Comparison and visualization of robot program modifications
- Applied on ABB industrial robots at Volvo Cars Corporation

Sergio Heras Aguilar
# A THESIS SUBMITTED TO THE DEPARTMENT OF ENGINEERING SCIENCE

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE WITH SPECIALIZATION IN ROBOTICS

AT UNIVERSITY WEST

2017

<table>
<thead>
<tr>
<th>Date:</th>
<th>July 4, 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author:</td>
<td>Sergio Heras Aguilar</td>
</tr>
<tr>
<td>Examiner:</td>
<td>Bo Svensson</td>
</tr>
<tr>
<td>Advisor:</td>
<td>Anders Appelgren, HV, Mathias Sundbäck, VCC</td>
</tr>
<tr>
<td>Programme:</td>
<td>Master Programme in Robotics</td>
</tr>
<tr>
<td>Main field of study:</td>
<td>Automation with a specialization in industrial robotics</td>
</tr>
<tr>
<td>Credits:</td>
<td>60 Higher Education credits (see the course syllabus)</td>
</tr>
<tr>
<td>Keywords:</td>
<td>KPI, Rapid, Robot, Application, Code, C#, Virtual commissioning</td>
</tr>
<tr>
<td>Template:</td>
<td>University West, IV-Master 2.7</td>
</tr>
<tr>
<td>Publisher:</td>
<td>University West, Department of Engineering Science</td>
</tr>
</tbody>
</table>

S-461 86 Trollhättan, SWEDEN
Phone: + 46 520 22 30 00  Fax: + 46 520 22 32 99  Web: www.hv.se
Summary

Volvo Cars Corporation creates robot programs off-line for all new robot implementations for virtual commissioning. These virtually created robot programs are then downloaded to the real robot, after the installation has been carried out, to be tested before they are fully operational. These tests are spanned from robot installation until full production, adjusting the robot programme according to Volvo Cars specification and correcting errors that the robot program may have. Changes of the robot programs will be saved each time it is modified, generating a series of backups for each robot until the robot is correctly adjusted along all the steps of the process. To improve the offline programming there is a necessity for visualize the modifications made during the physical robot commissioning.

The objective of this thesis is to identify, categorise, quantify and visualize modifications between each different backup of a robot. A software application has been developed using Microsoft Visual Studio using C#. The application is designed in windows for different types of data. It enables the user to compare two robot programs (two different backup programs from the tests) from one robot and see the result between them graphically. The graphs are designed interactively so that the user can filter the information to see the desired data from the robot programs comparison. Key performance indicators (KPIs) has been specified for RobTargets and Procedures according to Volvo Cars Corporation requests. These KPIs are implemented and visualised in a graphical representation.
Affirmation

This master degree report, Comparison and visualization of robot program modifications, was written as part of the master degree work needed to obtain a Master of Science with specialization in Robotics degree at University West. All material in this report, that is not my own, is clearly identified and used in an appropriate and correct way. The main part of the work included in this degree project has not previously been published or used for obtaining another degree.

Signature by the author

Sergio Heras Aguilar

Date

04/07/2017
Preface

SUMMARY .................................................................................................................. III
AFFIRMATION ........................................................................................................ IV
CONTENTS ............................................................................................................... V

Main chapters

1 INTRODUCTION .................................................................................................... 1
  1.1 AIM ..................................................................................................................... 2
  1.2 LIMITATIONS .................................................................................................. 2

2 BACKGROUND ...................................................................................................... 3
  2.1 VIRTUAL COMMISSIONING .............................................................................. 3
  2.2 KEY PERFORMANCE INDICATORS ................................................................... 5
  2.3 VOLVO CARS CORPORATION PRODUCTION SYSTEM ..................................... 6
      2.3.1 Offline programming ................................................................................. 7
      2.3.2 Volvo Cars Corporation approval process ............................................... 7
      2.3.3 Physical verification at Volvo Cars Corporation ....................................... 8
      2.3.4 Process stages at Volvo Cars Corporation ............................................... 8
  2.4 SOFTWARE DEVELOPMENT ............................................................................. 9
      2.4.1 Files: robot programming ........................................................................... 9
      2.4.2 RobotStudio ............................................................................................. 9
      2.4.3 Application Programming Interfaces ....................................................... 10

3 METHOD ............................................................................................................... 11
  3.1 PLANNING ....................................................................................................... 11
  3.2 KEY PERFORMANCE INDICATORS .................................................................. 11
  3.3 PROGRAMMING/DESIGN ................................................................................. 11
  3.4 TEST AND VERIFICATION .............................................................................. 11

4 SOFTWARE APPLICATION DEVELOPMENT ..................................................... 12
  4.1 APPLICATION DESIGN ................................................................................... 13
  4.2 KEY PERFORMANCE INDICATORS SELECTION ............................................. 14
      4.2.1 RobTargets .............................................................................................. 15
      4.2.2 Schedules ............................................................................................... 17
      4.2.3 Naming .................................................................................................... 19
  4.3 TEST AND VERIFICATION ............................................................................. 19
      4.3.1 Code test and verification ....................................................................... 19
      4.3.2 User interface test and verification ......................................................... 19

5 RESULTS AND DISCUSSION .............................................................................. 21
  5.1 APPLICATION ................................................................................................. 21
  5.2 KPIs monitored ................................................................................................ 24
  5.3 TEST AND VERIFICATION ............................................................................. 25
      5.3.1 Application background code test and verification ................................. 25
Appendices

A. USER MANUAL
1 Introduction

Modern production facilities, like manufacturing industries, have a continuously increasing complexity, as well as a growing difficulty during the engineering and operation phase of these plants due to the new challenges that this complexity creates. The automation system of production plants is critical to ensure a safe and productive operation. Therefore, the correct behaviour of the automation system is essential. The early test of the engineered automation solution is a critical task to reduce the risks within the real commissioning and plant operation [1].

Today, the testing usually takes place during the on-site commissioning within the real plant and therefore influences directly the critical path of the overall engineering process. These tests must be executed in a very short time to meet the start-up schedule. In addition, testing based on the real equipment sets some limitations on the executable test cases and the testing speed [1].

As a result, we can achieve a seamless transition from the virtual to the real environment. With a validation of the overall process on a virtual plant, shorter times will be needed in commissioning on the real one [2]. This process is called virtual commissioning, it has become a promising method in both securing work results created during the engineering process of automated plant and reducing time and effort of real commissioning [3].

Every project to change production layouts for car models at Volvo Cars Corporation (VCC) has four levels or steps (concept, station design, OLP and final model) of the simulation work, the overall process can be observed in Figure 1. VCC intention is that all programming will be done offline. VCC goal is to introduce the coming car models to reach full production with high quality within 8 weeks. To be able to reach this goal, with no or minimised production losses, offline programming (OLP) is a necessity. By making the programs offline, programming work, as well as simulation, can be done before the installation starts. The programs can be tested out and

![Figure 1 VCC project process from concept stage to full production](image-url)
when the robots are installed the programs are ready to start production\(^1\).

