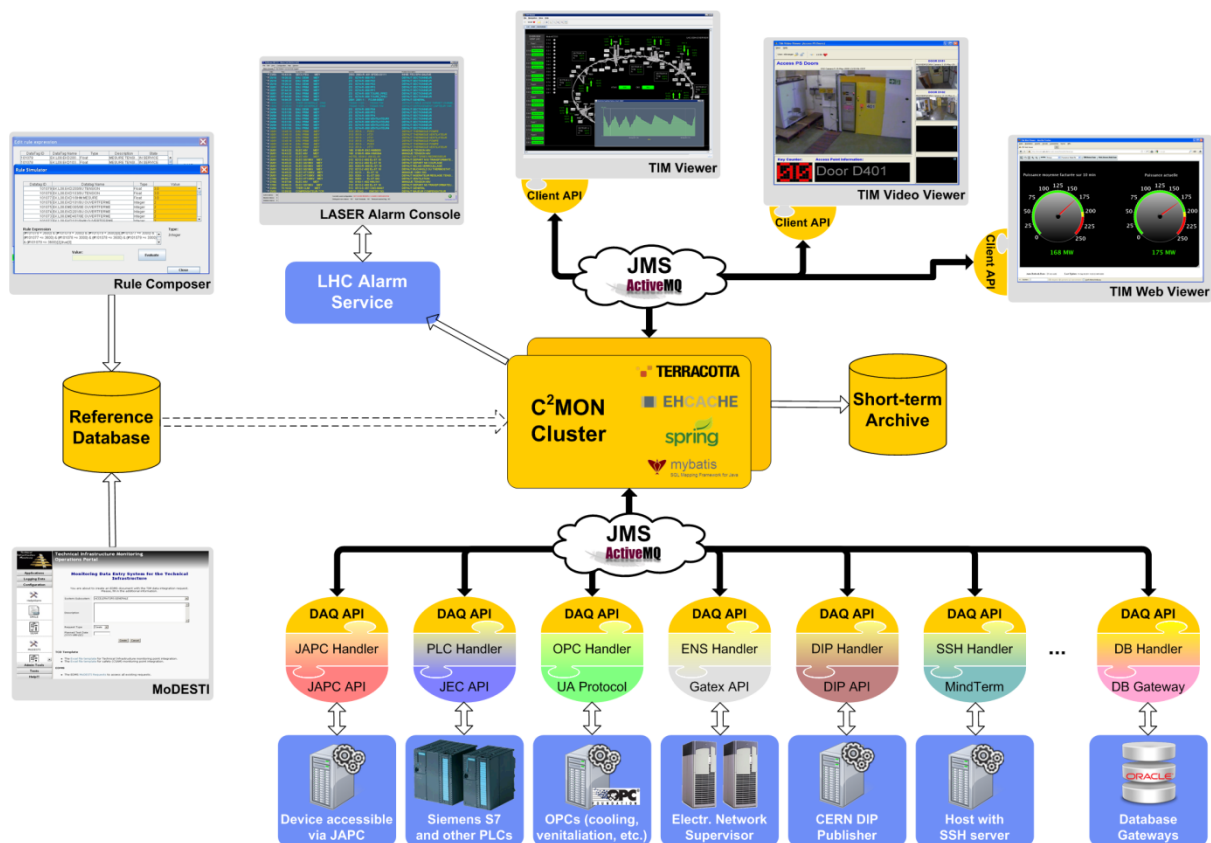


Figure 8 - The Technical Infrastructure Monitoring system architecture

1.4 Solution idea

The operators at the CCC (CERN Control Centre) want an efficient, functional and reliable system to efficiently show and analyze data they find useful, and to easily give commands to equipments. This system has already existed since July 2005, but there are many improvements to be made. The improvement of this existing system is the main task. To make the system easier to use and more functional they want to have bugs fixed, new features, and minor and major improvements on the system, mainly to the client applications.

One of the goals that need to be achieved is to create an intuitive and easy method for the operators at the CCC to choose what they want to see on the monitors. This should be achieved through communication with an operator and through agreement with Peter, who is my supervisor, Matthias, who is the TIM project leader, and the operator that is requesting the feature.

1.5 Project Method

Agile software development ^[3] is a group of software development methodologies based on iterative and incremental development. The Agile Manifesto ^[13] reads as follows:

«We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more. »

The TIM team does not use a specific software development process. But the way we work is very much like *Kanban* ^{[1][2]}, which is an *Agile software development* process. This process fits very well to the working environment I am currently in. I think this method is nice because it does not focus too much on bureaucracy, but rather on getting things done, and on rapid change of plans. This means that you do not need to create redundant diagrams and schemas, but rather what you think is useful to perform the task. It also gives room for having more than one ongoing task at once.

1.6 Evaluation of tools and programming language

CERN uses a system called *JIRA* from *Atlassian* to track issues and projects. Before fixing a bug currently in production, a *JIRA issue* providing information about the bug needs to be created. A *JIRA issue* must include the severity and in what version of the application it will be fixed, the estimated time consumption, and other parameters. When a code change is committed into SVN, a *JIRA issue*-number must be provided to connect the code changes to this issue. If doing a bigger task, like implementing a new feature, sub-tasks can be created for that *JIRA issue*.

This system gives a good overview of the tasks, and is very good when working in teams, as it is easier to see what tasks that need to be done.

When doing a small task, like a simple bug fix, an implementation plan is not required. But when doing a larger task it is recommended to have specifications and requirements document that specifies what is wanted and needed, with a suggestion on how to solve the task. This document should be sent to the people involved. When the requirements and specifications are agreed upon, an implementation plan can be created. Matthias, who is the project leader of the TIM System, should revise the implementation plan.

1.6.1 Software

The whole TIM system is written in *Java*. So the *Java* documentation ^[14] will be used frequently. The TIM Viewer and TIM Video Viewer use Java Web Start, which makes the applications very easy to maintain. With Java Web Start the application can be opened from an Internet page. When the application is starting, all the necessary jars / libraries are downloaded automatically from the Internet. If a new version of the program is released the user simply needs to restart the application to get the latest version. A newly released version will first be in a beta testing phase. When enough users have tested it over a period of time, it is released into *production*, which is the version that everyone runs by default.

Development

Remote desktop is used when developing. It includes a network storage system, which creates backups of all your files for the last 30 days. This is useful if files are deleted, or changed, by mistake. It can also be used just to compare what changes have been made. Eclipse is used for the development, together with SVN. This makes it easier to work on the same projects. I have prior knowledge using Eclipse, and for me to choose another tool for programming would be very impractical, as everything (Deployment using *Ants* and *Maven*, *JIRA*, *SVN*, etc.) is included in the Eclipse installed on the remote desktop. For Eclipse, a plug-in called *CheckStyle* is used. This is a tool that gives warnings if the code does not meet certain requirements. Some examples of what it will give warnings about:

- If java documentation comments is missing on methods, their parameters, or on the classes
- Certain coding, like inline if-statements are not allowed
- If the naming convention is not followed
- Magic numbers (numbers used in the code without being declared to a static final variable)
- Functions or classes with too many lines.

It is required to follow the guidelines from *CheckStyle* in most cases. It is enforced to make all the developers follow a set of coding standards, which in return gives a more readable code.

1.6.2 Hardware

Since everything is made in Java, the TIM Viewer can theoretically be run on any hardware where Java is supported. But it is only tested on computers running Windows, Linux and partly Os X (Mac). On a device with a small display, it will probably not work too well. In addition to this, some libraries used by the TIM Viewer are not supported on all devices.

Parts of TIM can be viewed using the TIM Web Viewer, which is supported on all devices with *Internet access*.

2. Task Definition

There are a lot of small tasks, like bug fixes, or small tiny improvements to the TIM system. I will present only one of the bigger tasks. In this section the requirements will be defined. I will also explain what is wanted, and what TIM currently offers of similar functionality.

2.1 The Combined View

The TIM Viewer has something called an *overview*, see Figure 9 below. This is a view containing several *SDM* views. An *SDM view* is one type of *view*, see Figure 10 below. The *SDM views* inside the *overview* can be clicked at, to zoom in on that *view*. To make changes to one of the *SDM* views, the changes will have to be made to both the *SDM view* and the *overview* containing it. So the exact same change must be made twice. This is of course very inefficient.

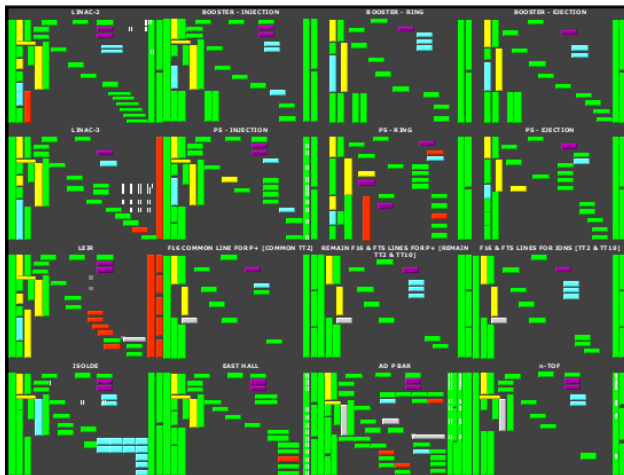


Figure 9 - An Overview in the TIM Viewer

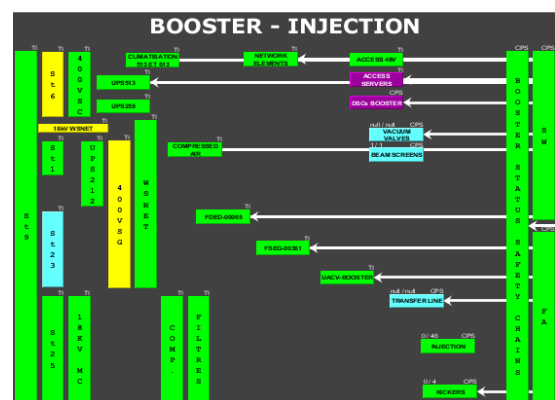


Figure 10 - A SDM view

The task is to create a new type of *view* that can contain several other *views*, but made so that the operators do not need to do the same changes twice when changing a SDM view. It should be made to replace the *overview*, but does not necessary have to have the exact same functionality. The important thing is that the operators can use this view instead of the current *overview*. So the other part of the task is to meet with one of the operators to discuss how they use the current *overview*, and what functionality they think are important to keep, or to add, to the new *combined view*.

