First name: Jean-Baptiste
Last name: LAMOTTE
Login: lamott_j

Internship report

From March 1st to August 31st

MERETHIS
12 avenue Raspail
94270 Gentilly – France

To validate the last year of Epitech, an internship is required. I did it in MERETHIS, monitoring company, editor of Centreon, and Centreon extension.
I have worked on the Centreon-Map extension, an advanced cartography tool enabling Centreon exploitation in a user friendly interface. My different tasks were project management, architecture, development, support, consulting, documentation, and packaging.

Acknowledgement:

First of all, I thank **Romain Le Merlus and Julien Mathis**, founder of MERETHIS, chief executive and technical officer, who trust me for 3 years.
I also thank Cédric Temple, project manager who oversees the consultants, for his experience and advice in project management.
Finally, thank to all the MERETHIS employees, for the general spirit of the company.

Table of contents

1. Summary

Monitoring is booming. Indeed, all companies today have growing need in computers, and the number of devices is increasing. Similarly, networks transmit more and more information. This makes it possible for computers to take a decisive place in the different companies. However, this makes infrastructure more sensitive.

Monitoring can survey the resources performance and prevent failures. It also enables us to identify them in the shortest possible time, before they have a wider impact.

MERETHIS has originally created Centreon, a complete open source monitoring solution. MERETHIS also develop extensions for Centreon, providing functionality basically unavailable in Centreon.

I have worked for almost 3 years on the first retail expansion, Centreon-Map. This tool enables advanced cartography, by representing resources supervised in Centreon, in views. These views are customized on demand. It is possible to create topology views, businesses views, networks views, technical views, etc... This tool is intended to experienced administrators as well as simple users only in charge of the Centreon data exploitation. No resources configuration is done In Centreon-Map, this part being reserved to Centreon.

During the internship, I was asked to add new features to the software, but also a variety of related tasks. I worked under the responsibility of my chief executive officer (CEO) and chief technical officer (CTO).

I was responsible for the architecture and development of Centreon-Map client. The server was developed by my CTO. We have worked together for each new functionality. The different frameworks study has been an important step for the project advancement. I got a part of project management too. I settled roadmaps, timeline, and suggested new features. I set up tools to protect the software source code against reverse engineering by obfuscation. I have optimized the building process, which saved a valuable time and developed adapted tools. These tools have prepared the automated test step.

Currently, Centreon-Map tests are done manually. We have created a test book, not to forget any test. I have also achieved support around Centreon-Map for users encountering use problems. I have been trusted some consulting missions to set Centreon-Map, and create network and business views.

Finally, I was responsible for maintaining Centreon-Map documentation for each new version. Other tasks have been added during the internship, such as the Centreon-Bi scheduler (another Centreon extension) architecture and development. For this project, I was under responsibility of the Centreon-Bi project manager

During the internship I have increased my skill in java language, with architecture and development, but I have also gained in experience on all the tasks making software development productive and secure. The project management is an important new step for my career.

2. About MERETHIS

2.1. Activity field

Today businesses and organizations have more and more computers and electronic equipment. To work at their best, all those equipments have to be monitored (status, load, quality etc…). All this survey cannot be done manually as their number is high. Here network monitoring software finds its place.
MERETHIS is at the heart of these surveillance systems, offering many tools to effectively monitor all theses equipments.

MERETHIS has customers in many different sectors, like services, banks, insurances, industry, editors.

2.2. Company

MERETHIS was born in 2005 from the will of engineers who wanted to answer to companies problems in the domain of IT monitoring.
Romain Le Merlus and Julien Mathis, founders of MERETHIS, have started an End of Study project in 2003, at Epitech. The initial project was to create a frontend to the famous Nagios scheduler, to simplify its configuration. This project was Oreon (renamed Centreon).

Today, MERETHIS is still developing Centreon for the open source community, but also offer extension modules for Centreon, for the businesses requiring further results from their monitoring.

- Centreon-MAP (Advanced Mapping)
- Centreon-BAM (Business activity monitoring)
- Centreon-BI (Business intelligence)

MERETHIS also offers complete preinstalled solutions, to monitor IT, with its "Meribox", including the various modules.

Finally, MERETHIS offers training, consulting and support to help companies to use more efficiently Centreon.

Over the years the company has grown and we are now approaching 20 employees.

Julien Mathis **Chief Technical Officer**		**Romain Le Merlus** **Chief executive officer**	
Consulting	**R & D**	**General Services**	**Commercial**
J.Mathis	J.Mathis	R.LeMerlus	R.LeMerlus
C.Temple	**J.B.Lamotte**	K.Sacouney	
M.Sugumaran	M.Sugumaran		
S.Sho	S.Sho		
N.Cordier			
D.Duponchelle			
L.Pinsivy			
A.Pellé			

We have also an administrator (T.Bernardini) and an executive secretary (K.Sacouney).

2.3. Services

2.3.1. Support

MERETHIS offers support to solve end-users' problems using Centreon. It includes bug-fixing on the Open source software (edited by MERETHIS).

There are 4 types of offers: bronze, silver, gold and platinum. Bronze support is only web based. Other offers include phone support.

It is based on a ticketing system with 3 types of problems: minor, major and blocking.

Problems	**Minor (p3)**	**Major (p2)**	**Blocking (p3)**
Acquittal - Workaround	8 hours	4 hours	2 hours
Fix	5 days	48 hours	24 hours
credits	6 *credits*	12 *credits*	18 *credits*

2.3.2. Consulting

The consulting goal is to listen to the specific needs of each customer. It consists in defining the architecture (server, processor, memory, etc…) necessary for the IT of the customer, setting up a functional solution with Centreon, expanding the number of monitored equipment or adding new indicators.

Often for the new customers, the complete deployment of Centreon is done by our consultants. Usually, they purchase a Meribox, including modules.

2.3.3. Training

MERETHIS offers different type of training, dedicated to the Open Source Monitoring, with Centreon / Nagios. Currently, there are 5 types:

- "Basics of monitoring" (3 days)
- "System administration" (2 days)
- "Advanced monitoring" (5 days)
- "Operator" (2 days)
- "On demand"

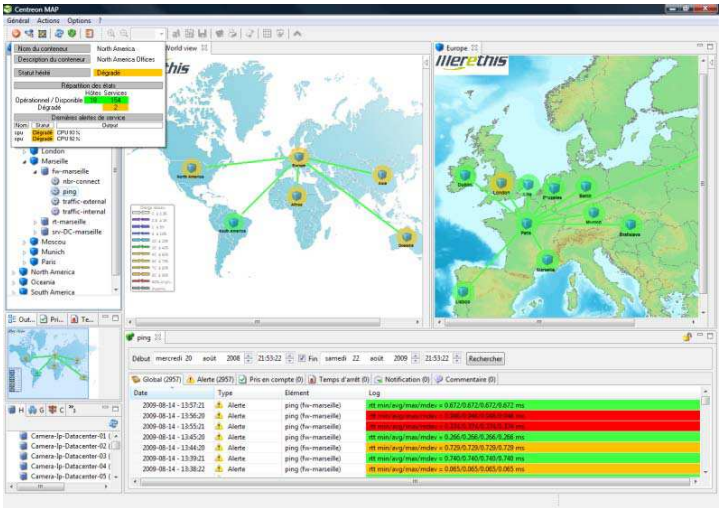
Each training is specific to the level of the customers. Best administrators appreciate the "Advanced Monitoring" training to increase their skills in very specific technical points. There are some requirements to access to each training.

2.3.4. Research and development

Since its foundation MERETHIS is developing Centreon software, and regularly adds new features.
For 3 years now, MERETHIS has also had R&D services, to create commercial extensions for Centreon.

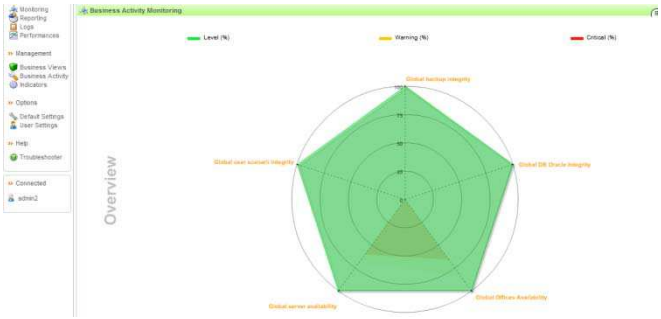
2.3.4.1. Centreon-MAP

Centreon Map is the first extension of the company. It is advanced mapping software that displays different indicators monitored by Centreon.
The "drill-down" navigation enables to create custom views with unlimited level(s). Each user can create a view corresponding to his need, like business, functional, technical, topological, geographical views.
Its use is drastically simple and easy, thanks to the drag and drop. It is based on the configured resources of Centreon. Each Centreon-Map resources are synchronized in real time with the Centreon resources.
This is a client / server software type.



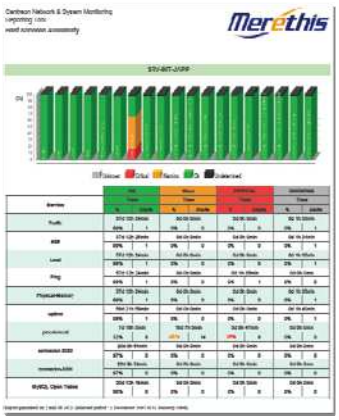
2.3.4.2. Centreon-BAM

Centreon-BAM (Business Activity Monitoring) is the second company extension. It is an advanced reporting software which offers real time measurement indicator, by aggregating the stats of various checkpoints monitored by Centreon. These indicators can be integrated in the Centreon-Map.



2.3.4.3. Centreon-BI

Centreon-BI (Business Intelligence) is the third company extension. It is a powerful reporting software, exploiting Centreon collected data. It offers the possibility to generate custom reports.



2.4. Positioning of the internship in the company

For 3 years now, I have been working on the Centreon-Map project (and more specifically, the Centreon-Map-Client aspect)).

First, I have worked as a trainee, during my Epitech 3rd year, only for development, and then during my Epitech 4th year, I have been a professionalizing contract employee. Finally, for my 5th year, I got hired with an open-ended contract (part-time during school, and full time since the beginning of the internship). My official status is software engineer.

3. Work and task

Due to the nature of the various products (commercial software, protected source code), no information concerning the technical development will be revealed.

3.1. Objective

3.1.1. General purpose

The specifications were not defined at the beginning of the internship. They were built day after day, according to new needs and these needs priorities.

However, my main task had to do with the Centreon-Map project, and more specifically, the Centreon-Map-Client. Here is a list of my different tasks.

- Project management (partial)
- Architecture
- Development
- Obfuscation
- Packaging
- Functional tests
- Support
- Consulting
- Documentation

Others tasks were added between Centreon-Map release, as the Centreon-Bi scheduler.

- Architecture (partial)
- Development
- Functional tests

3.1.2. Detailed explanation of the results to obtain

3.1.2.1. Centreon-Map

3.1.2.1.1. Context

I have been working for nearly 3 years on the Centreon-Map project. Initially, I was only a developer. As years went by, my responsibilities have increased, enabling my chief executive officer and chief technical officer to focus on others tasks.