A full simulation of the station will be developed during concept and station design steps and it will be ready by the end of the station design stage, this full simulation includes station layout, car-part assemblies, tools, fixtures, routing of cables, sensors, etc. It also includes robot poses and motions preparation and verification for the subsequent OLP implementation that will be completed in the next step of the process, a first event-based simulation study should be completed during this stage to verify feasibility and cycle time\(^1\).

The OLP will contain the final programs for all robots in the station with a correct configuration, after a quality verification of the programs (robot program approval process), a series of test will be carried out on site, to adjust the robot programs to specification and optimisation\(^1\). These changes on the robot program will be saved each time the program is modified, generating a series of backups for each robot until the robot is correctly adjusted at the end of the tests.

VCC is interested in analysing these different backups of the robot programs, from the first download to the definitive program, to measure how good are their generated OLP for the stations within the process of testing them. Key performance indicators (KPI), a type of performance measurement, to evaluate the success of an activity, will be used measuring different parameters to compare the initial programme that is downloaded for the first time in the robot and the modified ones to indicate what types of changes have been made in the program. These changes will be measured by the KPI with some predefined parameters to monitor the adjustments made in the robot programme. A report will be created to show the results of the comparison in a graphical way.

### 1.1 Aim

The purpose of this project is to:

- Develop a method to compare two robot programmes.
- Find the KPIs that will be monitored.
- Find a way to present the results graphically.

### 1.2 Limitations

The project scope can be wide in terms of work load, for that reason, some limitations have been set to adequate this thesis work to the estimated time and proper work load. The limitations correspond to the method used and what will be compared on it:

- The method developed will be used for comparison of the same robot program.
- The compared characteristics will be RobTargets.

---

\(^1\) Internal Volvo Cars Corporation document
2 Background

2.1 Virtual commissioning

Today, in an industrial environment, the design of manufacturing systems has a progressively reduced time for engineering of the process while the quality and accuracy of the planning is increasing due to the fact that manufacturing systems are defined by the cost of the operation, the product life-cycle and the time spent for getting the product to the market [4].

The development of a manufacturing system consists of several stages: facility design, mechanical engineering, electrical engineering and automation engineering (programming of robots, programmable logic controllers (PLCs), human-machine interfaces (HMI), etc.). These stages are usually implemented sequentially. They are implemented within a large combination of different parts such as storages, conveyors, transportation systems, machining or assembly, for naming some of them, control and HMI systems, are also parts of the design [4]. In addition to that, process plant complexity is increasing across all industries. The hurdles for plant design, engineering, construction, commissioning and operation are also growing in difficulty. Another important factor to add is the safety aspects in the process. Furthermore, errors found early in the design stage are easier to correct than those found later in the process, errors have a higher cost when they are found when the plant is operational [5]. A solution is the introduction of Virtual Commissioning (VC).

Virtual commissioning is the process by which a physical manufacturing system can be simulated, tested and optimized in a hybrid or purely virtual environment that includes both hardware and software. This environment enables realistic robot and

![Pyramid of automation](image)

**Figure 2** Pyramid of automation
device validation, signal mapping of robot programs to the system controller, testing and optimization of robot programs, reliable verification of safety procedures and fault logic, cycle-time verification and debugging against real production hardware (PLC, HMI) and executed PLC code [6], allowing a reliable simulation of the real process, without the need of any physical device to test all the systems involved.

Virtual Commissioning is a reliable method in both securing work results created during the engineering process of automated plant and reducing time and effort of real commissioning. Virtual Commissioning focusses on testing control-code and validating PLC programs to different standards, VC as a method can be deployed in different ways, providing a wide spectrum of possible applications. In Figure 2, the pyramid of automation is displayed, VC can be seen as a testing and validation of control level taking into consideration that field and process level have appropriate model to be able to simulate them [3].

It enables to do commissioning with a full simulation model of the manufacturing system before building it. The intention is the detection and correction of errors generated in the early stages during planning, design and programming of the manufacturing system. Virtual commissioning can be applied on all engineering stages, but completed before the physical installation began, to be able to validate all results [4].

The virtual commissioning of a production system can be considered as a simulation involving a real or virtual controller and a virtual plant model. Modern production systems are usually controlled by PLCs. The modelled plant should have the same behaviour as the real production system, see Figure 3. Since this model has various devices, such as robots, conveyors or sensors, a plant model can be considered as a set of device models. These device models should have a mechanical behaviour such as kinematics and an electrical aspect, in terms of signalling [7].

Conventionally, the mechanical and electrical design activities have been performed sequentially, wasting time. Following to complete the simulation of the virtual plant, a validation must be done for verify the results. The objective of virtual commissioning is to have a virtual plant where the validation can be done with all systems without any physical equipment, prior to real implementation. This entails a seamless transition from a virtual environment, the simulation, to the real system improving the overall quality in the process [2].

Doing a virtual commissioning procedure at the end of the automation engineering can reduce the risks to deploy an automation program containing errors on the
real plant. Furthermore, the faster an error is detected, the lesser cost it will have since
the error has been detected in a virtual environment, easy to reconfigure. To ensure
simple and proper testing a clear separation between simulation and automation
should be contemplated. Once the tested automation program is validated in the vir-
tual environment, it can be deployed within the real plant without any adoptions [8].

As advantages of VC we can see the work carried out in [9] where latest advances
in virtual commissioning are studied and the whole workflow of these techniques is
applied to an industrial assembly cell. The results show that virtual commissioning is a
reliable way for validating the operation of a cell prior to the installation of it, all in a
virtual plant simulating the real cell. The main benefits described in the paper are
ramp-up time reduction, affecting total installation time, reduction in investment cost
and enhancement of the reconfigurability of assembly equipment [9].

Also, in [6] a setup for virtual commission is proposed and described, the results
investigating the approach are that virtual commissioning provides a reliable process
for validation of robotic manufacturing systems prior to physical installation as in [9].
The benefits named in this paper are decreasing ramp-up time which affect the total
installation time up to 20–30 percent, Reduce cost of change with early detection and
communication of product design issues, optimize cycle times, minimize productions
risk by simulating several manufacturing scenarios, early validation of the mechanical
and electrical integrated production processes and early validation of production
commissioning in a virtual environment, where the PLC programs can be tested in a
virtual environment [6].

As virtual commissioning is currently seen as one additional step with in the engi-
neering process, using this approach it can be expected that a more common use of
virtual commissioning in the future becomes reality. The benefits of this integrated
process are very promising and an adaption within industry is very likely to occur
more often [8].

2.2 Key performance indicators

Contemporary companies have a need of establishing measurement programs to moni-
tor their operations to have a better overview of how everything is going within the
project, products or units they may have been developing. After initial stablish of a
measurement program, questions arise on which measures should be collected to lead
to actions and effectively make decision to redirect the operation in case those mea-
surements show a not desirable effect [10]. These measures or indicators serve to mon-
itor the status of an operation and plan long-term evolution based on the results, one
of the tools used for this purpose are the key performance indicators (KPIs) [11].