There must also be created functionality for creating a *Combined View*. Currently, it is a very complex process to create an *overview*.

2.1.1 Toolbars

If the *Combined Views* will support all kind of *views*, not limited to *SDM Views*, a solution must be provided for how to show the toolbars. The question is how the user can access the toolbar(s) for one of the views inside the *Combined View*. See Figure 11 and Figure 12 below for two different views with their toolbar(s). Figure 11 have two toolbars, one for zooming, and another one for doing special commands on the view.

If choosing to only show SDM views, this is only a minor problem, as a SDM view always has the same toolbar.

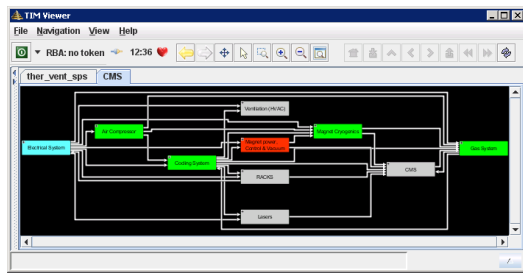


Figure 11 – A GTPM view with its toolbars

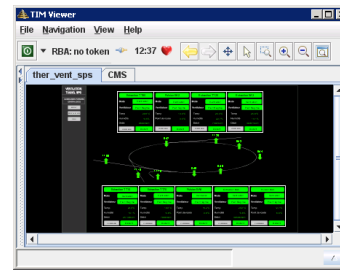


Figure 12 – A dashboard view with its toolbar

2.1.2 Links

After the first release of the *Combined View* functionality, the operator also wanted to have the possibility to add links from one view to another. So that when the link is pressed, it will open the linked view in a new tab.

2.1.3 Sketch

Figure 13 shows a sketch of how a *Combined View* might look like when it is finish.

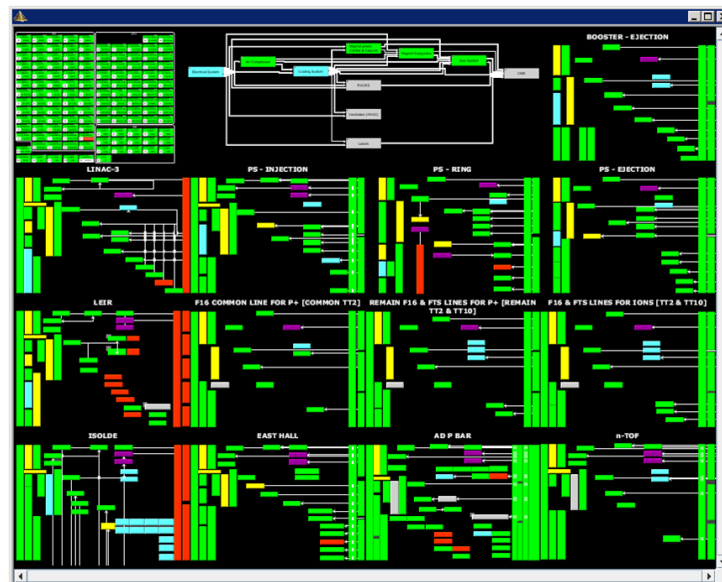


Figure 13 – A sketch of how a Combined View might look like

2.1.4 Requirements

This section will define the requirements that are specified together with the people involved.

High priority

- A replacement for the *overview*.
 - Should be able to contain several other *views*.
- Combined Views must be supported from the TIM Viewer. Look at how other types of *views* are implemented to see how to do this.
- If changes are wanted on a *SDM View*, it must not be necessary to make the changes twice.
 - All included views should only be referenced, not copied into the xml file. This will result in that any changes to a *view* will automatically be reflected in the *combined view*.

- The paths to the included *views* should be relative to the *next/production/old* path that the combined view is opened from.
- Functionality for easily creating a *Combined View*.

Medium priority

- All views have one or more toolbars that must somehow be available.
- All the *views* contained in a *Combined View* should support having a title.
 - Should also support not having a title.
- The toolbar should somehow be available for the views in the *Combined View*.
- The *Combined View* file should be saved in xml format.
- Possibility to add links to the *views* in a Combined View.

Open for discussion (will be discussed in chapter 3.1 Design solutions)

- All kind of views that exist in TIM Viewer must be supported in a combined view, not limited only to SDM views.

3. Problem Analysis

3.1 Design solutions

In this section I will attempt to present different solution ideas to problems. Each problem tries to solve one or multiple requirements. For each problem I will present a few possible solutions, and I will try to evaluate which solution is the best to pick.

3.1.1 Types of views in a Combined View

In the TIM Viewer there are 5 different types of views. On the right you can see three of the types. In this section I will discuss what types of views a Combined View should support.

Only SDM views

If a Combined View supports only SDM views, I would use the ILOG libraries from IBM to create it, because SDM views are created using this library. I am not too familiar with ILOG, and therefore I consider it to be a risk to use it. There might be problems that I cannot predict without more knowledge of the ILOG libraries.

All types of views

The alternative is that a Combined View could support all types of views supported by the TIM Viewer. It would then be very flexible compared to if it would only supports the SDM views, as it could be used to much more.

I would use Swing^[4] to implement it, which is a huge advantage since my colleagues has experience with it, and it is easy to get help from Google with most problems related to Swing.

The drag and drop from the tree menu in the TIM Viewer (see on the left of Figure 2) for adding views to the Combined View is also probably much easier to implement, because the TIM Viewer is created using Swing.

The ILOG libraries currently manage the toolbar for zooming, selecting and so on. Using Swing, and not the ILOG libraries, is therefore a disadvantage, as some work will need to be done for the toolbars.

Discussion of alternative solutions

The final decision is for the *Combined View* to support all the views supported by TIM. Because I think this have more advantages, and also less risk involved.

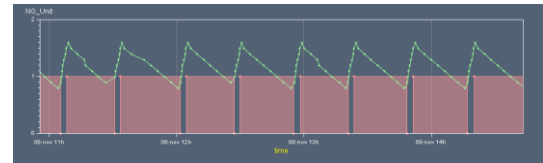


Figure 14 - Trending view

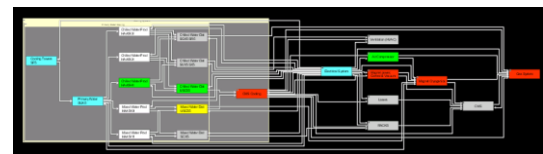


Figure 15 - GTPM view / Partly SDM view

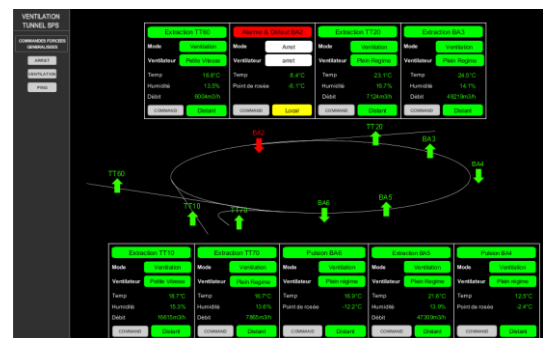


Figure 16 - Dashboard view

3.1.2 Creating a *Combined View*

In this section I will discuss where the user can edit or create a new *Combined View*.

From an editor

A new application called *Combined View Editor* could be made where the user can create and edit *Combined Views*. The process of creating a *Combined View* will then be separated from the TIM Viewer, thus the user does not need to open the TIM Viewer in order to create a *Combined View*. The reason I mention this solution is because it was suggested in the very beginning of the planning of this. But there was not argued with any reason for why it would be better to separate the editor from the Viewer. I cannot see any advantages for this.

Web page

The current *Overview*, which the *Combined View* will replace, can be created using a web page. You specify how many rows and columns, and which views you want to include in the view. A similar web page could be created for creating new *Combined Views*. Then no application needs to be run in order to create the view. The disadvantage is that it is quite some work to make this solution user friendly. It is also more work to maintain a web application than a Java Web Start application. I don't think the operators would be too happy about this solution.

From the TIM Viewer

A new *Combined View* could be created from the *TIM Viewer* by clicking in the menu, as in figure 17. This is very intuitive for the operators, as the operators can view the views without needing to open another application.

By implementing it into the TIM Viewer, a lot of functionality is already there. For example the menu in the TIM Viewer can be used to drag views into a new *Combined View*. This solution is probably easier to implement than any of the previous solutions.

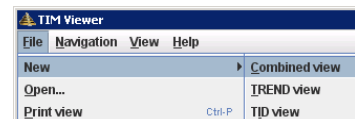


Figure 17 - Illustration of how to create a new *Combined View* in the *TIM Viewer*

Discussion of alternative solutions

Creating *Combined Views* from the TIM Viewer is more user friendly and is easier to implement than both of the other options. This solution will be used.

