technische universität dortmund

Programming an Application for Vehicle Prototype Planning

Master Thesis

Raúl Armero Almazán

NIF: 45788384F

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona (Spain)

08.07.2011

Tutors: Prof. Dr.-Ing. Uwe Clausen

Dipl.-Inform. Christian Tesch

Institut für Transportlogistik
Prof. Dr.-Ing. Uwe Clausen

Project summary

This project consists on programming an application, which is able to compute a list of Prototypes to be built by working with a list of Tests and given component constraints.

The created “Prototype Planning” application generates different solutions throughout three different programming methods (each one of them was improved from the previous one). It also has three sub-applications integrated to test these versions with different input times. The application is written in C# using “Microsoft Visual Studio 2010 Ultimate”.

The lists of Prototypes generated with version 1 gives a correct solution but the solution is not satisfactory, because there are many Prototypes that have a low duration at the end of the list. This problem is solved by creating version 2 with the implementation of a new programming method. As a result, the solution by version 2 is rather accurate to what the vehicle manufacturer needs. In version 3, not only the Prototype components are chosen, but also a time schedule is implemented. Despite this version giving a higher number of Prototypes, it is expected to find this result because the time limit of each Prototype was increasing. Finally, in version 4, a new hypothesis is computed to reduce the amount of Prototypes and to create a programming method more accurate to the reality.

Prototype planning and scheduling is a complex field, which much research have been based on. Consequently, this application is far from producing the optimal solution to the problem, but by adding more improvements to the application, the final aim of obtaining the definitive application is possible.

Table of Contents

1. Glossary

The following words are used frequently in the report:

- **Test**

A “Test” is a trial procedure or experiment that has to be performed to verify that the vehicle model meets all required technical specifications. As the word is considered important within the whole research, is written with capital letter.

- **Test Duration**

The “Test duration” is the time that a Test needs to be performed. The duration consists on prepare the Test, perform it and obtaining the results.

- **Vehicle Variant**

A “vehicle variant” is a combination of different components that theoretically could be built later as a Prototype.

- **Prototype**

A “Prototype” is a real vehicle variant built by hand where some Tests will be applied. As the word is considered important within the whole research, is written with capital letter.

- **Constraint**

A “constraint” is a compulsory component that all Tests need to be performed. Therefore, if the Prototype is allocating one Test, the Prototype is assimilating its Test constraints.

- **Requirement**

A “requirement” is each component variant that a Prototype has. The requirements of a Prototype are the constraints of the applied Tests in that Prototype.

- **Start of Production (SOP)**

The “start of production” is the date where all Tests have to be finished because the vehicle manufacturer is starting the production of the vehicle model.

- **Sub-Application**

A “sub-application” is each application integrated in the created program. This denomination is used to differentiate the “Prototype Planning” application from all the running methods or versions.

- **Version**

A “version” is each programming method created in the application. A “version” is also here a “sub-application” because of being performed under the main application.

2. Preface

The performance of this research is not only based on personal motivations, but to also continue with previous studies done by workers, students and researchers at “Technische Universität Dortmund”. In this research, the combination of my personal interests and institutional objectives in the planning of Prototypes made this possible.

2.1. Project Origin

This project is a part of the research that is being conducted by the “Institute of Transportation and Logistics (ITL)” on Prototype planning at TU Dortmund. In January 2008, Dacheng Chen performed a theoretical Prototype planning based on mathematics and heuristic methods. In addition, the “Verkehrssysteme und Logistik (VSL)” department of TU Dortmund was a part of a collaboration with a truck manufacturer and a car manufacturer, with the goal of giving an approximate number of Prototypes that will be built. Before these projects, the VSL department worked on the Prototype planning in collaboration with more car manufacturers. This project is the first programming solution given to the Prototype planning research after all the previous theoretical solutions.

2.2. Motivations

During an internship at the car manufacturer SEAT, I was up close and personal with the design and development of a new car model. By the time I joined, this new car model was in the Prototype phase and because I was incorporated so closely to the prototyping, I was able to understand the significance to why good prototyping is important for car manufacturers. Because of this, I knew the importance (regarding to budget and technical procedure) of defining the Prototypes, allocating the correct Tests and being on time with the schedule.

3. Introduction

In the automobile industry, the prototyping manufacturing process is a large percentage of the development costs of a new vehicle model. These Prototypes are a corner stone in the vehicle design because through the Tests, which are applied to these Prototypes, the company must certify that everything is working as expected from the computer simulation.

These Prototypes are handmade and their costs can be up to 1 Million Euro each. Because of these high costs for Prototypes, the company created a goal to reduce the amount of Prototypes made. This is significant, because the decision to build one less Prototype has a big impact on the total value of the designing process. However, to approve the design of the vehicle and to start the production, several Tests must be applied. Nevertheless, the Prototype planning still has the goal of completing all of its Tests with the minimum amount of Prototypes. This is why most Prototypes must be used for several Tests; however, the Tests have to be arranged in an appropriate order, so the Prototypes can be used multiple times.

4.1. Project Aim and Objectives

The main goal of this project is to generate an application, which provides the user a solution with the number of Prototypes to build and their requirements. According to the number of Tests, the time in which all Tests should be performed and their component constraints, the number of Prototypes will be placed within a high range of values. Obviously, we can only allocate a Test to a Prototype if this Prototype satisfies the component requirements of the Test. For instance, while a Prototype with a gasoline engine is not suited to execute a diesel engine Test, it does not matter which kind of engine is used to perform a brake system Test. Therefore, the “Prototype Planning” application must select appropriate variants of Prototypes so that all Tests can be assigned. Consequently, the aim is not only to provide a solution, but also to have the least amount of Prototypes as possible.

4.2. Scope

To set the limits of this project it is compulsory to look at the variables and hypothesis used. The variables of the Tests, which the application is going to work with, are only the duration and the constraints that the Test needs to be performed. Consequently, related issues to place the Prototypes in the calendar (such as releasing dates, first day of start or last day of start), additional Test properties (such as Test dependencies or priority order), working teams (such

as capacity teams to perform the Tests or working days) or associated costs (such as profit taken of building one Prototype with some related Tests or an overspending due to the difficulty of building the Prototypes with some given requirements) are not integrated in the program.

5. Application "Prototype Planning"

In other researches done a few years ago, the ITL Institute laid the foundations of the Prototype planning taking into consideration a Test database. From that point further investigations and larger develops should be taken part. However, the problem was taken always as a theoretical or mathematical problem, instead of thinking as a real programming problem. Therefore, this program is the first one that takes the theoretical knowledge into a real testing system and pretends, then, to be the forerunner for further developments to achieve an application, which would be able to provide the lowest amount of Prototypes with as many conditions and the external limitations as possible.

As we can see in *Figure 1*, the current problem has only been performed from the left side and this application takes place in the right side, where programming tools and an automated solution get together.

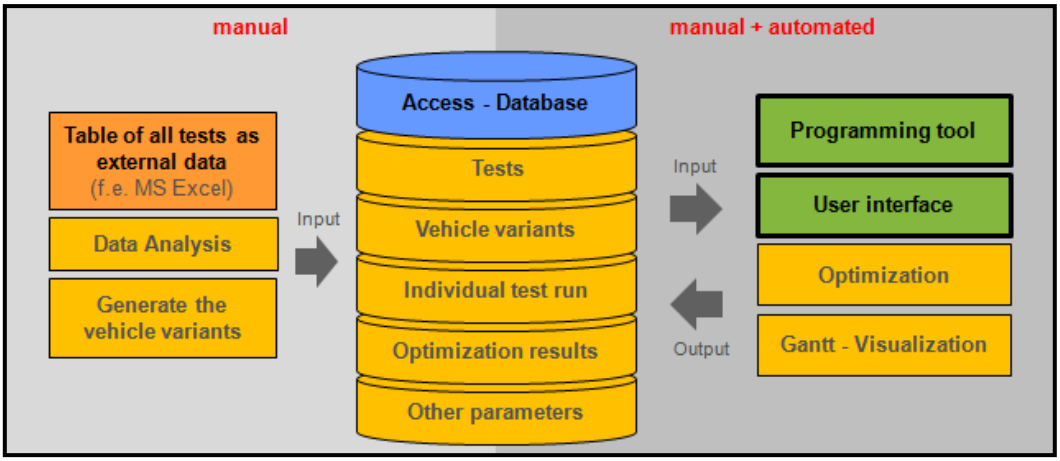

Figure 1. Current situation of the research in Prototype planning.

Consequently, the application "Prototype Planning" is an input-output application which works with an input database of Tests, which need to be performed, and returns an output solution of the Prototypes that have to be built. However, the user has not only the opportunity of obtaining a solution, but also testing the designed versions, as it is explained in Section 5.3.

Moreover, the Tests have some required components that are implemented in the programming like constraints. It is important to say that those constraints are the programming motor, because the program runs through the compatibilities search and its assignment. In the working Test database, the component requirement possibilities are 12. In *Figure 2* can be appreciated the meaning of the constraints and, although the Tests are not

described individually and the constraints are not assigned to one component variety in particular, this research can be better understood by taking a look at the components possibilities than only thinking of mere mathematical constraints. Furthermore, the reason of not revealing the assignment of each constraint to the real component is due to privacy.

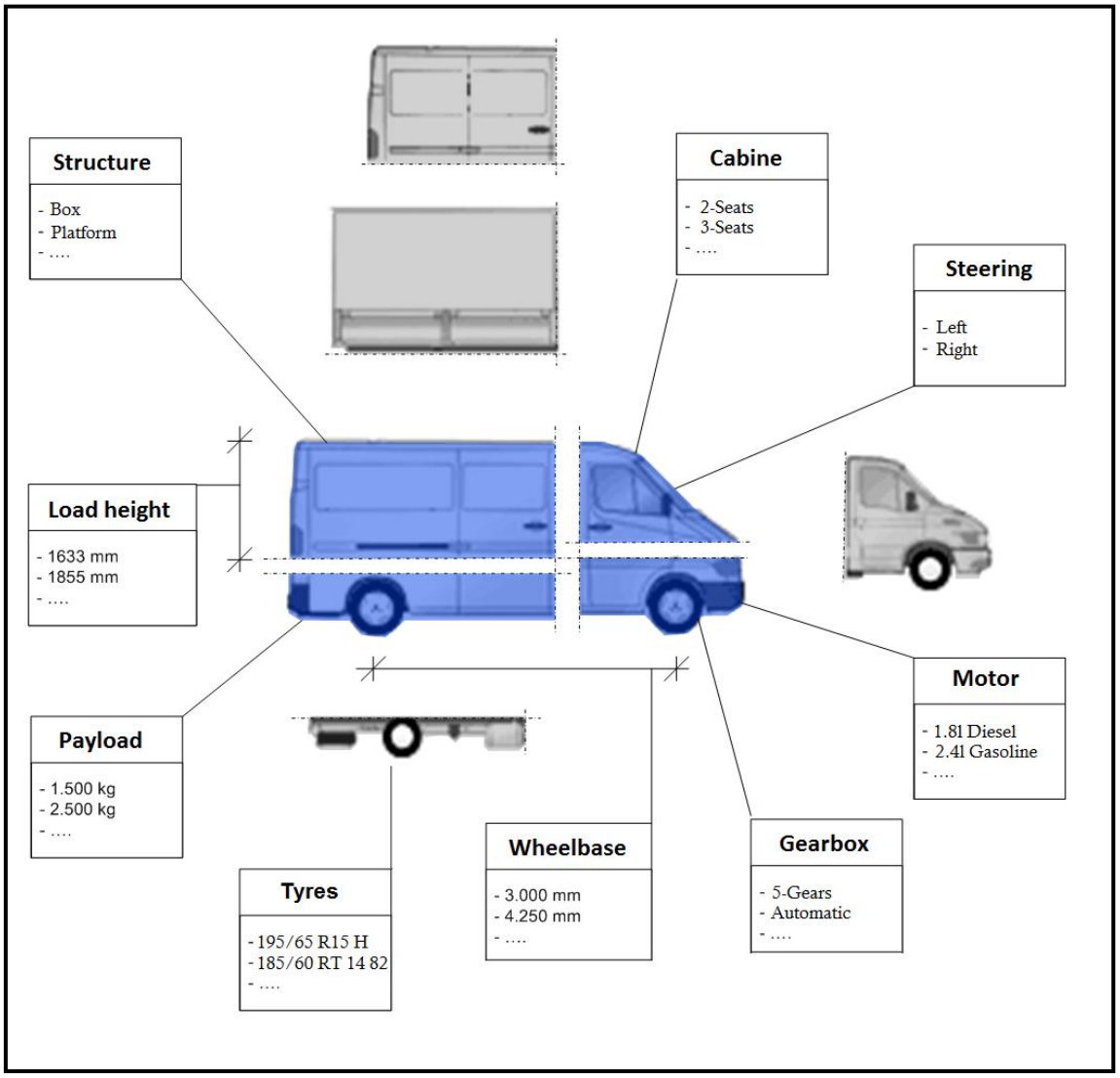

Figure 2. Possible components of a vehicle variant.

If we go deeper in this application, the working data from a vehicle manufacturer is a list of 1091 Tests to be allocated in the Prototypes and a Graphical User Interface (GUI) was designed to manage this input data. In addition, this GUI also allows the user to select the desired running application within the “Prototype Planning” potential, and save and manage the results or statistics to work further with them.

The explanation of the program designed is organized as follows. Section 5.1 provides the background information about the programming tools and the code issues. Then we present the GUI, the meaning of every button and the steps the user has to perform to get the desired results in Section 5.2. The potential of “Prototype Planning” with all the integrated sub-applications are formally shown and presented in Section 5.3. After all sub-applications are introduced, we can find their programming methods, sequence diagrams, and used functions in Sections from 5.3.1 to 5.3.8. Subsequently, in Section 5.4 is explained everything regarding to the results and validation of the versions. Therefore, in Section 5.4.1 are presented all the results and charts to give details about why version 2 improves the first one. Continuing with the analysis, in version 5.4.2 and 5.4.3 are presented on the one hand, the changes that version 3 implements, and on the other hand, the new variation that version 4 implements. Finally, in Section 5.5 is presented the room for improvement of this application and programming methods.

5.1. Issues Related to the Programming

The program is designed with the Microsoft "Visual Studio Ultimate 2010" software and the code used is C#. On the one hand, this software has been chosen for its help in the writing code procedure, the organized way of the programming and visualization and the easy debugging system that offers to the user. On the other hand, the C# code is also a new code growing up due to the several improvements that own, combining the advantages from C or C++, the programming structure of Java and the graphical solutions of "Visual Basic". From this point, the combination of both "Microsoft Visual Studio" and C# offers a plenty of possibilities to the application programming that were years ago was only a restricted field to "Microsoft Visual Basic".

If we go deeper in the input database, it has to be said that the application "Prototype Planning" is working with the data from an Excel file, which contains 1091 Tests and the information of duration and constraints of every Test.

For a better understating of the input data and therefore of the programming operation, when a Test does not need a specific component from the list of 12 components that can have, then is represented with a "0". In that case, all component possibilities are suitable to that Test in that component. This is not only useful for a reader of the input data, but also for an easier programming.

An example of some Tests of the input data is shown in *Figure 3*:

ID	Duration	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
516	4	103	3	6	0	7	0	6	2	0	2	0	0
517	4	103	3	6	0	7	0	6	2	0	2	0	0
518	4	102	3	6	0	8	0	6	2	0	2	0	0
519	4	102	3	6	0	8	0	6	2	0	2	0	0
520	4	103	3	5	0	8	0	6	2	0	2	0	0
521	4	103	3	5	0	8	0	6	2	0	2	0	0
522	15	0	0	2	0	0	0	0	0	1	0	1	1
523	15	0	0	1	0	0	0	0	0	2	0	2	1
524	15	0	0	1	0	0	0	0	0	2	0	2	2
525	15	0	0	1	0	0	0	0	0	2	0	2	3
526	15	0	0	1	0	0	0	0	0	2	0	2	4

Figure 3. Tests from 516 to 526 of the working database.

For example, if we look at Test 521, it needs a specific component in C1, C2, C3, C5, C7, C8 and C10 but in the components C4, C6, C9, C11 and C12 it does not require anything. Moreover, in order to complete the information, the duration of this Test is of 4 days.

Trying to explain the data type, which we are working with, and show the main property of a Test (its duration) within all the Tests, in *Figure 4* we can see a graphical presentation of the duration of the Tests.

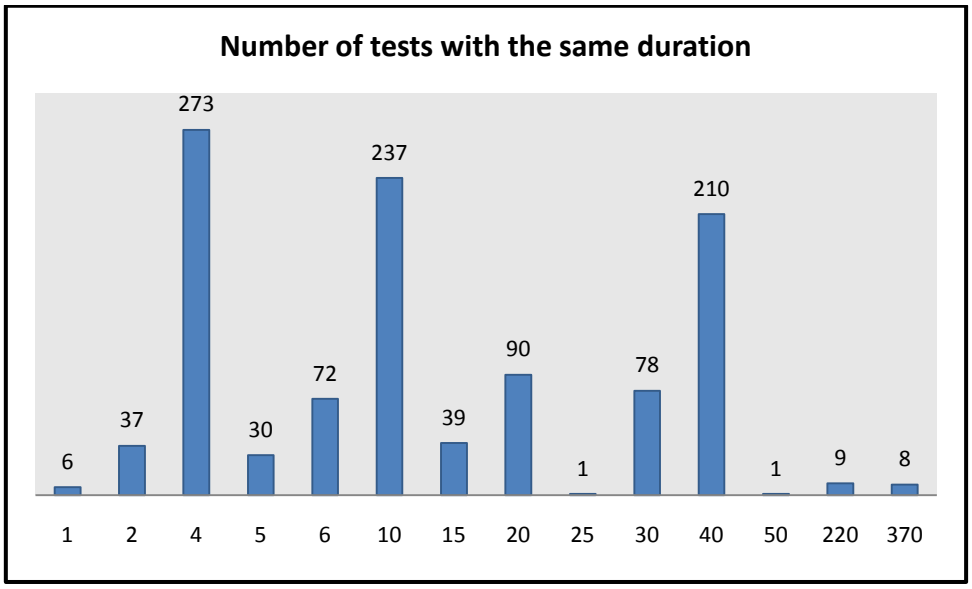

Figure 4. Duration statistics of all the Tests in the working database.