Inside an industry there are several types of equipment and processes that are a
challenge to control and maintain to achieve the highest performance and profit in the
process. KPIs are fundamental in the performance of an operation and its evolution
across time, they can provide information about the capacity of succeeding in differ-
ent areas. For instance, they are used in the energy industry, raw-material, control &
operation, maintenance, planning & scheduling, product quality, inventory, safety, etc.
[12].

As a definition, in [13], KPIs are described as the tool used by enterprises to moni-
tor the performance of their processes and business strategies relative to their objec-
tives. KPIs are traditionally defined taking into account a business strategy or a busi-
ness objective using management systems that show how to accomplish these objec-
tives and how it is going across the entire organization, to indicate what is monitored in different areas but still having an overview of the enterprise status. To monitor KPIs, enterprises rely on dashboards presenting one or more KPIs together with contextual information in order to help decision makers identify deviations and their root causes [13]. It can also be said that KPIs measure activities that have a significant effect on future performance, which are causal roots of the outcome [14].

In [11], a review is made of several papers, the author states that there are three major features that occur in all papers studied on how to approach KPIs:

- Ways to define KPIs under certain circumstances.
- Lists of KPIs which are relevant to a specific task or subject.
- Utilization of KPIs to accomplish defined goals.

From the bullet points mentioned above, the initial focus is to describe how to define the KPIs of importance to the objectives of the operation, according to the type of these, they can be defined differently. Afterwards, it is introduced the list of KPIs that will be monitored. Deducing KPIs from the objectives and study which indicators are relevant to the cause and how they can be used. Finally, it is important to understand how the KPIs selected can be relevant to measure the performance to accomplish the goals [11].

KPIs should have certain characteristics to fulfil their goal, measure the success of an organization or operation, in [14] some important characteristics found in KPIs:

- Sparse: the fewer KPIs the better.
- Drillable: user can go into detail.
- Simple: users can understand the KPIs.
- Actionable: user know how to affect outcomes, have a significant impact in the success factor.
- They can indicate trends over time.
- Easy to collect and process in terms of data.

Once KPIs have been defined and assessed, there should be a way to show the results clearly to the final user, this can be done with the implementation of a dashboard, these address the challenge of display the data gathered and support intuitive monitoring and visualisation of business performance information.

2.3 **Volvo Cars Corporation production system**

Every project to change production layouts for car models at VCC has four levels or steps, see Figure 4 for the overall process of a project at VCC. VCC intention is that all programming will be done offline. VCC goal is to introduce the coming car models to reach full production with high quality within 8 weeks. To be able to reach this goal, with no or minimised production losses, offline programming (OLP) is a necessity.

The project contains different milestones that have to be assessed along the process, they are called official deliveries. These are based on the simulation models, they are also shown in Figure 4 as “SDs”.

---

2 Internal Volvo Cars Corporation document
2.3.1 Offline programming

Offline programming (OLP) is a robot programming method to create and simulate the robot program without the need of the physical robot, in a virtual environment. By making the programs offline, all the programming work including the simulation can be finished before the installation of the cells starts. The OLP-completion includes the preparation of the final programs with Volvo Offline Program (VOLP) with correct parameters for download such as optimisation, additional signals, robot configuration, interlocking detection etc. To ensure good program quality the simulation on this level must be done in cooperation with robot programmers and PLC programmers. Download must be done before the first machine try out (MTO). Before any download is made into a robot there is a process of approval that must be done, so the status of all the programs are verified and deviations are agreed by VCC.

2.3.2 Volvo Cars Corporation approval process

There are two approval processes done by VCC, one before first download of the virtually generated robot program to the real robot and another before the second test of the robot program against the real world, called tooling trial (TT). The main reason for the first approval process is to decrease the time on the shop floor, since this verification is focusing on program syntax and the ability to implement the robot.
program in the physical robot without any major issues, while in the second is to maintain a high quality of the robot programs in the factory, see Figure 5.

A complete robot backup with modules and system files should be verified and approved in the second approval process. Eventual changes in the robot program on the shop floor still needs to be according to the specifications and past experiences have shown that this is not the case always.

2.3.3 Physical verification at Volvo Cars Corporation

When the OLP has been verified and downloaded to the robot, physical verification on the cells will be carried out after installation:

Machine Try Out (MTO) is the verification of manufacturing equipment to specification, the simulation studies and the corresponding installed stations must be compared and verified by the contractor that the models are according to specification and at correct position.

Tooling Trial (TT) has the purpose of verifying the product in the production plant process. The results of TT are that production tools have been verified, as well as equipment, facilities, systems and processes with hard-tooled production parts.

Pilot Production (PP) has the purpose of verify and confirm process capability of assembly production tools, equipment, facilities & processes.

During all these processes changes in the robot program may occur, these changes will be stored as backups in every step of the process and they will be available.

2.3.4 Process stages at Volvo Cars Corporation

In the VCC specification document is stated the following process for VCC projects: the first part of the process is to complete a conceptual design of tools, fixtures and other process equipment and simulate car-part assemblies, rough paths, station layout (SD1), see Figure 4 for reference of simulation deliveries.

The production feasibility must be verified according to demands from e.g. smoke exhaust, fixture and tool design including layout of hoses, approximate routing of cables, sensors, cable channels, valve arrangements, fastener etc. OLP-preparation regarding process and service paths are done here (SD2).

Before installation of the stations documentation in form of station layout, guidelines and directions for media installations, the simulated process flow must be prepared and communicated to the installation personal. OLP-preparation regarding process and service paths are done here, such as “Frames”, “Tools”, “Motion parameters” etc. Robot poses and motions to tip dressing, synchronisation, maintenance, tool changer, TCP check/calibration etc. must be verified and prepared. (SD3).

Installation and OLP-completion are two parallel processes that are done simultaneously. Download must be done before the first machine try out (MTO). Before any download is made into a robot the “robot program approval process” must be worked through by the contractor and have status OK, deviations must be agreed by VCC. (SD4).

During installation and after machine try out series, the physical verification starts, the simulation studies and the corresponding installed stations must be compared and verified by the contractor that the models are according to specification and at correct position. The different try outs are carried out: MTO, TT and PP (SD5, SD6 and SD7)

---

3 Internal Volvo Cars Corporation document
Comparison and visualization of robot program modifications - Background

After first downloads, some online changes are done to the programs (optimizations, adaptations to "bad" product parts to run the series or corrections due to installation errors). To not lose the changes done online, modifications that are wanted to be kept, all programs that has been modified after download shall be uploaded to return these changes to the simulation model. For each station will be 4 different robot programs corresponding to the original robot programs and 3 different modification that can occur during the process.

2.4 Software development

The files provided as the Rapid robot code are .mod files, readable by RobotStudio, but they also can be read as a .txt file. This opens a wide spectrum of possibilities in term of developing an application that reads the file and displays information. There are two main issues when approaching this project, see Figure 6: extract the needed data from the file/files and show the data in some way to the user. The final goal is to develop an application that can handle these problems and can be developed further in the future.

