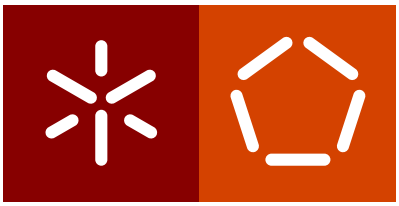


Universidade do Minho
Escola de Engenharia
Departamento de Engenharia Eletrónica
Laboratórios e Práticas Integradas 2

Integrator Project: Final Report

Radio Frequency Camera Assisted Rover (RFCAR)

Group 7

Nuno Rodrigues	A85207
Hugo Carvalho	A85156
Hugo Ferreira	A80665
João Faria	A85632
João de Carvalho	A83564
José Mendes	A85951
José Pires	A50178

Supervised by:
Professor Doutor Vitor Silva

June 30, 2020

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

Notation	Description	Page List
FGM	Functionally Graded Material	2
LBAM	Laser-based Additive Manufacturing	2, 3
MMAM	Multi-Material Additive Manufacturing	2
MMS	Multi-Material Sintering	35–38
OMT	Object-Modeling Technique	8
OOSE	Object Oriented Software Engineering	8
SLM	Selective Laser Melting	2, 5
SLS	Selective Laser Sintering	2, 5
UML	Unified Modeling Language	8

Symbols

Symbol	Description	Unit
ω	angular velocity	rad/s
π	ratio of circumference of circle to its diameter	

1. Introduction

One of the ultimate goals of engineering is the ability to provide sustained development, by efficiently using the available resources — materials, energy, knowledge, time. This is a "Sisyphus's stone"[1], a perpetual quest for optimisation of efforts and resources. One such example in the engineering field is the ability to produce components with the minimum amount of material needed for its function, i.e., functional design of components without restrictions to geometry or type and number of different materials; instead the focus should be on the desired properties of such components, customising them for the specific field of application.

1.1. Context

The functional design of components is a complex topic, with a myriad of questions to be answered: what is the function of the component?; what design criteria must be met to fulfil its function?; how will the component be produced, and what data does it require?; how will the component's performance be measured?, among others. The answers are often not clear or simple as they dictate the use of several materials and several manufacturing technologies, increasing severely the complexity of producing such components: how to effectively combine two or more materials into a single component in a synergistic way?

1.2. Motivation

The possibility of controlling composition or structure and thus obtain components with desired local properties, as regarding mechanical, tribological, thermal properties, and others are of great interest, as material is only added where it functionally needed, minimising waste and enhancing the overall properties of the component being built.

[1] King of Ephyra, condemned to push a stone uphill for eternity — immortalised in "The Myth of Sisyphus" by Albert Camus

1.3. Main objectives

The goal of the present work is to close the gap between design and fabrication of multi-material components from metallic/ceramic materials using Selective Laser Sintering (SLS)/Selective Laser Melting (SLM) technology. To this end several main objectives have been outlined:

1. Develop a design methodology for multi-material fabrication of metals/ceramics;
2. Instantiate a practical workflow and respective toolchain from the design methodology;
3. Develop and build a proof-of-concept equipment capable of producing such components;
4. Test the production of multi-material components using the proposed workflow/toolchain and the equipment built.

This is π
This is ω

1.4. Thesis organisation

This thesis is organised as follows: In Chapter 2, the state of the art of the additive manufacturing technology, Laser-based Additive Manufacturing (LBAM), and Multi-Material Additive Manufacturing (MMAM) is presented, with special focus on the last, namely on Functionally Graded Material (FGM) structures. Lastly, a brief overview of the available methodologies in these fields are presented.

In Chapter 3 the theoretical foundations are presented, namely the project development methodologies and associated tools, and the SLS/SLM process in detail.

In Chapter **??**, the multi-material design and production problem and its challenges using SLS/SLM technology are presented. The methodology devised for multi-material production through the LBAM technology is presented to tackle the high complexity of the process and the lack of a supporting methodology, taking into account the key agents of the process and leveraging the process information.