Taking a look at *Figure 4*, can be appreciated how all the durations are distributed and it is important to observe that Tests with a performing duration of 4, 10 and 40 days represent the 66% of all Tests in the database. In addition, the Tests with 370 days of duration play a boundary role, because the minimum time to get all Tests performed has to be over those 370 days.

After observing the Test database, the next step is to give details about how this application is working. From this point, the class types, which this application is using, have to be presented and those ones are *Prototype* and *Test* as in *Figure 5* is shown:

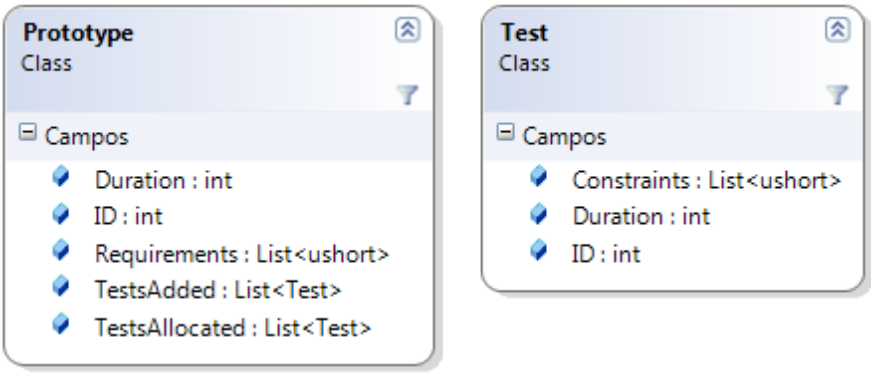

Figure 5. Classes of "Prototype Planning".

In order to give some details about both classes, it has to be said that both *Requirements* (from *Prototype*) and *Constraints* (from *Test*) are a list of 12 items with a number and each of these numbers belongs to one requirement/constraint. In addition, inside Prototypes there are two similar fields: *TestsAllocated* and *TestsAdded*. The first one contains the applied Tests through the basic programming method and the second one contains the reorganized Tests applied to a Prototype by an additional programming method, which belongs to version 4.

5.2. Designing a Graphical User Interface (GUI)

The application "Prototype Planning" needs a Graphical User Interface (GUI) so the user can interact with the application. Because of this need, it was chosen the combination of "Microsoft Visual Studio" and C# as it was explained in Section 5.3. From this point, the designing of the GUI is completed through a tool's editor, with which the programmer can add every single button or text, and then apply some interaction with the code (in most cases, calling a function).

This GUI and its entire buttons are presented below.

First, in *Figure 6* is shown the global aspect of the GUI with its two main parts. These divisions are the toolbar to interact with the application and the running window to select the type of sub-program we want to run.

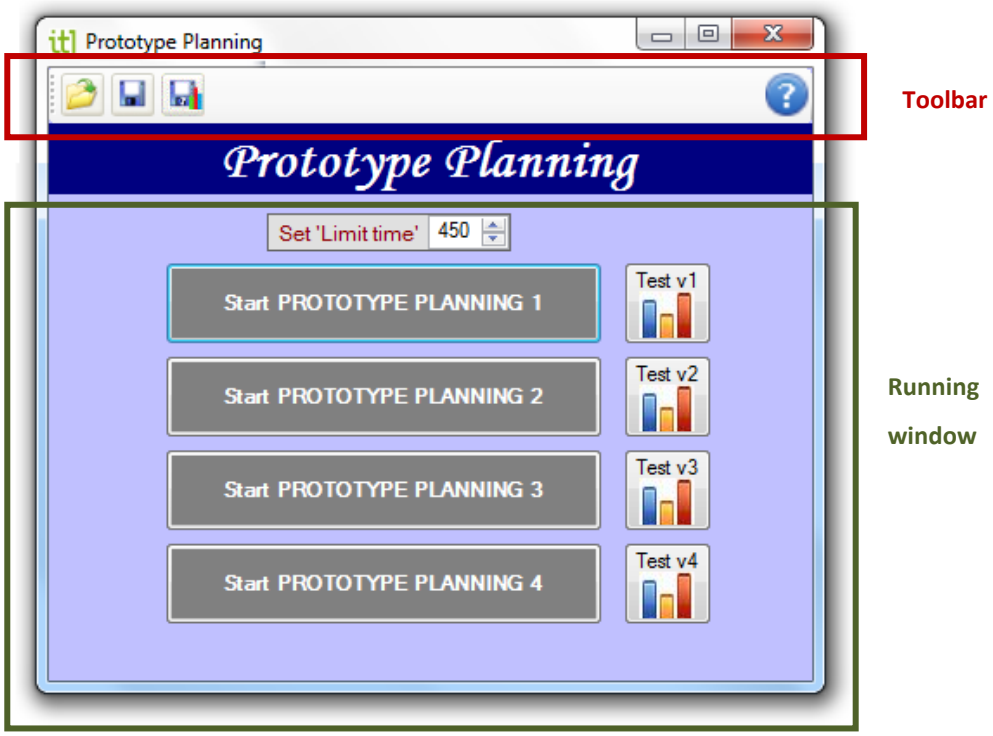

Figure 6. General view of the "Prototype Planning" GUI (Graphical User Interface).

To describe every button that is used in this interface, below is presented all buttons from the toolbar and all buttons from the running window, with a description of the actions they perform and how can the user manage them to get the desired solution.

On the one hand, the element on the top of the application is the **toolbar**, which is the entrance gate of the database with all the Tests and the exit gate through the buttons of saving the list of Prototypes or the statistics of one solution.

This toolbar is detailed in *Figure 7*, and here can be appreciated how the buttons are displayed.

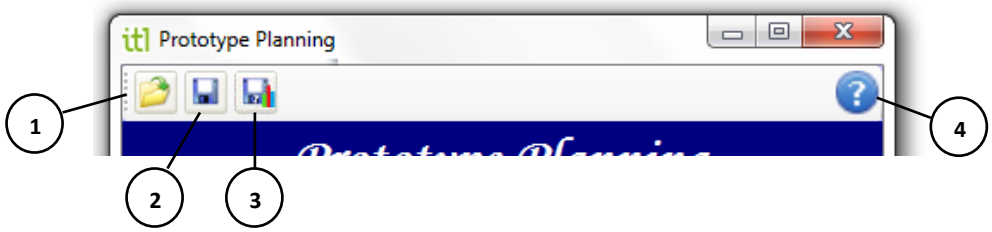

Figure 7. "Prototype Planning" toolbar.

1 Open Test database

2 Save Prototype list

3 Save Prototype list statistics

4 Display information about the Program

By clicking in 1 an "Open File Dialog" will appear and the user could select the input database to work with. The file has to be in format .csv to be read. After the simulation of one solution has been performed, the application will enable the buttons 2 and 3 a "Save File Dialog" will appear and the user could select the path of the saved file. Finally, by moving the mouse on button 4 the user can see all the information about the application project.

On the other hand, the main section of the application is the **running window** and consists in a plenty of running methods possibilities (described in Section 5.3) and it is the main part of the "Prototype planning" application inasmuch as is where all the programming code and the solution generation is.

The running window is detailed in *Figure 8*, where all buttons are presented for a later explanation.

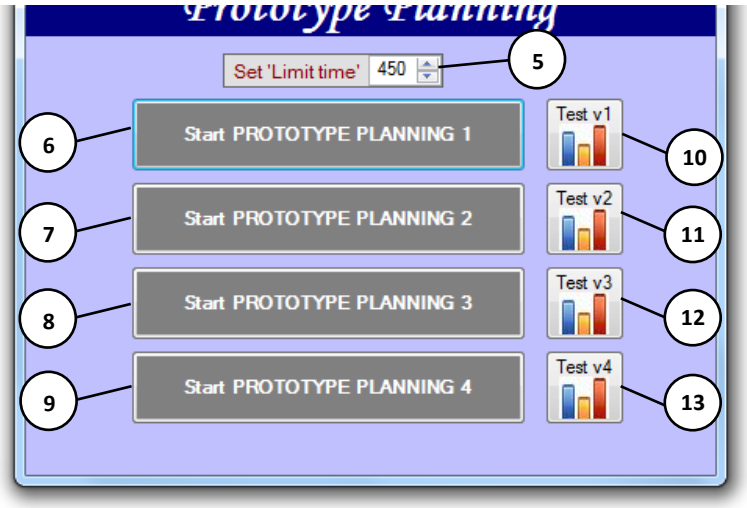

Figure 8. "Prototype planning" running window.

5 Set limit time

6 Run v1

7 Run v2

8 Run v3

9 Run v4

10 Generate stats of v1

11 Generate stats of v2

12 Generate stats of v3

13 Generate stats of v4

After all buttons are presented, here is explained how this interface is working. It is important to say, that the user can run the designed sub-applications in the order or way the user requests but following some steps. From this point, there are two different ways to proceed:

If the user wants to generate a list of Prototypes with one version, the steps are as follows:

1. Click on "Open Test database" (presented in *Figure 7*) and select the input lest of Tests.
2. Select time limit by setting the desire time in 5.
3. Select the desired version by clicking on the buttons 6 7 8 or 9
4. Save the list of generated Prototypes by clicking on 'Save Prototype list' (presented in *Figure 7*) or save the statistics of the simulation by clicking on 'Save Prototype list statistics" (presented in *Figure 7*).

5. Wait for the confirmation message, which says that the list of Prototypes has been generated.

If the user wants to test one version and generate the statistics of every generated Prototype list, the steps are as follows:

1. Click on “Open Test database” (presented in *Figure 7*) and select the input lest of Tests.
2. Select the desired version by clicking on the buttons (10) (11) (12) or (13)
3. Select the output file by the time the “Save file dialog” is displayed.
4. Wait for the confirmation message, which says that the statistics have been generated. A progress bar will be displayed during the solution is being generated.

As can be appreciated in the designing and the explanation of the steps, the application “Prototype Planning” has been conceived to be easy to use. In order to support this, every step the user performs, the buttons become enabled or disabled, so a mistake cannot be generated. For example, before saving a list of Prototypes, the user has to run one of the versions offered. That is the reason why “Save Prototype list” button in disabled until the user runs a version. In addition, the user also gets a confirmation message when a sub-application has generated the solution. Furthermore, a progress bar is displayed (if a testing sub-application is being run) so the user can see how long is going to take to get the results.

5.3. "Prototype Planning" Potential

The potential of this application is not only reduced to a simple solution based on one programming method. As the user is able to check in Section 5.2, the "Prototype Planning" user can generate four different solutions by running four different programming methods. Below are briefly described the three different versions:

- **Version 1**

  This version has the minimum programming constraints. To generate each Prototype is taking into account only the duration and the constraint of the Tests. This version is described in Section 5.3.1.

- **Version 2**

  This version integrates an optimization from version 1, which consists of taking away the Tests with no component constraints and allocating those Tests at the end of the simulation. This version is described in Section 5.3.3.

- **Version 3**

  This version integrates another optimization from version 2. In this case, it is known that not all Prototypes can be built and tested, starting all of them in the first day of prototyping and testing. Therefore, each Prototype is going to be tested with an offset of some days. This version is described in Section 5.3.5.

- **Version 4**

  This version integrates a new hypothesis. This hypothesis consists in reallocate the Prototypes, after all of them are generated. The reallocating system is based on changing some components, which are easy to change (for example, the tyres), from a Prototype to make it compatible with another one. This version and its premises are described in Section 5.3.7.

As it is explained in Section 5.2, all these four versions are run by selecting a limit time (by default, its value is 450 days) and it has to be higher than the maximum duration Test. In this case, the working Test database has a highest-duration Test of 370 days and as a result, the user can set a minimum limit time of 380 days.

In addition, after running any of the versions proposed, the user is able to save the generated Prototype list and/or save the simulation stats in an extern file by clicking on the saving buttons.

On the other hand, the user can also Test how is the generated solution of a version by clicking on one of the “testing buttons” or “statistics generators” presented in Section 5.2.

These testing possibilities are four and each one belongs to one version, as is presented below.

- **Test version 1**

  It generates the Prototype list statistics for the version 1. This sub-application is described in Section 5.3.2.

- **Test version 2**

  It generates the Prototype list statistics for the version 2. This sub-application is described in Section 5.3.4.

- **Test version 3**

  It generates the Prototype list statistics for the version 3. This sub-application is described in Section 5.3.6.

- **Test version 4**

  It generates the Prototype list statistics for the version 3. This sub-application is described in Section 5.3.8.

These sub-applications generate a solution for different limit times and save the statistics of each solution to an extern file. In these sub-applications, the user cannot see how every Prototype should be built, but only the statistics of each simulation.

5.3.1. "Prototype Planning" v1

This first version of the application, called "Prototype Planning" v1 is running by clicking on the 

button and takes into account these hypotheses:

- ✓ The maximum duration of a Prototype is the limit time given by the user
- ✓ The Prototypes are assigned to one Prototype if the constraints are compatible with the Prototype's requirements and if the combined duration does not exceed the limit time.

The programming method is as follows:

1) Search for the Test with the maximum duration. (`SearchForMaximumDuration`)
2) Assign this Test to a new Prototype. (`OpenPrototype`)
3) Search for compatibilities through the whole Test list. (`CheckCompatibilities`)
   a) If there is a compatible Test:
      i) Assign the compatible Test to the current Prototype. (`AssignTestToPrototype`)
      ii) Repeat this sequence (from 3) until there is no compatibility with the current Prototype.
   b) If there is no compatible Test:
      i) Tell the program to restart the method. (`NoCompatibleTestFound`)

❖ Repeat this method (from 1 to 3) until all the Tests are assigned

For a better understanding, in *Figure 9* there is the sequence diagram of version 1 with a deepness characterization of only one level, which means that in the diagram are only appearing the calling functions. Hence, the functions or instructions, which are inside the primary calling functions, and the instructions integrated in "Microsoft Visual Studio" (for example, calling the amount of the allocated Tests in a Prototype) are not displayed.

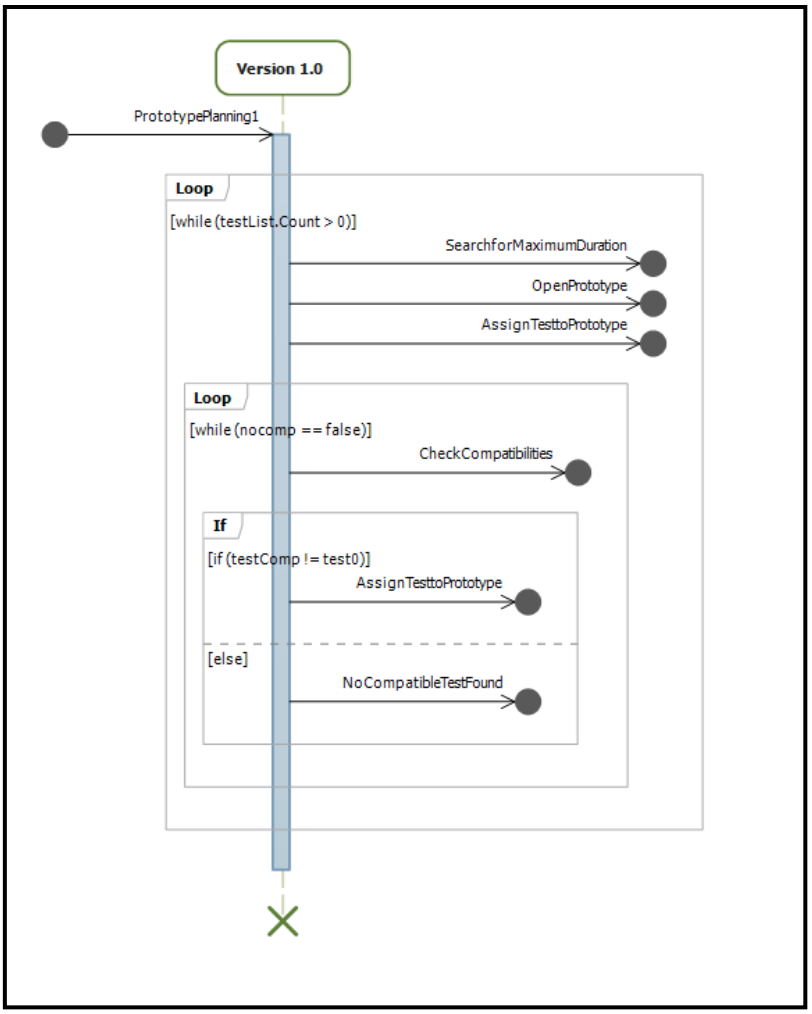

Figure 9. Sequence diagram of "Prototype Planning" v1.

The main intern functions, which this sub-application is working with, are shown in *Figure 10*.

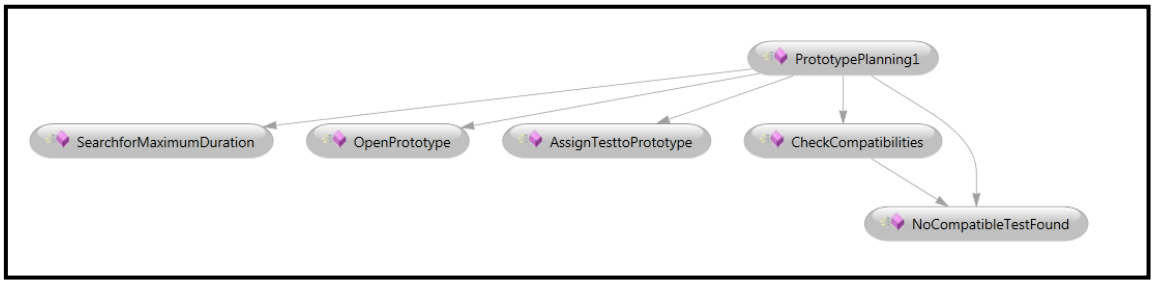

Figure 10. Functions used in "Prototype planning" v1.

As these functions are used all over the other running methods, they are going to be described individually.

- `static Test SearchForMaximumDuration()`

It searches in the whole Test list for the Test with the maximum duration and returns that Test.

- `static void OpenPrototype()`

It creates a new Prototype in the Prototype list to be filled further with Tests.

- `static void AssignTestToPrototype(Test t, Prototype p, List<Test> tl)`

It assigns the constraints of one Test (t) to the requirements of Prototype (p). Inside the function is a calling to `AssignConstraintsToRequirements`, to make all the assignments. After the Test (t) is applied, it is removed from the Test list (tl) so cannot be used further in this simulation.

