The aim of this page is being a **summary**.

See the attached part of the memory for more detail.

Project context

Explained in the first chapter of the document.

Planning

The original planning was:

- GEP
- Requirement analysis, architecture and debugging
- Iteration 1: Pseudo-boolean minimization
- Iteration 2: Timeout
- **(Optional) Iteration 3: Multi-threading**
- **Finalization stage**

The current planning is:

- GEP
- Requirement analysis, architecture and debugging
- Iteration 1: Pseudo-boolean minimization
- Iteration 2: Timeout
- **Finalization stage**
- **(Optional) Iteration 3: Multi-threading**

Once the second iteration was finished, instead of starting the third one, the finalization stage was started. The decision was made in order to guarantee that the required stages were finished. Because the third iteration is optional, it has been delayed until the documentation is finished and will be done if there is enough time.

The project is expected to be finished without delays according to the planning.

Methodology and rigour

The methodology followed during the development has been Agile with TDD (Test Driven Development).

TDD implied that before writing code all the requirements had been defined i.e the behaviour of that component is fully defined (but not closed to extension). Then, a set of tests are written for each requirement/method and then the

For each requirement/method, a set of tests were written and after that, the implementation starts. The only goal of the implementation is pass tests with the minimum functional code. When the tests pass, a refactor step is done in order to optimize the code.

With TDD, the written code is the only one necessary which helps to simplify its logic. Also, Category-Partition testing has been used, especially in the search strategy implementation, to identify relevant values for the implementation and tests special cases.

Altogether, in particular the test suite, guarantees that the software has been developed with quality, rigour and that works.

It is important to remark that there is no perfect test suite because not all possible combinations of input values can be tested.

Alternatives analysis

This section is explained in more detail in the Development section of the memory.

In a nutshell, for the first iteration there were two alternatives: use PBLib or implement all the functionalities from scratch.

Use PBLib was the selected alternative because it would have taken too much time to implement all the functionalities and the quality would be lowered as PBLib has been developed by experts.

For the second iteration, there were also two alternatives about how to approach the timeout: using processes or threads.

The behaviour wanted was that one part of the program counts for the timeout while the other is executing the solver.

Finally, thread was the selected alternative because creating a thread is cheaper than creating a process.

Knowledge integration

Until now, the following knowledge has been integrated:

- Computer science due to the project’s topic and context, search algorithms and methods time complexity.
- Software engineering because TDD has been followed for the implementation and there has been a special emphasis in architecture design to make code reusable and easy to add extensions.
- Operative systems and parallelism have been applied for the second iteration to implement the timeout.

Laws and regulations

For this project, no laws or regulations are applied.