In Chapter **??** is presented all the development phase of the project. A specific workflow was instantiated from the methodology, attending to the specific requirements and constraints of the project. Based on this workflow, a toolchain was assembled, designing the required software components. Finally, based on the requirements and constraints of the process itself, the mechanical and electronic infrastructures were designed, and on top of the last, the control software was designed.

In Chapter **??**, the workflow and equipment were put to the test to verify their suitability to the process and their performance for multi-material component production. Additionally, production manufacturing

tests were also performed. Tests were used to validate the workflow and equipment, pointing out also straightforward ways to adapt and implement custom paths for the multi-material LBAM process.

The Chapter 9 gives a summary of this thesis as well as prospect for future work.

Lastly, the appendices (see Section 9) contain detailed information...

2. State of the art

In this chapter a review on the state of the art is presented. Additive manufacturing is discussed briefly as a preliminary topic. Then, Laser-based additive manufacturing processes are discussed as viable solutions for metallic and composite manufacturing.

3. Theoretical foundations

In this chapter some background is provided for the main subjects. The fundamental technical concepts are presented as they proved its usefulness along the project, namely the project development methodologies and associated tools, and the SLS/SLM process in detail.

3.1. Project methodologies

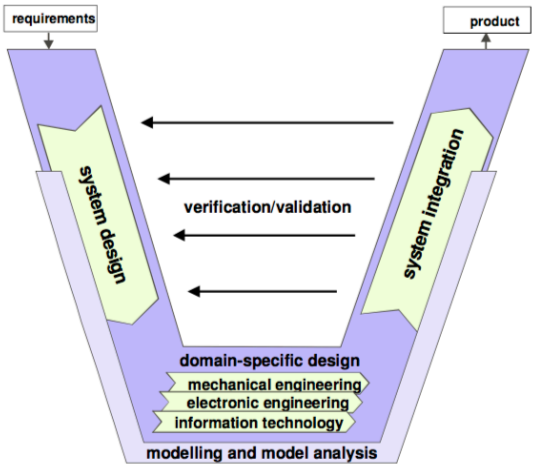
The methodologies used for the project development are briefly described next.

3.1.1. Development methodology of mechatronics — VDI 2206

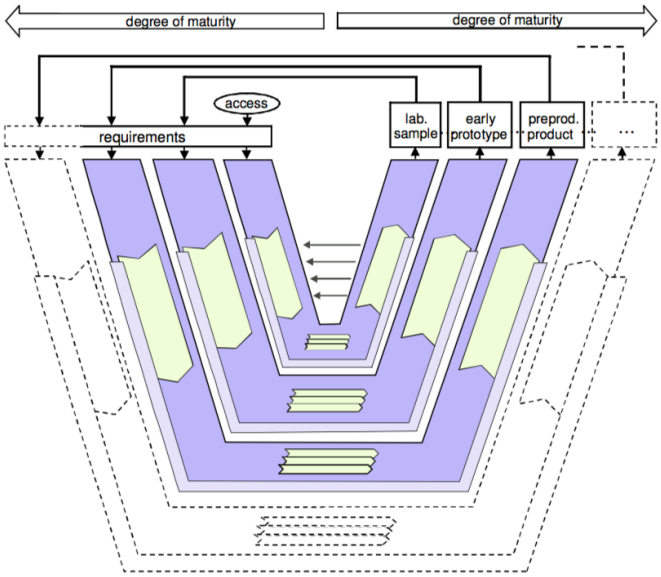
Mechatronics, was defined by Harashima et. al [1] as: "the synergistic integration of mechanical engineering with electronic and intelligent computer control in the design of industrial products and processes". This definition has more than twenty years, and nowadays it could be extended to other domains beyond the industrial sector, but a central idea remains: the synergy between these fields of knowledge, able

3.1.1.1. Process modules for recurrent working steps

The main phases in the v-model are not yet detailed: this needs to be done by the designer. However, some design procedures occur regularly during the design and were identified by field practitioners in terms of predefined process modules, representing procedures and methods for different design tasks, organised in a data base (fig. 3.2[2]). The VDI 2206 guideline provides detailed process modules for the V-model macro cycle. One such example is given in fig. 3.2 for the system design, which can be described phase-milestone-diagrams, checklists, process-flow diagrams, etc..