- `static void AssignConstraintsToRequirements(Test t, Prototype p)`

  It assigns every constraint of one Test (t) to the requirements of Prototype (p). If one constraint of the Test is "0" (that Test can be run with every variety of that component), the requirement value for that component remains as it is.

- `static Test CheckCompatibilities(Prototype p, int limit)`

It searches for a compatible Test with one Prototype (p). This procedure is done through comparing the requirements of the Prototype (p) with the constraints of every Test and checking that if a compatible Test is added to the Prototype, the combined duration does not exceed the limit time (limit).

- `static void NoCompatibleTestFound ()`

  It gives the main function the instruction that there is no compatible Test found within the list of Tests.

5.3.2. Testing "Prototype Planning" v1

This sub-application is running by clicking in the 

button. Then, the user has to select the path of the saving stats. Finally, the application will generate a file with all the statistical information.

If we go deeper, this sub-application is generating the statistics of all the Prototype lists, which are generated with the method of "Prototype Planning" v1 and with a limit time between 380 (the maximum duration of a Test is 370 days) and 1000 (random number, but quite high so the user can see the evolution of the number of Prototypes). It is therefore like an evaluation of the version 1 through 620 simulations with a different limit time.

The programming method is as follows:

1) Simulate the first Prototype list (by running the "Prototype Planning" version 1 with the first limit time. (`PrototypePlanning1`)
2) Generate the statistics of this simulation and write them in the saving file given. (`GeneraTestats`)
3) Delete the Prototype list and regenerate the Test list so can be used in the next simulation. (`RegenerateData`)
4) Update the value of the Progress bar. (`PerformStepInProgressbar`)

❖ Repeat this programming method (from 1 to 4) until all the limit times are used.

For a better understanding, in *Figure 11* there is the sequence diagram of the evaluation of version 1.

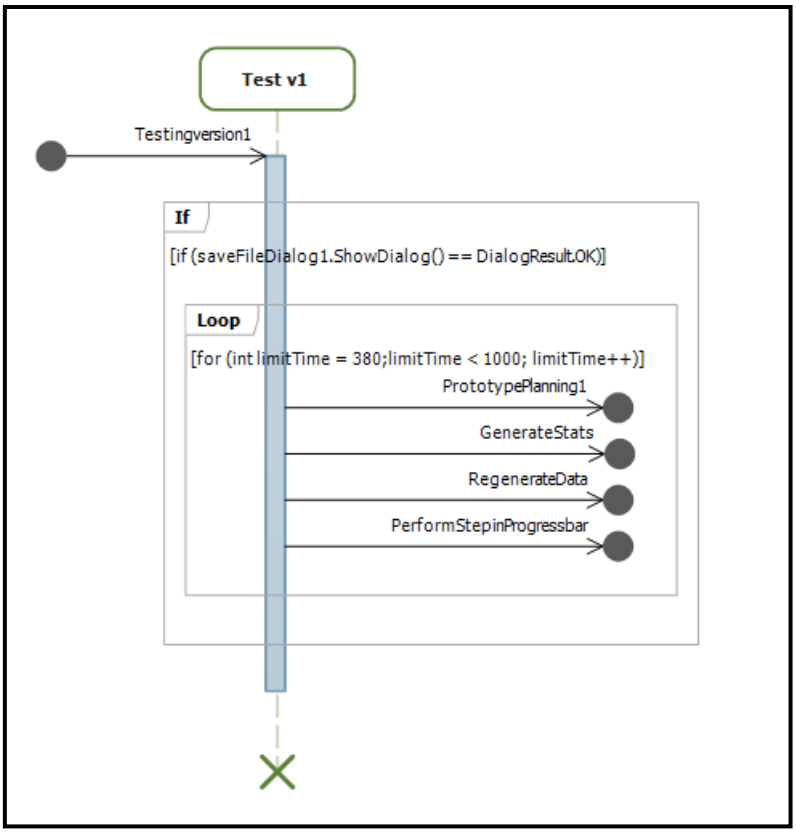

Figure 11. Sequence diagram of Test "Prototype Planning" v1.

The main intern functions, which this sub-application is working with, are shown in *Figure 12*.

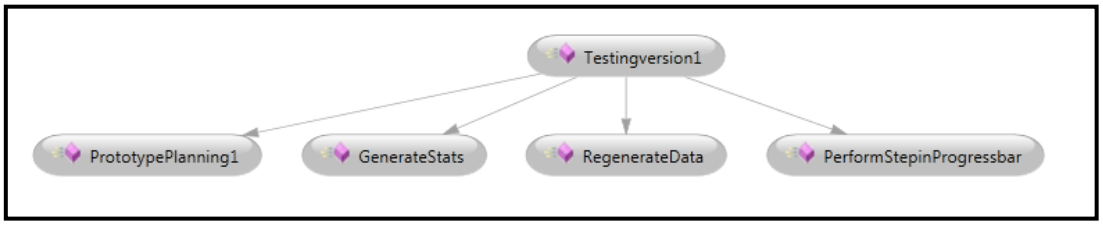

Figure 12. Functions used in Test "Prototype planning" v1.

Two new functions are used in this sub-application and they are described below:

- `static public void GenerateStats()`

It creates the statistics from a Prototype list solution given. The statistics this program creates are: number of Prototypes, maximum Prototype duration, minimum Prototype duration, average Prototype duration, maximum number of Tests that one Prototype has, average number of Tests that one Prototype has and number of Prototypes with only one Test applied.

- `static public void RegenerateData()`

It clears the Prototype list generated and reopens the Test list file to start a new simulation.

- `static public void PerformStepInProgressbar(Progressbar pbar, int step)`

It updates the value of a Progress bar (pbar) every number of iterations (step). In this case, the Progress bar is always the same one is the one, which is shown in the window application.

5.3.3. "Prototype Planning" v2

This second version of the application, called "Prototype Planning" v2 is running by clicking on the 

button and takes into account these hypotheses:

- ✓ The maximum duration of a Prototype is the limit time given by the user
- ✓ The Prototypes are assigned to one Prototype if the constraints are compatible and the combined duration does not exceed the limit time.
- ✓ **The Tests with no component constraints are taken away from the Test list and are used after all the other Tests are applied.**

This method purports to optimize the first version of the application by removing the Tests without constraints and add them after the other Tests are assigned to the Prototypes. By doing this, these "removed" Tests can be applied in any Prototype, because the constraints will match always (if there is no component constraint, there is nothing to match) and the only think to care about is the combined duration of the Prototype and the working Test.

The programming method is as follows:

1) Search in the working Test list for all the Tests with no constraints and move them to a new list of Tests called *TestListaux*. (`InitilializeListAux`)
2) Run version 1 with all the remaining Tests in the original Test list. (`PrototypePlanning1`)
3) Apply the Tests from *TestListaux* to the created Prototype list. The only variable lo look at is the duration, so if the combined duration of a Prototype and one Test is under the limit time, the Test can be applied there. (`OptimizationWithListAux`)

For a better understanding, in *Figure 13* there is the sequence diagram of version 2.

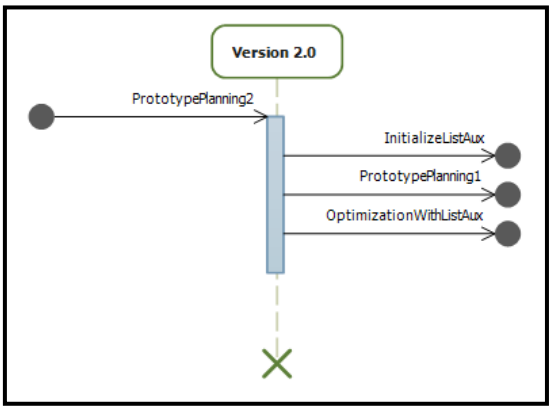

Figure 13. Sequence diagram of "Prototype Planning" v2.

The main intern functions, which this sub-application is working with, are shown in *Figure 14*.

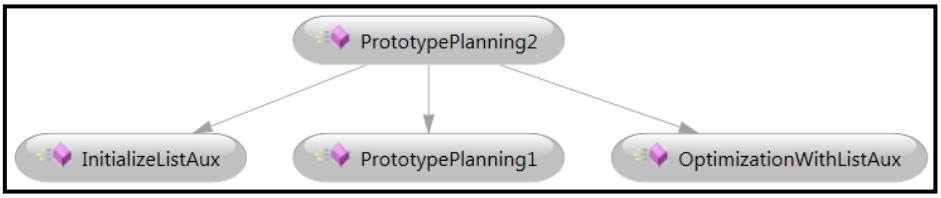

Figure 14. Functions used in "Prototype planning" v2.

Two new functions are used in this sub-application and they are described below:

- `static public void InitializeListAux()`

It searches for all the Tests in the working Test list, which do not have constraints in their components and moves those Tests to another list of Test called *TestListaux*.

- `static public void OptimizationWithListAux()`

It implements the Tests from *TestListaux* to the generated Prototypes. As the method of this function is more complex than the other ones presented, it is below deeper analyzed.

The programming method of this function is as follows:

1) Take a Test from the list of Tests with no constraints (*TesListaux*) and search if there is space for this Test in one of the Prototypes created, by only looking at the property of the duration. If the duration of the Test and the examined Prototype does not exceed the limit time, the Test can be applied to that Prototype. (`SearchForSpace`)
    a. If the Test has no space inside a created Prototype, create a new one and allocate this Test in it. (`OpenPrototype`)
2) If the Test can be applied inside a created Prototype, apply the Test to that Prototype. (`AssignTestToPrototype`)

❖ Repeat this programming method (from 1 to 2) until all the limit times are used.

For a better understanding, in *Figure 15* there is the sequence diagram of the "Optimization with List Aux" function.

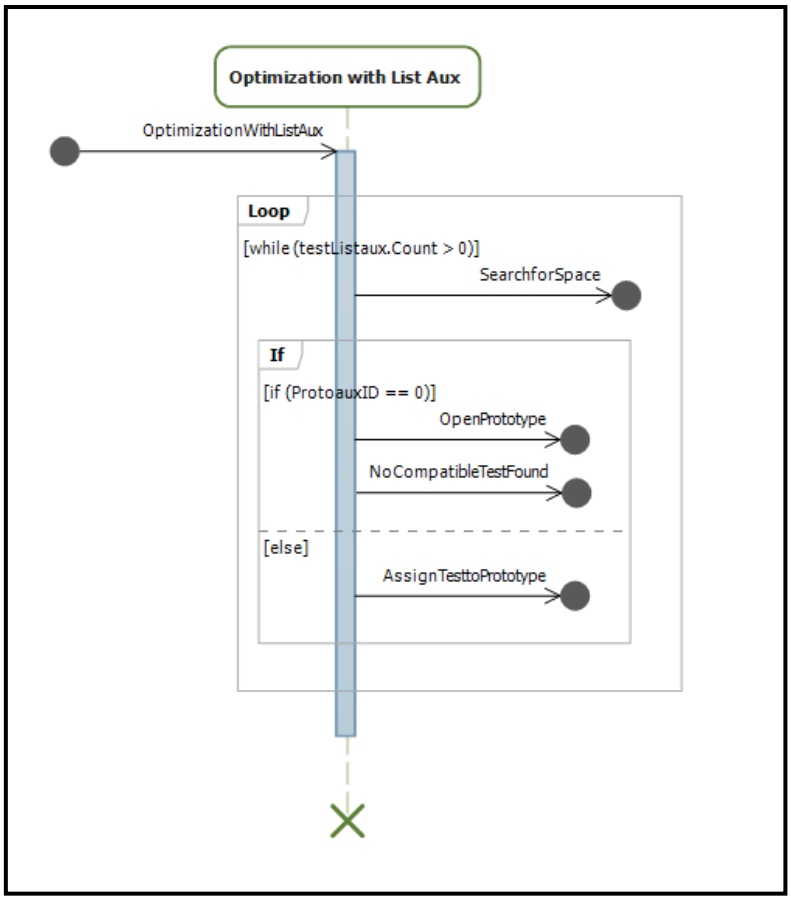

Figure 15. Sequence diagram of the function "OptimizationWithListAux".

A new function is also used in the function `OptimizationWithListAux` and, therefore, it is described below:

- `static int SearchForSpace(Test t, int Time)`

It takes the Test (t) from *TestListaux* that the master program selects and it looks for a Prototype where this Test can be applied. The only restriction that a Test has to be placed in a created Prototype is that the combined duration does not exceed the limit time given (Time). If the combined duration of the Test and one Prototype is under the limit, the function returns the ID of the compatible Prototype. If not, it returns the value "0" that will be further processed like "no compatible Prototype found".

5.3.4. Testing "Prototype Planning" v2

This sub-application is running by clicking in the 

button. Then, the user has to select the path of the saving stats. Finally, the application will generate a file with all the statistical information.

If we go deeper, this sub-application is generating the statistics of all the Prototype lists, which are generated with the method of "Prototype Planning" v2 and with a limit time between 380 (the maximum duration of a Test is 370 days) and 1000 (random number, but quite high so the user can see the evolution of the number of Prototypes). It is therefore like an evaluation of the version 2 through 620 simulations with a different limit time.

The programming method is as follows:

1) Simulate the first Prototype list (by running the "Prototype Planning" version 2 with the first limit time. (`PrototypePlanning2`)
2) Generate the statistics of this simulation and write them in the saving file given. (`GenerateStats`)
3) Delete the Prototype list and regenerate the Test list so can be used in the next simulation. (`RegenerateData`)
4) Update the value of the Progress bar. (`PerformStepInProgressbar`)

❖ Repeat this programming method (from 1 to 4) until all the limit times are used.

For a better understanding, in *Figure 16* there is the sequence diagram of the evaluation of version 2.

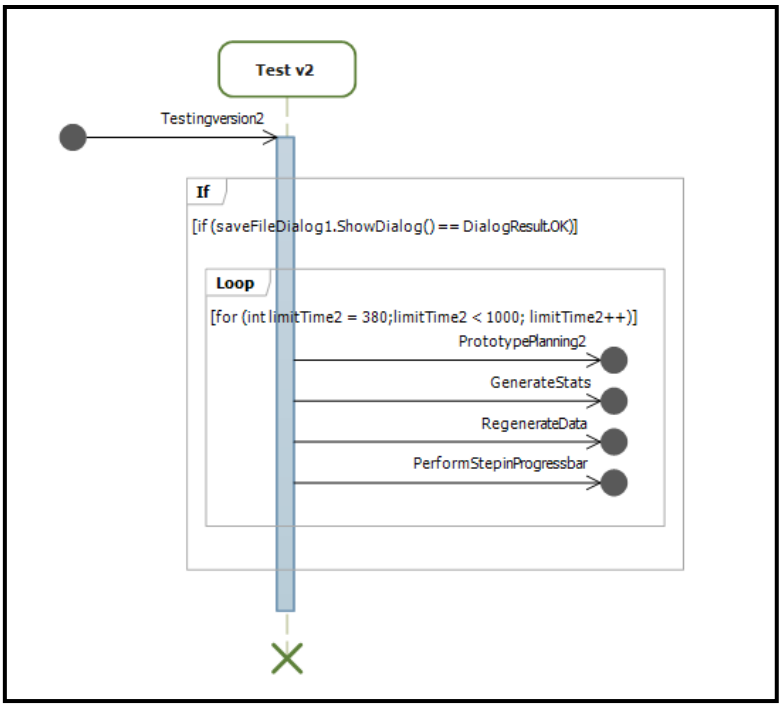

Figure 16. Sequence diagram of Test "Prototype Planning" v2.

The main intern functions, which this sub-application is working with, are shown in *Figure 17*.

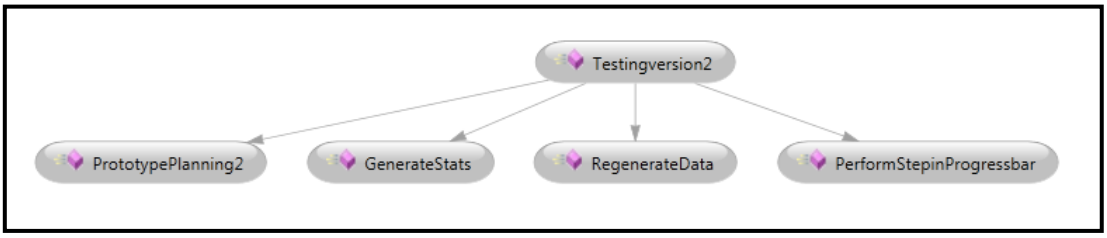

Figure 17. Functions used in Test "Prototype planning" v2.

5.3.5. "Prototype Planning" v3

This third version of the application, called "Prototype Planning" v3 is running by clicking on the Start PROTOTYPE PLANNING 3.0 button and takes into account these hypotheses:

- ✓ The maximum duration of a Prototype is the limit time given by the user
- ✓ The Prototypes are assigned to one Prototype if the constraints are compatible and the combined duration does not exceed the limit time.
- ✓ **The starting point of every Prototype has an offset delay with the previous Test.**

In the field of Prototype planning, it is also important to bring the simulation closer to the reality as much as possible. Consequently, this version of the application integrates another optimization to have a solution more accurate to the real Prototype planning by taking into account the fact that in the Prototype building and testing exist work-teams and all the job cannot be simultaneously with other one similar. The teams have a capacity, thus the Prototypes cannot be all built in one day.

In this simulation, the Prototypes are built with an offset of 3 days and this value has been taken by knowing how a Prototype building team can operate. Anyway, it is a random value and a more accurate value needs information from the vehicle manufacturer. Besides, perhaps the start-up curve has a different inclination or a curving function, but that information is private and therefore is not applied here.