2.4.1 Files: robot programming

An industrial robot can be defined as “an automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications”. The definition includes the manipulator, the controller and any communication interface (hardware or software) [15]. These robots can either be programmed online or offline.

Online programming is done by setting the desired points by jogging manually the robot and create the desired paths using the recorded positions. This method requires to be physically with the robot and use a teach-pendant to teach the robot the positions, this may be simple, it has several downsides [16].

On the other side, offline programming (OLP) utilises 3D CAD data of the environment surrounding the robot to generate and simulate its behaviour, this method is widely used in automation systems. Thus, the user can test all the aspects of the robot programming on a computer and then it can be downloaded to the robot. OLP offers some advantages over online programming: programming process does not require be present with the robot, minimize production robot down time, robot programs can be developed earlier in the design stage, programs are more flexible than in the other method, etc. [16].

2.4.2 RobotStudio

The robots can be programmed by a high-level programming language depending on the robot brand. For ABB robot, the programming language is called Rapid and it
is programmed in RobotStudio, a simulation software that allows offline robot programming to be done on a PC.

A rapid program contains instructions describing the work process of the robot grouped in procedures, routines inside the Rapid code. The instructions have a number of arguments or parameters that defines how the instruction will be executed in the program. Information can also be stored in variables of different data types that work with given parameters as the instructions, for instance, tool data contains all information about of a tool like the pose of the tool or the load of it [17].

Robot programs provided for this thesis work are from ABB robot, so there are written in Rapid, different backups will be provided as .mod files. A .mod file is an extension file readable by RobotStudio, it also can be opened with normal text readers.

### 2.4.3 Application Programming Interfaces

RobotStudio provides an Application Programming Interface (API), a set of subroutines definitions, protocols and tools for building application software, it is a defined method of communication between different software components, it provides building blocks that are put together by the programmer. RobotStudio API can be used to develop add-ins, an attached application that can be run inside RobotStudio software [18].

RobotStudio API uses Microsoft .NET and Microsoft Visual Studio. Most of code examples are programmed in C#. C# is a high level common programming language and one of object oriented languages group that was developed inside the .Net framework of Microsoft. The .Net framework includes a large class library named Framework Class Library (FCL) and provides the user with interoperability options within different programming languages. Software networks may include support programs, compiler, code libraries, tool sets and APIs [19], like RobotStudio API previously mentioned, that enables the development of a project.
3 Method

The project’s initial part was a literature study to get a better overview of the whole project, also have a clear idea where is located this project inside a VCC projects. What are the options in terms of developing an application, the most suitable way to accomplish the aims of the project in an efficient and clean way with the resources available.

3.1 Planning

The information collected served to establish what could be the options for developing the project. Two tests were made with RobotStudio API and Excel API in C# to check feasibility of using these programs or discard them. The planning was carried out for how to tackle the project and the software solution.

3.2 Key performance indicators

The most important part of the project is to select the most relevant features in the data given in form of Rapid code files. The selection is made analysing the different parts of the code in variables and study each one of those separately and this way select the appropriate KPIs to finally show in the application. However, this selection is influenced by the own specifications and ideas from VCC, which knows what data may be more relevant to them.

3.3 Programming/Design

A decision has to be made to select which platform will the application be programmed in and which programming language will be used for this purpose. A program will be created according to programming standards to make it readable and understandable by everyone. A friendly interface will be designed in order to facilitate the understanding of the information shown.

3.4 Test and verification

The program developed is tested with the files given by VCC. These files are the Robot programs for the different robots located in the factory, they will be introduced as an input to the application and the data should be displayed accordingly to the information within the files. The KPIs monitored should be verified, so the information is feasible and trustable. The testing was made in parallel with the developing of the application to check that everything is correctly programmed.
4 Software application development

The initial idea was to use RobotStudio API to develop the project, a small application in C# in Microsoft Visual Studio was created to check the feasibility of using RobotStudio as the main platform to develop the entire project as an add-in to RobotStudio with PC SDK. The application would take a .mod file from a folder in the computer and synchronize it automatically to RobotStudio. There are advantages in using this API, all the types of variables are already created and defined and the information can be accessed much easier once the robot program is loaded to RobotStudio.

There are certain requirements in order to extract information from the robot program files. A virtual controller must be up and running on RobotStudio software and the .mod must be loaded and synchronized to the virtual controller. The main problem when synchronizing is that the robot programs provided are from a real station and real controller and some data variables and instructions in the code are issued as rapid errors by RobotStudio. As a result, there is a loss of information in the process of load and synchronize the robot program to the station and controller previously created in RobotStudio. A test was made, see Figure 7, with a file with 102 RobTargets, after the process only 67 were available to get information from. There is a lack of control in RobotStudio and for this type of task the PC SDK is not viable since we cannot control everything that is happening in RobotStudio.

The decision was to keep working on C# in Visual Studio, since there were already some usable parts on the code, but developing the program without RobotStudio to have full control of what is happening in the process. The other side of the solution is the representation of the results, .Net framework has libraries to develop

![Image](image-url)

Figure 7 RobotStudio test where of a robot program file with 102 RobTargets, only 67 were successfully transferred to the station.
an API for Microsoft Excel, a commonly used program to display information in tables and charts. An Excel file can be created or opened and data can be written in it. A small demo was made to display some data from the position parameter of the RobTargets, but the processing time of writing data on Excel is slow, furthermore, when updating the graphs in terms of different visualization it gets messy to try to put at the same level both programs. Also, a lack of control was noted in the process of adjust the data and display the graphs. As a result, a decision was made, of not using external programs and to develop an entire project using the tools Visual Studio and the .NET framework offer, creating the entire application in C#.

4.1 Application design

The objective of the application is to display information about different parts of a robot program, that is written in Rapid code. The application will be divided in windows to separate different types of information, see Figure 8. A main window will be designed from where it will be possible to access different type of information according to the data that will be compared.

From the main window, after introducing the robot program files, it will be possible to select what data the user wants to see, a new window will pop up, where all the data is displayed, for instance, if we select “RobTargets”, the new window will contain the data of all the RobTargets comparison between the two robot programs.

It is possible to go further if it necessary to display relevant information, a new window can be chosen to open, in this case, it is possible to open a window to see RobTargets that have not been matched as equal, this window show additional information about RobTargets, relevant since it is possible to choose new variables to add to the graphs.

This interface makes it easier to have an order when writing the code, since everything is separated in windows for each type of data. It also serves to be more intuitive for the final user, since it can be confused to have all data displayed in the same window, it can lead to wrong assumptions, this way, everything is ordered in windows that separate data from one another.

Figure 8 Distribution of windows in the application
An initial approach is made based on RobTarget data for the code running in the background, see ¡Error! No se encuentra el origen de la referencia.. The main window, mentioned above, is displayed where it is possible to select the paths of the two robot programs for later comparison, once the robot programs are selected the files are read and the data related to RobTargets is stored.