(a) One macro cycle



(b) Iterating over a number of macro-cycles with increasing process maturity

Figure 3.1.: Mechatronics design V-model, VDI 2206 2003 [2]

3.1.2. Waterfall

For the domain-specific design of software the waterfall methodology is used. The waterfall model (fig. 3.3) represents the first effort to conveniently tackle the increasing complexity in the software development process, being credited to Royce, in 1970, the first formal description of the model, even though he did not coin the term [3]. It envisions the optimal method as a linear sequence of phases, starting from requirement elicitation to system testing and product shipment [4] with the process flowing from the top to the bottom, like a cascading waterfall.

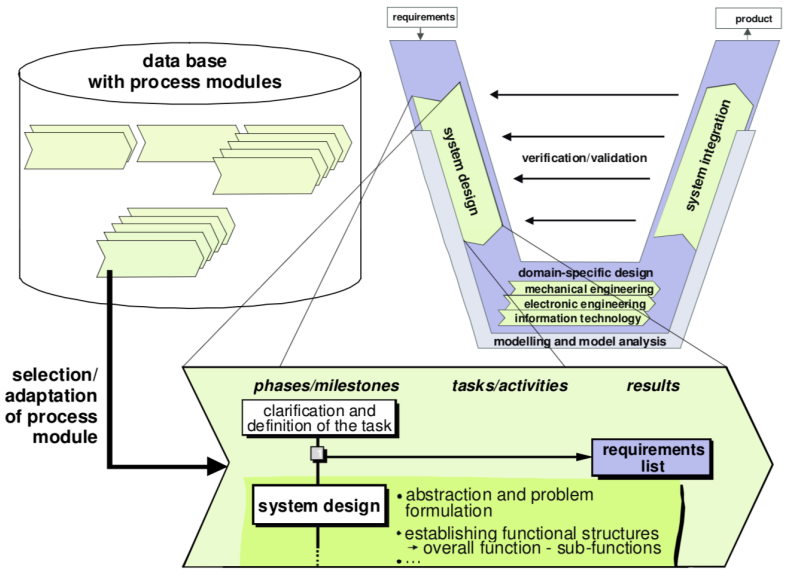


Figure 3.2.: Configuration of process modules for individual operation steps

In general, the phase sequence is as follows: analysis, design, implementation, verification and maintenance.

1. Firstly, the project requirements are elicited, identifying the key requirements and constraints the system being developed must meet from the end-user perspective, captured in natural language in a product requirements document.

2. In the analysis phase, the developer should convert the application level knowledge, enlisted as requirements, to the solution domain knowledge resulting in analysis models, schema and business rules.

3. In the design phase, a thorough specification is written allowing the transition to the implementation phase, yielding the decomposition in subsystems and the software architecture of the system.

4. In the implementation stage, the system is developed, following the specification, resulting in the source code.

5. Next, after system assembly and integration, a verification phase occurs and system tests are performed, with the systematic discovery and debugging of defects.

6. Lastly, the system becomes a product and, after deployment, the maintenance phase start, during the product life time.

While this cycle occurs, several transitions between multiple phases might happen, since an incomplete specification or new knowledge about the system, might result in the need to rethink the document.

The advantages of the waterfall model are: it is simple and easy to understand and use and the phases do not overlap; they are completed sequentially. However, it presents some drawbacks namely: difficulty to tackle change and high complexity and the high amounts of risk and uncertainty. However, in the present work, due to its simplicity, the waterfall model proves its usefulness and will be used along the project.

As a reference in the sequence of phases and the expected outcomes from each one, it will be used the chain of development activities and their products depicted in fig. 3.4 (withdrawn from [5]).