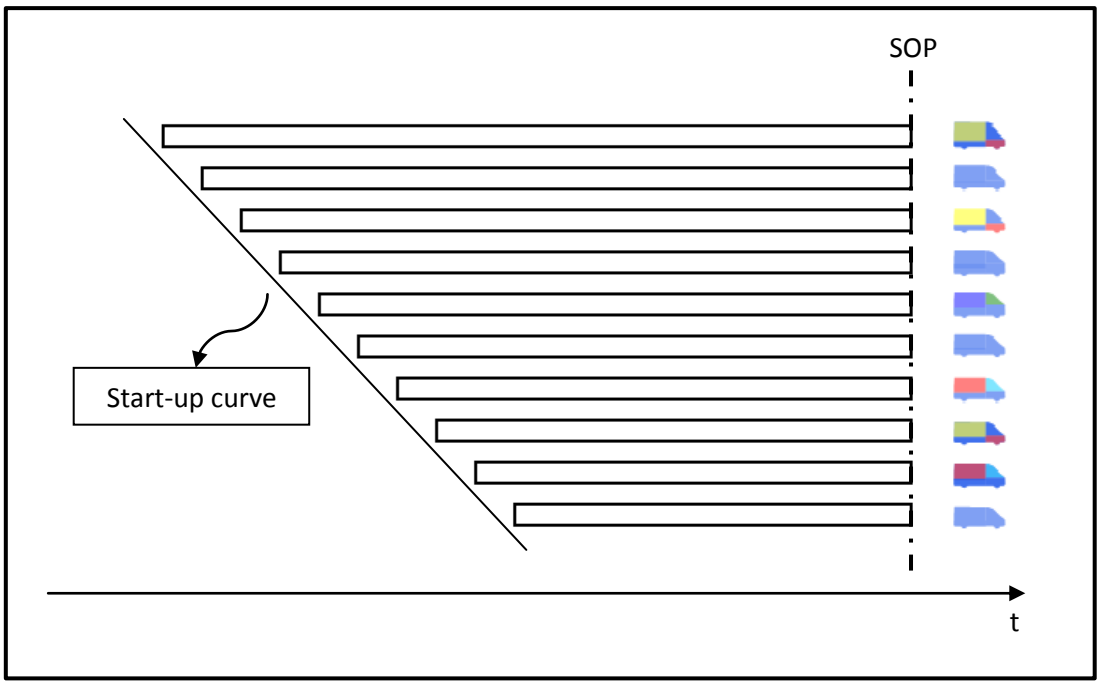

Figure 18. Start-up curve of the Prototype planning

The *Figure 18* shows the new hypothesis that this third version is working with: the line of the start-up curve is going to be like an offset of 3 days between each Prototype releasing.

The programming method of "Prototype Planning" v3 is based on the second version in that it uses the auxiliary list of Tests *TestListaux*, as the aim of a complex programming application like this one, is to implement in every version a new optimization to be, every version, closer to the reality.

The programming method is as follows:

1) Search in the working Test list for all the Tests with no constraints and move them to a new list of Tests called *TestListaux*. (`InitilializeListAux`)
2) Search for the maximum duration of both of the Test lists. (`SearchForMaximumDuration`)
    a. If the maximum duration Test belongs to the main list of Test, create a new Prototype and assign this Test. (`OpenPrototype & AssignTestToPrototype`)
    b. If the maximum duration Test belongs to the auxiliary list of Test, search if the Test can be applied in one of the Prototypes created. (`SearchForSpace`)
        i. If there is space in one created Prototype, assign the Test to that Prototype. (`AssignTestToPrototype`)
        ii. If there is no space in any created Prototype, create a new one and assign the Test to that new Prototype. (`OpenPrototype & AssignTestToPrototype`)
3) Search for compatibilities through the whole Test list. (`CheckCompatibilities`)
    a. If there is one compatible Test:
        i. Assign the compatible Test to the open Prototype. (`AssignTestToPrototype`)
        ii. Repeat this sequence (from 3) until there is no compatibility with the current Prototype.
    b. If there is no compatible Test:
        i. Decrease the limit time. (`UpdateLimitTime`)
        ii. Tell the program to restart the method. (`NoCompatibleTestFound`)

❖ Repeat this method (from 2 to 3) until all the Tests are assigned.

For a better understanding, in *Figure 19* there is the sequence diagram of version 3.

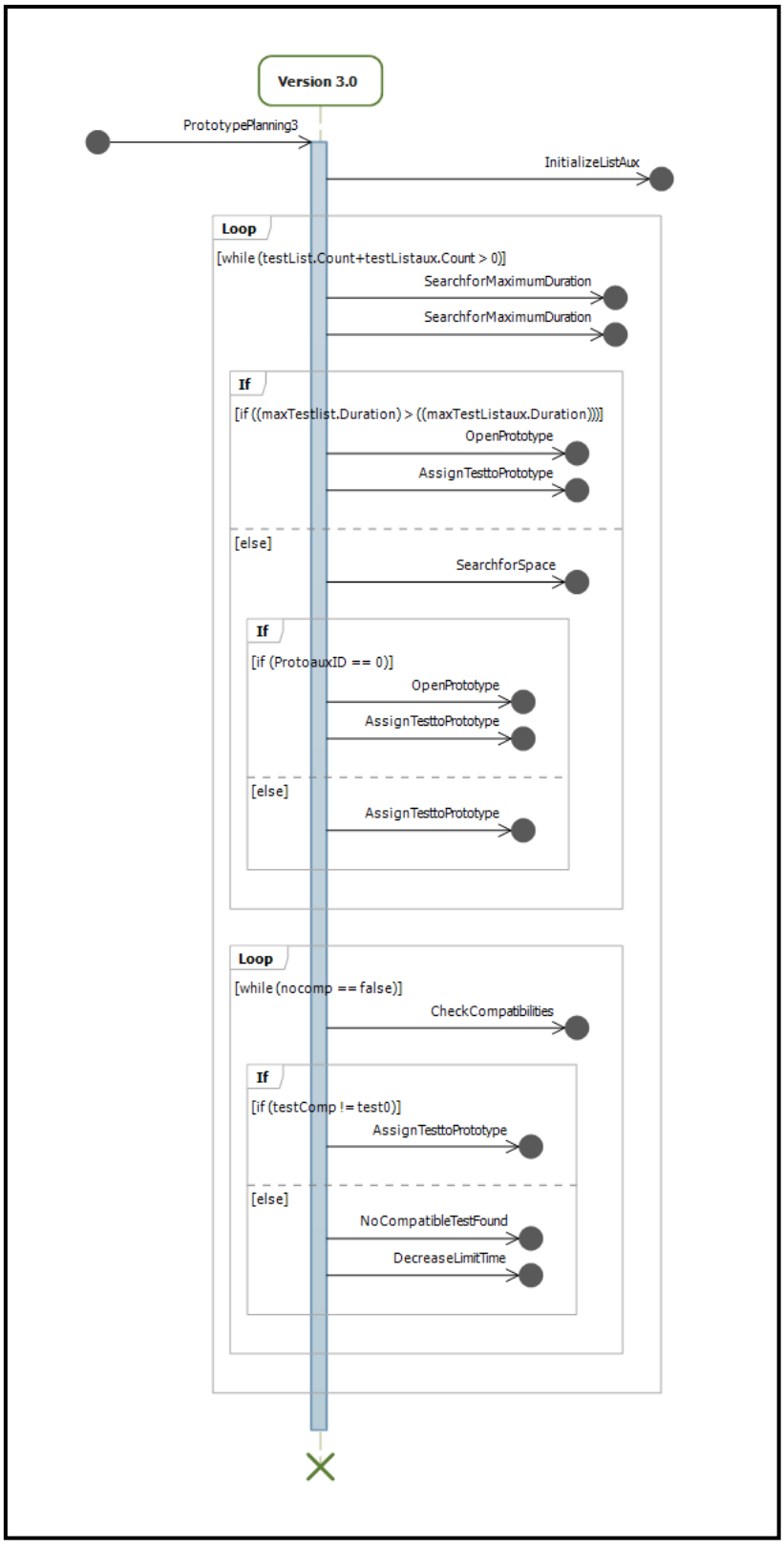

Figure 19. Sequence diagram of "Prototype Planning" v3.

The main intern functions, which this sub-application is working with, are shown in *Figure 20*.

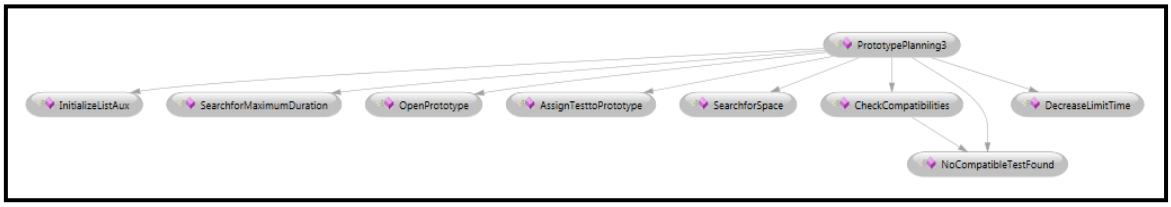

Figure 20. Functions used in "Prototype planning" v3.

A new function is also used and it is described below:

- `static int UpdateLimitTime(int r, int l)`

It decreases the limit time by 3 days every new Prototype is built. This function can be changed by having more accurate information from the vehicle manufacturer. A polynomial function can also be implemented here.

5.3.6. Testing "Prototype Planning" v3

This sub-application is running by clicking in the 

button. Then, the user has to select the path of the saving stats and the application will generate a file with all the information.

If we go deeper, this sub-application is generating the statistics of all the Prototype lists, which are generated with the method of "Prototype Planning" v3 and with a limit time between 400 (the maximum duration of a Test is 370 days) and 1000 (random number, but quite high so the user can see the evolution of the number of Prototypes). It is therefore like an evaluation of the version 3 through 600 simulations with a different limit time.

The programming method is as follows:

1) Simulate the first Prototype list (by running the "Prototype Planning" version 3 with the first limit time. (`PrototypePlanning3`)
2) Generate the statistics of this simulation and write them in the saving file given. (`GenerateStats`)
3) Delete the Prototype list and regenerate the Test list so can be used in the next simulation. (`RegenerateData`)
4) Update the value of the Progress bar. (`PerformStepInProgressbar`)

❖ Repeat this programming method (from 1 to 4) until all the limit times are used.

For a better understanding, in *Figure 21* there is the sequence diagram of the evaluation of version 3.

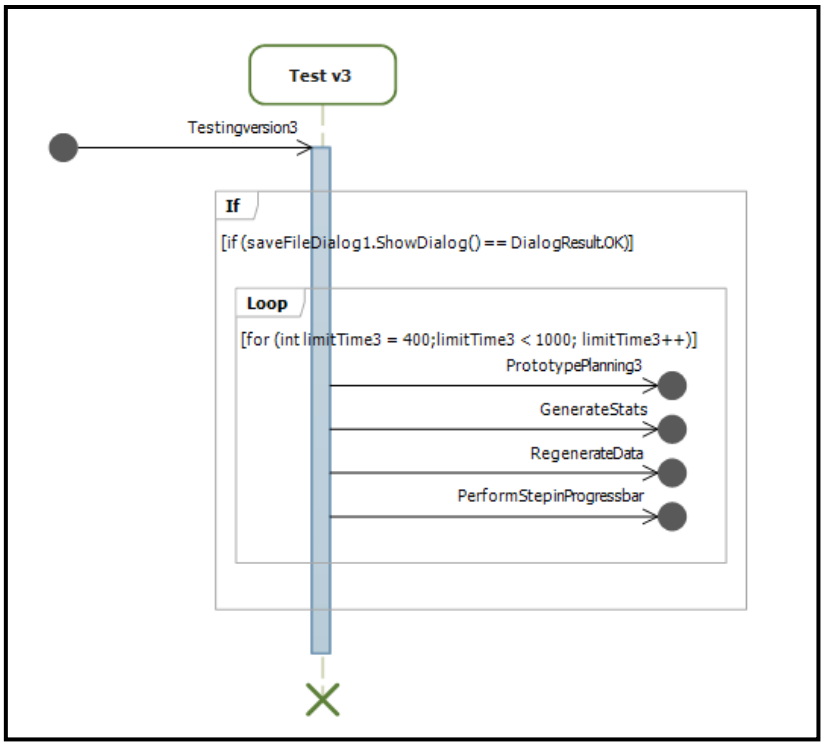

Figure 21. Sequence diagram of Test "Prototype Planning" v3.

The main intern functions, which this sub-application is working with, are shown in *Figure 22*.

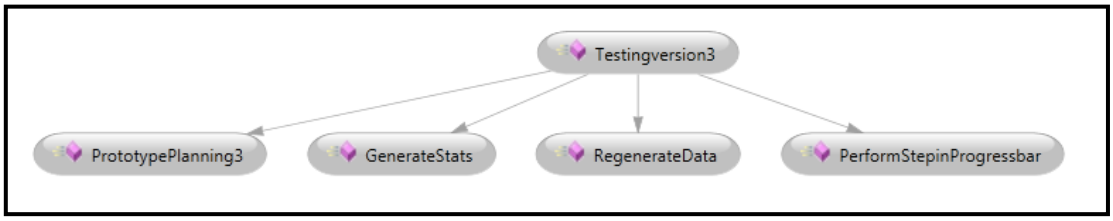

Figure 22. Functions used in Test "Prototype Planning" v3.

5.3.7. "Prototype Planning" v4

This fourth version of the application, called "Prototype Planning" v3 is running by clicking on the 

button and takes into account these hypotheses:

- ✓ The maximum duration of a Prototype is the limit time given by the user
- ✓ The Prototypes are assigned to one Prototype if the constraints are compatible and the combined duration does not exceed the limit time.
- ✓ The starting point of every Prototype has an offset with the previous Test.
- ✓ **After all Prototypes are generated, by making some changes, some Tests can be reallocated in other Prototypes.**

In this version is performed one optimization to get a solution closer to reality and it is executed through reallocating some Tests that are easy to place in a built Prototype. For example, if there are two braking Tests, which the only difference between them is they use different tyres, it is easy to change them, without building a new Prototype for that. In this way, with this version we are trying to say that, if some constraints are the difference between a built Prototype with a low duration and another one, then can be reallocated in the high-duration Prototype.

In this Test database, the constraints that are easy to change are 1, 2, 3, 4 and 8. If two Prototypes have some differences only in these constraints and the combined duration does not exceed the limit time, they can be joined.

The programming method is as follows:

1) Simulate the Prototype list by running the "Prototype Planning" version 3. (`PrototypePlanning3`)
2) Reallocate the Tests with a low duration. (`ReAllocateTests`)

For a better understanding, in *Figure 23* there is the sequence diagram of version 4.

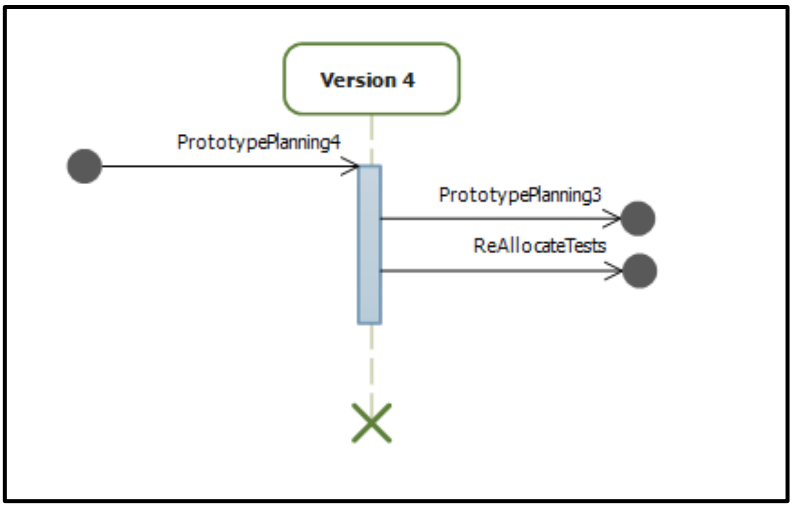

Figure 23. Sequence diagram of "Prototype Planning" v4.

The main intern functions, which this sub-application is working with, are shown in *Figure 24*.

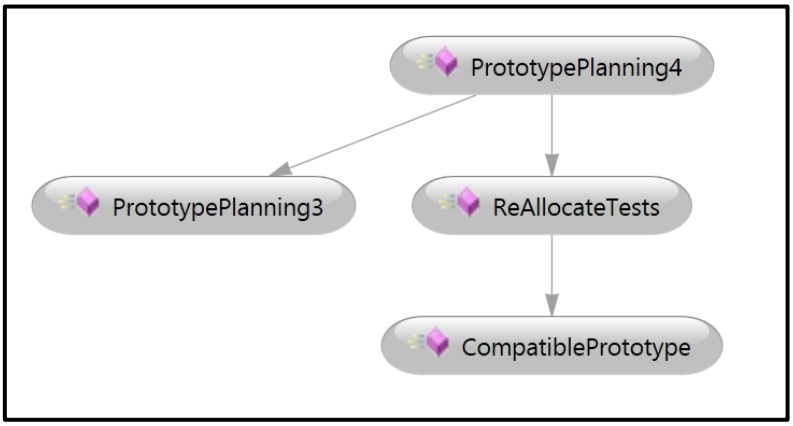

Figure 24. Functions used in Test "Prototype Planning" v4.

Two new functions are also used and they are described below:

- `static void ReAllocateTests (List<ushort> consselect, int lim)`

It reallocates the Tests with a low duration to some compatible Prototypes with a high duration. These compatibilities are guided by an exception list (consselect) and the time limit used (lim).

- `static Prototype CompatiblePrototype(Prototype p, int Limit, List<ushort> exception, int pnow)`

It searches for a compatible Prototype for the Test allocated in a low duration Prototype (p), through to comparing the restrictions to each built Prototype. If the restrictions exceptions (exception) are the only divergence between two Prototypes and the combined duration is lower than a limit time (Limit), there is compatibility.

5.3.8. Testing "Prototype Planning" v4

This sub-application is running by clicking in the 

button. Then, the user has to select the path of the saving stats and the application will generate a file with all the information.

If we go deeper, this sub-application is generating the statistics of all the Prototype lists, which are generated with the method of "Prototype Planning" v4 and with a limit time between 400 (the maximum duration of a Test is 370 days) and 1000 (random number, but quite high so the user can see the evolution of the number of Prototypes). It is therefore like an evaluation of the version 4 through 600 simulations with a different limit time.

The programming method is as follows:

5) Simulate the first Prototype list (by running the "Prototype Planning" version 4 with the first limit time. (`PrototypePlanning4`)
6) Generate the statistics of this simulation and write them in the saving file given. (`GenerateStats`)
7) Delete the Prototype list and regenerate the Test list so can be used in the next simulation. (`RegenerateData`)
8) Update the value of the Progress bar. (`PerformStepInProgressbar`)

❖ Repeat this programming method (from 1 to 4) until all the limit times are used.