Centreon-Map team is reduced. I now have multiples roles. I am project leader (co leaded by J.Mathis (CTO), and R.LeMerlus (CEO)), architect, developer, packager, tester, researcher, supporter, and occasionally consultant. Regarding development, I only work on the Centreon-Map-Client, the server being developed by Julien Mathis.
Some developed features are presented in appendices.

3.1.2.1.2. Project management

This task consists in defining new features to implement. We must take account of the user feedback about desired changes, and make some choices about these new features priorities.
We must also inquire about the competitors and try to always be one step ahead of their products.
For each new feature, it is necessary to estimate the time development and group feature related to common parts of the source code (so as to destabilize as little as possible the software).
The time development assessment and grouping features are done during the architecture step.

3.1.2.1.3. Architecture

Thinking about architecture is an important step of the project. It enables to define the best framework to use, to obtain a powerful and assessable software.
In several people teams, the architect makes the link between project manager and developer.

3.1.2.1.4. Development

From the specifications determined during the architecture step, development can begin. There is usually an upstream work on the different API frameworks.

3.1.2.1.1. Obfuscation

To avoid a potential hacker to decompile / reuse Centreon-Map, source code must be obfuscated. For this purpose, I have tested different tools, and integrated the result in the building process.

3.1.2.1.2. Packaging

Once the release is generated, it is necessary to build a package to be readily and easily readable by users. To this end, I have optimized the process, and developed several tools to save valuable time.

3.1.2.1.1. Functional tests

Functional tests are essential for each release. For major release, complete tests are performed, whereas for minor version (also called bugfix version), only bugfixes are tested. Usually, I accomplish this task with the help of other employees.

3.1.2.1.1. Support

Most of the time, support help end users to resolve their problems with Centreon-Map. Tickets are usually resolvable using Kayako, however, sometimes they need to t ake remote control with WebEx. In the most complex cases, I have to intervene directly into the office customer.

3.1.2.1.1. Consulting

After the purchase of Centreon-Map, some customers need help from MERETHIS to achieve their mapping. The missions can last from 1 day to 1 week, depending on the preparations made before by the customers.

3.1.2.1.2. Documentation

I am in charge of maintaining Centreon-Map documentation, in French and in English. After each release, I incorporate the new features.

3.1.2.2. Centreon BI

For this project, I was under the responsibility of a collaborator (M.Sugumaran). I was in charge of creating the Centreon Bi scheduler, a component which launches report generations at specified times, defined in the Centreon-Bi Web UI.

My versatility in the java world and my fluency in multithreaded environments have enabled us to economize valuable time in the progress of this project. I have also advised M.Sugumaran on its architecture on many points.

3.2. Activity report

3.2.1. Real studies display

Development of Centreon Map is the activity where I spent the most time during the internship.

3.2.1.1. Project management of Centreon-Map

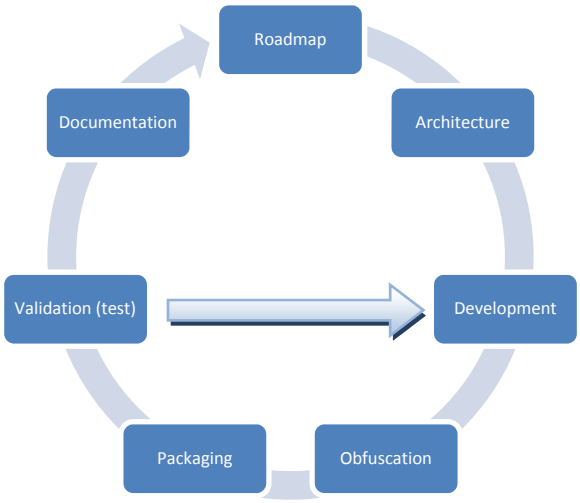
Project management was one of my tasks, but I was not fully engaged. I worked mostly on the client roadmap, suggesting changes and estimating time. The main decisions were taken or validated by my chief technical officer (J.Mathis) and chief executive officer (R.LeMerlus).

3.2.1.1.1. History

During my training, I have established several Centreon-Map roadmaps, for versions 2.4.x and 2.5.x. Version 2.4.x had already started at the beginning of the internship. Versions prior to 2.3.3 were performed before the internship.

3.2.1.1.2. Life cycle

Releasing a version consists into several steps. The first step is to establish the roadmap, containing the various features.



During the validation phase if bugs are found, we fall back to the development step. It is much inspired by the agile method (XP).

3.2.1.1.3. Roadmap

Generally, I estimate the development time of various client features, after having grouped them according to the code parts they affect.
Once the development time set, I validate the features to add with my CTO's and CEO's, and possibly report some to the next versions.
I am not charged to set deadlines for the server side.

Deadlines are generally respected; however, often functional tests reveal some bugs that take a long time to be found, delaying the release.

3.2.1.2. Centreon Map architecture

Centreon-Map is a 2 components application: client and server. The server is developed by J.Mathis, whereas I am responsible for the client development.
Studies of the architecture are generally performed by both J.Mathis and I.

3.2.1.2.1. Global architecture

3.2.1.2.1.1. Server

There is only one server type, which integrates into Centreon

- Centreon-Map-Server.

It includes the client Centreon-Map-Client-JWS.
The server is developed in PHP. It uses data from the Mysql database, containing the different resources from Centreon. (Host, host group, service, service group, status, log etc…)

3.2.1.2.1.1. Client

Client is divided in 2 categories:

- Centreon-Map-Client

This client is directly settled on the user workstation as with any application (Word, Excel etc ....)

- Centreon-Map-Client-JWS

This client is automatically spread on the user's workstation, using a simple browser. After the user has clicked on the Centreon link, the execution of Java Web Start is triggered. Java retrieves the client, and executes it in the JVM. Features are identical to Centreon-Map-Client.

There are also other clients, for the exclusive use of MERETHIS:

- Centreon-Map-Client-Demo

This client is a limited version of Centreon-Map-Client. It can only connect to our demonstration server.

- Centreon-Map-Client-Demo-JWS

This client is a Java Web Start version of the Centreon-Map-Client-Demo.

- Centreon-Map-Client-Debug

This client incorporates a debug console, used to view the transactions between the client and server.

Each type of client is available for different architectures and OS (Windows 32 & 64 bit, Linux 32 & 64 bit, Macos PPC & x86) for the 6 types of clients. (30 different client in all).

3.2.1.2.2. Centreon-Map-Client architecture

The internal functioning of Centreon-Map-Client cannot be explained in the report because of the confidentiality of this product.

However, Centreon-Map-Client is developed in Java and is based on Eclipse RCP framework, GEF, DRAW2D, JDOM, JFACE and EQUINOX. It respects the OSGi specifications.

Centreon-Map architecture is powerful and able to handle heavy loads. It can be used with a cluster of Nagios servers and a lot of control points. (Centreon is able to handle multiple Nagios).

3.2.1.2.3. Integrating new framework

I am responsible to determine the framework to use for the Centreon-Map-Client developments.
Frameworks are very rich in functionality, and all features are not used. For each new Centreon-Map feature, I have to study deeply those, and check if it is possible to use them, before searching a new framework overloading the current architecture.

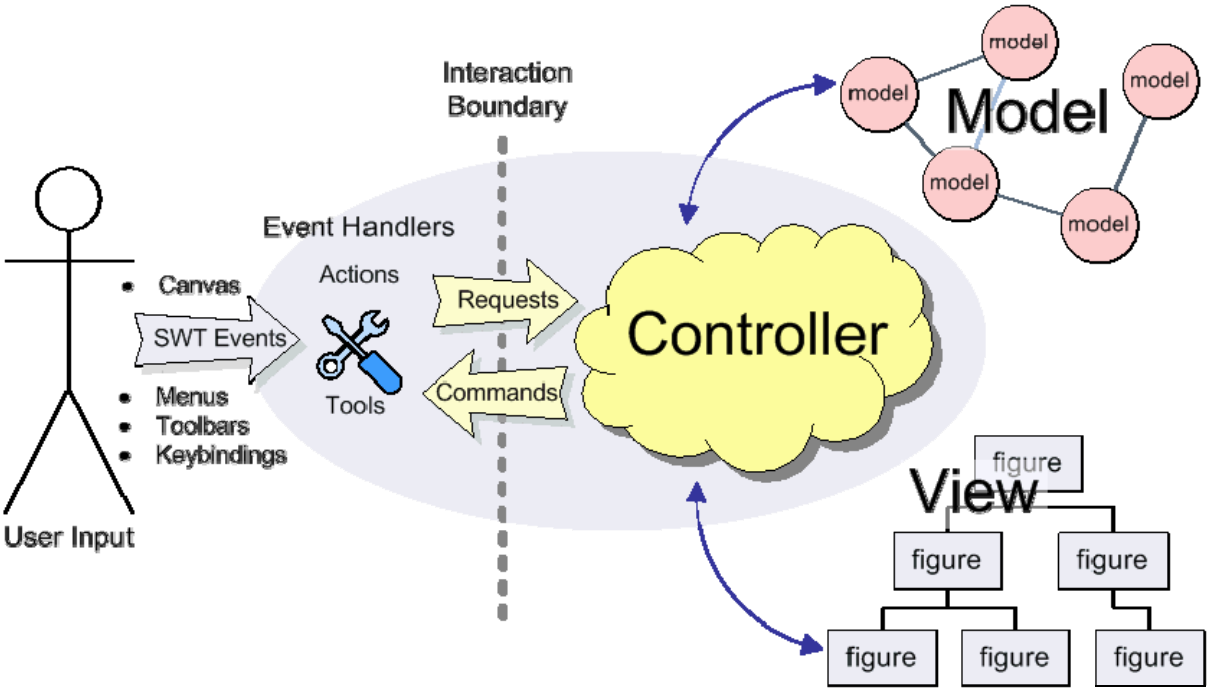
For each choice, I must refer to my CTO (J.Mathis), to whom I explain the development time required and functionality. He validates or invalidates the choices made.

3.2.1.2.4. GEF & DRAW2D framework study

Centreon-Map-Client is a very visual software; so, the greatest part of the work is done with GEF & DRAW2D.

These frameworks have been studied during the last 3 years. However, they suffer from a severe lack of documentation. The only available documentation is the official API of GEF. No mechanism is explained, and it for the architect and developer to understand their workings. Their vast complexity makes the task even more difficult.

During the internship, I studied the "tools" mechanism that interact with the user action, for the functionality of adding link. The objective was to preserve the spirit of Centreon-Map-Client: Its very simple use.



The tool can interact with the user at different levels.
To summarize the operation, they can capture events from mouse and keyboard, directly linked to the object controller of each resource.

After having mastered these tools, the use of the palette (a GEF component) could be diverted to contain a list of links.

3.2.1.3. Centreon-Map development

After having set the roadmap and defined the architecture, development can begin.