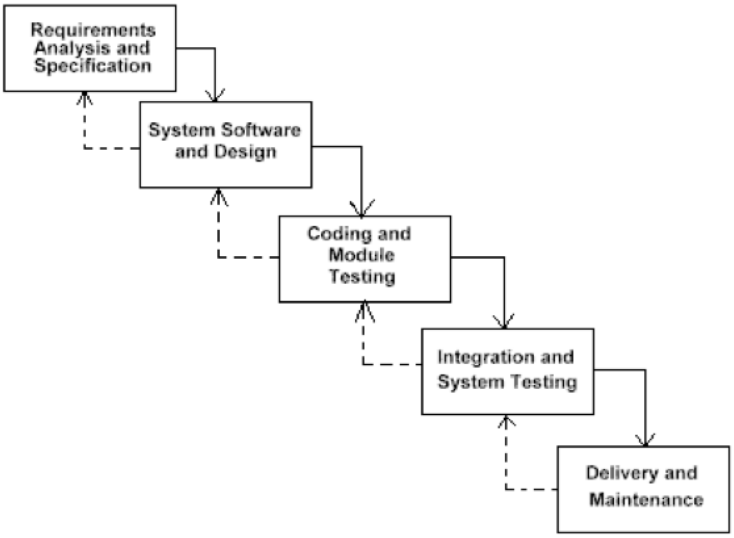


Figure 3.3.: Waterfall model diagram

3.1.3. Unified Modeling Language (UML)

To aid the software development process, a notation is required, to articulate complex ideas succinctly and precisely. The notation chosen was the Unified Modeling Language (UML), as it provides a spectrum of notations for representing different aspects of a system and has been accepted as a standard notation in the software industry [5].

The goal of UML is to provide a standard notation that can be used by all object- oriented methods and to select and integrate the best elements of precursor software notations, namely Object-Modeling Technique (OMT), Booch, and Object Oriented Software Engineering (OOSE) [5]. It provides constructs for a broad range of systems and activities (e.g., distributed systems, analysis, system design, deployment). System development focuses on three different models of the system (fig. 3.4) [5]:

1. **The functional model**: represented in UML with use case diagrams, describes the functionality of the system from the user's point of view.

2. **The object model**: represented in UML with class diagrams, describes the structure of the system in terms of objects, attributes, associations, and operations.

3. **The dynamic model**: represented in UML with interaction diagrams, state-machine diagrams, and activity diagrams, describes the internal behaviour of the system.