When both files have been read it is possible to go to the following window, where all the information is displayed. The window loads, the data stored is processed and the comparison is run according to the KPIs selected for this variable type. The carts is updated every time any of the selectors is changed, so that the used will get the desired information every time the selector are updated. This can be extrapolated to other type of data inside the robot programs, but in this case, just RobTarget data is shown.

4.2 Key performance indicators selection

The application programmed will give the option for accessing two different files, the robot program files from the back up and compare them. The KPIs to monitor will be selected according to the fact that it is a comparison between two files that ideally should be the same, so any variation in the program should be monitored in an appropriate way. The data types inside the Rapid code can give us information about how the files have changed, choosing specific parameters of each data type and dis-
play them in the most representative way to show the differences will be the criteria to follow.

4.2.1 RobTargets

General information about RobTargets will be displayed in the form of a table, it will be divided according to the different files and according to two types of RobTargets that can be found in every robot program. The information in these tables will be based on the naming in both files. How many RobTargets there are on each of the files, how many of those are not equal, and how many have remained the same in both programs, see Table 1.

Table 1 General information displayed about RobTargets in the RobTarget window

<table>
<thead>
<tr>
<th>Data</th>
<th>Robot program 1</th>
<th>Robot program 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RobTargets</td>
<td>Number of RobTargets in program 1</td>
<td>Number of RobTargets in program 2</td>
</tr>
<tr>
<td>Obsolete/New</td>
<td>Number of RobTargets that are not used in program 2</td>
<td>Number of RobTargets that new in program 2</td>
</tr>
<tr>
<td>Equal RobTargets</td>
<td>Number of RobTargets that remain the same in both robot programs</td>
<td></td>
</tr>
</tbody>
</table>

The RobTargets that are the same are divided into two categories: “via location” and “welding spot”. Been the motion to go to a welding position and the weld position itself, respectively. The differentiation of type is based in the naming of the RobTarget in the program: Via location RobTargets (ToB640wp8865_10) have a “to” or “fr” at the start of the name, followed by the weld spot name the want to reach, while weld Spots don’t have this nomenclature (B640wp8865). The separation is due to see differences in changes between both types of motions, to see clearer how the changes are made in the program, see Table 2.

Table 2 General information about the RobTargets that remain equal in both robot programs, divided in two different types

<table>
<thead>
<tr>
<th>Type</th>
<th>Via Position</th>
<th>Weld Spot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>Total number of RobTargets of the type Via Position</td>
<td>Total number of RobTargets of the type Weld Spot</td>
</tr>
<tr>
<td>Not Modified</td>
<td>Number of Via Position RobTargets not modified</td>
<td>Number of Weld Spot RobTargets not modified</td>
</tr>
<tr>
<td>Modified</td>
<td>Number of Via Position RobTargets that have been modified</td>
<td>Number of Weld Spot RobTargets that have been modified</td>
</tr>
</tbody>
</table>

RobTargets have different parameters, the following illustrates an example of the data type RobTarget: CONST robtarget name= [ [x, y, z], [q1, q2, q3, q4], [cf1, cf2, cf3, cf4], [ eax_a, eax_b, eax_c, eax_d, eax_e, eax_f] ]; They are used to define the position of the robot and additional axes. As the robot is able to achieve the same position in several different ways, the axis configuration is also specified, the position is defined based on the coordinate system of the work object, including any program displacement [20]. The components of any RobTarget variable are the following:

Translation: the position of the tool center point (TCP) expressed in mm. The translation parameter contains three numerical values (x, y, z) corresponding to the translation in each axis [23].
The difference is calculated considering the value of a filter that will help to group the values in intervals, for instance, if the filter value is 1, the intervals will be of 1 mm each, from 0-1, 1-2, etc. grouping the RobTargets inside these intervals depending on the result of the difference. This difference will be calculated for each RobTarget by

\[
\text{diff} = \frac{(\text{Mod1Value} - \text{Mod2Value})}{\text{filterValue}}
\]

Now the resulting absolute value is rounded to the upper integer value to group it in the filter interval and then multiplied by the filter value to get the final number inside the specified interval. The result is measured in mm, this difference will be represented in all axis separately and in combination showing the distance that the RobTarget has been displaced. As an example of this procedure, see Table 3.

**Table 3** Example of operations to calculate the intervals where the differences are grouped. FV = filter value, RV = RobTarget value in position, D = difference, R = round, Abs = absolute, I = interval

<table>
<thead>
<tr>
<th>FV</th>
<th>RV program 1</th>
<th>RV program 2</th>
<th>D / FV</th>
<th>R upper integer</th>
<th>R * FV</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3090.89</td>
<td>3090.50</td>
<td>0.39</td>
<td>0.39</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>481.29</td>
<td>476.19</td>
<td>5.10</td>
<td>1.7</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2000.12</td>
<td>1997.9</td>
<td>2.22</td>
<td>1.11</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2001.12</td>
<td>2001.12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Rotation:** the orientation of the tool, expressed in the form of a quaternion \((q_1, q_2, q_3, q_4)\). Typically, the rotation is described by a rotational matrix that expresses the direction of the axes of the coordinate system in relation to a reference system. A quaternion is a more concise way to describe this rotational matrix. Instead of 9 terms, there are just 4, these quaternions are calculated based on the elements of the rotational matrix [20].

In this way, the representation is more compact but less representative. However, it is possible to convert the quaternions to degrees with respect of the axis, a more much representative and intuitive way, the angles respect of each axis can be calculated by

\[
\begin{bmatrix}
RX \\
RY \\
Rz
\end{bmatrix} =
\begin{bmatrix}
\text{atan2}(2 \cdot (q_1 \cdot q_2 + q_3 \cdot q_4), 1 - 2 \cdot (q_3^2 + q_4^2)) \\
\text{asin}(2 \cdot (q_1 \cdot q_3 - q_4 \cdot q_2)) \\
\text{atan2}(2 \cdot (q_1 \cdot q_4 + q_2 \cdot q_3), 1 - 2 \cdot (q_2^2 + q_3^2))
\end{bmatrix}.
\]

Having the angles, it is possible to compute the difference in the same way that with the position, being able to measure the difference in angle in absolute value with respect of every axis and in combination, this difference will have a filter to determine how the intervals of degrees are grouped.

**Robot configuration:** the axis configuration of the robot (cf1, cf4, cf6, cfx). This is defined in the form of the current quarter revolution of axis 1, axis 4 and axis 6. The first positive quarter revolution 0 to 90° is defined as 0. The meaning of the component cfx is dependent on the robot type.

The robot configuration is specified using four axis values. For a rotating axis, the value defines the current quadrant of the robot axis. The quadrants are numbered 0, 1, 2, and so on (they can also be negative). The quadrant number is connected to the current joint angle of the axis. For each axis, quadrant 0 is the first quarter revolution, 0 to 90°, in a positive direction from the zero position; quadrant 1 is the next revolu-
tion, 90° to 180°, and so on. For instance, quadrant -1 is the revolution 0 to (-90°) [20].