3.2.1.3.1. Code conventions

I try to respect the CCJPL (Code Conventions for the Java Programming Language) for developing Centreon-Map-Client. These conventions enable to keep a source code clean and make development more enjoyable.

3.2.1.3.2. Javadoc

Centreon-Map-Client source code is documented. Each method has in header a short description of the way they operate.

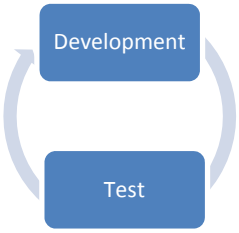
3.2.1.3.3. Refactoring

During development, I regularly carry out a general refactoring of the code. Functionally they are not changed, but it enables the application to improve performance, and to be more adaptable for later change. If this work wasn't done, the software development would become slower and slower, because of an accumulation of anti pattern.

For example, refactoring enabled to gain performance on the service status loading time. A small existing bottleneck slowing the overall functioning of status update. This problem was visible only with a huge amount of elements loaded in Centreon-Map (tested with a powerful testing server that emulates Centreon objects). Currently, nobody uses Centreon-Map with such charges, but the gain is still available.

3.2.1.3.1. Extreme programming

Principles used in the project advancement are close to those of extreme programming. The generation of a functional release occurs per cycle (as indicated in the project management).

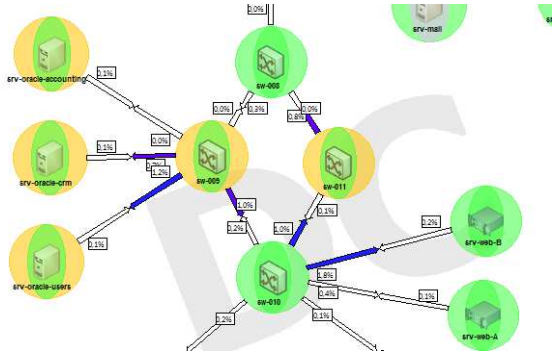


However, during development, I already apply a little cycle to pre-validate the different developments.
For Each development, I check it's correct, before proceeding to the next.

3.2.1.3.2. Non-exhaustive list of functionalities

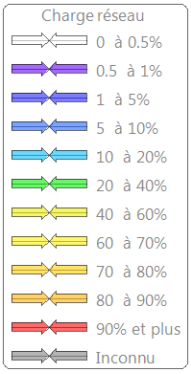
Here is a partial list of functionalities developed during the internship. Others are available in the appendices.

3.2.1.3.2.1. Load link, status link and legend

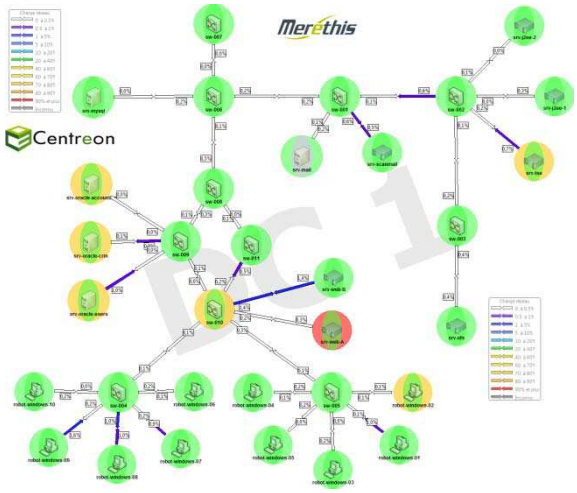


Load link and status link make possible to connect 2 resources together, and associate a service from Centreon. Load links are used to display the network traffic between different resources. Visualization of the data in this way is much more meaningful than in Centreon.

The legend that can be added to all view level, making possible to interpret the usage of a load link.

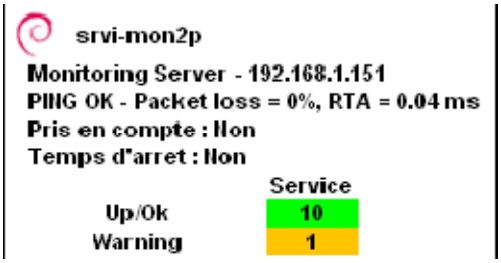


3.2.1.3.2.2. Recording in PNG format

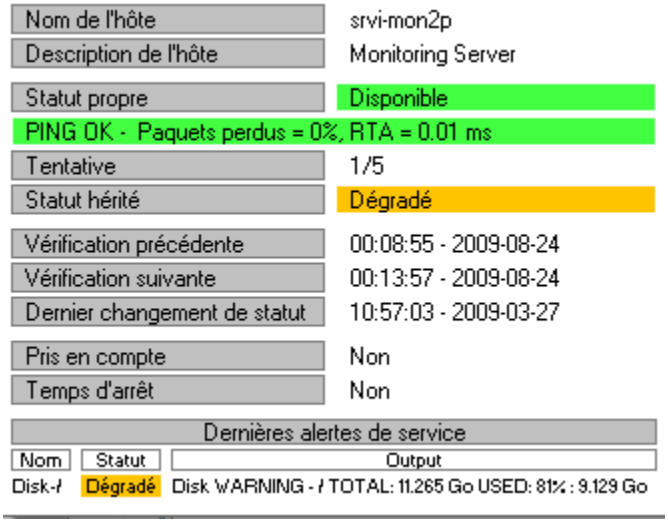


Views can now be exported into PNG format. It can take a kind of snapshot at a specific time to show the seriousness of the situation for example. This feature is also popular for the demonstrations.

3.2.1.3.2.3. Improved tooltips



Tooltips appear in flight over a link or resource.
Old tooltips were not very nice. These have been reworked and bring now much more information.



This kind of tooltip is difficult to achieve. Unlike web programming, it is necessary to calculate the exact coordinates of all elements. (Width and height of frames and text, total size of the tooltip, etc ...).

Above is the old version (2.4.x), and on right, the new version (2.5.x).
Other examples are available in the appendices.

3.2.1.3.2.4. Multiple languages

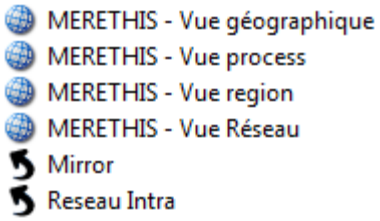
The software was only available in French before. Multiple languages support has been added. Centreon-Map-Client automatically detects the user OS language and switches to the good language. If the language is not available, English is displayed by default.

3.2.1.3.2.5. New Os support

Centreon-Map only worked on Windows x86, Linux x86, MacOS x86 and PPC. I have added support for Windows Linux x86_64 and x86_64 after request from users.

3.2.1.3.2.6. View shortcut

To improve timely access to views, I have added a feature to create shortcuts to objects contained in a view. These appear in the list of views available at the software launch. This feature has required a lot of refactoring.



3.2.1.3.2.7. Other features

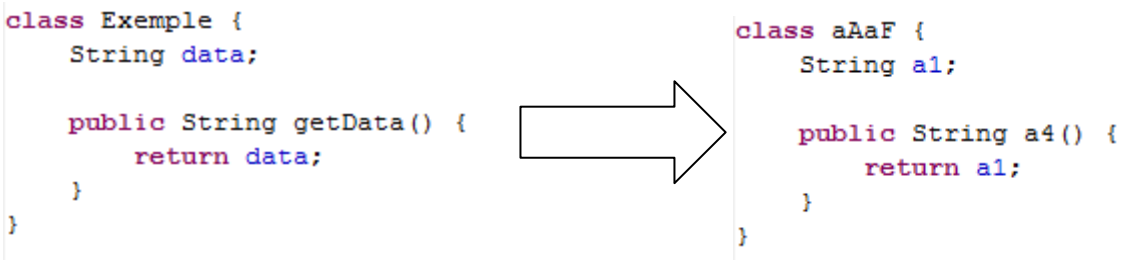
Other features are available in the annexes.

3.2.1.4. Centreon-Map obfuscation

Obfuscation is a procedure to "encrypt" the source code of a software to make it very difficult to analyze after having decompiled an application code.

3.2.1.4.1. ¨Operating principle

This operation consists in replacing the classes, methods and variables names by data without any sense.
Here is a very simple example:



The first image is the original code, while the second image is the decompiled code. The name of the class “Example”, the variable “data”, and the method “getData” have been changed.

Applied throughout the source code, decompiled code has no more meaning. The names of methods that were previously explicit don’t allow anymore a hacker to understand the meaning of the methods. Especially, obfuscation does not only rename classes, methods and variables, but also optimizes the code by merging some existing methods that were created for readability. And finally, it adds false methods at various locations to cover the tracks.

3.2.1.4.2. Tool choice

To implement the obfuscation of Centreon-Map, it was necessary to test different tools. The selection criteria were that it had to be open source, and embeddable in an automated building process.
3 choices were revealed to me:

- Retroguard
- yGuard
- ProGuard

The first (Retroguard) was quickly removed because the obfuscation procedure couldn’t be automated.
The second (yGuard), based on Retroguard, included the possibility of automated building. However, its configuration was particularly complex and time consuming.

Finally, the last one, Proguard, included the possibility to be integrated in an automated building tool, and its configuration was simpler than yGuard.

3.2.1.4.3. Study and configuration of the tool

We need to take some precautions in the obfuscation procedure. Indeed, it is not possible to obfuscate libraries from the different frameworks used, but Centreon-Map-Client makes calls to classes and functions of these libraries.
If the names of these methods are processed, the software will no longer work.
In addition, the class implementing interfaces from the various frameworks should not be obfuscated, under penalty of becoming classes not respecting the contract defined by the interfaces.

After many hours of testing, discovery, and critical errors, these precautions were finally included in the configuration of the tool.