3.1.3 The user interaction for creating a *Combined View*

This section will discuss some solutions for how a new *Combined View* can be created. See chapter

3.1 for why it is decided that new *Combined Views* should be created from the *TIM Viewer*.

Select rows and columns

When the user chooses to create a new *Combined View*, they must select the number of rows, and the number of columns that the *Combined View* should have. Then the user can select what *views* to have for each cell, from a list containing all the views. A text box can be placed in the top of the cell for the user to set the title. This solution is not pretty, and requires that the user knows exactly how many and which views to include. It must also have other additional fields to enable a view to span over several columns or rows. This can easily become dirty, and adding new features at a later stage might be challenging. I consider this solution to be a hack; a solution that is easy to implement, but impossible to maintain.

The main reason for mentioning this solution is because a similar solution is used for creating the current *Overviews*, the feature that the *Combined Views* will replace.

Drag and drop

This solution will use drag and drop functionality to create a *Combined View*. When the user chooses to create a new *Combined View*, an empty *view* shows up. The user can then drag the *views* from the menu on the left (see Figure 18) into the empty *view*. Here comes a step-by-step description of how the user would create a new *Combined View*. All the steps refers to Figure 19.

Step 1. A *view* is dragged from the menu on the left into the empty *Combined View*. Two new rows and columns appear all around it. (The red box illustrates the *view* that were dragged in)

Step 2. The user drags in the bottom right corner to resize the *view*.

Step 3. Another column appears on the right to always keep two cells on all sides.

Step 4. The user drags a new *view* (aqua color) from the menu on the left into one of the available spaces.

Step 5. A new row is added to keep a margin of two cells.

Step 6. The user resizes the aqua *view*.

Step 7. Another column is added to keep the cell margins.

Step 8. The *view* is saved, and opened in normal mode. The *view* is then shown, without the margins of course.

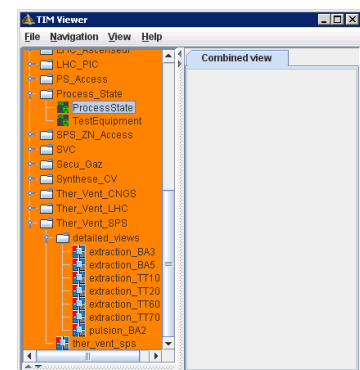
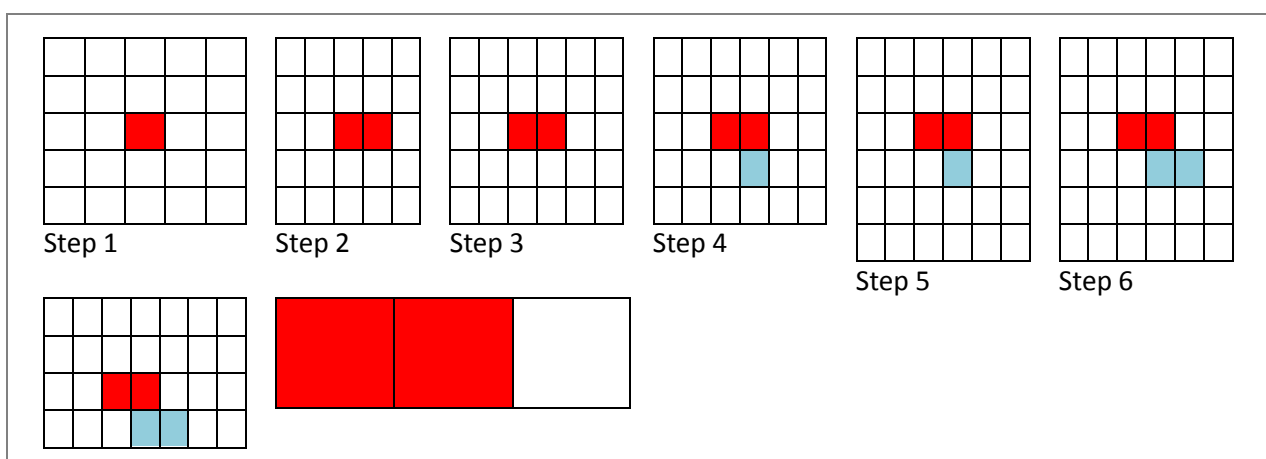


Figure 18 - TIM Viewer with menu and an illustrated empty *Combined View*



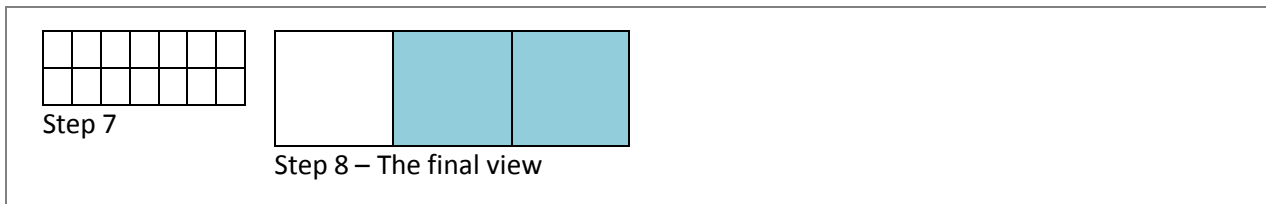


Figure 19 - Step by step how a Combined View can be created

While editing, you can right click one of the added views to get a menu of actions, like adding a title to the *view*, or removing it from the *Combined View*.

This solution is very user friendly and intuitive, and it is easy to add new functionality.

Discussion of alternative solutions

The drag and drop solution takes more time to implement, but have far more advantages than the first solution. It is very intuitive and user friendly. Therefore the drag and drop solution will be used.

3.1.4 Toolbars

This section will discuss what to do with the toolbars. All views have one or more toolbars that must somehow be available to the user. See chapter 2.1.1 for more details about the toolbar problem.

Here comes a few different solutions.

Toolbar(s) in each view

The toolbar(s) can simply be placed in the top or bottom of each *view*, as in the figure below.

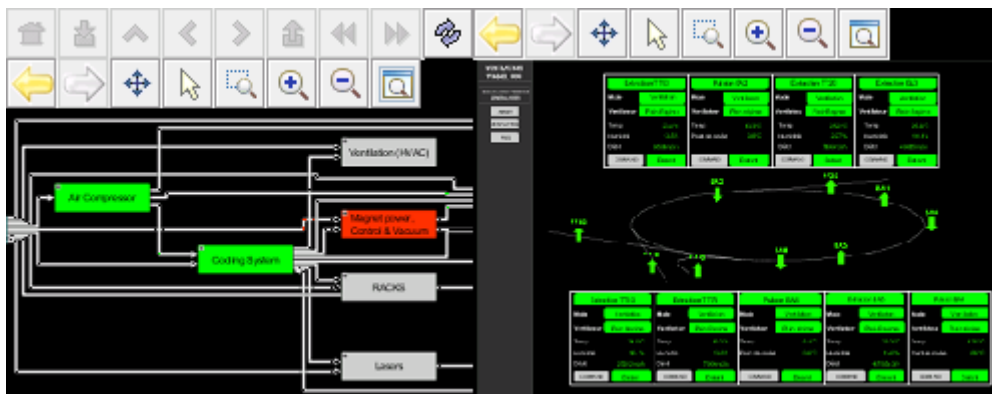


Figure 20 - Illustration of a Combined View with individual toolbars

This is simple to implement, but the toolbars takes up too much space if there are more than four views in the *Combined View*.

Click the view to activate

You will have to click on a view to activate it. The toolbar would then show up for the activated view. This solution was suggested in the beginning. But the thought was then that the view would be zoomed in at, and the interactions would be enabled.

This solution is easy to implement, and is user friendly. A minor disadvantage is that the user cannot interact with a view without first activating it.

Maximize button

A maximize button could be placed in the upper right corner of each view. When it is clicked, the view would be maximized and the toolbar would be activated for that view. When the view is maximized there can be a button in the upper right corner that *restores* the size of the *view* and deactivates the toolbar. This is nice, but has the slight disadvantage that the view has to be maximized for the user to use the toolbar for that view.

Discussion of alternative solutions

The solution of having the mouse click on the view to activate it, and the maximize button in the upper right corner, could be combined. Instead of clicking the whole view, a button can be added next to the maximize button, that can be pressed to activate the toolbar for that view. This gives the possibility for the user to activate the toolbar(s) for a view without maximizing it. And does also show the toolbar(s) when the user maximizes the view. In addition a view does not need to be activated to only interact with it (for example for sending commands).

3.1.5 Adding links

This section will discuss how the user can add links to one of the *views* inside a *Combined View*, and how to activate the link when the user opens the *Combined View* in real mode, not in edit mode.

Discussion of solution

As a maximize button already is in place in the upper right corner of each view, I suggest that while the user is editing a *Combined View*, all of the *views* can have a link icon, also in the upper right corner. The user can drag a *view* from the menu on the left, and drop it on this icon. This can present the user with a dialog for specifying a name for the link. To edit or delete a link, the user can right click the link icon to get a popup with a list of the links. When the user presses the link he can choose to edit the name of it, or delete it.

To activate the links when the view is being opened by the TIM Viewer, the user can right click the link icon to get the list of links that the user can click, or the user can directly left click the link icon to open the first link in the list.

This solution is intuitive, user friendly, and is easy to implement.

Alternative solutions will not be discussed for this matter because I cannot come up with any solution that is near as equally good.