The configuration supervision (cf1, cf4, cf6) will check that axes 1, 4, and 6 will not move more than 180 degrees, and that the ordered movement does not require a change in cfx (cfx is only used for serial link robots). Cfx is used to select one of eight possible robot configurations numbered from 0 through 7.

Since the changes expected in the programs are small adjustments, it is not expected to the robot configuration to be changed in most cases. For this reason, the KPI will be to measure if there is any kind of change in the robot configuration at all, the display will show the number of RobTargets that have change their robot configuration from program to program.

**External axes**: the position of the additional axes, they are defined as follow for each individual axis (eax_a, eax_b ... eax_f). Additional axes are logical axes. How the logical axis number and the physical axis number are related to each other is defined in the system parameters. For rotating axes, the position is defined in degrees and for linear axes, the position is defined in mm. The value 9E9 is defined for axes which are not connected [20].

For external axes, the fact that they can exist or not is important, so the number of RobTargets that contain any of the external axes will be count. From those RobTargets that have any external axis, will be counter how many remain the same and how many have changed, in each external axis. Finally, the difference will be measured in the same way that translation and rotation, displaying this difference in interval of change. In this case there is no units since an external axis can be rotational or lineal, so the change can be either in mm or degrees.

As a summary, the KPIs selected for RobTargets are shown in Table 4, they are chosen according to the parameters that a RobTarget variable have and what is the most suitable way to display the information contained in these parameters.

**Table 4 KPIs selected for RobTargets**

<table>
<thead>
<tr>
<th>RobTarget parameter</th>
<th>KPI selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Difference in mm intervals</td>
</tr>
<tr>
<td>Rotation</td>
<td>Difference in ° intervals</td>
</tr>
<tr>
<td>Robot configuration</td>
<td>Change in Robot configuration</td>
</tr>
<tr>
<td>External Axes</td>
<td>Change in external axes and Difference</td>
</tr>
</tbody>
</table>

### 4.2.2 Schedules

A schedule is the acronym VCC gives to procedures in Rapid code. A procedure is a set of instructions that are run once the program starts, the entry point for the robot is the main procedure, a procedure must be declared with the word “proc” followed by the procedure name. A procedure can be called from other procedure (except main, which is automatically called when the program starts) [17].

Every VCC robot program contains a main procedure, called “SchDefault” and a series of following procedures for different purposes, the application will read and save all schedules separately, having a name and a body.
When comparing two different schedules, it is important to note that the procedures can change in number of instructions, for instance, some instructions may have been removed or added, and other could have been shifted from their initial position. A differentiation is made according to this fact, see Figure 10, separating each instruction in three possible ways: equal in the comparison, shifted if the position in the procedure has changed, or removed or new if it is not found in the other robot program in the comparison.

The problem with the shifted instructions is to establish a reference so even if one instruction is removed, the rest of the procedure does not get affected by this, see Figure 11. The proposed solution is to establish as a reference the most normal instructions in Rapid code, such as a “TEST” or a “WHILE” instruction, to divide a procedure in smaller pieces to work with. Inside each piece of code, the program will read by line by line, if the lines are equal, continue with the next one, if not, test that the line has not been shifted to other position in the same piece of code, if the result is false, then line has been removed, otherwise it has been shifted.

The KPI selected then are the number of instructions in each procedure that are equal, the number of instructions which position in the code has changed, and the ones that have been removed or are new in the code.

![Figure 10](image1.png) Type of operations that can happen in a procedure

![Figure 11](image2.png) Result of the comparison of schedules in a small sample of code
4.2.3 Naming
One key function in the program is the one that reads the files and stores the data according to any specifications given. The problem when reading two files that might have changed, is that one variable name can be different, but the information contained on it still be the same, for instance, B640wp8865G is a RobTarget variable in the original robot program, in the updated robot program is named B640wp8865, but the information is still the same, it is then correct to say that even when the name is different, the variable is the same. This can lead to non-accurate information if the robot program files are read and it is just compared the variables or information that are the same according to the name, since there might be small changes like the one mentioned. These small changes will make the information displayed be wrong since variables that are the same are displayed as different.

An algorithm is used for these small changes, the Damerau-Levenshtein distance [21], which is a string metric for measuring the minimum number of changes between two strings (insertions, deletions or substitutions of a single character or transporting of two adjacent characters) to get the same sequence.

This algorithm allows to detect variable names that have small changes, and, in addition it is possible to compare also other parameter in the variables to be sure that the variable is the same but with a small change in the name. If the example taken above is run through the algorithm, the output will be 1. This means that one change is required to get the second name from the first. It is a good measure to detect these modified names in the program that on other instance, will be sorted as not equal variables and discarded from the comparison.

4.3 Test and verification
The verification of the application is done by two means, a first verification must be done to test that the code behind the application works fine and does not lead to error or wrong information displayed on it. A second is done to verify that the user interface is friendly and understandable for persons with no relation with the project. The two are carried out for different aspect of the programs, but both are important to get a final product that is accurate and user friendly.

4.3.1 Code test and verification
This test is carried out along the programming process with the robot program files given by VCC. The files used are the 4 backups of one of the robots to test and verify that the information displayed in the application is accurate in terms of real comparison between the two robot program files selected. With the help of a command windows it is possible to display the desired information about the different parts of the application to check everything is working as it should, with no bugs in the code.

A final verification was made with three different robot programs to verify the code accuracy against different Rapid codes. This verification consists in test the different backups of the robots and check the information shown in the application correlated to the information in the robot program files.

4.3.2 User interface test and verification
A user test is carried out with students that have previous experience with RobotStudio to get feedback about different parts of the application to check if the interface is understandable and intuitive, even if it is the first time the application is used. Some
background information is given to the students in order for them to have an overview of what the application does. A set of questions was prepared for them to answer at the end of the testing so it was possible for them to give feedback that the author could use to improve in every possible way the application interface to make it more intuitive and understandable.

The survey contains questions regarding the interface of all windows in the application, see Table 5. Most of the questions are punctuated from 0 to 10 being 0 the lowest score and 10 the highest. The last question in each window is used for feedback about what the user thinks can be improved or if some part of the application interface is confusing.