For a better understanding, in *Figure 25* there is the sequence diagram of the evaluation of version 4.

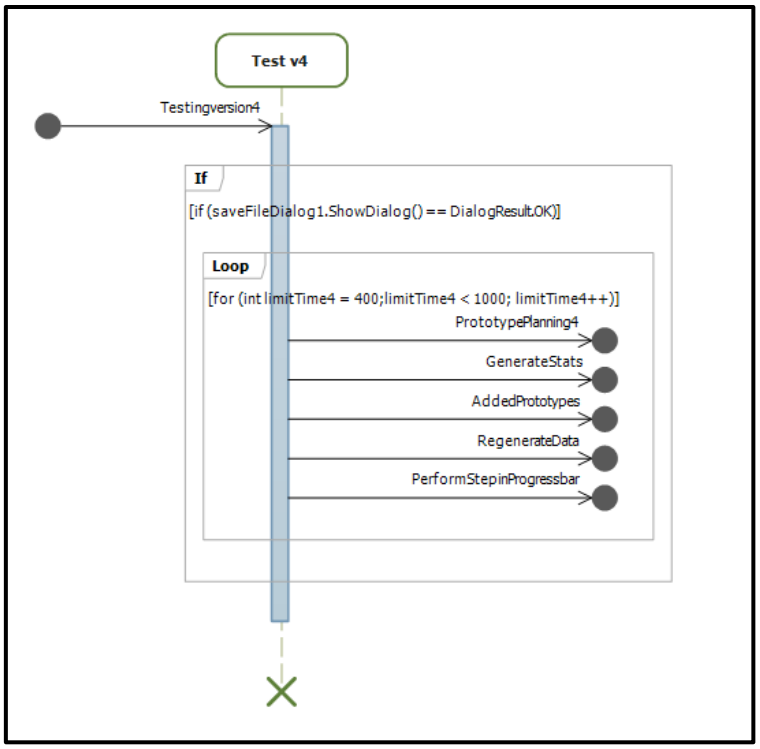

Figure 25. Sequence diagram of Test "Prototype Planning" v4.

The main intern functions, which this sub-application is working with, are shown in *Figure 26*.

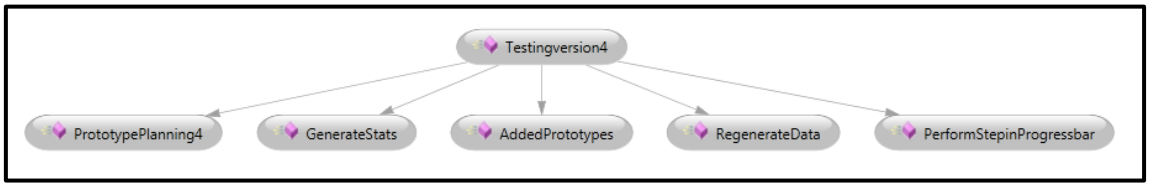

Figure 26. Functions used in Test "Prototype Planning" v4.

A new function is used and therefore is described below.

- `static void AddedPrototypes()`

It counts the amount of Prototypes with reallocated Tests.

5.4. Generating and Comparing Results

In this section, the lists of Prototypes are generated by running the different applications and all the results are going to be analyzed. The aim of this section is not only to present the results and statistics, but also to give a programming or practical explanation (whichever is applicable) of every phenomena occurring.

The results generated are based in a random solution with a limit time of 450 days. This value has been taking to be away from the starting limit time (380) and not so far so the solution is consistent.

When general statistics are compared is based on the statistics of the sub-applications Test "Prototype Planning" v1, Test "Prototype Planning" v2, Test "Prototype Planning" v3 and Test "Prototype Planning" v4. These sub-applications generate a plenty of statistics that are very useful to understand the behavior of the programming method and much easy to compare results and see how a new version is improving the previous one or how can influence the new programming method in the solution.

To analyze and interpret the general statistics, a working area has to be defined. Therefore, we run the first and second version of "Prototype Planning" with a sky-high limit time to note when the solution becomes unstable or monotone

The result of that simulation is presented in *Figure 27*:

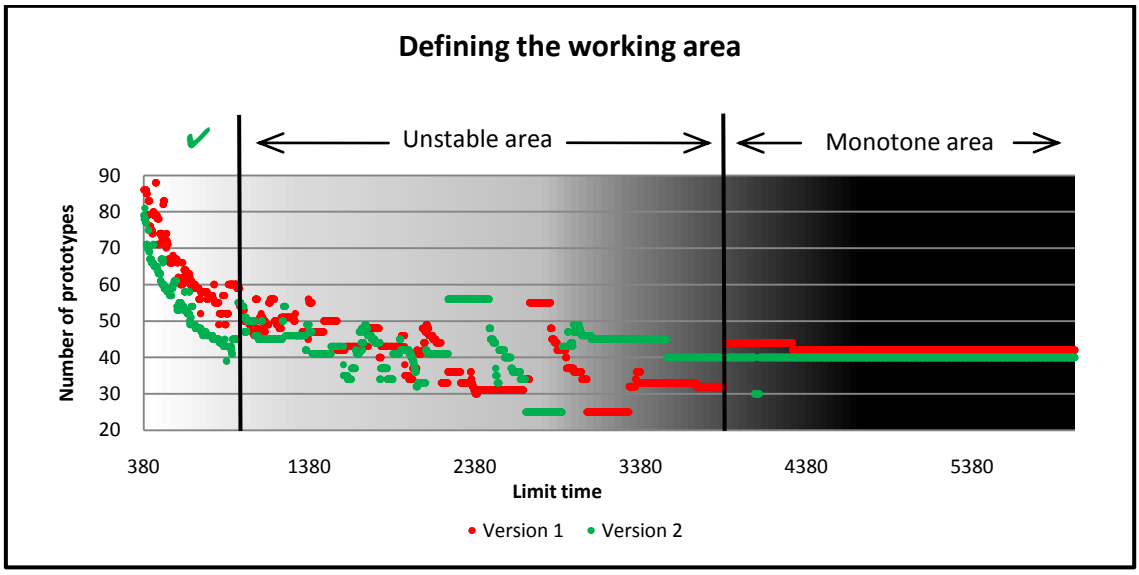

Figure 27. Defining the working area of the solutions.

In *Figure 27* can be appreciated that there is a monotone area, which can be said that starts around a time limit of 4000 days. From this point, the solution becomes always the same and it has no value for our study. However, it has mathematical sense as long as it is the limit value, which all further solutions are taking.

It can be also appreciated that between a limit time of 900 and 4000 days, the solution becomes unstable. The number of Prototypes almost does not depend on the limit time, but in the heuristic. This area has therefore no sense for the solution study.

Finally, we are defining our working area between 380 and 900 days, because here the solution in stable and with a mathematical regression. Consequently, all the study that is further coming is working within this limit time.

To break down the solutions and statistics, the analysis of the results is going to be divided in two sections:

- **Analysis of the Improvement of Version 2 Over 1.**
  In this section are compared the first two versions, through a plenty of statistics. Here is also explained the impact of the upgrading of version 2. The analysis is presented in Section 5.4.1.

- **Evaluating the Modification of Version 3.**
  In this section, the changes that version 3 incorporates are analyzed by being compared with version 1 and especially with version 2. The evaluation is presented in Section 5.4.2.

- **Evaluating the New Hypothesis of Version 4.**
  In this section is analyzed the impact to the solution of the new hypothesis. The explanation is based essentially on comparing version 3 and 4, but version 4 is also contrasted with version 2. The evaluation is presented in Section 5.4.3.

5.4.1. Analysis of the Improvements of Version 2 Over 1

The second version incorporates an optimization over version 1 that consists in applying the Tests without constraints at the end of the simulation. In this section, is going to be checked if this implementation has repercussion on the number of Prototypes to be built.

As it was said in the previous section, is important to remember that all the statistics and solution study is based in the working area delimitated by limit times between 380 and 900 days.

The number of Prototypes to be built according to each version is as it follows.

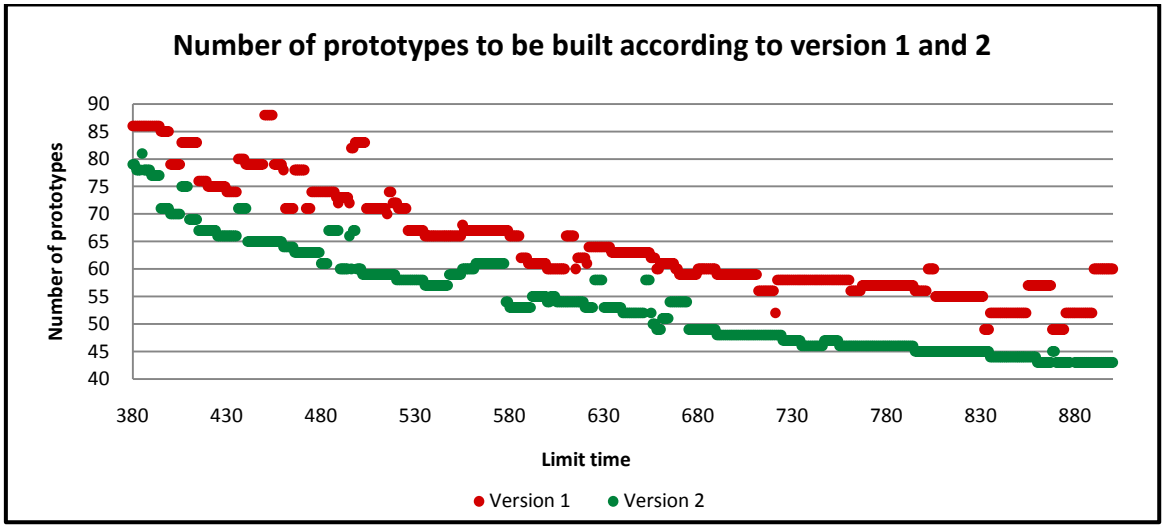

Figure 28. Number of Prototypes to be built according to version 1 and 2.

As it can be appreciated in *Figure 28*, the optimized version generates a better solution for every limit time. Through all the limit times tested, the version 2 has a lower number of Prototypes than version 1.

For a better understanding, a deeper analysis is going to be performed. First, we are looking at the solution behavior, where the value series does not draws a straight line, but one curve function. The next step is therefore to see which is the regression that better fits to the values and that is the polynomial regression (2nd grade) shown in *Figure 29*.

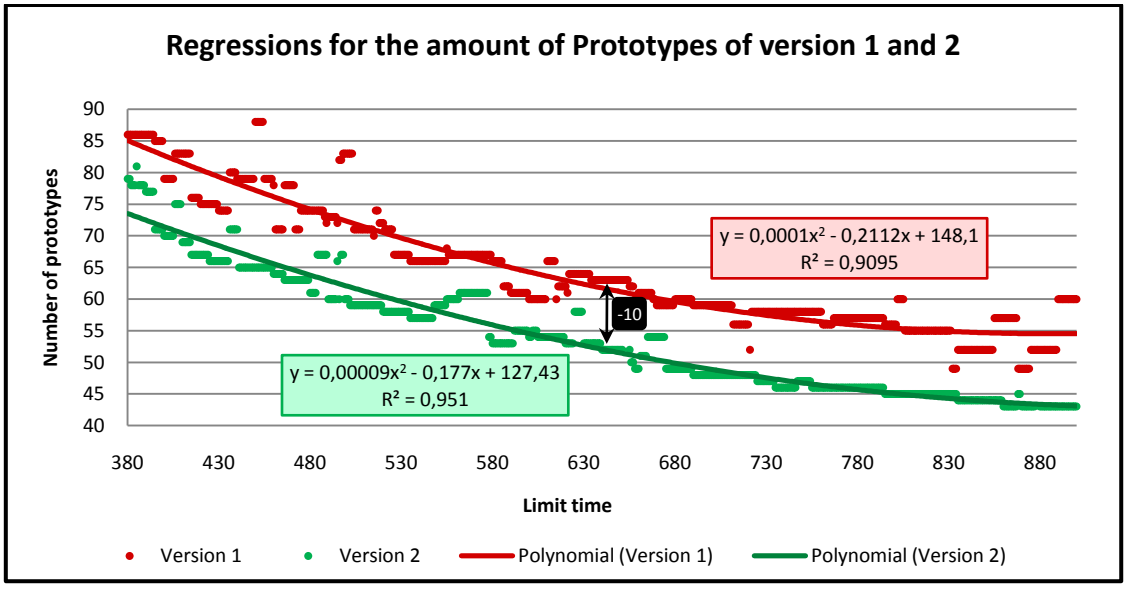

Figure 29. Polynomial regressions for the number of Prototypes from version 1 and 2.

Figure 29 shows the selected regression for the version 1 and 2 series. To decide which regression is the one that fit the best, all the other typical functions (logarithmic, exponential, linear, etc.) have been tried. Consequently, it has been selected the polynomial function because is the function, which describes better both version 1 and 2 series.

The mathematical functions are:

Version	Polynomial regression	R^2
Version 1	$y = 0{,}0001x^2 - 0{,}2112x + 148{,}1$	0,9095
Version 2	$y = 0{,}00009x^2 - 0{,}177x + 127{,}43$	0,951

Table 1. Polynomial regressions of the number of Prototypes of versions 1 and 2.

As can be appreciated in *Table 1*, the regressions are rather accurate to the series.

It can be also appreciated in *Figure 29* that there is an approximate difference of 10 Prototypes between both solutions. To best viewing the difference of the number of Prototypes to be built between version 2 and 1, in *Figure 30* it is shown the improvement that version 2 has over version 1.

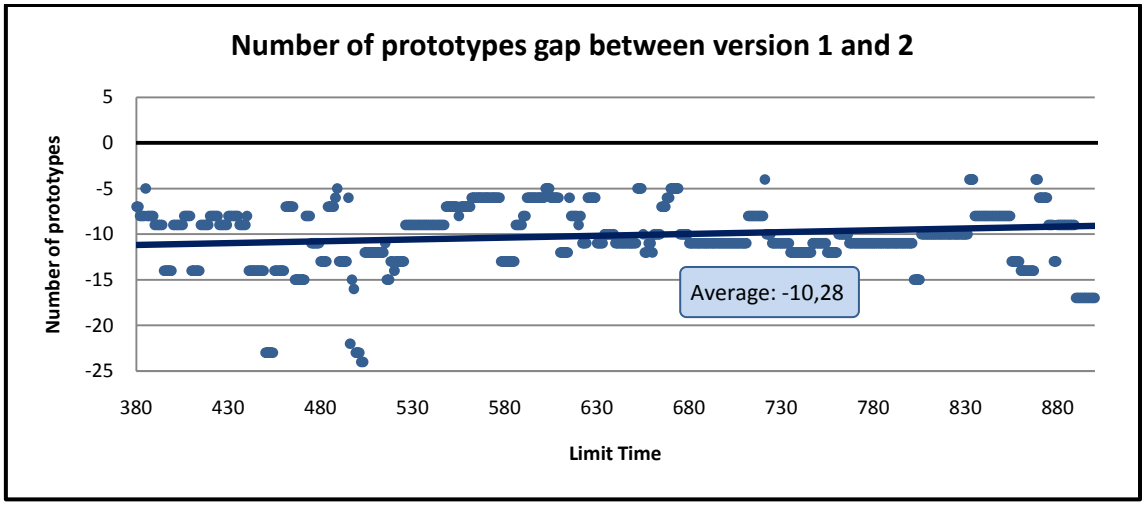

Figure 30. Number of Prototypes gap between version 1 and 2.

Now can be said without any doubt that version 2 gives a better solution than version 1. As the linear regression of *Figure 30* shows and as it has been calculated mathematically, there is an improvement of more than 10 Prototypes less to be built with the second version. In terms of economy, this fact can be a huge gap of budget between doing the planning of the Prototypes by following version 1 or version 2.

However, the solution cannot be approved by only heeding the number of Prototypes to be built and we can take more profit of the statistics, which testing sub-applications offer, to validate the solution and understand the reason why a programming method is better than another one. That is why we are looking now at the duration of the Prototypes generated with each version.

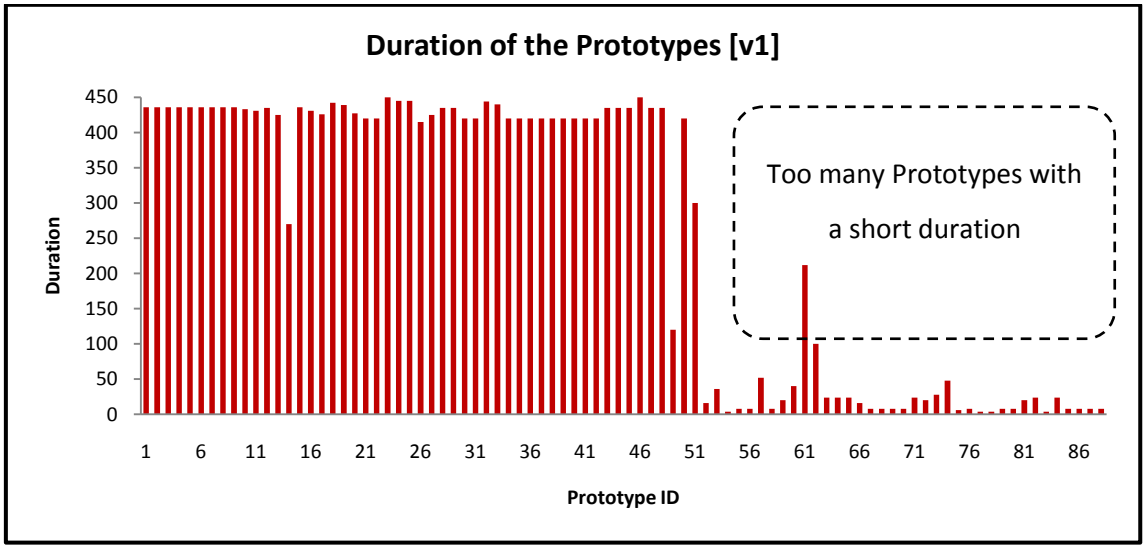

Figure 31. Duration of the Prototypes generated with version 1.