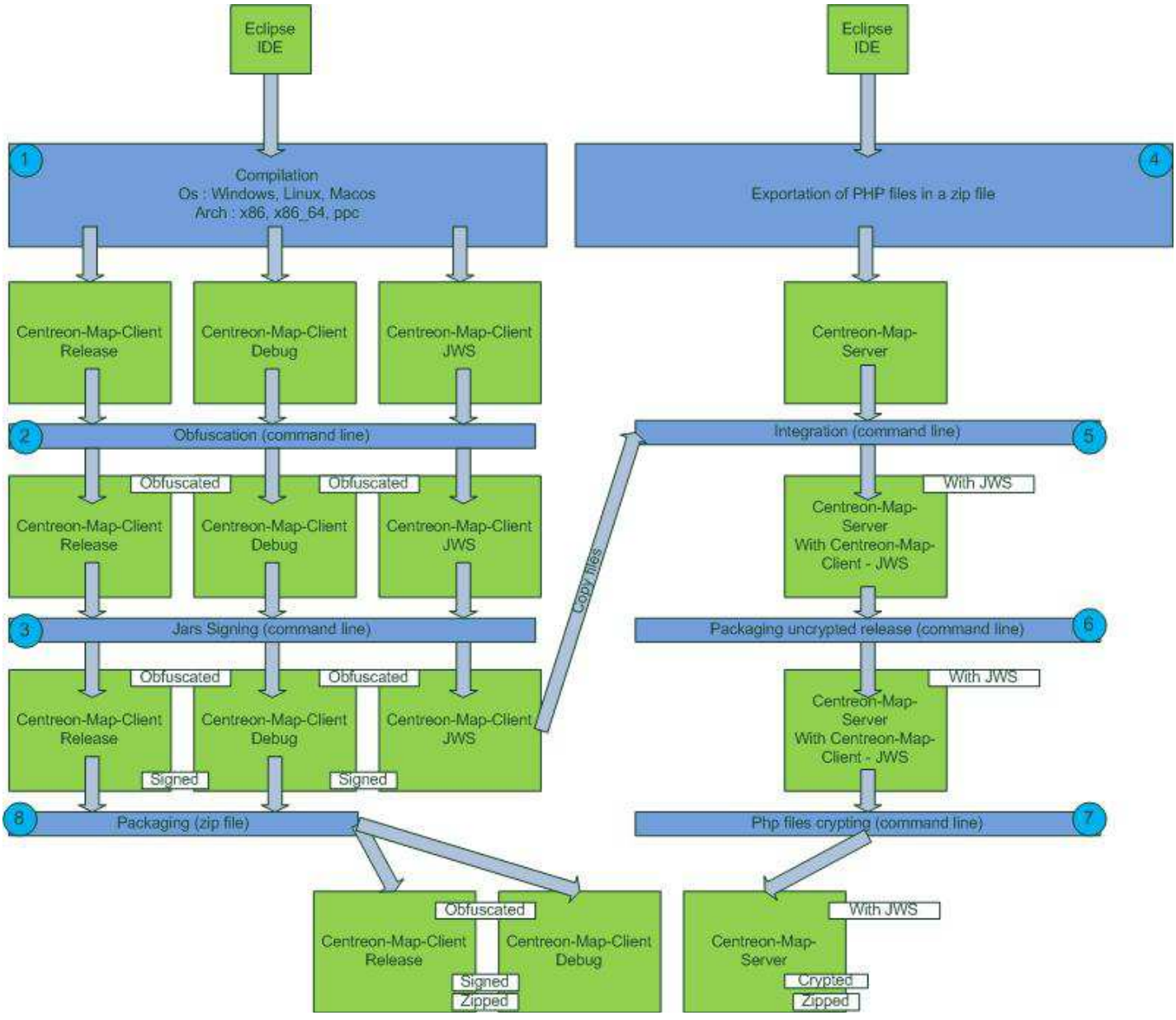
The obfuscation process was subsequently integrated into the software building process.

3.2.1.5. Paclaging tool for Centreon Map

For each new release, I am responsible for the Centreon-Map-Client & Centreon-Map-Server packaging.

3.2.1.5.1. Initial process for a complete Centreon-Map release

Complete generation of a product is time-consuming. Initially, for each product, it was necessary to perform all the steps by hand: half a day was needed. Sometimes, after the final test, some bug was identified, and a new product had to be built. The process had to be optimized.



➢ Step 1 : Compilation of Centreon-Map-Client

This step is the longest. 18 clients have to be compiled.

- 3 types of clients :
  - Release
  - Debug
  - JavaWebStart
- 6 different architectures :
  - Windows x86 (32 bits)
  - Windows x86_64 (64 bits)
  - Linux x86 (32 bits)
  - Linux x86_64 (64 bits)
  - Macos x86 (32 bits)
  - Macos PPC (PowerPC)

This step needed several hours. Different os & architecture were needed to compile the corresponding client.

- Eliminate the need for different architecture
- Eliminate human intervention during the compilation

➢ Step 2 : Obfuscation of Centreon-Map-Client

This step is also time-consuming. It is necessary to perform the obfuscation on each client, one by one.

- Integrate the process of obfuscation in the build, or automate the process.

➢ Step 3 : Signing Centreon-Map-Client

As for step 2, it is necessary to complete the signing of jars for each client, one by one.

- Integrate the process of obfuscation in the build, or automate the process.

- Step 4 : Export of Centreon-Map-Server

This step, very fast, consists in extracting from the SVN the server source code, and exports it in a zip.

- Eliminate human intervention to avoid error.

- Step 5 : Integrate Centreon-Map-Client-JWS in Centreon-Map-Server

This step, very fast, consists in copying the Centreon-Map-Client JWS files, in the Centreon-Map-Server.

- Eliminate human intervention to avoid error.

- Step 6 : Create Centreon-Map-Server zip

This step, very fast, consists to create a zip of the server, containing the client JWS.

- Eliminate human intervention to avoid error.

- Step 7 : Encryption of Centreon-Map-Server

This step is relatively fast, but lots of human commands are needed. The Centreon-Map-Server zip must be copied on a server, encrypting started, answer to some question from the encrypted script, and the encrypted server is copied back.

- Eliminate human intervention to avoid error.

- Step 8 : Create a zip of the different Centreon-Map-Client

This step, again, relatively fast, consists to zip each client. A lot's of human intervention are needed.

- Eliminate human intervention to avoid error.

The analysis of this step shows that a lot of human intervention could be eliminated, to avoid human error, and to gain time. The challenge was to automate step 1 to 8 without any human intervention.

3.2.1.5.1. Shell script

Steps 2 to 8 seemed possible to automate using shell script, so I worked in this direction.

3.2.1.5.1.1. Automating steps 2, 3 and 8

From the different compiled client, a shell script automatically launches the obfuscation and the signing of jars, then create the 12 zips containing debug and release client. It remains to upload JWS clients on SVN.

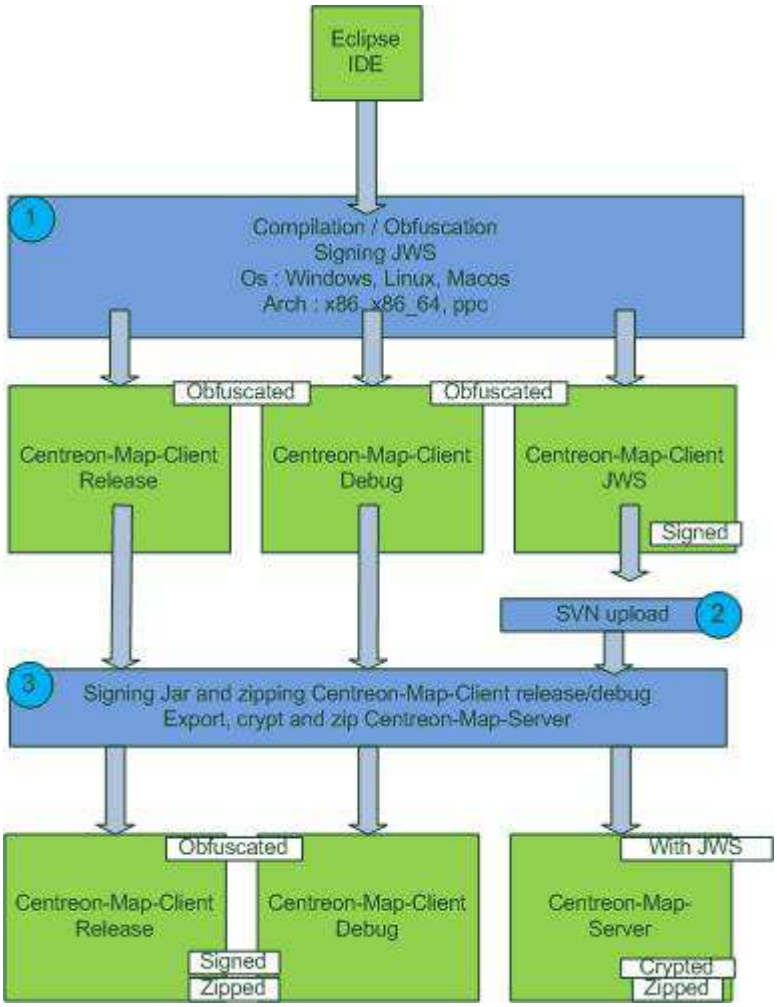
3.2.1.5.1.2. Automating steps 4, 5, 6 et 7

After the upload of the JWS clients on the SVN (manually), a shell script retrieves all the server components, including JWS client. Then, the shell script zips the files, copies to the encryption server, crypts, copies the encrypted files to the local host.
The removal of these 4 steps has created a new step: the JWS client upload on SVN.

3.2.1.5.2. Pde tools (headless build of Eclipse)

To automate Step 1, I have done some research, around the headless build of Eclipse.
Unfortunately, after many tests, some bugs were found in Eclipse. It is currently impossible to generate a JWS client with Eclipse, in command line. I had to wait a new Eclipse release, correcting the bug (long date bug …). Nevertheless, the debug and release client can be generated in command line.
During those tests, I found a way to integrate the obfuscation in the build process (in headless build or in Eclipse). I also found a way to sign the JWS clients jars, during the building (with graphical Eclipse).
Finally, I found an Eclipse extension, making possible to generate product for a different architecture with only one OS, reducing the 18 steps (for the 18 different clients) to only 3 steps (release, debug, JWS).
The compilation time has been reduced from many hours to 1 hour, with only 3 human interventions.

3.2.1.5.3. New process



The new process has eliminated many steps and human error risks. Tasks are performed much quicker (with a better reactivity than a human being). A complete package building requires now less than 2 hours

A complete package building can now be summed up in 3 stages:
Step 1 :
Various client compilation, obfuscation, and signature of the jars of Centreon-Map-Client-JWS.

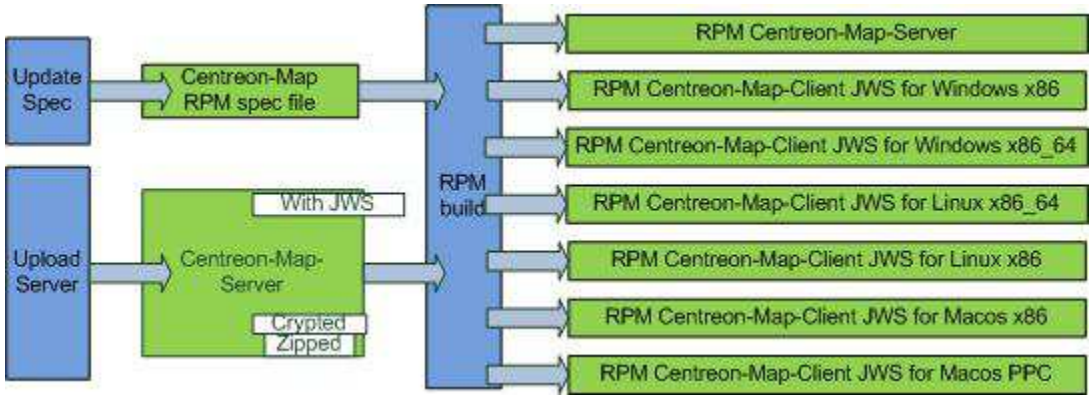
Step 2 :
JWS client upload on SVN.