3.1.6 Changes from the initial requirements

In the initial report there was a requirement: *“When showing a combined view, the toolbar is active for the view that was last clicked. The zoom in, zoom out, zoom rectangle, moving around and “fit to screen” toolbar buttons will be used by the combined view, and will not be managed by any of the single views in it. This will not be noticeable by the user compared to how the overviews are today.”*

This is something I came to realize is not a good idea after all, because it makes it impossible / very impractical to do zooming for only one *view* in the *Combined View*. In addition the user would need to click on a view first to be able to interact with it. The operator did not mind having it this way.

This could still be achieved by having an additional zooming toolbar, but it would be confusing having two sets of zooming toolbars. Anyway, if this is wanted at a later time, this is not difficult to implement.

4. Design and solution

In this chapter the solutions for how to design the project will be described and illustrated.

4.1 Description

The *Combined View* is made to be as intuitive as possible for the operators. They can now go to the *file* menu to create a new *Combined View*. They can then simply drag the *views* that they want from the tree menu on the left into the newly created *Combined View*. They are already very familiar with the tree menu on the left as they use it all the time to open *views*. While creating the *Combined View*, they can move and resize the *views* inside it. To make a link from one of the *views* inside the *Combined View* to another *view*, they can use simple drag and drop behavior, dropping a *view* on the link icon in the given *view*. The *view* can be right clicked to change or remove the title. When they are done editing the view, they can save it. A *Combined View* can also be opened for editing at a later time.

When the *Combined View* is opened for viewing, not for editing, all the *views* in it have several buttons in the upper right corner; *Activate toolbar*, *Links* and *Maximize*.

If a *Combined View* is moved from the *Next* folder, into the *Production* folder, the *Combined View* will start using the *Production views* instead of the *Next views*. This is because all the *views* included in the *Combined View* reference relative to the *Old*, *Next* or *Production* folder.

4.2 In pictures

To easily get an understanding of the solution and the design there are below several images that shows different kinds of features when creating a *Combined View*. Then some images of how the *Combined View* can be used when it is opened.

4.2.1 Edit mode

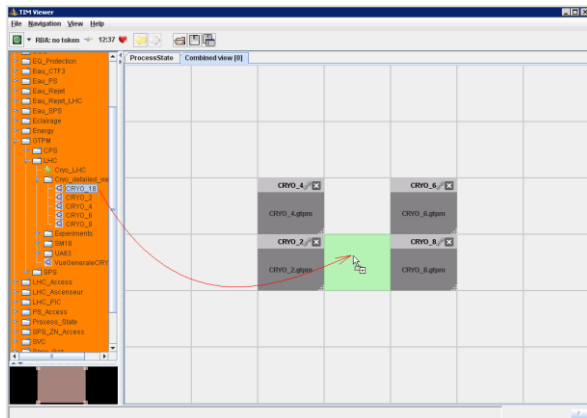


Figure 21 - The user drops a view from the menu on the left into a free spot

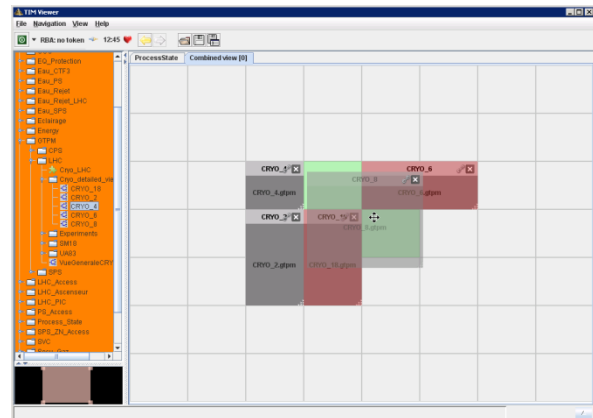


Figure 22 - The user tries to move a view

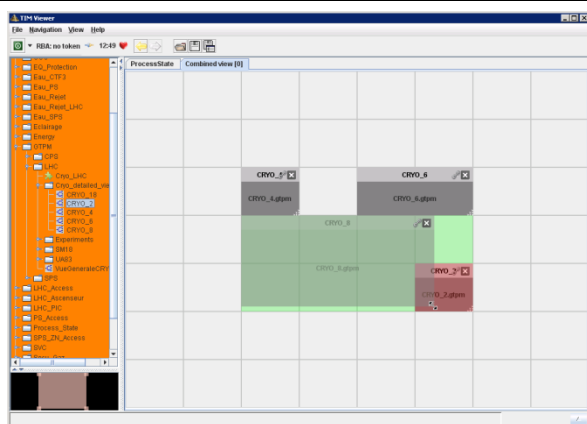


Figure 23 - The user tries to resize a view.

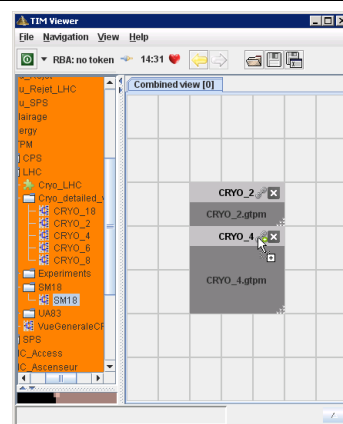


Figure 24 - The user adds a link by dragging a view from the menu on the left onto the link icon

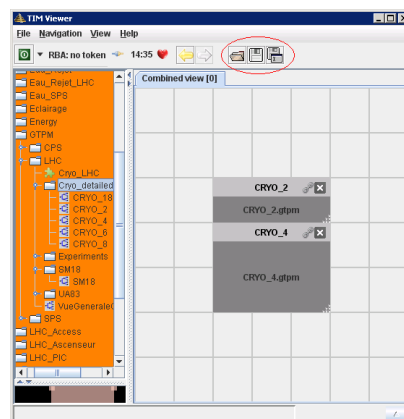


Figure 25 - The user uses the toolbar to save the view

4.2.2 The view in use

In this section two illustrations of how the *Combined View* looks like when it is opened by the user are presented.



Figure 26 - Showing a Combined View, the user is about to press the maximize button

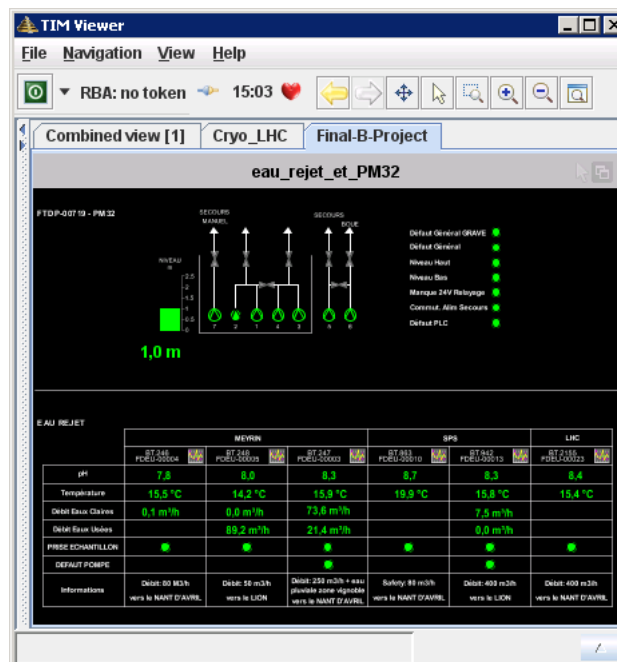


Figure 27 - A view is maximized; the toolbar is showing for that view

4.2.3 Generated XML

When creating a *view* it is saved as *XML*. Please see Appendix E for how the *XML* output looks like, and a screenshot of how it is interpreted by the TIM Viewer.

5. Implementation

The entire GUI (Graphical User Interface) in TIM is made using SWING ^[4], which is an API for providing GUIs in Java.

5.1 Introduction to the classes

In this section the relations and functions of the most important classes will be described.

5.1.2 Overview

Below is an overview of the most important classes for the *Combined Views*. The public methods are given for just a few classes.