As can be appreciated in *Figure 31*, the simulation with version 1 generates 88 Prototypes. If we start the analysis, can be observed that after Prototype number 51, all the Prototypes have a short duration. Therefore, there are 25 Prototypes (if we remove the Prototype number 61 and 62 because of their higher duration) that their duration is lower than 50 days. Therefore, it is a big problem to the Prototype planning, because many Prototypes have a low charge of Tests, but they still need to be built (with their cost impact).

By version 2, this problem is almost solved. As can be appreciated in *Figure 32*, only the last 8 Prototypes have duration under 50 days.

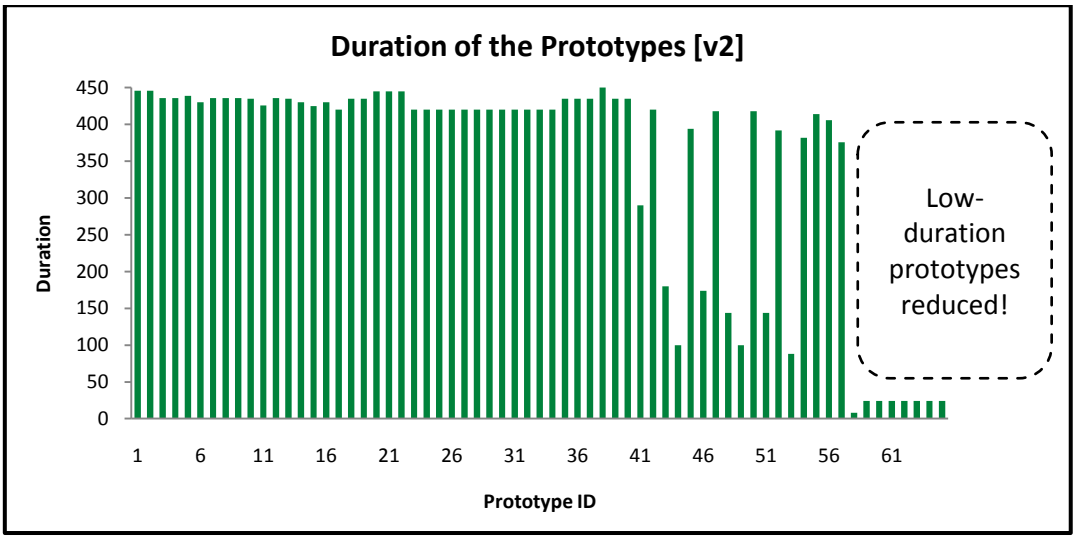

Figure 32. Duration of the Prototypes generated with version 2.

The version 2 generates for the same time limit that is used for version 1, a number of Prototype much lower: 65 Prototypes.

If we look at *Figure 31* and *Figure 32*, can be understood one of the reason of this improvement: the last Prototypes of the list have a higher duration in version 2 than in version 1. Therefore, by re-sorting the list of Tests and applying a right programming method, the duration of the last Prototypes can be really improved.

As this low duration Prototypes are the biggest problem of the simulation, in the next chart are analyzed the minimum duration Prototypes of every simulation from both version 1 and 2.

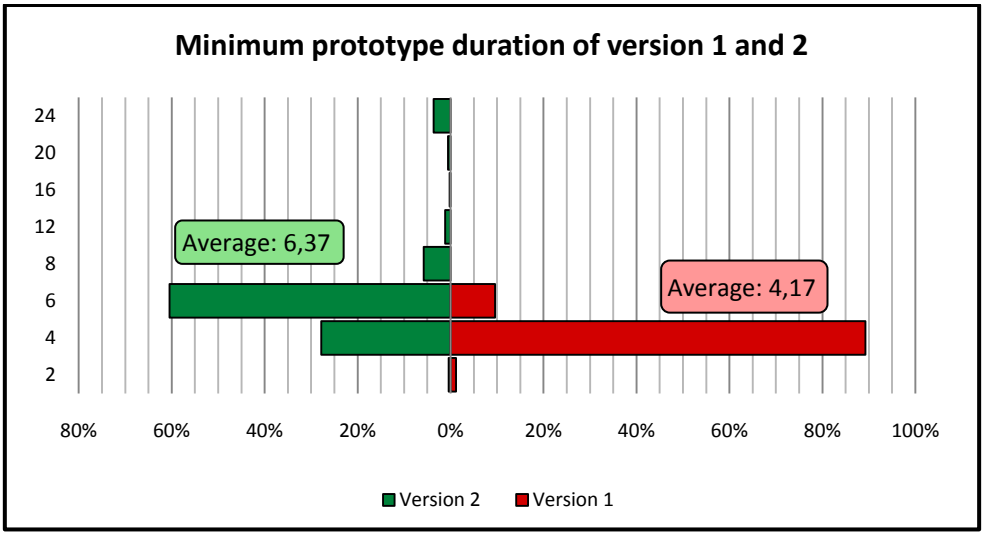

Figure 33. Analysis of the minimum duration Prototypes of version 1 and 2.

To support the graphical presentation is also attached *Table 2* with the information:

Minimum duration	Version 1	%	Version 2	%
2	6	1,15	2	0,38
4	465	89,25	145	27,83
6	50	9,59	315	60,46
8	0	0	30	5,75
12	0	0	6	1,15
16	0	0	1	0,19
20	0	0	3	0,58
24	0	0	19	3,65
Average value	**4,17**		**6,37**	

Table 2. Minimum duration Prototypes of version 1 and 2.

The minimum duration of a Prototype list also defines how the Prototypes are generated. If there is a high duration in the Prototype with the minimum duration, is possibly a symptom of good performance. For example, the version 2 has a 3,65% of the simulations (19 out of 521), with a minimum duration Prototype of 24 days. This means that the solution generated will be for sure a good solution.

By following the line of explaining why the second version improves the solution generated by the first version, now it is time to look at the number of Prototypes with only one Test applied. As it has been appreciated in the Prototypes duration charts, there are many Prototypes at the

end of the list with a low duration. If some of those Prototypes have a low number of Tests applied, we could decide that those Prototypes can be maybe created in a different way to optimize the solution. A Prototype with only one Test applied is detrimental to the planning in terms of time and budget.

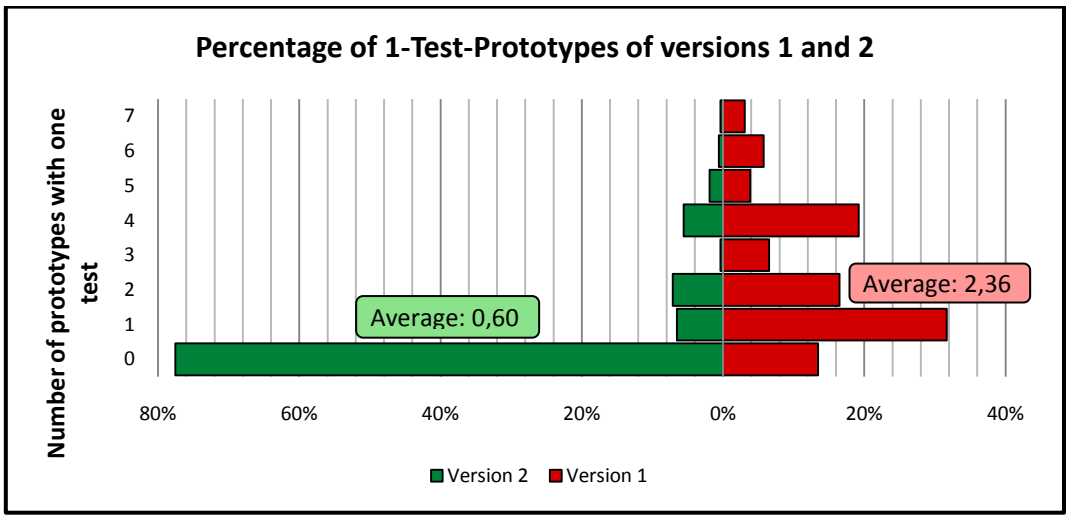

Figure 34. Percentage of Prototypes with only one Test applied of version 1 and 2.

Figure 34 shows an interesting observation: almost 78% of the Prototype lists generated with version 2 has no Prototype with only one Test applied. In addition, if we look at the average value, with the first version we get that a Prototype list has an average value of 2,36 Prototypes with only one Test, while in second version the average value is only 0,60. If we have many 1-Test-Prototypes in a Prototype list, the solution will not be as accurate as desired.

To support the graphical presentation is also attached the table with the information:

1-Test-Prototypes	Version 1	%	Version 2	%
0	70	13,44	404	77,54
1	165	31,67	34	6,53
2	86	16,51	37	7,10
3	34	6,53	2	0,38
4	100	19,19	29	5,57
5	20	3,84	10	1,92
6	30	5,76	3	0,58
7	16	3,07	2	0,38
Average value	**2,36**		**0,60**	

Table 3. Number of Prototype lists with Prototypes with only one Test applied of version 1 and 2.

The last parameter that is going to be used to explain both solutions is how close to the limit time is the length of the maximum duration Prototype.

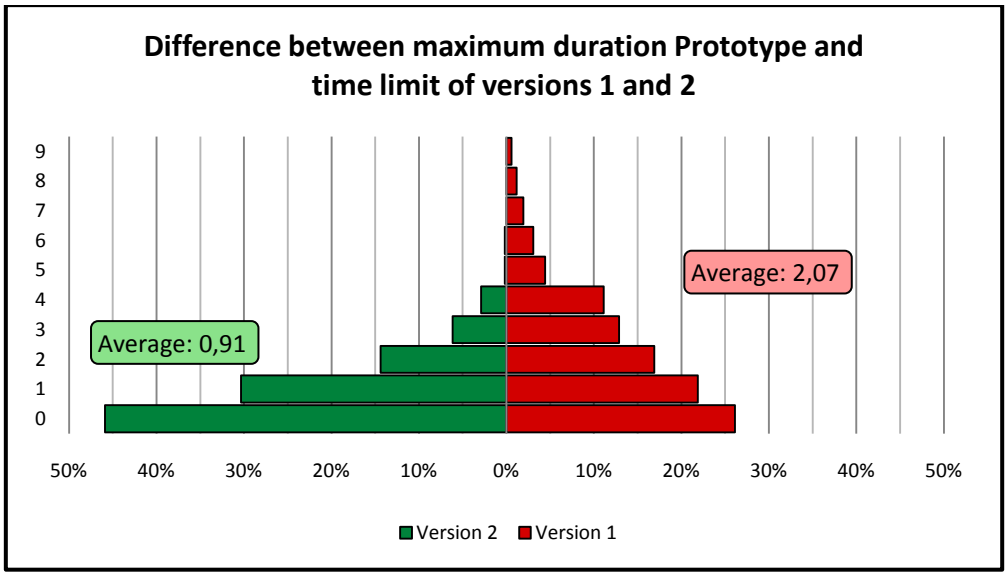

Figure 35. Difference between maximum duration Prototype and time limit of version 1 and 2.

With this last chart can be appreciated that with version 2, the difference between the duration of the maximum duration Prototype and the time limit is lower than in version 1. Consequently, version 2 gives a more accurate solution by being closer to the limit time than version 1. This can mean less low-duration-Prototypes at the end of the Prototype list.

To support the graphical presentation is also attached the table with the information:

Difference	Version 1	%	Version 2	%
0	136	26,10	239	45,87
1	114	21,88	158	30,32
2	88	16,89	75	14,40
3	67	12,86	32	6,14
4	58	11,13	15	2,88
5	23	4,41	1	0,19
6	16	3,07	1	0,19
7	10	1,92	0	0
8	6	1,15	0	0
9	3	0,56	0	0
Average value	**2,07**		**0,91**	

Table 4. Difference between maximum duration Prototype and limit time of version 1 and 2.

After the intensive analysis of the results, we can conclude that version 2 is much better than version 1, not only because of the number of Prototypes to be built, but also because the disposition of the allocated Tests in the Prototypes follows a better distribution. With this distribution, we avoid 1-Test-Prototypes, to be closer to the limit time and to have a higher duration of the lowest length Prototype.

5.4.2. Evaluating the Modification of Version 3

In this section is going to be evaluated the changes that implements version 3 in the solution. Although the solution that version 3 is much different from version 2 or 1, is going to be analyzed to understand the differences.

Version 3 generates a list of 91 Prototypes and their duration is as is follows:

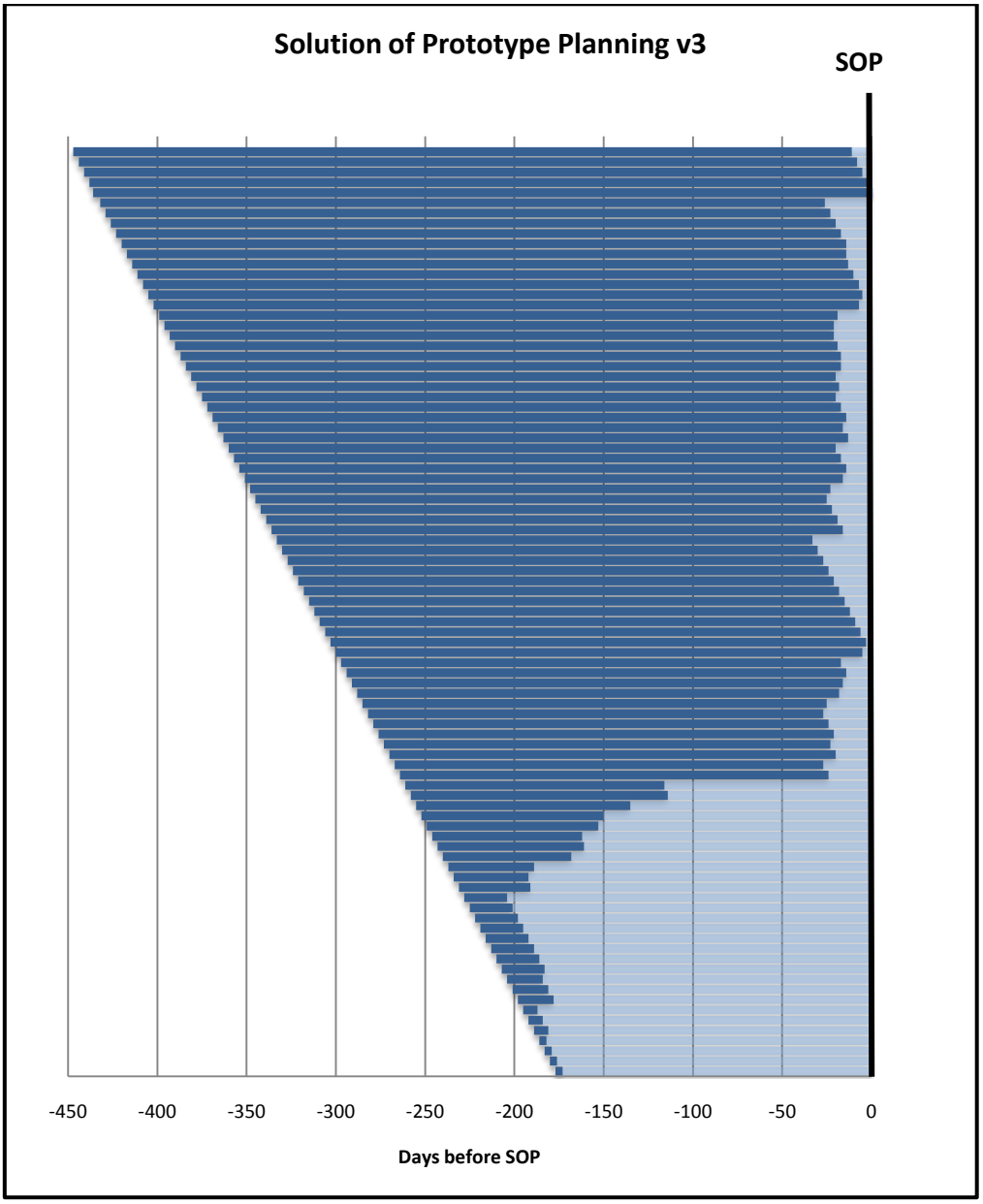

Figure 36. Prototype planning solution by running version 3.

In *Figure 36*, each horizontal bar represents a different Prototype to be built. The dark blue bars represent the duration of the Prototype and the light blue bar represent the days until the "Start of production" (SOP).

In version 3, not only the Prototype defining is done, but also the scheduling of the Prototypes. This implementation gives the generated solution an approach to the reality of the Prototype planning.

As can be appreciated in the next Figure, the Prototype's duration is very accurate until Prototype number 66, where the constraints of the Tests bring the last Prototypes to a low duration. Therefore, this becomes a problem to solve in following versions.

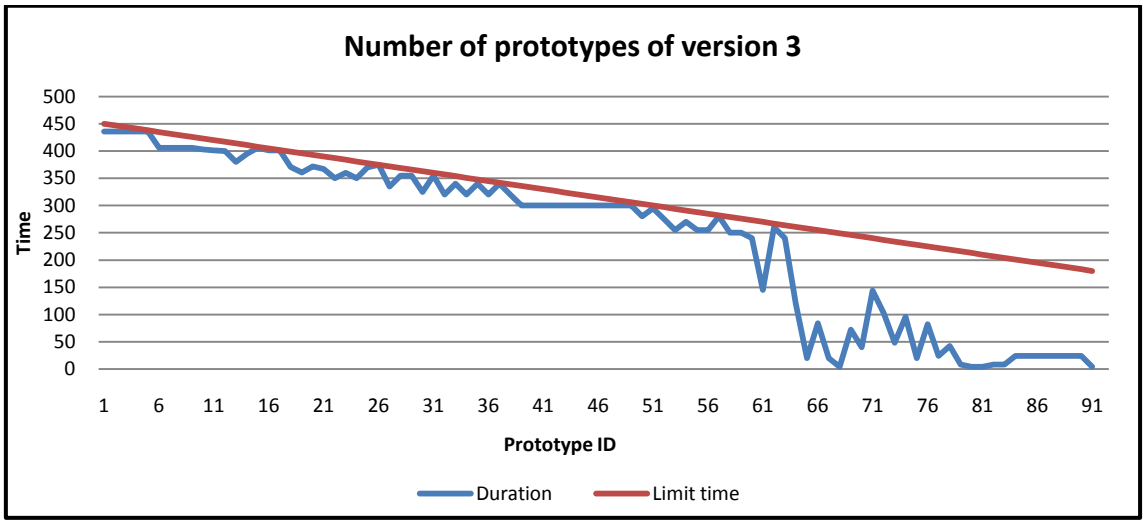

Figure 37. Number of Prototypes generated with version 3.