Step 3 :
Shell script:
Zip release and debug client.
Copy Centreon-Map-Server from SVN, and create a zip. (Including Centreon-Map-Client-JWS clients)
Copy zip on encrypt server, encrypt, and fetch the encrypted server files.
Centralize every zip in one location. (Release and debug clients, and server).

3.2.1.5.4. RPM build

A new step has been added recently. Now we perform RPM packages, for Centreon and our proprietary module, for the Centreon-Pro solution.

An external consultant has installed a solution to create RPM for different architectures. (CentOs, RedHat, etc…).
This solution needs specifications files describing the RPM to build.
I created specifications cutting the encrypted server files into different RPM. The first one contains the server file only, and 6 others RPM contains the JWS client files, for each client architecture. Each RPM exists for the different server architecture.
This task is not yet integrated in my building process.



3.2.1.5.5. Hudson

I have installed a Hudson server, for a complete building process, with a friendly user interface. It is able to detect automatically release tag on the SVN, and start an associated build.
Currently, the step 1 of the building process prevents me to fully use it. Once the step 1 is removed, the step 2 may also be automated.

Nevertheless, the goal is to be able to generate a full build (including RPM) in 1 click, from Hudson, deploy the build on a test server, and start automated tests.
The platform is ready, but there is still some work to do to be fully functional.

3.2.1.6. Functional test

During development, it is necessary to check that the application works. However, the final tests should be undertaken to check that everything works properly, and there is no bug. These tests are performed on different RC before the final version (which is also tested).

3.2.1.6.1. Manually

The tests are manually done. There is no tool that can check that the software behavior is correct.
These tests are usually carried out by several people. I set up a server for testers (myself included) equipped with Centreon, Nagios, Centreon-Map-Server, and the different versions of Centreon-Map-Client.
Everybody tests every functionality. When a bug is discovered, it is reported in our collaboration tool (Redmine) and corrected immediately upon discovery.

3.2.1.6.2. Test booklet

Despite all the tests, sometimes a bug is forgotten. For this, on the initiative of my technical director, we have started to develop a test booklet. Julien Mathis (CTO) has written the beginning of the test booklet, and I finished it. It will be available for the next version of Centreon-Map. It will permit to tests and validate all the functionality (old and new) without forgetting anything.

3.2.1.6.3. Automated test

The automated testing introduction would save valuable time during the tests. All tests can't be automated, however, but the entire communication between client and server could be. To do this, I thought about different possibilities. The most interesting would be to incorporate a hidden menu (such as the debug console) in the current client, to initiate automatic actions, such as the creation of containers, adding resources, creating link... That does not test the interfaces, but would nevertheless remove some tests, that are very long (and repetitive).
This would avoid creating another client dedicated to test, and who should be maintained in parallel with the client Centreon-Map-Client.
These automated tests have not yet been created.

3.2.1.7. Support

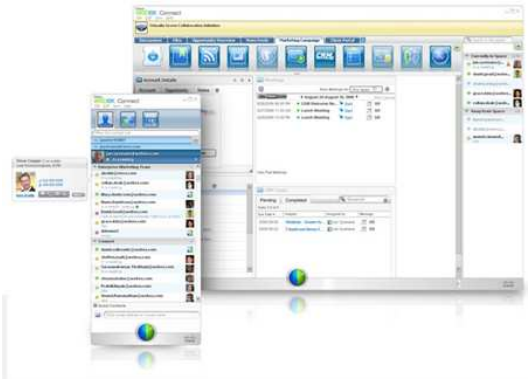
We receive a notification by mail for each ticket. I only resolve Centreon-Map tickets. Other tickets about Centreon are resolved by the rest of team. The ticketing system is managed by Kayako.
There are 3 levels of support: L1, L2 and L3.
Support L1, solve the simplest problems. If it is not able to solve problems, he diagnoses and escalates it to support L2.
L2 support solves problems from a list of procedure established by the support L3. If he cannot solve the problem, it is escalated to L3.
L3 support solves the most complex problems. Generally, it is problems that have never been encountered before. It may include correcting bugs or improving performance of the software.
I am supporting L3 within MERETHIS, but I also solve L1 and L2 problems.

3.2.1.7.1. Kayako

This tool manages the tickets issued by our clients, and notifies us by email for each new ticket.

Most of the time, I solve L1 & L2 tickets. This is a minor problem such as a bad configuration of resources in Centreon (impacting Centreon-Map), or misunderstanding of a functionality.
The more complex issues (L3) are often resolved through WebEx.

3.2.1.7.2. WebEx



This tool allows us to take remote control of the PC clients. The advantage is that it works on main Oss (Windows & Linux), and is able to cross complex networks filtering VPN connections.

Here are some examples of L3 tickets resolved by WebEx:

- Migration issues for non compliance of the migration procedure

The user had not respected the migration process from one version to another. Centreon-Map was unusable. For this it was necessary to perform the migration step by step, directly in the database, and determine the blocking step. It was a double index, blocking the migration. The problem solved, the rest of the migration was done properly.

- Incorrect handling of user

After using forbidden characters, the xml passed between the server and the client was corrupted. Views were impossible to load. A research in the database was used to identify the corrupted field, and correct it.
In some cases, support cannot solve the user's problems. For this it is necessary to move directly to the customer, as part of a consulting mission.

3.2.1.8. Consulting

I did few consulting missions; this task was rather reserved to consultants better trained for this type of work. However, there were several occasions when I went to set up the mapping network of the company.
Missions can be very fast, if the customer has prepared all the needed documents for carrying out the mission. Unfortunately, often, it is not the case.

3.2.1.8.1. Listening the needs

It is necessary to listen to the needs of the customer. Typically, customers do not know exactly what they want. It must first demonstrate them the advanced possibilities of Centreon-Map. The most common need is to create a network map, but they can also be of business type.

3.2.1.8.2. Autonomy

Once the needs were identified, I began the mapping building. Often some information was missing or inconsistencies in the documentation present. I had to go fishing information, to the right person. Sometimes they were not available and it was necessary to find the information by myself, which made the mission slower.

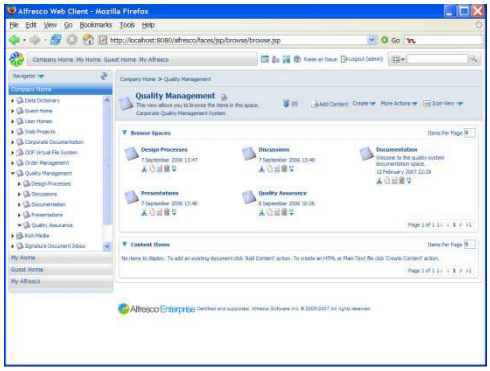
3.2.1.8.3. Evolution

Finally, the greatest interest as Centreon-Map project manager is to exploit the tool in real condition, and find its weaknesses, specifically in terms of ergonomics. When creating software, we end by not seeing its weaknesses.
Users can also ask for things that are not possible in the existing versions, which generally suggest new ideas, and are quickly integrated in the roadmap for future versions. For example, being able to make restoration points, or run ssh commands directly from Centreon-Map.

3.2.1.9. Documentation

I am responsible for maintaining Centreon-Map documentation. For each new version, major or minor, the new features must be integrated. The documentations are written with open office, and Gimp for graphics.
When the documentation is finished, it must be organized and converted into PDF. To do this, we have a document management tool.

3.2.1.9.1. Alfresco



Alfresco is a document management tool. It enables us to organize all the documentation available and turn them into PDF.

3.2.1.9.2. Création de règle de génération de PDF

For each new documentation, I must organize Alfresco, and create rules for generating PDF associated to the news storage space.
Sometime, I advise other employee on some functionality of this tool.

3.2.1.9.3. Documentation d'exploitation autre que Centreon-Map

I must also make operations documentation, enabling the monitoring implementation of a Tomcat and Jboss server using JMX. Unfortunately, due to a lack of time, these documents have not yet been realized this time.

3.2.1.10. Centreon BI

This project began one year ago. In its first version, the scheduling was run by cron task from Linux systems. This system set some problems for changing date generation. M.Sugumaran decided to rewrite a scheduler in java. I had to perform this task.

First, we have considered the scheduler architecture. Some points had to be improved. The first architecture suffered defects that unnecessarily complicated its development. We have redesigned the architecture.

Thanks to my experience over the years in multithreaded environments, the development was fast. Tests revealed no malfunction.

Finally, I have helped M.Sugumaran to integrate his reporting engine in the scheduler. Today, final tests are underway.

3.2.2. Analyze of the resultants

Analysis of the various issues above highlights some problems.

The main problem is the release of Centreon-Map is often after the scheduled date. Development times are good, and respect the initial deadlines. However the test step is very time-consuming, and sometimes some bugs are not immediately discovered. The setting of a test booklet will help to be more effective in the future release.

Before the test step, it is necessary to generate a complete package, and set up a test server. This operation takes time and sometimes needs to be done repeatedly, to eradicate all bugs.
Improved packaging process has divided by 3 or 4 the time generation. Moreover, it is now impossible to have errors because of human manipulation; a program never makes a mistake. This operation should have been done earlier to save time during testing step. Nevertheless, the process is not fully automatic. This is a point to improve in the future.

Now Centreon-Map-Client is protected against reverse engineering and hackers and its complete integration in the build process is a success.

Concerning architecture and developments, overall, everything goes well. Methodology acquired over the years is now good. Developments are effective, and the architecture is powerful. Now we spend more time thinking about the product architecture.

4. General conclusion

MERETHIS trusted me for almost 3 years. Thanks to my work on Centreon-Map, I gained experience on the Java language. My expertise scope has also increased. I initially worked only on development (and a little architecture), under the direction of my CTO and CEO. Today I have become very independent. I've been thinking about new product features, I established roadmaps, estimated development time and cared about improving processes around the product.

Centreon-Map is a pilot project for MERETHIS. Indeed as it is the first module of the society, we met problems in the end, as the project growed. These errors may not be reproduced on other projects of the company. The setting of a test booklet and automating the building has significantly improved production times. One point remains, it would be very productive to automate the simplest test.

Today, there are still many doodles in Centreon-Map to bring even more satisfaction to the customers.

5. Bibliography & useful websites

5.1. Bibliography

- "Eclipse 3 pour les développeurs Java"

  « Développez des plug-ins et des applications « client riche »
  Author: Berthold Daum, translated from German by Claude Raimond.
  Brand: Dunod
  Collection: InfoPro

5.2. Useful websites

- The official Eclipse website: www.eclipse.org, for its Eclipse IDE and framework required by the project Centreon-Map and its excellent online API..
- The French community of developer : www.developpez.com
- A lot's of small technical blogs, containing interesting analysis, or code exemple.

6. Glossary

These definitions are summarized from Wikipedia, when available.