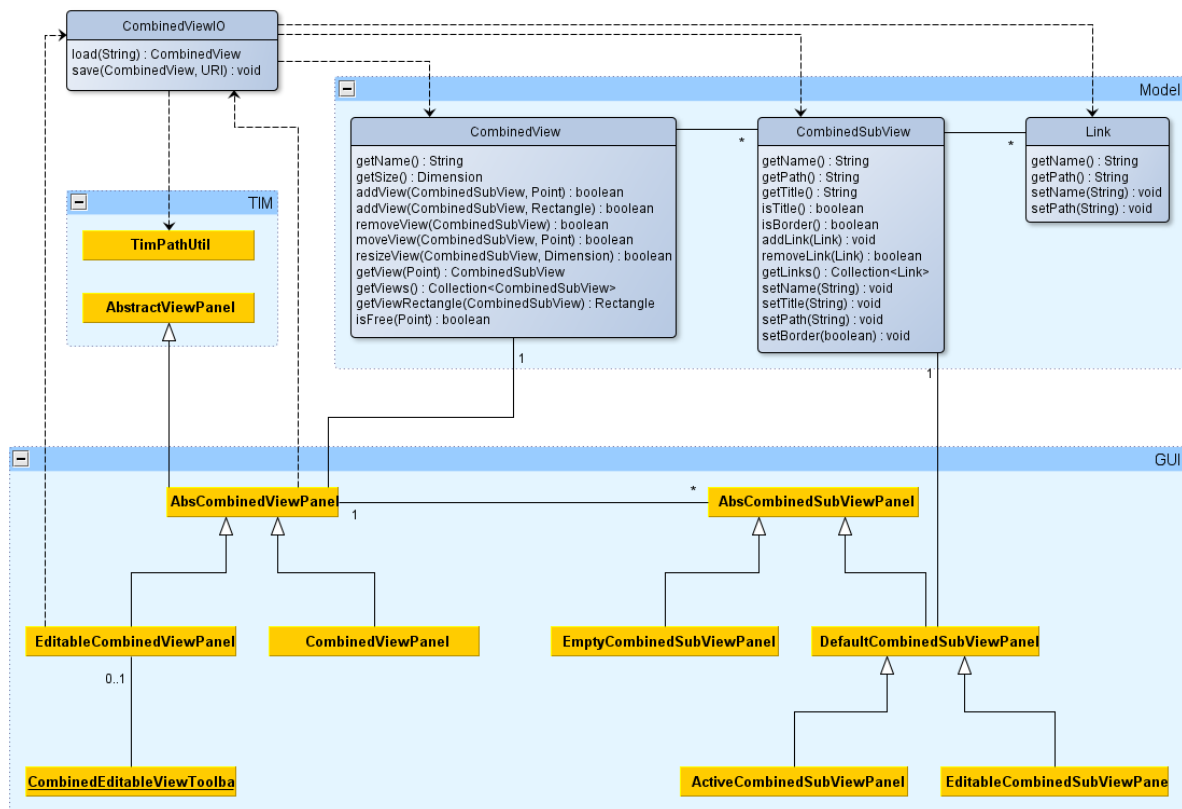


Figure 28 - Diagram of the most important classes

The GUI is split up into classes to make the loading of a *Combined View* as fast as possible. Also, if the methods used for editing a *Combined View* were mixed with the methods for displaying, it would become messy, and the code could become difficult to maintain.

5.1.1 Descriptions

In this section a short description for all the classes presented in Figure 28 are presented.

Model

CombinedView	Holds the attributes and the logic for a <i>Combined View</i> . It has functions for adding, removing, moving and resizing views.
CombinedSubView	Holds the attributes for a <i>view</i> .
Link	Holds the attributes for a link. A link holds information about a <i>view</i> that it is linking to.

Others

CombinedViewIO	Have functions for saving and loading a <i>Combined View</i> .
TimPathUtil	Is used to find the paths relative to the <i>Old</i> , <i>Production</i> and <i>Next</i> folders.
AbstractViewPanel	All views in TIM must extend this abstract class. This class extends JPanel, which is a container where graphical components can be added. This abstract class forces you to implement some functions that all <i>views</i> must implement.

GUI (Graphical User Interface)

AbsCombinedViewPanel	Functions and attributes that is needed both for displaying and editing the view.
CombinedViewPanel	The Combined View when it is actually shown to the user.
EditableCombinedViewPanel	A Combined View that is being edited. Listens for mouse events for moving or resizing of a view. Does also activate highlighting of views when the user tries to add, move or resize a view.
CombinedEditableViewToolbar	Provides buttons for saving and loading on the toolbar.
AbsCombinedSubViewPanel	Contains methods for making the view glow green or red, which is used in user interactions.
EmptyCombinedSubViewPanel	Contains methods for supporting dropping of <i>views</i> on it.
DefaultCombinedSubViewPanel	Contains methods for drawing a <i>view</i> , with title and control buttons (the buttons in the upper right corner)
ActiveCombinedSubViewPanel	Contains methods for loading a <i>view</i> and responding to mouse interactions from the user. For example when the control buttons are pressed (the buttons in the upper right corner)
EditableCombinedSubViewPanel	Contains methods for loading a view for editing. Handles mouse interactions for when the user presses any of the control buttons, and when right clicking the title bar. Does also handle drag and drop events for adding links.

5.2 Methods

In this section some of the methods that are used to implement the *Combined View* functionality into the existing TIM system will be described.

5.2.1 The graphical user interface and the interactions

In this section some of the methods that are used to implement the graphical user interface and the user interactions will be described.

Drag and drop

Drag and drop is used when the user drags a *view* from the menu bar on the left and drops it into the *Combined View*, like in Figure 21. This functionality is built into SWING, which makes it much easier to create the drag and drop functionality.

Layout manager

A layout manager decides where the different containers (graphical user interface components) in a window should be placed. For the *Combined View* a layout manager called *TableLayout*^[5] is used.

This is a perfect layout manager for this task. It partitions the container into a set of rows and columns where containers can be added. Spanning several rows and columns are supported.

Layers

The *Combined View* is built using layers^[6]. In the *EditableCombinedViewPanel* two layers is used; the *default layer* and the *drag layer*. All components are normally put into the *default layer*, but when the user wants to move or resize a *view* it is removed from the *default layer* and added to the *drag layer*. No layout manager is used for the *drag layer*, which means that the positions of the containers must be set with absolute positioning. The *view* can therefore be moved freely according to where the mouse is being moved. The layers also give the functionality to put the containers in different Z-orders. For example the *default layer* is at the bottom, and above it, comes the *drag layer*, which has a transparent background.

5.2.2 Input and output

Simple xml is used for saving and loading a *Combined View*.

5.3 Changes to the TIM system

In this section I will describe how the *Combined View* is integrated into the existing TIM projects, and what changes were needed.

Some refactoring and changes had to be done to implement the *Combined View* functionality properly without hacks.

To create TIM views, a factory called “*ViewPanelFactory*” can be used. It would return an instance of a requested *view*. This factory was in a java project called “*tim-jviews-viewer*”, which is also where the entry point of the application is. The *Combined View* classes are in a project called “*tim-jviews-panels*”, which is a dependency of the “*tim-jviews-viewer*”. This caused a problem; because the class “*ActiveCombinedSubViewPanel*”, which is inside “*tim-jviews-panels*”, needs to access the “*ViewPanelFactory*” to create the *view* requested, but this cannot be done, since the “*tim-jviews-panels*” project cannot depend on the “*tim-jviews-viewer*” project. See Figure 29.

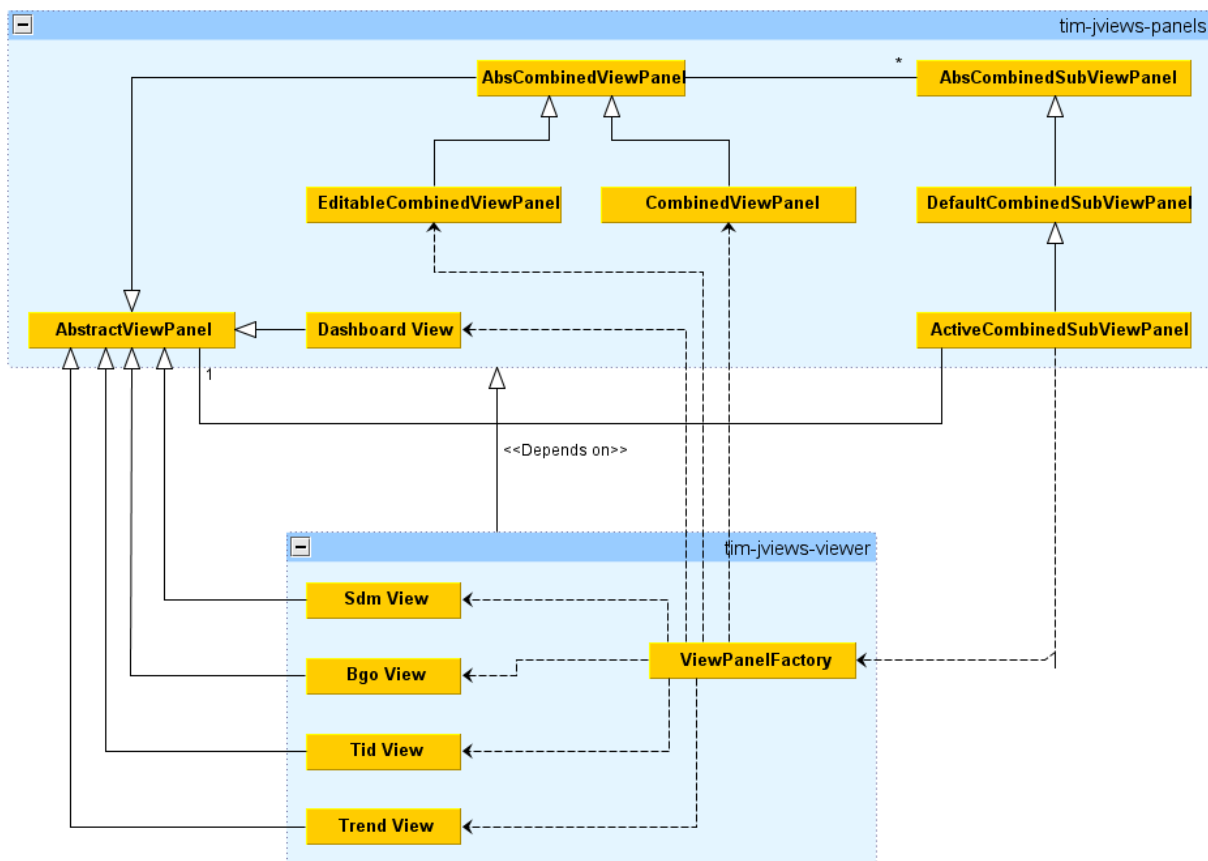


Figure 29 - Dependencies before the changes

The TIM project leader said that the “*ViewPanelFactory*” and the views should have been moved to the “*tim-jviews-panels*” earlier, but they did not have the time then. Therefore I moved the classes that were necessary to move, and it now looks like in Figure 30. All the view implementations, its dependencies and the factory are now moved to the “*tim-jviews-panels*”. I understood why it had been delayed when I started moving things. There were many circular dependencies of the classes that had to be separated before moving them into the other project.