Table 5 Questions asked in the survey for the different windows in the software application

<table>
<thead>
<tr>
<th>Window</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>Was the interface intuitive?</td>
</tr>
<tr>
<td></td>
<td>Do you think something can be improved to be clearer?</td>
</tr>
<tr>
<td>RobTargets</td>
<td>Was the interface intuitive?</td>
</tr>
<tr>
<td></td>
<td>Was it easy to understand the information in the tables?</td>
</tr>
<tr>
<td></td>
<td>Was it easy to understand the information shown in the graphs?</td>
</tr>
<tr>
<td></td>
<td>Do you think something can be improved to be clearer?</td>
</tr>
<tr>
<td>Not Matched RobTargets</td>
<td>Was the interface intuitive?</td>
</tr>
<tr>
<td></td>
<td>Was it easy to understand how works the information in the boxes?</td>
</tr>
<tr>
<td></td>
<td>Was it easy to understand how it worked?</td>
</tr>
<tr>
<td></td>
<td>Do you think something can be improved to be clearer?</td>
</tr>
<tr>
<td>Schedules</td>
<td>Was it easy to understand how works the information in the boxes?</td>
</tr>
<tr>
<td></td>
<td>Was it easy to understand the information in the tables?</td>
</tr>
<tr>
<td></td>
<td>Do you think something can be improved to be clearer?</td>
</tr>
</tbody>
</table>

The survey was answered for 11 persons that had previous knowledge about RobotStudio and Rapid code so it could be given proper feedback about the interface of the application.
5 Results and discussion

The application developed allow the user to check the similarities and differences of two rapid programs. The work shows how the application is capable of reading two robot program files and display the desired data according to the KPIs selected for each type of information in the files. The comparison made will show how both robot programs are related, and what has changed from the first to the second robot program, allowing the user to interpret the data and analyse what is a good result since the ideal case of the comparison is that the two module files are equal. This interpretation is made by the user, but tools and dynamic graphs are displayed in order to help to see what can be improved in the programs.

5.1 Application

The software was programmed in Microsoft Visual Studio using C#, it was programmed following a scheme of using windows for each type of data, that way the software can be further developed with other variable types using this method.

The main window of the application consists of controls for browsing the robot program files in the computer and once loaded, the buttons for the different data types that are displayed in different windows, see Figure 12. There are two boxes that show basic information that is written on the top of the program for each robot program loaded. From this window, it is possible to access to the RobTarget window, display the data about the difference and similarities of this variable type, and the schedule window, that shows the user the differences and similarities in the procedures of the programs.

The windows mentioned will display the information from the two robot pro-
gram, in case the user wants to see other robot programs, browsing the new files will load the new information into the application.

The RobTarget window, see Figure 13, shows general information about the RobTarget comparison between both robot programs loaded in the form of a table on the left, while on the right is displayed the information regarding the KPIs selected for these variable types. The table displays how many RobTargets have been modified and how many remain equal in both robot programs. RobTargets are then divided into two different types according to what they are used for.

For the RobTargets that are not equal, another window is designed to check manually the names and parameters of these RobTargets, allowing the user to select those which might have changed naming, but still are the same RobTarget in the robot program and add them to the RobTarget Window graphs, see Figure 14. It is possible to do this RobTarget by RobTarget or use an automatic mode, that depending on two parameters, difference in position and the number of letter that it is needed to convert the name in the robot program 1 to the same name in robot program 2, it automati-
cally will add to equal those who match the conditions proposed. After the selection is made, apply changes will updated the information in the RobTargets window displaying the new RobTargets as equal.

In the main window, the other option is to check the schedules of the robot programs, a new window will pop up with this information, see Figure 15. This window shows the schedules (procedures) of both robot programs and it is possible to select which ones the user wants to compare. The information displayed is about number of instructions that have been modified in the schedule, having three possible outputs: the instruction is equal, it has changed its position in the code, or there is no match in the comparison.

As an extra functionality to the application, an export window was added, from this window it is possible to create an output Word document with the information from the comparison for the program, being able to write the different tables that are shown in the application an also the desired graphs that the user can choose to save,

Figure 15 Interface of schedules window

Figure 16 Export window and its functionality
5.2 KPIs monitored

The KPI monitored in this software, see Table 6, are chosen to show in the most intuitive way the information extracted from the robot program files, these KPIs are displayed in a graphical form with charts or tables so the user can see the information straightforward.

Table 6 Key performance indicators for each type of variable measured in Rapid code

<table>
<thead>
<tr>
<th>Parameter</th>
<th>KPI selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>RobTarget: position</td>
<td>Difference in mm intervals</td>
</tr>
<tr>
<td>RobTarget: rotation</td>
<td>Difference in ° intervals</td>
</tr>
<tr>
<td>RobTarget: robot configuration</td>
<td>Change in Robot configuration</td>
</tr>
<tr>
<td>RobTarget: external Axes</td>
<td>Change in external axes and Difference</td>
</tr>
<tr>
<td>Schedules: equal instruction</td>
<td>Same instruction in both robot programs</td>
</tr>
<tr>
<td>Schedules: shifted instruction</td>
<td>The instruction has shifted position</td>
</tr>
<tr>
<td>Schedules: removed instruction</td>
<td>The instruction has been removed</td>
</tr>
</tbody>
</table>

The KPIs monitored for RobTargets are separately represented in different graphs, both position and rotation data are represented with a pie chart where it is displayed the percentage for each interval of change measured in mm and degrees for position and rotation respectively, a column chart is also displayed where more specific information is shown about the number of RobTargets within each interval. This interval can be changed via a selector to group the RobTargets on different intervals, for instance, in Figure 17, position data (left) is measured of intervals of 3 mm, while in the rotation data (right) is measured on 1 degree intervals.

Figure 17 Graphical representation of position and rotation KPIs
Figure 18 Graphical representation of Robot configuration changes and External axes

Robot configuration and External axes are displayed in the same graph, see Figure 18, since the robot configuration KPIs only measure how many RobTargets have changed their configuration, it will not happen often. The external axes are divided in 3 charts, displaying general info on external axes (top left), which external axis has changed (top right) and selecting the specific external axis, how they have changed (bottom).

The schedules KPIs (procedures) are measured according on how the lines in the robot program have changed, there are three possible outcomes for a line, it is equal in both robot programs, it has shifted its position to somewhere else or it has been completely removed from the program, in Figure 15 this information can be seen in the form of a table. It will give an output depending on which schedules are chosen in both robot programs.

5.3 Test and verification
Two tests have been carried out to verify the software application, a test to verify that the background code runs properly through all the application and a survey to test the functionality and understandability of the application interface.

5.3.1 Application background code test and verification
Given different robot programs backups from different robots by VCC, a test was carried out using all backups given to verify that the application does not break in any circumstance. Also, 3 robots were introduced in the application and the data shown in the interface was verified against the real data in the Rapid code of the robot programs to debug any possible bug and error. After testing with the different backup combinations of the 3 robots, it can be said that the application can be trusted and the
data shown correspond to the information contained in the Rapid code of the robot programs.

5.3.2 Application interface functionality test

A survey was made for analysing the interface of the application. The objective of the survey was to test the application functionality and interface to user with previous knowledge of RobotStudio, but without any previous relation to the developed application. The persons who tested the application were asked to give feedback about the different windows about how the thought they could be improved to be more intuitive and user friendly.

Eleven persons tested the application, the result of the survey was summarized in Figure 19, the questions were about different aspects of the interface, the surveyed person had to answer based on their opinion in a scale from 0 to 10, being 10 the highest score. The answers were evaluated based on the average value of all the scores. Also, a final question for each window was asked to receive written feedback about what could be improved in the interface according to the surveyed person.