Name	Definition
API	**Application Programming Interface** (**API**) is a documentation that defines the ways by which an application program may communicate with framework. It contains calling conventions used by programmer. Ref : http://en.wikipedia.org/wiki/Api
CCJPL	Code Conventions for the Java Programming Language. Ref : http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html
Centreon	**Centreon** is a frontend for Nagios. It provides a lot's of functionality not available in Nagios, and simplify its configuration.
Centreon-BAM	**Centreon Business Activity Monitoring** is an extension for Centreon, providing the ability to aggregate the states of different checkpoints.
Centreon-BI	**Centreon Business Intelligence** is an extension for Centreon, providing the ability to exploit data collected by Centreon, and generate custom made dashboards.
Centreon-Map	**Centreon Map** is an extension for Centreon, providing the ability to create advanced cartography with the resources issued from Centreon.
Drag-and-Drop	**DnD** is the action of clicking on a object, drag it in a different location or another object, and drop it, to invoke an actions, like moving object or associating 2 objects. Ref : http://en.wikipedia.org/wiki/Drag-and-drop
DRAW2D	**DRAW2D** is framework developed for the Eclipse platform. It is the view part of the **GEF** framework.
Framework	A **framework** is an abstraction in which common code providing generic functionality is provided in a set of libraries. Ref : http://en.wikipedia.org/wiki/Software_framework
GEF	**Graphical Editing Framework** (GEF) is a framework that was developed for the Eclipse platform. It is a Model-view-controller (MVC) concept. It is known as a framework with a very steep learning curve. Ref : http://en.wikipedia.org/wiki/Graphical_Editing_Framework
Hudson	**Hudson** is a continuous integration tool written in java. It can execute Ant or Maven task as well as arbitrary shell script. Ref : http://en.wikipedia.org/wiki/Hudson_%28software%29
IDE	An **integrated development environment** (IDE) also known as integrated design environment or integrated debugging environment is a software application that provides comprehensive facilities to computer programmers for software development. Ref: http://en.wikipedia.org/wiki/Ide

Java	**Java** is a programming language. It derives much from C and C++, but has a simpler object model. It can be run on any **JVM** regardless of computer architecture. Ref : http://en.wikipedia.org/wiki/Java_language
Jboss	**Jboss** is an application server based on Java. It is cross-platform, usable on any operating system that Java supports. Ref : http://en.wikipedia.org/wiki/Jboss
JDK	The **Java Development Kit** (**JDK**) is a set of tool for Java, to compile, generate API, debug, analyze execution etc… It also contains an embedded JRE. Ref : http://en.wikipedia.org/wiki/JDK
JFACE	**JFACE** is a framework developed for the Eclipse platform. It is a widget container, providing classes for handling common UI programming tasks. Ref : http://en.wikipedia.org/wiki/JFace
JMX	**Java Management Extension (JMX)** is java technology permitting to monitor and manage Java application. Ref : http://en.wikipedia.org/wiki/JMX
JRE	**Java Runtime Environment (JRE)**. See JVM. Ref : http://en.wikipedia.org/wiki/JRE#Execution_environment
JVM	A **Java Virtual Machine (JVM)** is a set of computer software creating a virtual computer, where the Java byte code is executed. There is JVM for most of the existing architecture, permitting to launch Java on any system. Ref : http://en.wikipedia.org/wiki/Java_virtual_machine
JWS	**Java Web Start (JWS)** is a framework developed by Sun that allows users to start application software directly from the internet, using a browser. Ref : http://en.wikipedia.org/wiki/Java_web_start
Obfuscation	**Obfuscated code** is a source code that has been made difficult to understand. It's a form of security through obscurity, to deter reverse engineering. Ref : http://en.wikipedia.org/wiki/Obfuscated_code
Oreon	Old name for **Centreon**. See **Centreon**.
Proxy	A **Proxy server** is a server that acts as go-between for requests from clients seeking resources from other servers. It's common in big company, with a complex network. Ref : http://en.wikipedia.org/wiki/Proxy_server
Roadmap	A **roadmap** is a list of functionality for a release. It's a list of objectives to accomplish for the release.
RCP	**Rich Client Platform (RCP)** is software consisting of the following components: core, bundling framework, portable widget toolkit, text handling & editor, workbench (views, editors, perspective, and wizards), data biding, and update manager. Ref : http://en.wikipedia.org/wiki/Rich_Client_Platform
RPM	A **RPM** is a package containing software, and instruction to automatically install it. It also contain dependences to other RPM, needed by a RPM. Ref : http://en.wikipedia.org/wiki/RPM_Package_Manager

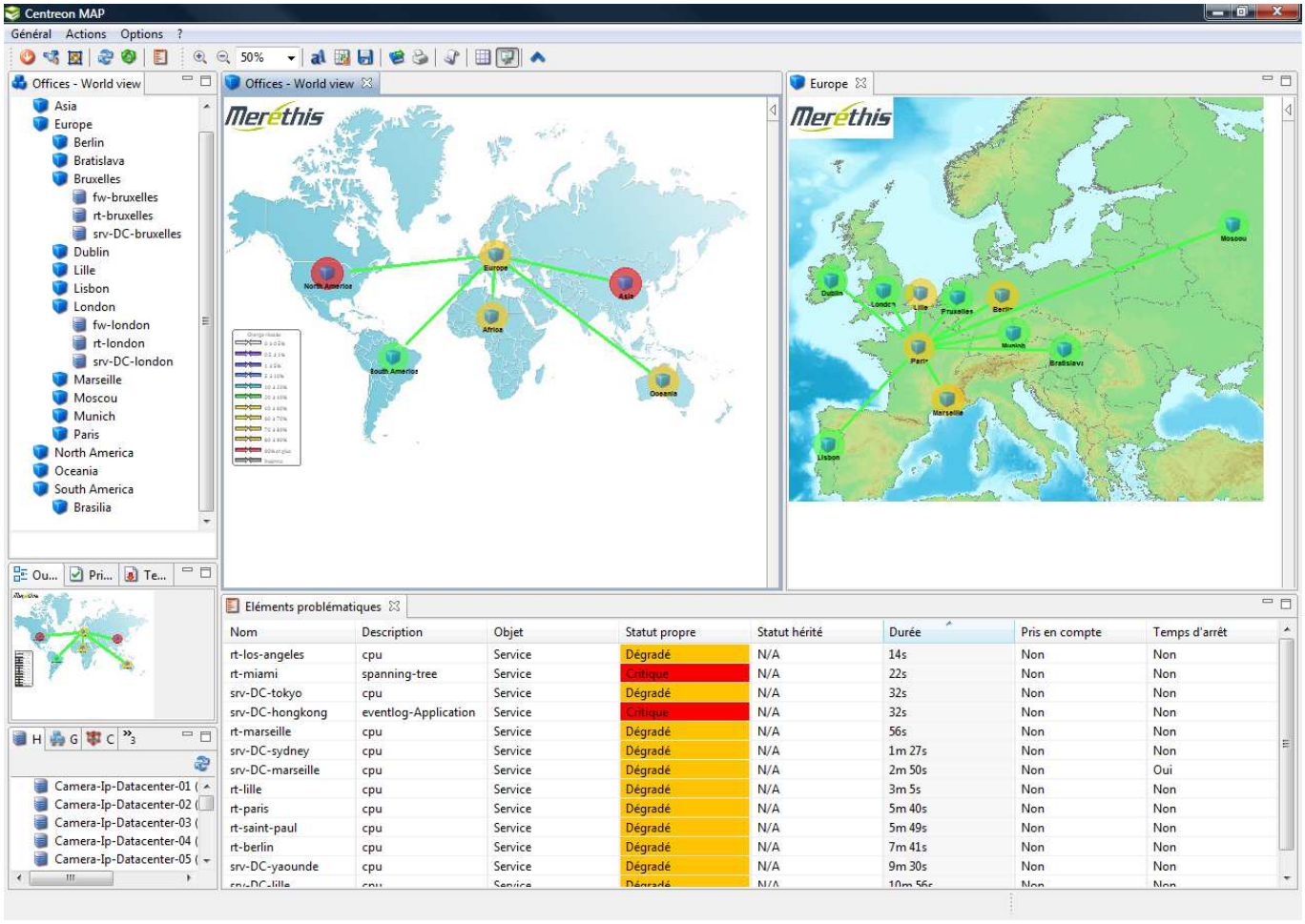
Socks	A **sock** is like a **proxy**, but permit to connect to any port, oppositely to proxy that can only connect to http and ftp protocol. Ref : http://en.wikipedia.org/wiki/SOCKS
SVN	**Subversion (SVN)** is a version control system. It is used to maintain current and historical versions of files such as source code, web pages and documentation. Ref : http://en.wikipedia.org/wiki/Subversion_%28software%29
SWT	**Standard Widget Toolkit (SWT)** is a graphical widget toolkit for Java. It is maintainer by the Eclipse foundation, in tandem with the Eclipse IDE. Ref : http://en.wikipedia.org/wiki/Standard_Widget_Toolkit
TAG (svn)	A **Tag** is a snapshot of a version on the SVN. It is used to keep a history of software, when a version is released, to eventually modify it for bugfix. Ref : http://en.wikipedia.org/wiki/Subversion_%28software%29
Tomcat	Apache Tomcat is an application server like Jboss. Ref : http://en.wikipedia.org/wiki/Apache_tomcat
Thread	A **thread** is a software task. Multiple tasks can be executed in parallel in multithreaded environment. This type of programming is the most complex, because every resource must be protected to avoid parallel reading or writing of a variable leading to corrupted result. Ref : http://en.wikipedia.org/wiki/Thread_%28computer_science%29
VPN	A **Virtual private network** is a virtual connection established between 2 computers over internet, simulating a local connection. It permit to access to computer not visible on the internet. Ref : http://en.wikipedia.org/wiki/VPN
XML	**Extensible Markup Language (XML)** is a set of rules for encoding or transfer data. The goal is to have very structured contents. Ref : http://en.wikipedia.org/wiki/XML
ZEST	**ZEST** is a framework developed for the Eclipse platform. It contains a set of libraries permitting to display object with different appearance. It is not well integrated with the other framework.
Zip	A **zip** file is a compressed file containing files and folders.