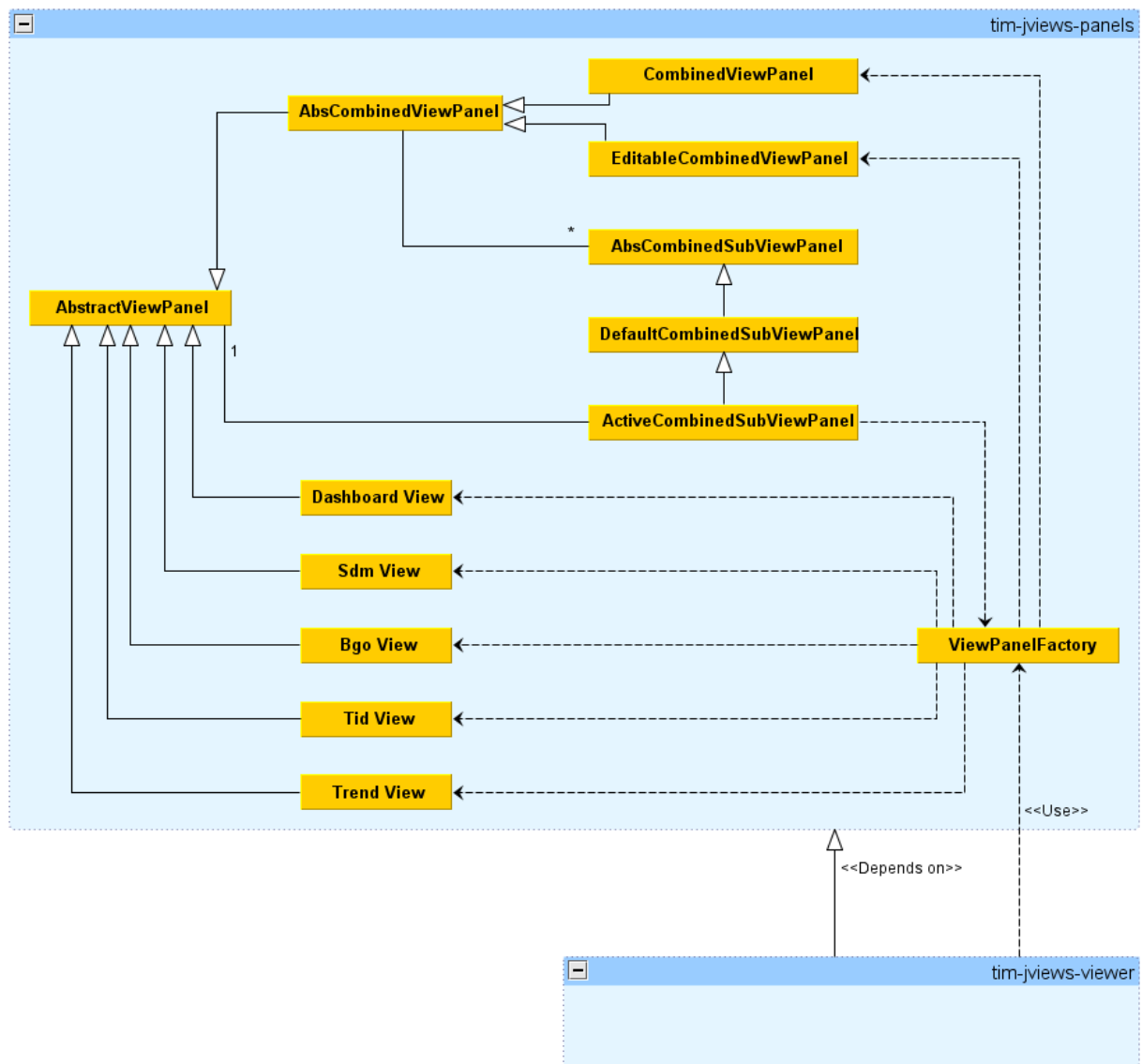


Figure 30 – Dependencies after the changes

6. Testing

This chapter will describe how testing was carried out.

6.1 Testing for the *Combined View* project

Testing was done by creating a few *junit* tests, but most by testing it manually. This is because the biggest part of the program is the graphical user interface. Automated tests can easily be made for the model, but for the graphical user interface it is not that easy.

6.1.1 Testing of the model

After creating the model and a graphical user interface to represent it, some strange behavior appeared. A *junit* test was then created to test the model. Several problems were found and solved. Tests were also made for the classes that transfer the model into xml, and that transfer xml into a model. After looking at the xml, some changes were made to make it more readable by not including attributes that are not changed by the user.

6.1.2 Testing of the graphical user interface (GUI)

While doing the implementation, constant testing was done by trying the features. Testing on the performance of the *Combined View* was also done. Some changes were made to the implementation to improve the performance.

6.1.3 Showing it to an operator

After the implementation one of the operators tried the *Combined View*. He liked it, but came with a few minor improvements. He also then told that he would like to have the link functionality.

After making the changes, the operator tried it again, and did not have any more requests.

6.2 Testing in general

During the work at CERN, a lot of bug fixing and small tasks have been completed in addition to the *Combined View* project. When doing these tasks it is mostly manually testing that have been done. It is mostly when creating new classes for the model or data access objects that *junit* tests are created. Manually testing have often been used to try all features that might be affected by code changes, most often closely related to the GUI.

7. Project management

In this chapter I will describe my role at CERN; where the tasks come from, who I work for, and where I get advice or help.

7.1 My role at CERN

I work in the BE/OP/TI section (Beam department / Operation / Technical Infrastructure). The TI section monitors the entire technical infrastructure at CERN and looks after energy consumption, heating, cooling, ventilation, air conditioning equipment, safety installations, high vacuum, cryogenic systems, coordination and reporting after major infrastructure disruptions, and much more.

My supervisor, Peter Sollander, have sent me to work with the TIM group (Technical Infrastructure Monitoring) who has created the TIM system. The TIM group is a part of the GS/ASE/SSE section (General Infrastructure Services Department / Access, Safety and Engineering tools / Safety Systems Engineering). I get tasks both from Matthias Braeger and from Peter Sollander, but it is Peter Sollander who have the final decisions, and who decides the priority of my tasks. Matthias Braeger is in charge of the TIM system, so I can talk with him if I need advice about anything regarding the TIM system. When changes are made to it, I will first show it to Matthias, and then he will tell if anything needs to be changed before I commit it. Matthias and his group are the people I meet every day, as I work in the TIM office. See Figure 31 to see my role at CERN.

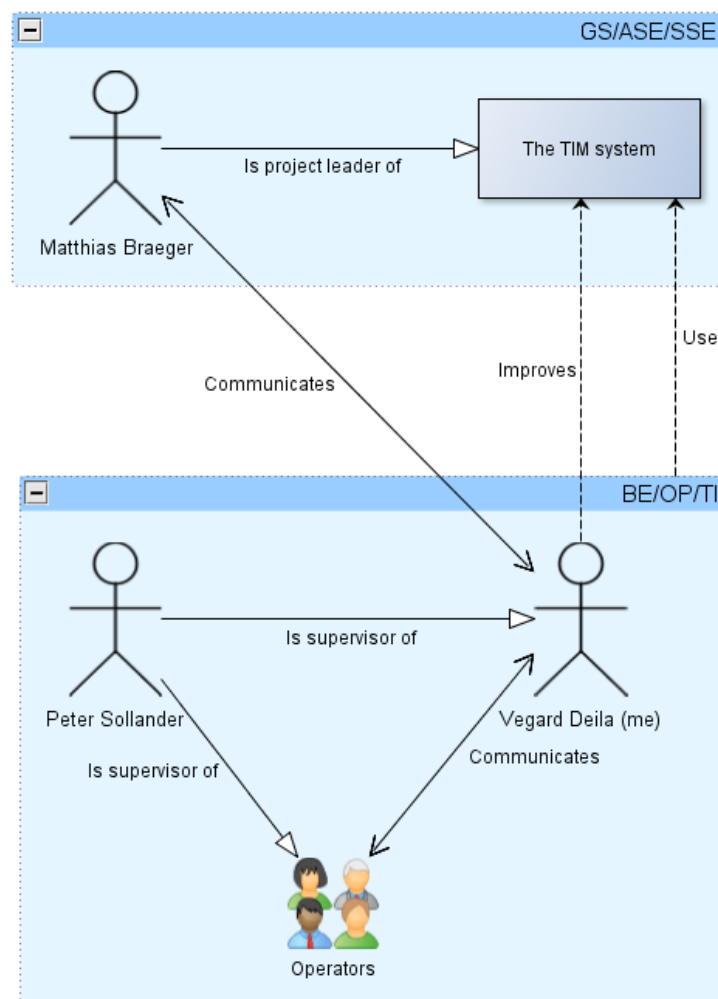


Figure 31 - My role at CERN

8. Summary, discussion, conclusions

In this chapter a summary of the outcome of the project work will be presented. The following questions will be answered: Are all the requirements fulfilled? Are there any limitations with the solution? Did the progress go as planned? Was the risk analysis from the initial report sufficient?