The result shows that three windows have a punctuation above 8, this could be considered as good, while the unmatched RobTarget window has slightly less score. During the tests, all participants understood what was happening in the application, but this one was confusing for some of them. The following section will explain what feedback was given in the different windows and how the interface was updated according to this feedback.

5.3.3 Survey feedback and improvements

As it is mentioned above, the program is divided in 4 windows, each of them with a different structure since the displayed information is different and they have different purposes. In a general way:

- The letter size was too small in some parts.
- The windows were not resizable so depending on the resolution of the monitors it may causes problems to display the window in a proper way.

![Survey results](image)

**Figure 19** Results of the survey carried out to test the application
There was an inconsistent way of relating the two robot programs across the application.

Some parts of the application were confusing, like the unmatched RobTargets window.

The charts were not addressed clearly.

Taken the feedback, several changed were made in the interface of all windows, the following explain what was done to solve all these issues found by the users:

- All text in the application were augmented to a more comfortable size.
- All windows were remade to make them resizable so the user can put the most comfortable size in the window, no matter what resolution the monitor where the application is running has.
- The different references to the robot programs were renamed to make clear across the application which information’s is according to which robot program.
- Some information boxes where placed in the more confusing parts of the interface, to help the user to use the application without any problem.
- The charts where addressed properly so the user can see clearly what each chart refers to and what information is being displayed.
6 Conclusion

All robot programs at VCC are created virtually. Once the robots have been installed, the robot programs are downloaded to the real robots and tested different times against the VCC specification. These tests aim to verify that the programs have been generated correctly and that there are no errors in it, the robot program might suffer modifications along this process. In every one of these tests the robot program is saved as a backup for posterior reference. In the end, 4 programs will be available for each robot that belong to the different tests carried out in the process of the physical commissioning of the robots, the ideal case is that the different programs for the same robot are equal, but that is not the case. In order to improve their robot programs VCC requires of a method to compare these robot programs from the same robot and measure the differences on it. An application is created in Microsoft Visual Studio, using C#, to be able to read and store information from the robot programs.

The application was designed to allocate each variable type or characteristic in a separate window and display a series of key performance indicators (KPI) that would be unique to each characteristic that wants to be monitored. The work was focused on the information within the RobTarget variable types and the Procedures and how to extract the information related to the comparison of both robot programs. Specific KPIs where established for each parameter to measure in a graphical way the difference between the two robot programs selected previously.

The output of the application is in form of tables and graphs where the information related to the comparison of the two robot programs is displayed. The information shown is related to the KPIs selected for the variable type and the comparison of both robot programs. The output graphs were designed interactively to let the user display the information that is needed, having filtering options for this purpose. The information extracted from the robot programs can then be exported to a word document with all the graphical representation of the comparison.

A user test was carried out to test and verify the ease of use of the interface and the application in general, 11 persons tested and gave feedback related to what could be improved to be more intuitive or less confused, after the user test, all the opinions were considered and the application interface was redesigned according to the feedback given.

The final version of the application is able to extract and display information of two robot programs written in Rapid that correspond to the same ABB robot. It displays the data in an interactive and representative way to clearly see the differences between the two robot programs.

6.1 Future Work

The software application developed is able to successfully extract information from the RobTarget variables of two different robot programs from the same robot, it also does the same for the procedures data of the robot programs. The application was designed in windows on purpose so it is possible to extend the work to other variable
types. That way the application would show all the information contained on the robot programs, to have a more extensive view of the changes in the robot programs.

As it is mentioned along this report, there are 4 robot programs for each robot, the application designed was implemented to read two robot programs and display the data related to these. A future development is to have the information displayed over time being able to read the 4 different robot programs, from start of the process to the finished program. A comparison over time considering the 4 robot programs at the same time can be very useful to check how the program have been modified in each step of the process and propose actions according to the data of the comparison.

6.2 Generalization of the result

This work was carried out for a specific purpose at VCC to compare Rapid data from robot programs that are from the same robot program and are modified over time. The application created is a tool to measure graphically the changes of these robot programs from the same robot program.

RobotStudio has a tool to make a comparison of equality line by line between two different robot programs displaying what has changed, but this tool is not accurate at all and present several problems when lines are added or removed from the code. The purpose of the application created in this project is the same of the tool in RobotStudio, but it is much more flexible and the output is graphical instead of code. It could be used to compare any kind of robot programs where it is needed to see what has changed, the only restriction is that the software application developed is created according to VCC specification, but with some adjustments in the code, it could be used for any kind of robot program.
7 References


A. User Manual

Purpose:
A tool for high level comparison of robot programs, the comparison checks RobTargets and Schedules between two module files from the same robot. The tool will display the information related to these parameters in interactive graphs with some filtering options.

It is possible for the user to save the data and export it to a Word document as a report.

Usage:
Installation
The folder provided contain the necessary files to run the software, place the folder in “C:\Program Files\Rapid CodeComparer” in the computer, then run the RapidCodeComparer.exe in the Rapid Code Comparer folder to start the application.

Main window
To load module files, select ‘browse’ and browse the path of the module file, the file need to be a .mod file or .txt file, the application will reject other type of files. Once both files have been entered correctly a message will display: “files loaded”, then it is possible to go to the next windows.

RobTarget window
Information about the robot programs loaded is displayed, to see unmatched RobTarget click in the button with the same name in the top left part or the window.

A new window will pop up, this window allows the user to see the RobTarget that are considered not equal, it is possible to select RobTargets in both files and add them to the equal RobTarget manually pressing “add to equals”, “remove from equal” will erase the selection previously made for the selected RobTargets.

The automatic recognition automatically will do the job depending on two parameters:
- Difference in position: if the difference in position is equal or less to the selector
- Minimum number of changes in names: the names will be checked, if the amount of changes to get the second name from the first in characters is equal or less than the selector.

Schedules window
Information about the modules loaded is displayed, the schedules from both modules will be shown, selecting the desired schedules the number of instructions will be displayed and the information about how they have changed. Two methods can be used for the compari-
son, line by line, that is self-explanatory, or a smart comparison that is more accurate when the procedures are more similar.

**Export Window**

Through the RobTarget window is it possible to export the results, either selecting which tables the user wants to export, or saving the chart clicking the “save” button on the bottom left of them, when the user is satisfied, browse the folder and name where the Word document will be created and click “export” button.

**Good to know:**

- It is necessary to have .Net Framework 4.0 version or newer in order for the application to work, if not updated, a message will pop when trying to load the module files in the main window.
- The Office Interop API dependency is required for exporting data from the application.
- It is necessary to have the ‘.’ As the separator for decimal in the operating system setting, having ‘,’ as the decimal separator will cause the application to not work properly giving wrong outputs. This is achieved by using English (United States) as regional settings.

**About**

Developed by Sergio Heras Aguilar ([sergioherasaguilar@gmail.com](mailto:sergioherasaguilar@gmail.com)) for Volvo Car Corporation 2011-2017.

Copyright VCC 2017.