To perform a deeper analysis it is necessary to see what the number of Prototypes generated by this version, is compared to versions 1 and 2, in *Figure 38*.

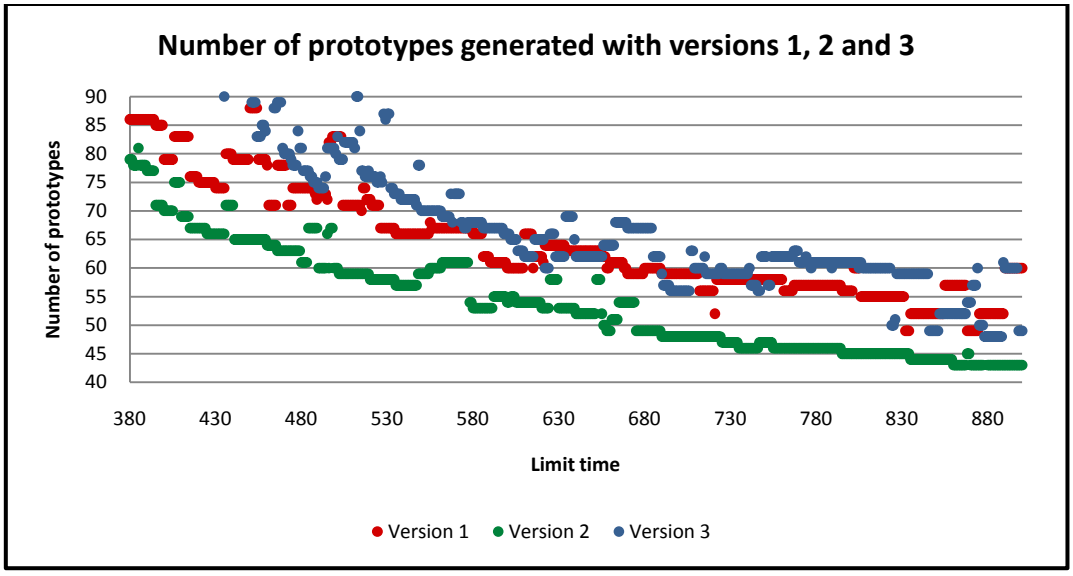

Figure 38. Number of Prototypes generated with versions 1, 2 and 3.

As can be observed, the number of Prototypes to be built with this version is much higher than version 2 and even higher than version 1.

Anyway, this third version is difficult to compare with the other ones because is giving a different solution. In fact, is giving a solution for another problem. Because of this, the number of Prototypes to be built is much higher than in the previous versions. This can be explained because the limit time is being reduced through the whole simulation and therefore the solution has to be higher.

A regression with the values can also be performed as it is shown in *Table 5*.

Version	Polynomial regression	R^2
Version 3	$y = 0{,}0002x^2 - 0{,}3717x + 210{,}9$	0,9

Table 5. Polynomial regression of the number of Prototypes of version 3.

In *Figure 39* is incorporated the polynomial regression to the results scenario.

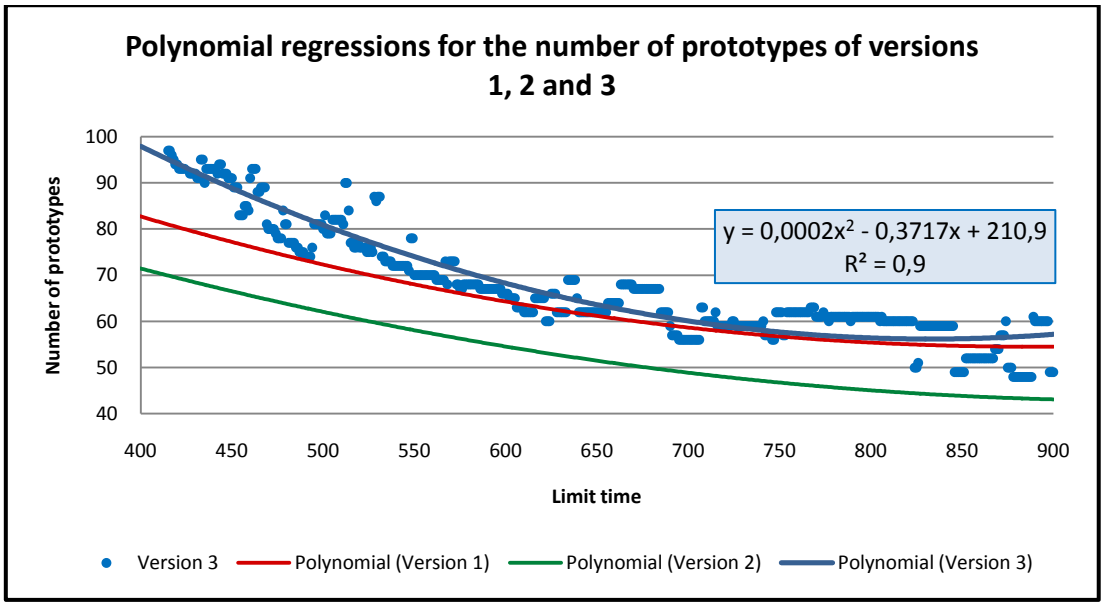

Figure 39. Polynomial regression of the number of Prototypes generated of versions 1, 2 and 3.

This third version gives an amount of Prototypes much higher because of the point commented. However, in *Figure 40* can be appreciated the difference regarding 1-Test-Prototypes.

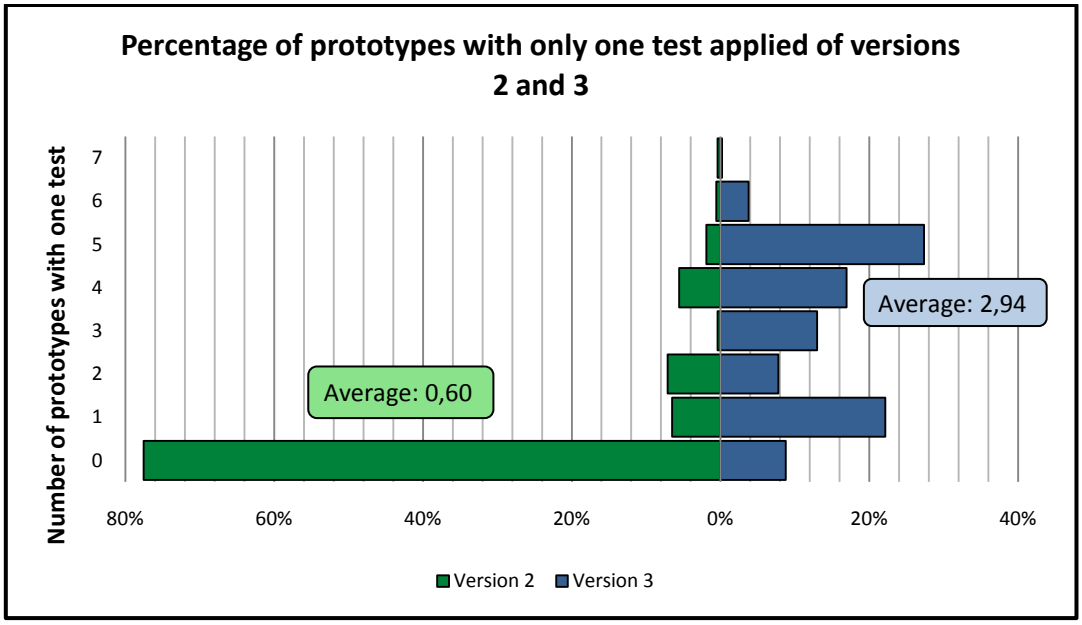

Figure 40. Percentage of Prototypes with only one Test applied of versions 2 and 3.

The nearly perfect optimization of the space that has version 2 cannot be reached by version 3. Therefore, there are many Prototypes with only one Test applied.

For a better understanding, the table with all the values of the Prototypes with only one Test applied is presented below. It has also the information regarding 1-Test-Prototypes of version 1, although is not shown in *Figure 40*.

1-Test-Prototypes	Version 1	%	Version 2	%	Version 3	%
0	70	13,44	404	77,54	404	77,54
1	165	31,67	34	6,56	34	6,56
2	86	16,51	37	7,10	37	7,10
3	34	6,53	2	0,38	2	0,38
4	100	19,19	29	5,57	29	5,57
5	20	3,84	10	1,92	10	1,92
6	30	5,76	3	0,58	3	0,58
7	16	3,07	2	0,38	2	0,38
Average value	**2,36**		**0,60**		**2,94**	

Table 6. Number of Prototype lists with Prototypes with only one Test applied of version 1, 2 and 3.

The study of the minimum duration Prototype is also done, by comparing version 2 and 3 in *Figure 41*.

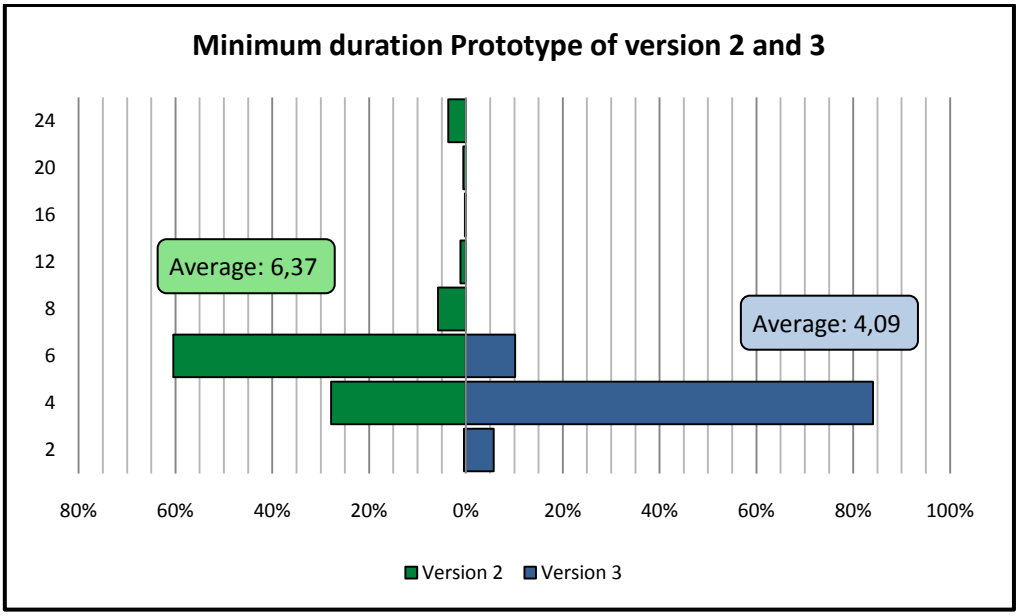

Figure 41. Length of the minimum duration Prototype of version 2 and 3.

To support the chart, below are presented the values of the minimum duration Prototypes of versions 1, 2 and 3.

Minimum duration	Version 1	%	Version 2	%	Version 3	%
2	6	1,15	2	0,38	30	5,76
4	465	89,25	145	27,83	438	84,07
6	50	9,60	315	60,46	53	10,17
8	0	0	30	5,76	0	0
12	0	0	6	1,15	0	0
16	0	0	1	0,19	0	0
20	0	0	3	0,58	0	0
24	0	0	19	3,65	0	0
Average value	**4,17**		**6,37**		**4,09**	

Table 7. Minimum duration Prototypes of version 1, 2 and 3.

As the analysis of the version 3 comes to an end, it is remarkable that the provided solution fits rather good in the planning and scheduling of Prototypes, after taking a look at the generated solution and how was developed. It cannot be compared with the other versions in terms of results (for example, putting side by side the amount of Prototypes that every version generates) because the boundary condition that implement version 3 is too restrictive and therefore, gives a solution much different.

However, the procedure of a programming application is to incorporate betterments and optimizations that make the solution every step more complex and sophisticated. There can be also steps, where the solution becomes worse than in the previous one, but if it is well planned, will be an improvement for next versions.

The Prototype planning is, as it is said in the name, a planning of Prototypes. Therefore, a planning without a time scenario is not possible. That is why this third version is important to the further development of the application. Consequently, version 3 represents the start of the planning and scheduling of the “Prototype planning”, which is the searched final goal.

In addition, version 3 is based on the programming method of version 2 in that it uses the idea of the auxiliary list of Tests. The trouble that version 3 finds in the way is that although it can be based on version 2, the condition of decreasing the limit time, brings the programming to a new dimension. From version 1, it is relatively easy to implement some improvements to the code by adding new functions or improving some existing ones, but the incorporation of the “scheduling factor”, which version 3 implements, makes hard the improvement from version 2. Therefore, a new programming method had to be thought to solve this difficulty.

5.4.3. Evaluating the New Hypothesis of Version 4

This last version of the "Prototype Planning" application incorporates the new hypothesis based on reallocating the lowest duration Prototypes to obtain a more realistic solution. As it is based on version 3, all further explanation is going to be through comparing version 4 with that previous one.

If we start with the evaluation, in *Figure 42* is shown the Prototype list generated with this version.

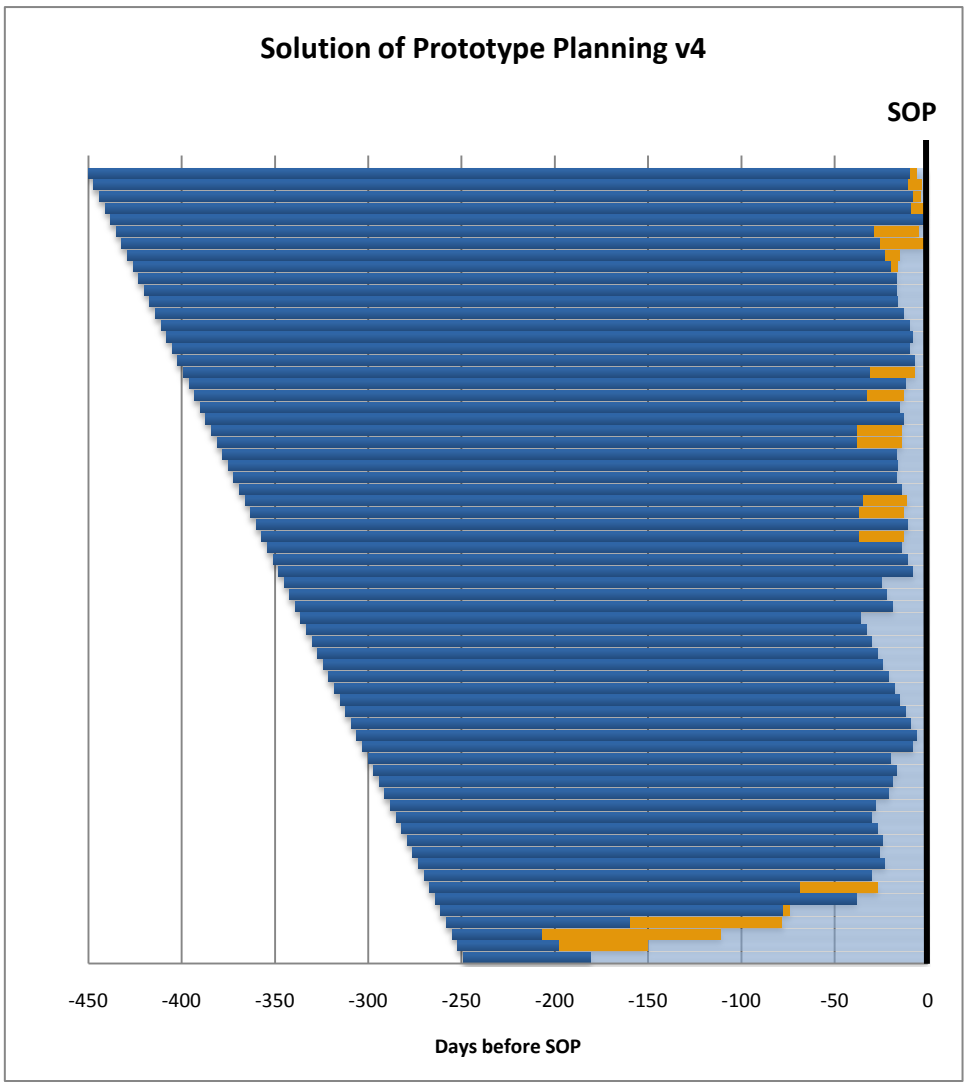

Figure 42. Prototype planning solution by running version 4.

First, it is important to define what every color means. As it was explained in Section 5.4.2, dark blue means the Prototype duration and light blue is the unused time, but this version

incorporates a Prototype reallocation, which has been represented with orange. This orange bar represents the reallocated Tests that had a low duration on version 3.

Consequently, it can be easily appreciated that version 4 has a huge improvement in the generated solution, over version 3. If in version 3 we had 91 Prototypes with the limit time of 450 days, this version provides us a solution of 69 Prototypes. Therefore, the reallocation system has reduced the amount of Prototypes in 22 less. However, we are going to analyze this decreasing deeply.

In *Figure 43* is shown the difference that version 4 presents over version 3, regarding the amount of Prototypes to be built.