8.1 Changes and limitations

All the requirements in the initial requirement list is completed for the *Combined View* project, except one requirement that was changed; the zooming toolbar was supposed to be for the whole *Combined View*, not for the individual view inside it, see “3.1.6 Changes from the initial requirements” for more details. Changing it to do the zooming for each individual view gives one limitation; you cannot use the zoom buttons for zooming in/out on the entire *Combined View*. If this is wanted at a later time, it is easy to implement this functionality. But the reason for not having it is that it would be confusing for the user to have two sets of zooming toolbars; one for the entire *Combined View*, and one for the current active view inside the *Combined View*.

Because of how the *views* included in the *Combined View* are referenced, it is not supported to save a *Combined View* outside of the TIM folder structure. This could of course be done differently, but the operators did not mind. This behavior is from the requirements in chapter 2.1.4.

8.2 Progress

I may have failed to make it clear in the initial report that the dates in the Gantt diagram never were meant to be correct, but rather the amount of time spent on the project. The dates could never be planned because I all the time get tasks with higher priority that cannot be predicted to come. The amount of days used is almost as planned.

One of my colleagues started in 2009 to create a new TIM server and TIM client API from scratch. This new version he is creating is called TIM2. In the end of July 2011 we started to migrate all the old TIM projects to support the new TIM2 client API and server. TIM2 is now running in beta stage, and will be the official version when TIM1 is terminated in the end of January. Until then TIM1 and TIM2 is running in parallel. So the operators at the CCC (CERN Control Centre) are running both versions, using TIM1 if TIM2 somehow fails.

The *Combined View* is implemented in TIM2, so that I did not need to implement it twice, for both TIM1 and TIM2. This has caused the beta testing phase and the release of the product phase to melt together. But the *Combined View* is being tested, as the operators run TIM2 in parallel with TIM1.

8.3 Risks

There were not many problems. But I see that I could have added another item to the risk list; there is a risk that I get so many higher priority tasks that I would not have time to finish the *Combined View*, although the probability of it happening would be very low.

8.4 Further development of the *Combined Views*

The *Combined Views* is currently not available in the TIM Web Viewer, this could be done.

If you put a combined view inside another combined view the control buttons in the upper right corner would overlap each other if no title bar is used. This does work correctly if a title bar is used, but without the title bar only the buttons for the outer most *combined view* would work. It is currently not a solution for this if you do not want the title bar.

It is not possible to load a view that is outside of the TIM folder. This could maybe be fixed by having the user somehow select what version folder to get the views from (*Old*, *Production* or *Next*)

9. References and training

9.1 References

Description	Link	Accessed
1 - Introduction to Kanban	http://drdobbs.com/architecture-and-design/225702051	14.11.2011
2 - Kanban, Wikipedia	http://en.wikipedia.org/wiki/Kanban_(development)	14.11.2011
3 - Agile, Wikipedia	http://en.wikipedia.org/wiki/Agile_software_development	14.11.2011
4 - SWING (Java)	http://en.wikipedia.org/wiki/Swing_(Java)	15.11.2011
5 - TableLayout (Java)	http://java.sun.com/products/jfc/tsc/articles/tablelayout/	15.11.2011
6 - Layers in Swing	http://docs.oracle.com/javase/tutorial/uiswing/components/roootpane.html#layeredpane	15.11.2011
7 - About CERN	http://public.web.cern.ch/public/en/About/About-en.html	09.11.2011
8 - CERN, Wikipedia	http://en.wikipedia.org/wiki/CERN	09.11.2011
9 - About TI section	https://espace.cern.ch/be-dep/OP/TI/default.aspx	10.11.2011
10 - CERN budget	http://indico.cern.ch/getFile.py/access?contribId=60&sessionId=4&resId=0&materialId=0&confId=114484	18.11.2011
11 - Stack overflow Forum	http://stackoverflow.com/	
12 - Oracle documentation	http://download.oracle.com/docs	
13 - Agile Manifesto	http://agilemanifesto.org/	12.12.2011
14 - Java documentation	http://download.oracle.com/javase/1.4.2/docs/api/	

9.2 Training

In this section I will inform about the training I have completed at CERN. The training was not necessary needed to fulfill my tasks.

Safety

I have done the following training (online on internal CERN pages)

Course	Description
Basic Safety (Levels 1 and 2)	Must be followed by any persons working on the CERN site.
Computer Security	Training needed for using any computer at the CERN site
Specific Risks (level 3)	
LHC Machine (Level 4)	
ATLAS Safety (Level 4A)	
LHCb Safety (Level 4b)	
CMS Safety (Level 4C)	
Electrical Safety Awareness	

Other

Two levels of French courses.

Appendices

Appendix A: Dictionary

The list is alphabetical

BE/OP/TI	Beam Department / Operation / Technical Infrastructure, a section at CERN
C²MON	CERN Control and Monitoring Platform. The new platform for TIM2.
CCC	CERN Control Centre
CERN	The European Organization for Nuclear Research
GUI	Graphical User Interface
LHC	Large Hadron Collider
TIM	Technical Infrastructure Monitoring
UI	User interface
XML	Extensible Markup Language

Appendix B: Progress plan

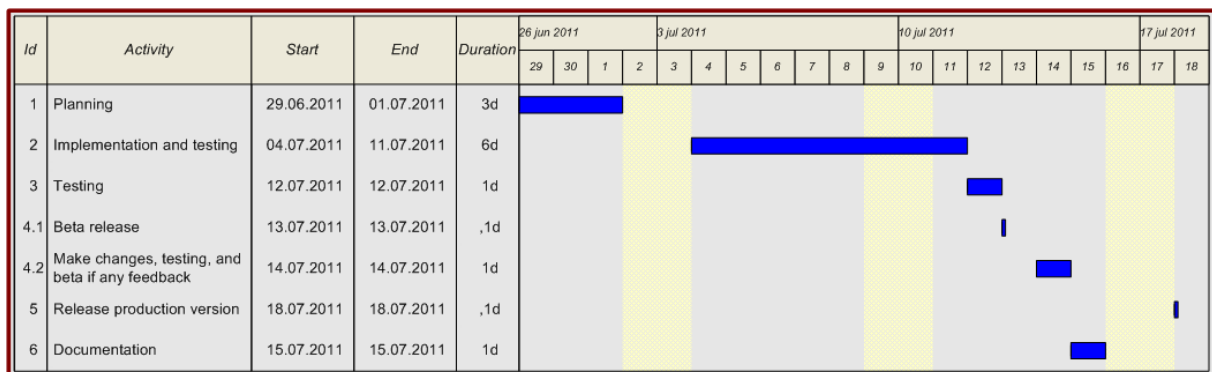
In this appendix the progress plan from the initial report, and how it ended up being, is presented.

The initial progress plan

In this section is the initial progress plan for the Combined View project, which was presented in the initial report.

Progress

The task is divided into 6 main activities, where each number is a milestone (the end of number 4.2 is the milestone for the beta release). There is also some time waiting for feedback after the beta release.



Sub tasks with details

1. Planning
 - a. Have meeting with Chris Wetton, which is one of the operators at the CCC and is the one that is requesting this feature for TIM. Discuss the requirements and propose solutions.
 - b. Make a document with the requirements and proposals of how the user will interact with the different features. Send this document to Chris, Peter and Matthias for responses.
 - c. After agreement with the last document, make a plan for how to implement the features into the TIM system. Show it to Matthias who is in charge of the TIM system, to get feedback.
2. Implementation and testing
 - a. Implement the changes while constantly doing testing on new implementations. Also do documentation of the code continually while coding. Also create *JUnit tests* if considered useful.
3. Testing
 - a. Do testing and change the code as necessary.
 - b. Show the new features to Matthias, if changes are suggested, do it and do point 3 again until it is accepted by Matthias.
4. Beta release
 - a. Make a *next* release (beta version) to show it to Peter and the operators. If requested, do additional changes and start over from point 3 again.
 - b. Show the new functions in the next release to the operators. Then let them run the next release until we consider it stable.
5. Release the production version (the version that is used by everyone using TIM Viewer)

- a. Make the release.
 - b. Send release e-mail to the operators containing what new features are added, and how to access them.
- 6. Documentation
 - a. Make class diagrams to give an easy overview of how the function works.
 - b. Be sure that all classes are well documented.

The final Gantt diagram

	Activity	Start	End	Duration	June		July					September								
					6th	7th	12th	13th	14th	15th	6th	7th	9th	12th	13th	14th	15th	19th	20th	30th
1	Planning																			
1.1	Meeting with Chris	06.06.11	06.06.11	1 hours																
1.2	Create proposal for a requirements document and send it to the people involved.	07.06.11	07.06.11	3 hours																
1.3	Planned the implementation	12.07.11	13.07.11	2 days																
2	Implementation and testing	14.07.11	14.09.11	7 days																
3	Testing	14.07.11	14.09.11	1 days																
4.1	Beta release + meeting with an operator	14.09.11	15.09.11	3 hours																
4.2	Make changes, testing, and beta again if any feedback	19.09.11	20.09.11	2 days																
4.3	Presented beta version again to an operator	30.09.11	30.09.11	2 hours																
5	Release production version	-	-	-																
6	Documentation - check that the code is well documented	20.09.11	20.09.11	2 hours																

Please note that some dates are skipped, because there was not any work on the project those days.
The darker the color is on the days, the more time is spent that day on the project.

You might have seen that in the final Gantt diagram, the dates does not correspond to the dates in the initial Gantt diagram, also the *"Release production version"* does not have any time set; see chapter 8.2 for the explanations.