7. Appendices

7.1. Centreon-Map

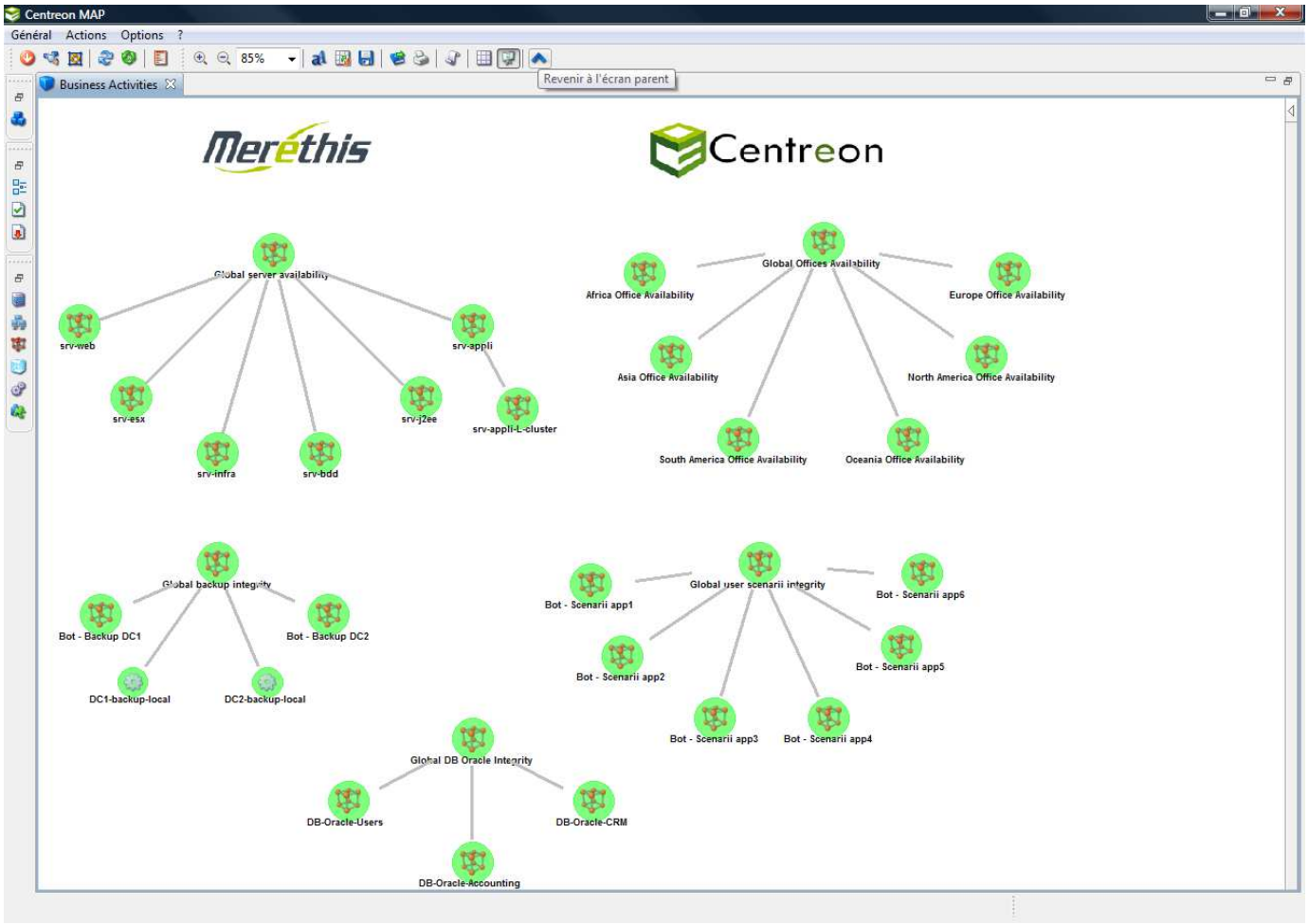
Here some screenshot of different functionality of Centreon-Map.

7.1.1. A geographical view



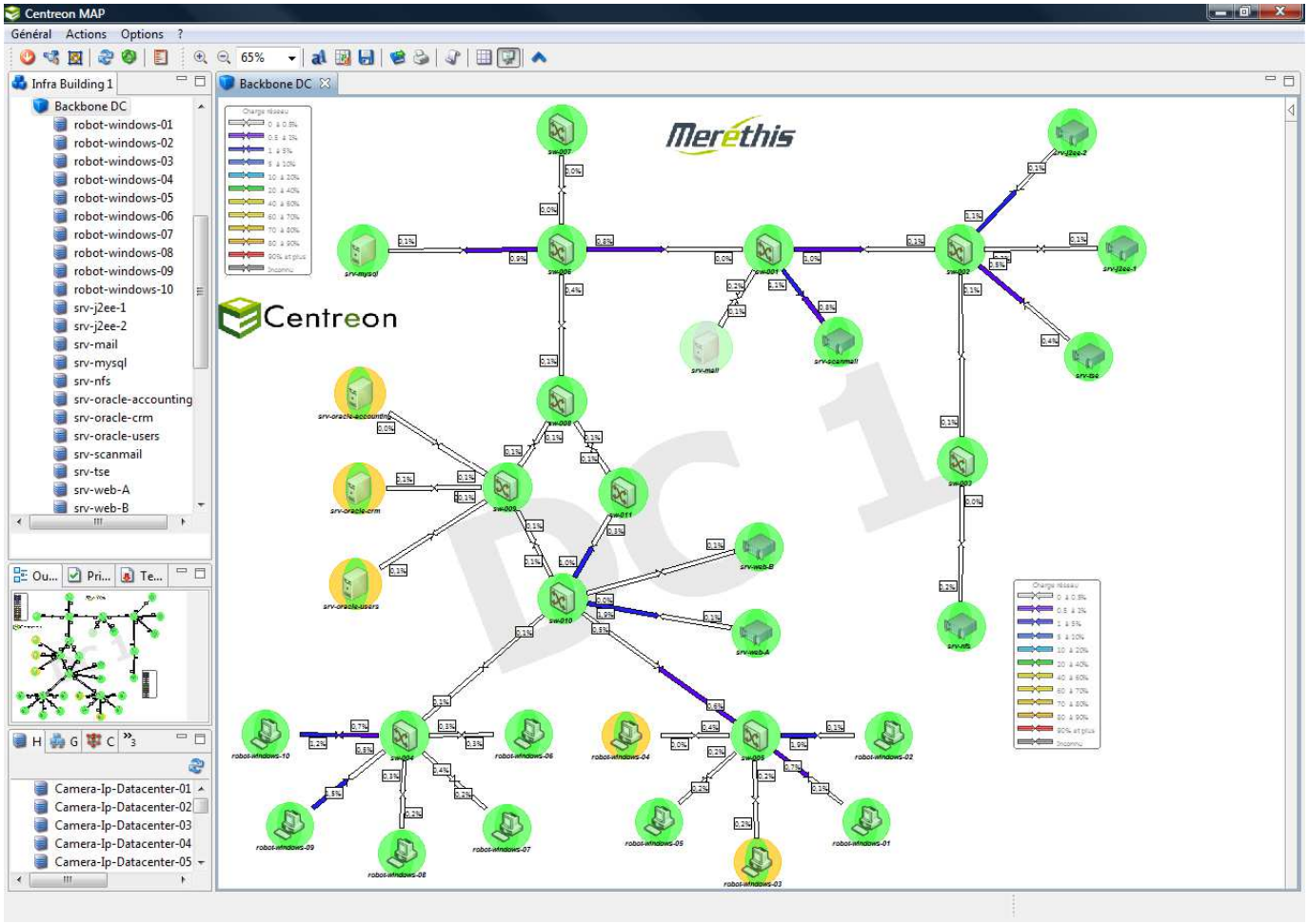
A geographical view, with 2 views level opened

7.1.2. A business view



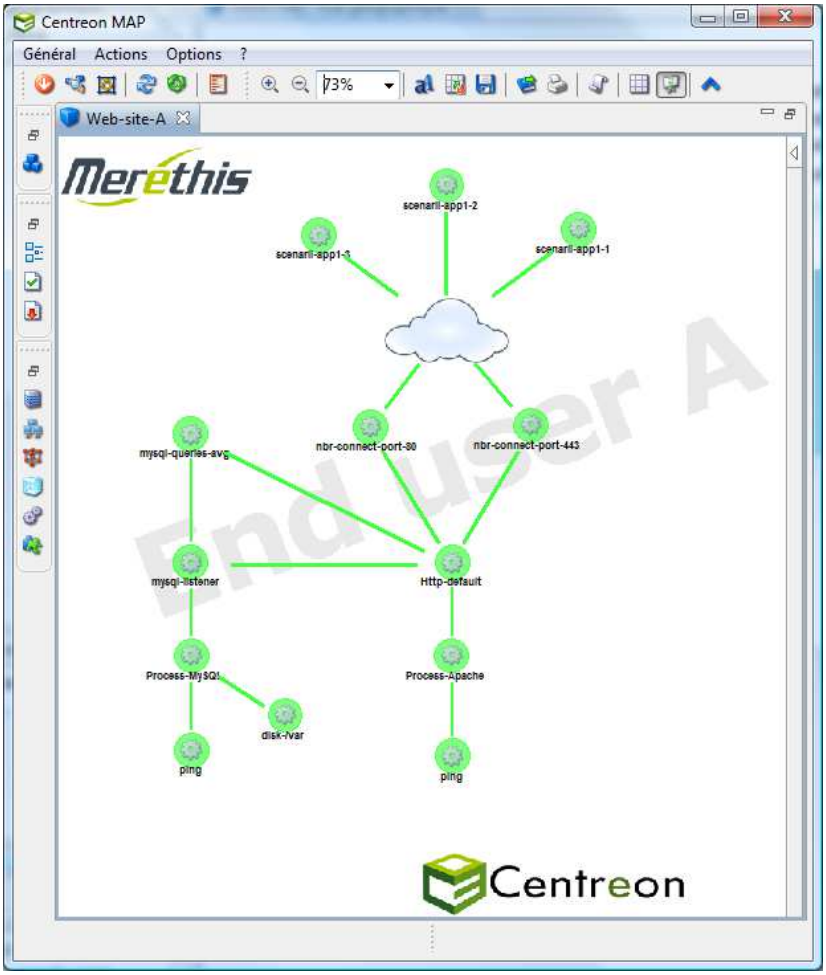
Centreon-BAM object, embedded in Centreon-Map, representing the general availability of web site

7.1.3. A network view



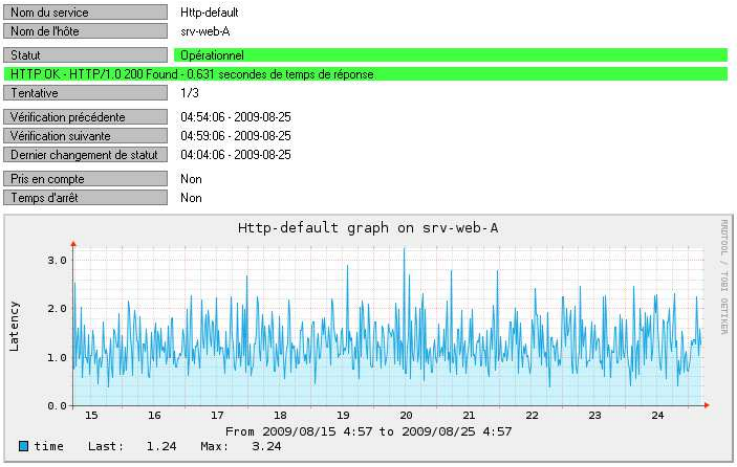
A network view with load link

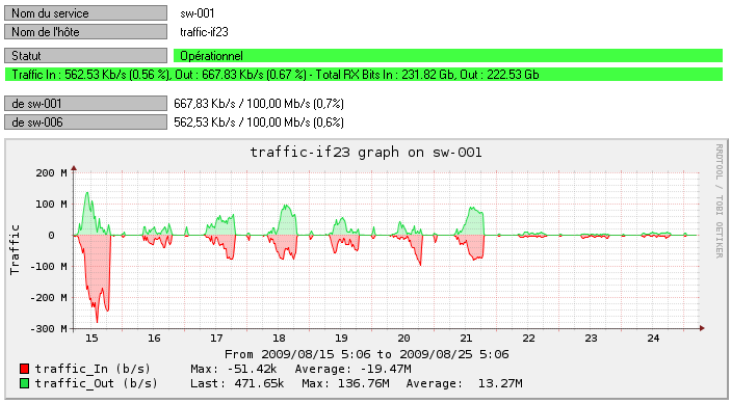
7.1.4. An other business view



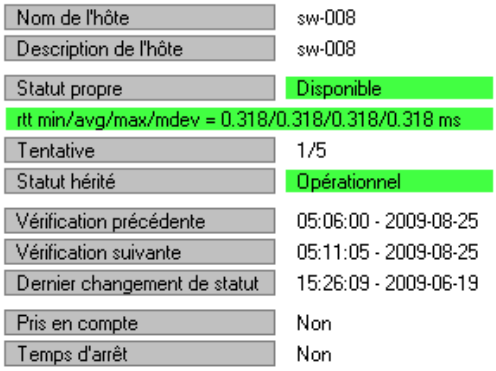
The availability of different components of a web server