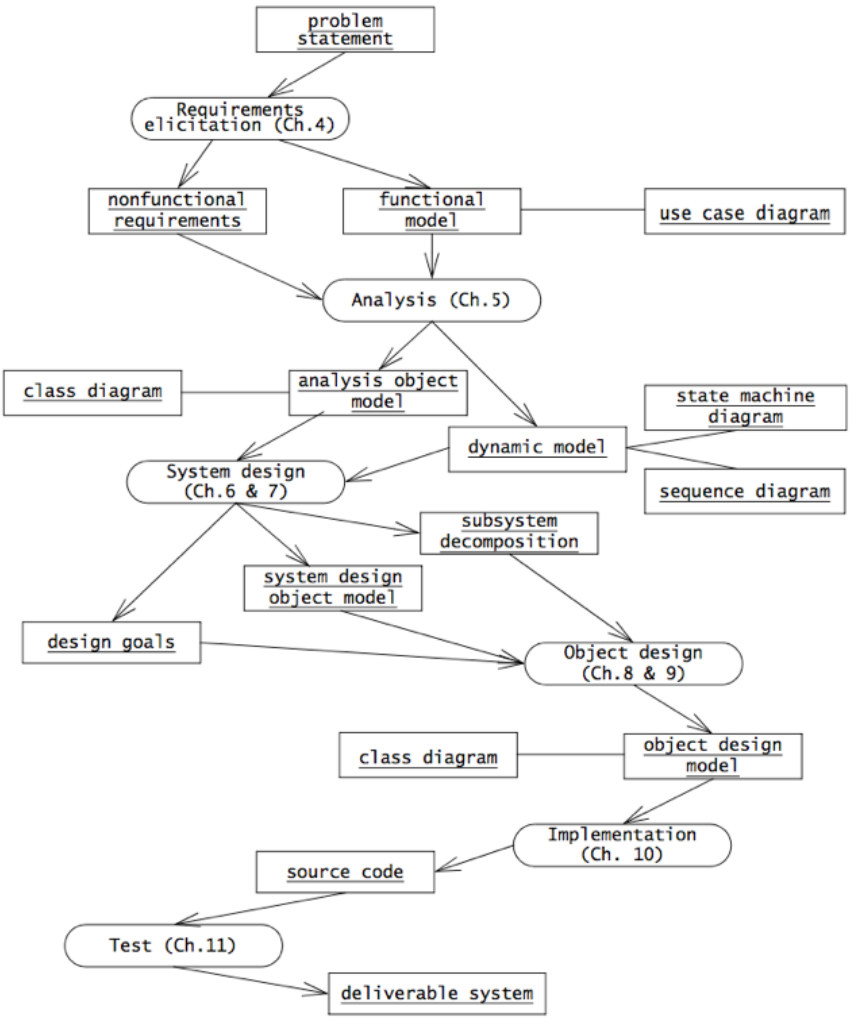


Figure 3.4.: An overview of the object-oriented software engineering development and their products. This diagram depicts only logical dependencies among work products (withdrawn from [5])

4. Analysis

The multi-material part fabrication is a complex topic and most current commercially available systems have been designed for mono-material part fabrication [6] and are unprepared for multi-material processing due to the lack of flexibility and processing capability.

4.1. Product concept

The envisioned product consists of a remote controlled car used to assist exploration and maintenance domains, hereby, denominated as Radio Frequency Camera Assisted Rover (RFCAR). To satisfy such requirements, the vehicle must contain a remotely operated camera that provides a live video feed to the user. Additionally, the vehicle must include an odometric system that assists the driving and avoids unintentional collisions when remote control is compromised, e.g., when connection is lost. The vehicle provides means for exploration and conditions assessment in critical or unaccessible areas to human operators, such as fluid pipelines and other hazardous locations.

4.2. Foreseen specifications

In this section the foreseen product specifications of the system to be developed are provided. Such specifications were obtained through the intersection of customer, functional requirements and project restrictions.

4.2.1. Quality Function Deployment

The customer requirements are usually abstract and can collide with the functional requirements, compromising the fulfilment of the project. Thus, it raises the need of a methodology which converts abstract requirements into a series of concrete engineering specifications.

An efficient quality assessment methodology is the use of a Quality House (QFD). In this method, the desired requirements are laid out as rows and the engineering specifications/restrictions as columns. In

the intersections lies a symbol representing the strength (weak, moderate or strong — Figure 4.2) of the relationship requirement-specification. This symbol is one of the many tools that allow the quantification of relations existing between the customer requirements and engineering specifications. For instance, the 'engine power' specification and the 'fast' requirement have a very strong correlation (9) since the power of the engine is directly responsible for the speed of the car.

Along with the requirements, the importance given to each is also specified, ranging from 1 (lowest importance) to 5 (highest importance) these, along with the number at each intersection, will be used to calculate the importance of each specification and thus assign priorities for the Design Team.

Lastly, the triangle shape (the 'roof' within the house metaphor) serves as another way of measuring relationships, this time between each specification: such is achieved by placing a symbol (ranging from very negative to very positive, see Figure 4.1) in the diagonal intersection of two specifications. I.e., the battery life will have a very negative correlation with the battery temperature, due to the fact that the increase of the temperature will cause a decrease in life time. As such a 'very negative' correlation was placed in the diagonal intersection betwixt 'Battery Life' and 'Battery Temperature'.