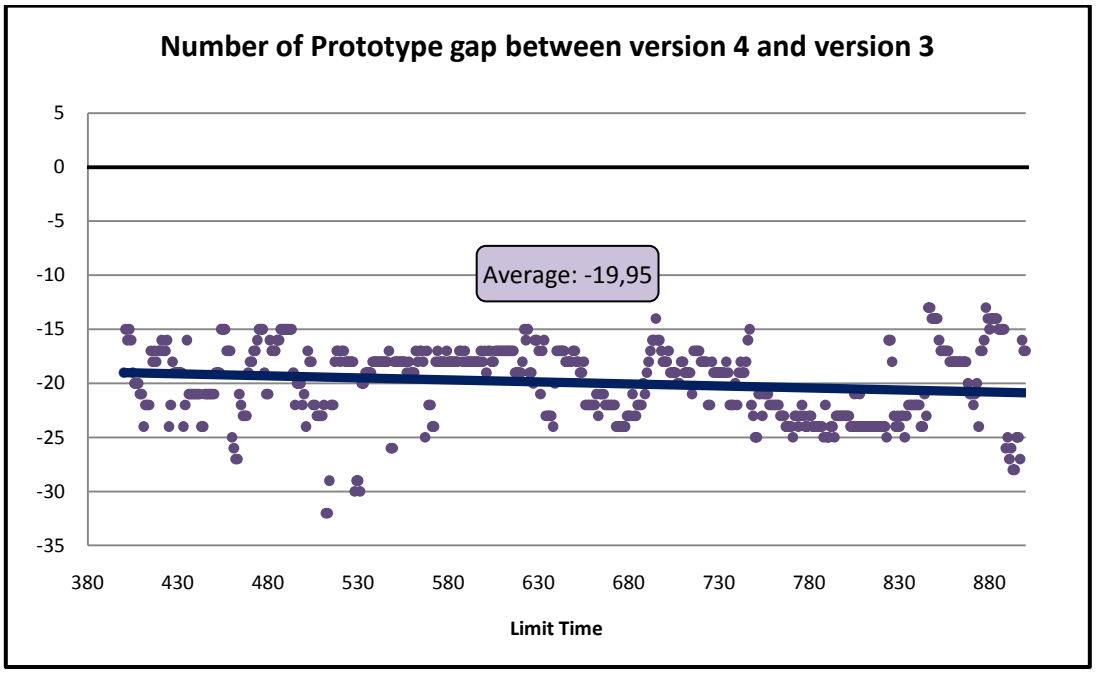

Figure 43. Number of prototype gap between version 4 and 3.

As can be appreciated in this figure, the generated solution with version 4 provides a much better solution, as long as the decreasing on the number of prototypes is nearly of 20 Prototypes less.

In addition, as it has been performed for the previous versions, we can draw a regression in order to be easier compared with the other versions, as it is shown in *Figure 44*.

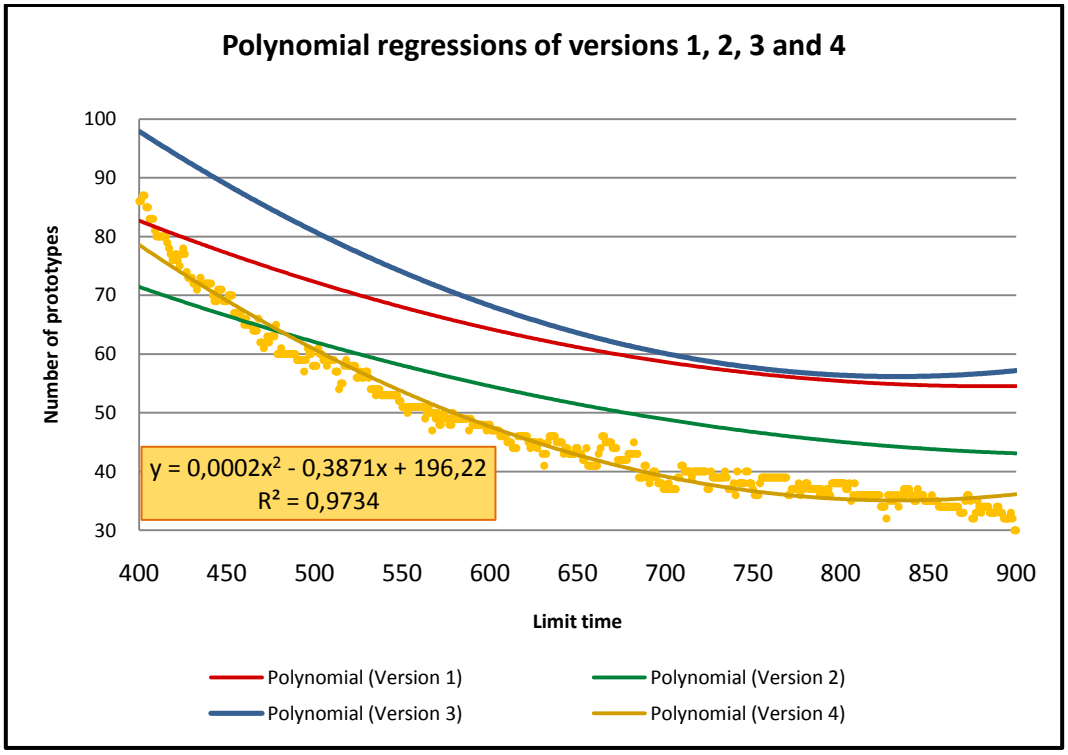

Figure 44. Regressions of version 1, 2, 3 and 4.

This figure shows that the amount of Prototypes has decreased and consequently, all the statistics should be better. Moreover, to support the graphical presentation below is the table with the regression applied:

Version	Polynomial regression	R^2
Version 4	$y = 0,0002x^2 - 0,3871x + 196,22$	0,9734

Table 8. Polynomial regression of the number of Prototypes of version 4.

In addition, this decreasing can be appreciated in the difference between the limit time and the duration of the Prototypes, as it is shown in *Figure 45* .

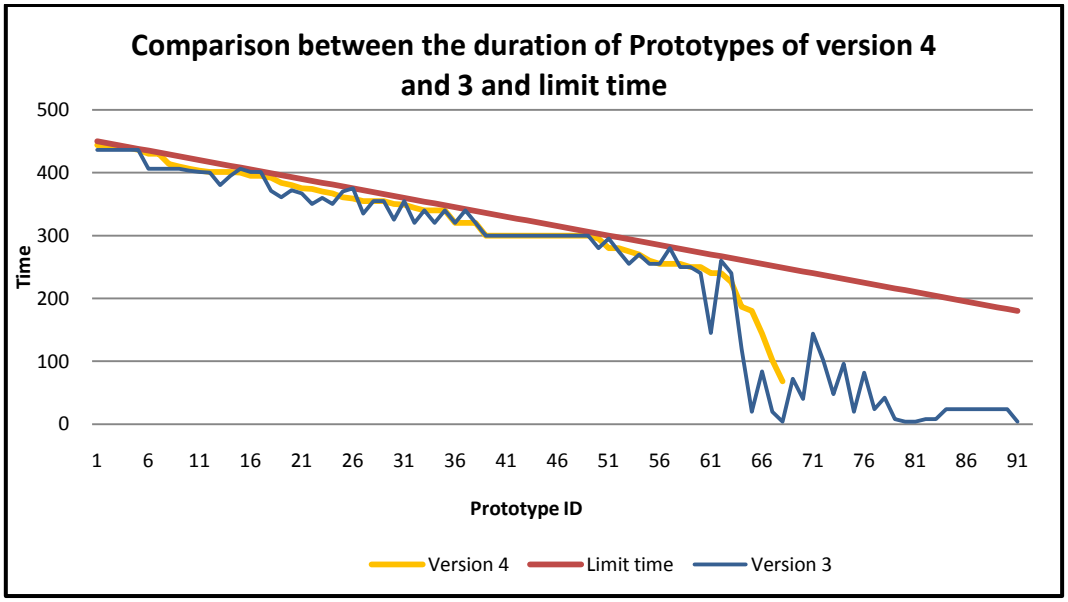

Figure 45. Comparison between the duration of Prototypes of version 4 and 3 and limit time.

Looking at the figure, can be easily appreciated that version 4 optimizes the duration of the Prototypes to be closer to the limit time.

For that reason, it has less low-duration-Prototypes as it is shown in *Figure 46*.

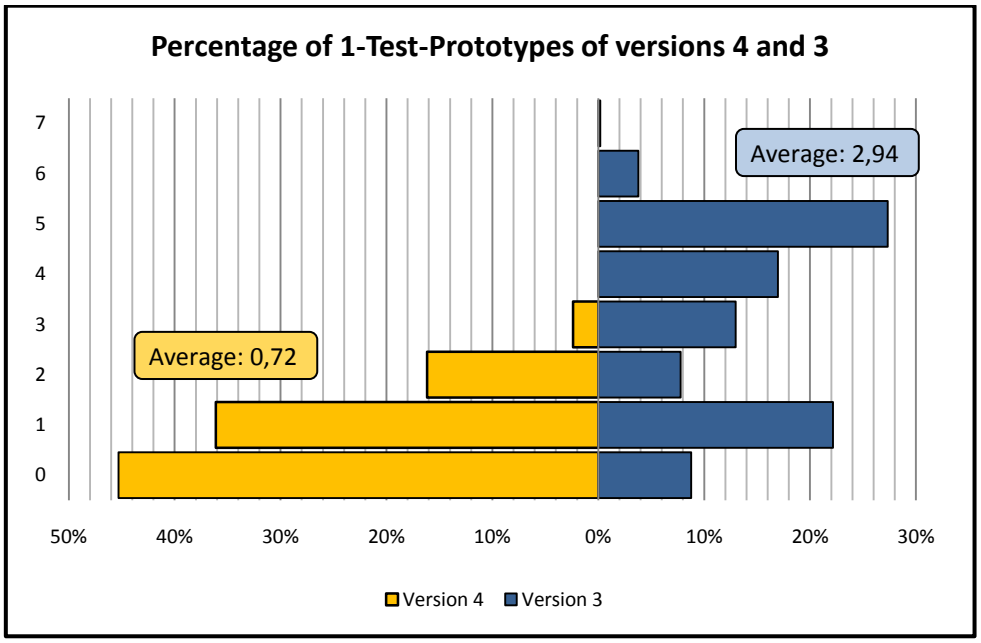

Figure 46. Percentage of 1-Test-Prototypes of versions 4 and 3.

The amount of Prototypes with only one test applied is close to zero. In fact, the average value (as it is shown in the figure and in XXX) is of 0,72 1-Test-Prototypes in a Prototype list. This

supports the idea commented before that this version pretends to remove this low duration prototypes.

1-Test-Prototypes	Version 3	%	Version 4	%
0	404	77,54	227	45,31
1	34	6,56	181	36,13
2	37	7,10	81	16,17
3	2	0,38	12	2,39
4	29	5,57	0	0
5	10	1,92	0	0
6	3	0,58	0	0
7	2	0,38	0	0
Average value	**2,94**		**0,72**	

Table 9. Number of Prototype lists with Prototypes with only one Test applied of version 3 and 4.

After all this analysis, we can conclude that the generated prototype list of version 4 is much better than version 3, but it is important to remember why is improving the previous version: the new hypothesis is a big change to the solution, so this result was expected.

Nevertheless, this fourth version gets closer to the reality and therefore, is a step to the front for further developments. If the aim of further improvements for this application is to get the solution closer to the reality, this hypothesis implemented in version 4 is the right technique to do it, as it improves version 3 in that way.

5.5. Ways of Improvement

If the final goal of the "Prototype Planning" is to create an application to become the optimal solution by taking all the possible parameters into account, the steps that have to be performed is the implementation of optimization of the programming method by taking the code to a more realistic solution every new version.

In my opinion, the next step in the way to create the perfect application for planning and scheduling Prototypes is to implement more conditions to the Tests such as:

- **Releasing dates**

  The Tests usually have a releasing date or maximum date in the calendar for a Test to be performed. That is done because perhaps a Test has to be performed before a day to start a new testing procedure or because some crashing Tests are going to be performed and everything has to be tested before.

- **Test dependencies**

  It is known that Tests have dependencies on other ones. Some Tests have to be performed before others are done. For example, a long driving Test has to be performed after a braking Test or a tyre control.

- **Priority order**

  A Test can have priority to be performed before another one. Maybe if a braking Test is performed, many Tests can be performed after that point and not before.

- **Differentiation between the time needed to perform a Test and the time to obtain the results**

  A Test can have a performing time of only a few hours but a hard work at the office after. It is important to know it, because another Test can start while an engineer is working at his desk.

- **Determinate if human participation is needed**

  If a Test can be run without one engineer performing it, human resource can be used in another job.

The next step can be in terms of scheduling. Some new parameters that I think they can be useful are:

- **Capacity teams to perform the Tests or build the Prototypes**
  If the programmer could know some information about the working days of the people, involved in the Test performing or the Prototype building, the solution would be more realistic.
- **Working days**
  Knowing the working calendar can be important to schedule some Tests.

Moreover, the final implementation that I would do is related to associated costs or any other quantification way to consider building a Prototype better than another one in terms of budget or required time.

- **Profit taken of building one Prototype for some defined Tests**
  Maybe building a Prototype for a purpose becomes a save of money because can be done in some specific conditions.

- **Overspending to build a defined Prototype**
  A Prototype can also have some impediments to be built in a defined way and doing it can become an overspending or maybe some particular components have incompatibilities.

By adding all this new parameters, the solution can be almost real. However, implementing these conditions can create a too difficult program so a powerful computer should be used.

The problem that these types of applications with so many variables and methods have is that they required powerful computers to run the application and a long time to become a solution is needed.

The perfect application is that one that gives an optimal solution within the shortest running time.

Conclusions

It can be concluded that finding the optimal solution is a long-haul travel, but as a first solution, it gives a rather good solution in its second version. The implementation of the scheduling system of version 3 supposed a big trouble for the improving of the amount of Prototypes, but it becomes a great step for further improvements of the application, as demonstrate fourth version, which gives the lowest amount of prototypes than previous versions.

The initial goal was to create an application that, using a list of Tests, was able to return a list of Prototypes to be built, and this goal was accomplished and some improvements were also made.

It has to be said, that the importance to get a low amount of Prototypes lies, specially, in the way they are initially assigned, because that point sets the base for the following programming method. Therefore, the more sophisticated the Tests assignment is, the better the solution the user gets.

To conclude, this application is far from being the definitive one, as more developments and conditions implementation should be done, but by adding more conditions, as it was said through the previous section, the final program can be created.

Acknowledgment

I would like to thank my supervisor, Dipl.-Inform. Christian Tesch, who has guided me throughout this project and whose encouragement was always there to support, not only my project, but also my integration in the city of Dortmund and the German culture.

Thank you to the Erasmus Program and the "*Escola Tècnica Superior d'Enginyeria Industrial de Barcelona"*, in special the people of the International Office of the university, who allowed me to accomplish my Final Project in Germany, as I always wanted, and also to the "*Technische Universität Dortmund"*.

I would also like to thank the support my family has given to me, not only financially but above all morally, during the months I have been abroad, in special to my mother and couple who have been a crucial support at all times.

Finally, thanks to those, who have been standing by my side in Dortmund, making my daily life in Germany easier and unforgettable.

This experience has taken me, not only to acquire knowledge about programming and the Prototype planning, but also to get in touch with a different culture and to improve my German and English.

References

[Ada98] ADAM, DIETRICH: *Produktionsmanagement*. Gabler, Wiesbaden, 1998.

[Che08] CHEN, DACHENG: *Mehrkriterielle evolutionäre Reihenfolgeplanung bei der bedarfsorientierten Prototypenerprobung im Fahrzeugbau*. Dortmund, Lehrstuhl für Algorithm Engineering, 2008.

[BPN01] BAPTISTE, P., PAPE, C.L. AND NUJITEN, W: *Constraint-Based Scheduling: applying constraint programming to scheduling problem*. Kluwer, 2001.

[BNR06] BEUME, NICOLA, BORIS NAUJOKS and GÜNTER RUDOLPH: *Mehrkriterielle Optimierung durch evolutionäre Algorithmen mit S-Metrik-Selektion*. In: R. MIKUT, M. REISCHL (Herausgeber): Proc. 16th Workshop Computational Intelligence, Pages 1–10, Karlsruhe, 2006. Universitätsverlag.

[Bru81] BRUCKER, PETER: *Scheduling. Akademische Verlagsgesellschaft*, Wiesbaden, 1981.

[CMM67] CONWAY, R.W., W.L. MAXWELL and L.W. MILLER: *Theory of scheduling*. Addison-Wesley, Reading, 1967.

[CW06] CLAUSEN, U. and J. WEBER: *Prototypenplanung im Nutzfahrzeugbau*. In: Automobiltechnische Zeitschrift : ATZ 108 (2006), Nr.9, Pages 740–744, 2006.

[DAPM00] DEB, K., S. AGRAWAL, A. PRATAP and T. MEYARIVAN: *A Fast Elitist Non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II*. In: Proceedings of the Parallel Problem Solving from Nature VI Conference, 16-20 September., Pages 849–858, Paris, France, 2000.

[DP01] DANNA, E. AND PAPE, C.L.: *Constraint and Integer Programming: Toward a Unified Methodology. Chapter Two: generic schemes for efficient and robust cooperative algorithms*. Pages 33–58. Kluwer, 2004.

[Gol89] GOLDBERG, DAVID E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, 1989.

[Hoo04] HOOKER, J.: *A hybrid method for the planning and scheduling*. Tepper School of Business, Paper 164. Pittsburgh, 2004.

[Lim08] LIMTANYAKUL, KAMOL: *Scheduling Tests on Vehicle Prototypes using Constraint Programming*. Dortmund, Robotics Research Institute, Dortmund University, 2008.

[MSDN] MSDN LIBRARY. URL: http://msdn.microsoft.com/es-es/library/ms123401.aspx. Consulted: May 2011.

[Meh05] MEHNEN, JÖRN: *Mehrkriterielle Optimierverfahren für produktionstechnische Prozesse*. Vulkan Verlag, Essen, 2005.

[Pin02] PINEDO, MICHAEL: *Scheduling - Theory, Algorithms, and Systems*. Prentice-Hall inc., Upper Saddle River, New Jersey, 2002.

[SCP05] SCHEFFERMANN, R., CLAUSEN, U. AND PREUSSER, A.: *Test-scheduling on vehicle prototypes in the automotive industry*. Manila, 2005.

[SD94] SRINIVAS, N. and KALYANMOY DEB: *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*. Evolutionary Computation, 2(3):221–248, 1994.

[Wei02] WEICKER, KARSTEN: Evolutionäre Algorithmen. Teubner, Stuttgart / Leipzig / Wiesbaden, 2002.

[Zac00] ZAKARIAN, A.: *Performance analysis of product validation and test plans*. Annual progress report, Center for Engineering Education and Practice, College of Engineering and Computer Science. University of Michigan-Dearborn, 2000.

Appendix I. List of Figures

All the figures in this document are of own production.

Appendix II. List of Tables

Statutory declaration

I herewith declare that I have completed the present thesis independently making use only of the specified literature and aids. Sentences or parts of sentences quoted literally are marked as quotations; identification of other references with regard to the statement and scope of the work is quoted.

Date:_____ Signature:_____

I grant the Technische Universität Dortmund the right to publish, reproduce and distribute my work, especially when it is to be presented to a third party for inspection.

Date:_____ Signature:_____