7.1.5. Tooltips



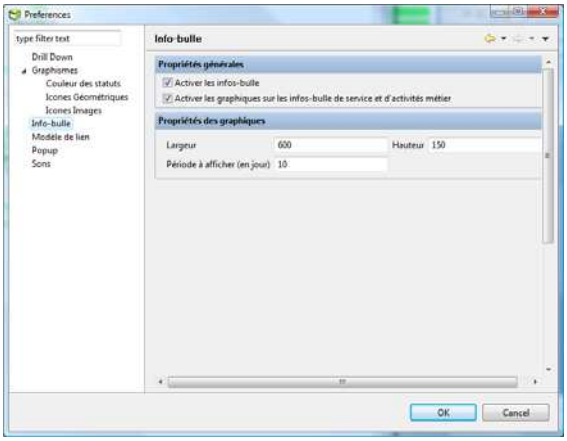


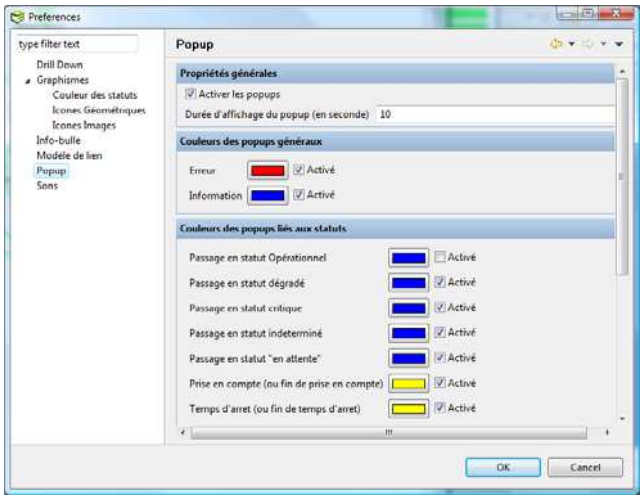
A service & load link tooltip



Host tooltip

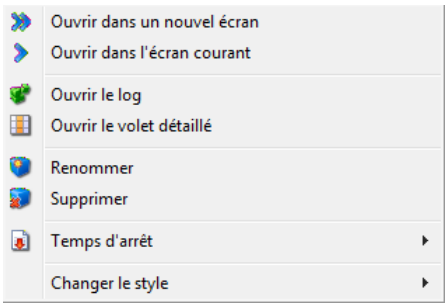
7.1.6. Preferences windows

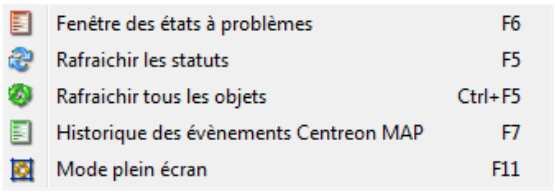




Some windows to configure Centreon-Map-Client options

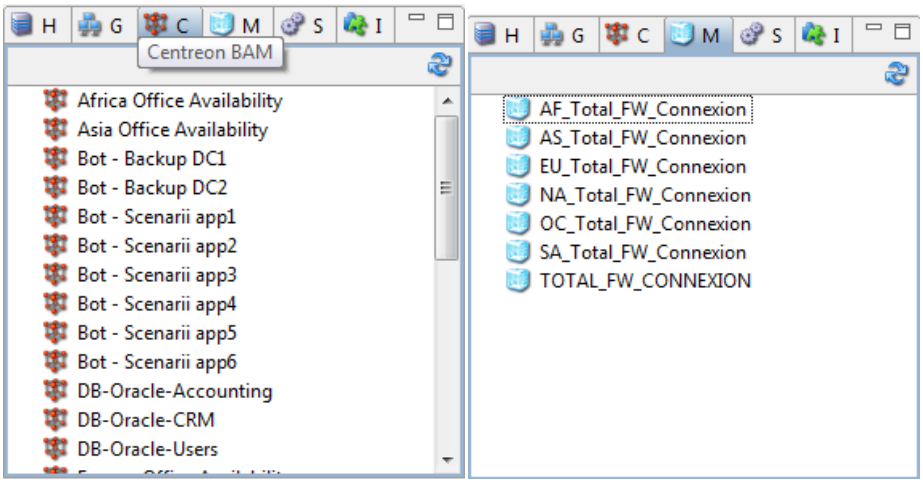
7.1.7. Toolbar & menu





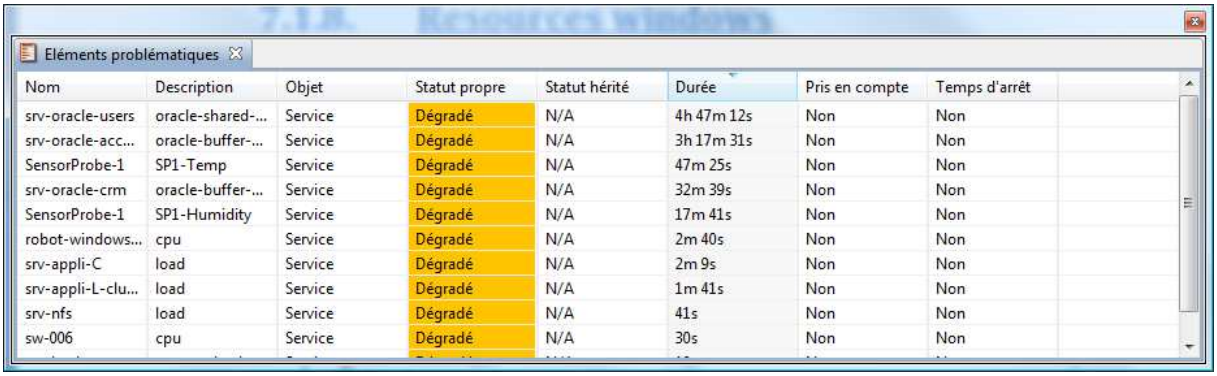
Menu & toolbar for different actions

7.1.8. Resources windows



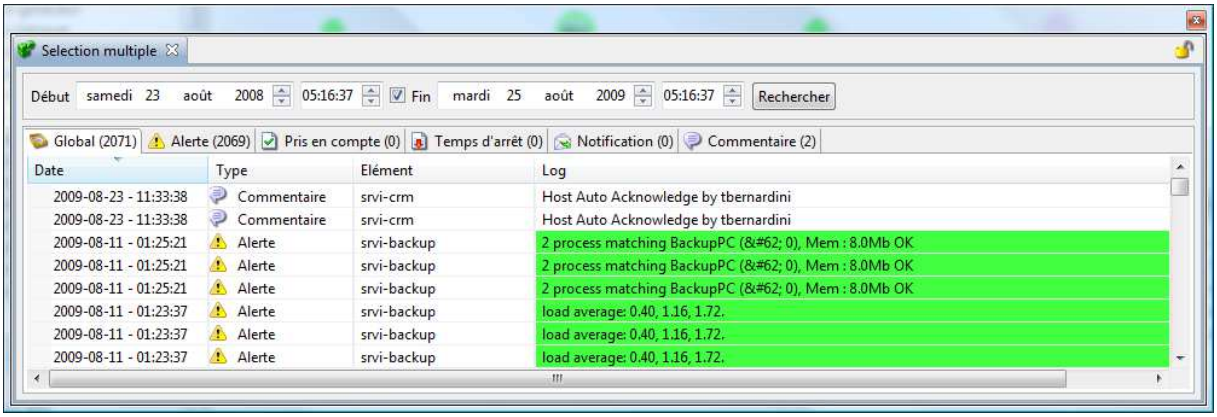
A list of Centreon-Bam object and a list of metaservice

7.1.9. Object with dysfunction



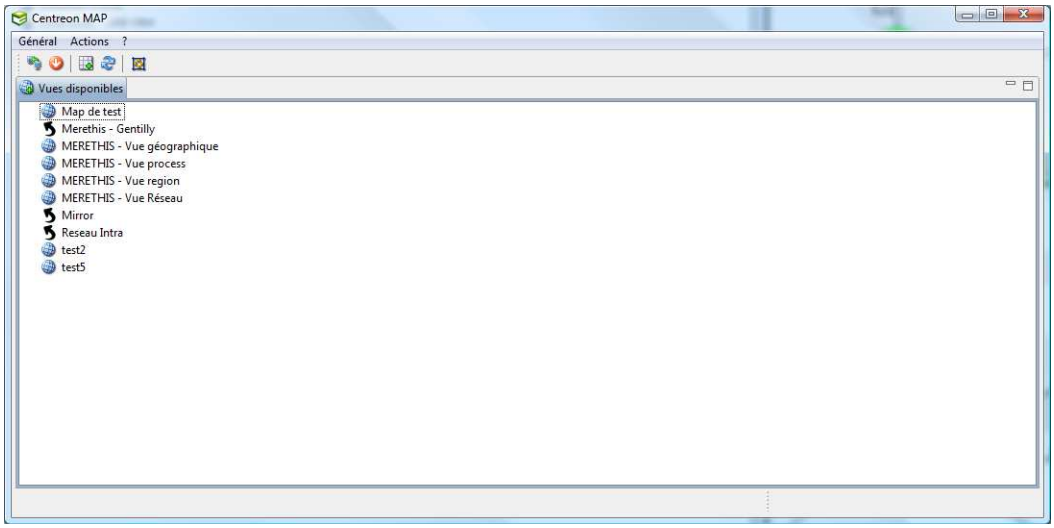
Windows listing object with malfunction

7.1.10. Logs



Windows to search in the history of an object

7.1.1. View selection



The window to select view