Appendix C: Risk list

This is the list of risks that was presented in the initial report

Risk	Consequence	Probability	Severity	Precautions to be taken to prevent it, or actions to fix it
Miscommunication with the user	The user doesn't get what they asked for and time is used on something that is not needed.	Medium	High	Make sketches; both for the implementation and for the GUI. Show the sketches to the user to see if you have understood it right.
Sickness	Will not be able to do some or more of the tasks given.	Low	Medium	Make plans that can be understood by others. And make documentation WHILE coding, not after the implementation is complete.
Using too much time on one problem or task	Using more time than necessary	High	Medium	Make good implementation plans before starting the implementation. This makes the implementation itself easier and faster. Ask for advice if faced with a problem that seems difficult.
Introducing bugs to existing functions	Making existing function(s) inaccessible or difficult to use	Low	High	Do testing of all functions using the changed code. Also notify the users when a Next release is out so they can try the new version before putting it into production.
Introducing bugs to new functions	Making new function(s) inaccessible or difficult to use	Medium	Low	Do testing of the new functions. Let someone else also test the new functions to see if they think as expected when using the new function. Also notify the users when a Next release is out so they can try the new version before putting it into production.
Human error when it comes to releasing a product	Can make the program or functions in the program inaccessible or difficult to deal with	Low	High	Always test the version that the user will open when a new release is out. Also test the new features to see that everything works as expected. If a problem has occurred we can go back to the previous version.
Loss of data	Loss of data can in the worst case make the project impossible to complete.	Very low	High	Save everything on the network storage space and not locally on any computer. The network storage is managed by the IT people and is always backed up.

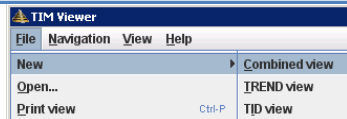
Appendix D: User manual

This appendix will describe how to create a *Combined View*, and is sending you in the right direction to learn how to use it.

How to create a new *Combined View*

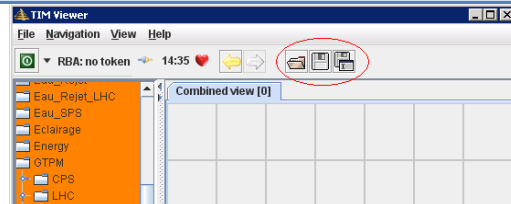
Follow these steps using the TIM Viewer

Step 1



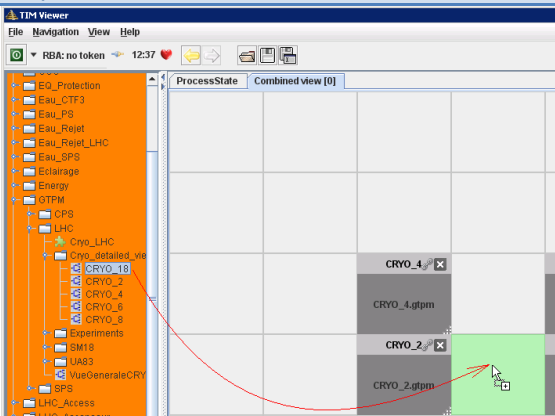
In the menu go to File->New->Combined view

Step 2



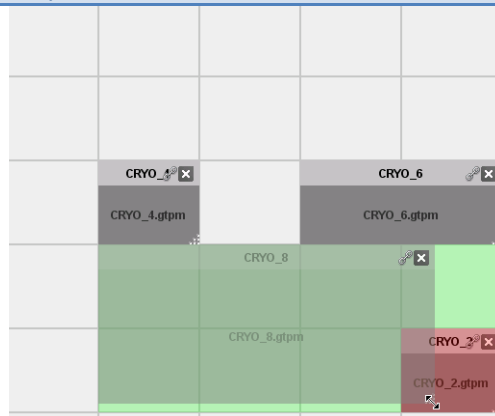
A new empty combined view appears. While creating a combined view, a toolbar will be on the top of the view where you can load an existing combined view, or save the current combined view.

Step 3



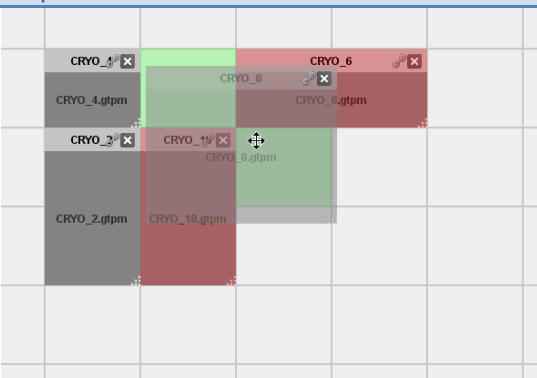
To add a view to the *combined view*, drag the views from the tree menu of the left into the new *combined view*.

Step 4



To resize a view, press and drag in the bottom right corner of the view you want to resize.

Step 5



To move a view, press anywhere in the view you want to move and drag it to the square you want to put it in. If any red color appears it means you cannot move the view there because the space is taken up by another view.

Step 6

If you would like to have a border on a view, right click it and press "Use border".

Step 7

If you would like to change or remove the title of a view, right click it and press "Edit title".

Step 8

To save the combined view, use the toolbar as shown in step 2. Regardless of where you save it, you must open it while it is in the TIM folder structure. (Inside the *Old*, *Pro* or *Next* folder)

How to use a Combined View

Using the *combined view* is very simple. The easiest way to learn how to use it is by trying it. Notice the buttons in the right corner of each of the views inside the *Combined View*. Hold the mouse over the buttons to get a description of what it does. Try also to right click the title bar. If a view does not have a title bar, a button with a plus sign will replace it, try right clicking that button instead.

Appendix E: XML output

In appendix you will see the XML output generated by the *Combined View* functionality and a screenshot of the xml loaded by the TIM Viewer. The indentations in the XML are manually added.

Generated xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<combinedView name="ExampleView">
  <views>
    <view rectangle="3, 2, 1, 1">
      <combinedSubView name="LHCUnder.ivl" path="${TIM_HOME}/Ther_Vent_LHC/LHCUnder.ivl" title="LHC Under">
        <border use="false" />
      </combinedSubView>
    </view>
    <view rectangle="2, 2, 1, 1">
      <combinedSubView name="Eclairage%20SPS.ivl" path="${TIM_HOME}/Eclairage/Eclairage%20SPS.ivl" title="">
        <border use="false" />
      </combinedSubView>
    </view>
    <view rectangle="2, 4, 2, 2">
      <combinedSubView name="ther_vent_sps.idbd" path="${TIM_HOME}/Ther_Vent_SPS/ther_vent_sps.idbd" title="">
        <border use="false" />
        <links>
          <link name="extraction_BA3" path="${TIM_HOME}/Ther_Vent_SPS/detailed_views/extraction_BA3.idbd" />
          <link name="extraction_BA5" path="${TIM_HOME}/Ther_Vent_SPS/detailed_views/extraction_BA5.idbd" />
          <link name="extraction_TT10" path="${TIM_HOME}/Ther_Vent_SPS/detailed_views/extraction_TT10.idbd" />
          <link name="extraction_TT20" path="${TIM_HOME}/Ther_Vent_SPS/detailed_views/extraction_TT20.idbd" />
          <link name="extraction_TT60" path="${TIM_HOME}/Ther_Vent_SPS/detailed_views/extraction_TT60.idbd" />
          <link name="extraction_TT70" path="${TIM_HOME}/Ther_Vent_SPS/detailed_views/extraction_TT70.idbd" />
          <link name="pulsion_BA2" path="${TIM_HOME}/Ther_Vent_SPS/detailed_views/pulsion_BA2.idbd" />
        </links>
      </combinedSubView>
    </view>
    <view rectangle="4, 5, 2, 1">
      <combinedSubView name="Energy-1-day.trend" path="${TIM_HOME}/CombinedView/trend/Energy-1-day.trend" title="">
        <border use="false" />
      </combinedSubView>
    </view>
    <view rectangle="4, 2, 2, 3">
      <combinedSubView name="ProcessState.ivl" path="${TIM_HOME}/Process_State/ProcessState.ivl" title="">
        <border use="false" />
      </combinedSubView>
    </view>
  </views>
</combinedView>
```

```

</view>
<view rectangle="2, 3, 2, 1">
  <combinedSubView name="LHCodhOverview.idbd" path="$${TIM_HOME}/Secu_Gaz/LHC-ODH/LHCodhOverview.idbd" title="">
    <border use="false" />
    <links>
      <link name="Sector12" path="$${TIM_HOME}/Secu_Gaz/LHC-ODH/detailed_views/Sector12.ivl" />
      <link name="Sector23" path="$${TIM_HOME}/Secu_Gaz/LHC-ODH/detailed_views/Sector23.ivl" />
      <link name="Sector34" path="$${TIM_HOME}/Secu_Gaz/LHC-ODH/detailed_views/Sector34.ivl" />
      <link name="Sector45" path="$${TIM_HOME}/Secu_Gaz/LHC-ODH/detailed_views/Sector45.ivl" />
      <link name="Sector56" path="$${TIM_HOME}/Secu_Gaz/LHC-ODH/detailed_views/Sector56.ivl" />
      <link name="Sector67" path="$${TIM_HOME}/Secu_Gaz/LHC-ODH/detailed_views/Sector67.ivl" />
      <link name="Sector78" path="$${TIM_HOME}/Secu_Gaz/LHC-ODH/detailed_views/Sector78.ivl" />
      <link name="Sector81" path="$${TIM_HOME}/Secu_Gaz/LHC-ODH/detailed_views/Sector81.ivl" />
    </links>
  </combinedSubView>
</view>
</views>
</combinedView>

```

Screenshot

This is a screenshot with the XML above loaded into the TIM Viewer.