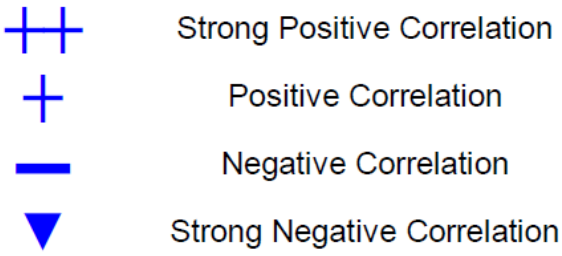


Figure 4.1.: Quality House - Specification Correlation Strength Symbols

Figure 4.3 shows the 'Quality House' for the RF CAR containing:

- **Customer Requirements**: Vehicle Integrity; Obstacle Avoidance; Reliable Feedback; Fast Response; Fast; Budget Friendly; Low Consumption; Small.
- **Functional Requirements or Restrictions**: Autonomy; Battery Temperature; Minimum Distance to Obstacle; Maximum Velocity; Motor Expectancy; Cost of Production; Motor Power; Ramp-Up Speed Time; Frame Rate; Camera Range; Resolution; Communication Range; Communication Speed; Dimensions; Mass.
- **Intersection Values** (referencing the strength of the requirement-specification correlation) — see Figure 4.2.
- **Analytical Data**, depicting, in a quantifiable manner, the aims of the project and the relevance of each entity:

- Target or Limit Value: The metrics the design team will be based on, white spaces are left for either further discussion and refinement.
- Difficulty: Allows a subjective input to be added so that 'importance' can be changed to balance unforeseen circumstances.
- Importance and Relative Weight: The main conclusion for which the QFD was used, it assigns the priorities for the design team in an objective manner.

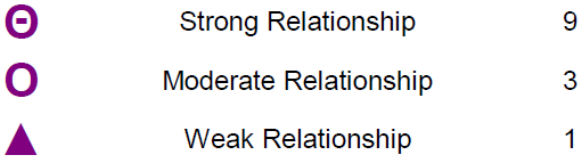


Figure 4.2.: Quality House - Relationship Strength Symbols

With the QFD, the prioritized ranks and specification targets were obtained and diffused within the Design Team with a straightforward guideline. For instance, the low cost requirement should be prioritized over all other specifications, followed by the maximum speed, Ramp-Up Speed Time and so on. On the other hand, the engine expectancy is of little to no consequence (note that the importance added up to a mere 3%), followed by the camera-related specifications. This could be regarded as a point of discussion, which should be prioritized? The functionality of the car or the the feedback provided by the camera?

With the last point in mind, the QFD has the advantage of promoting further discussion, simply by changing the importance of a requirement the priority ranking will change, ergo the priorities can be altered, easily and efficiently, if deemed appropriate.

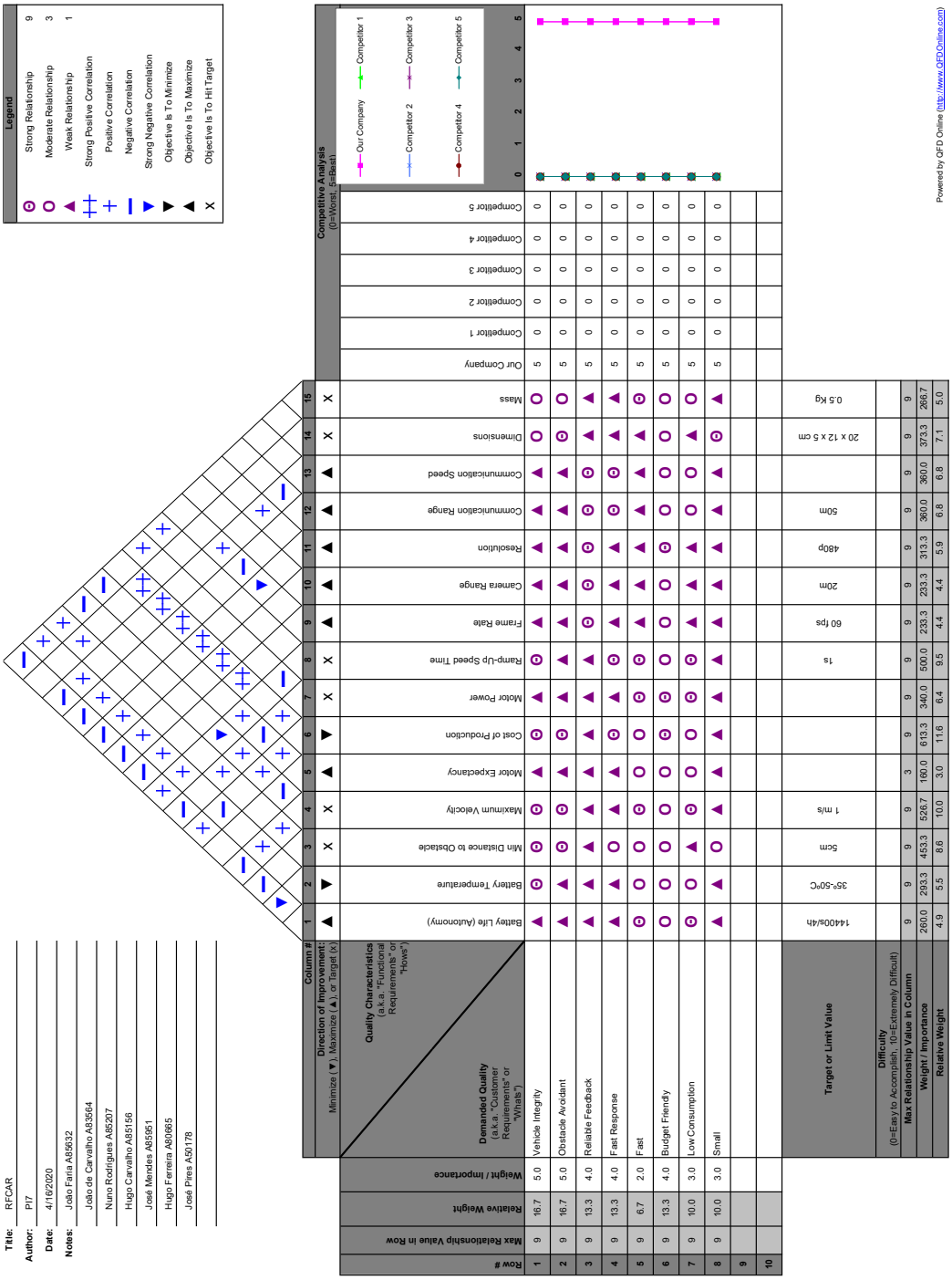


Figure 4.3.: Project Study – RFCar Quality House

4.2.2. Vehicle Autonomy

The vehicle is operated in wireless mode, thus, a portable power source must be included. The autonomy refers to the vehicle operating hours since the battery is fully charged and safely discharged and should be observed for the following scenarios:

- No load and vehicle operating at maximum speed;
- No load and vehicle operating at mean speed;
- Maximum load and vehicle operating at maximum speed;
- Maximum load and vehicle operating at mean speed.

4.2.3. Speed

The vehicle must be operated within a safe range of speed, while also not increasing excessively the power consumption. Thus, these speed boundaries should be tested in the absence of an external load and in the presence of the maximum load.

4.2.4. Safety

Vehicle self integrity protection is a requirement in product design, especially considering the vehicle is to be remotely operated. The safety in the operation can be analysed in two ways, and considers the preservation of people and goods. For the former, it is important to assure safe interaction as well as user operation — the vehicle may encounter several obstacles along its path, but it must not inflict any damage. For the latter, the vehicle under operating conditions must not inflict any damage to goods. Thus, in the presence of conflicting user commands violating the safety of people and goods, the local system should override them, taking corrective measures to prevent it. The same holds true if the communication between user and system is lost.

4.2.5. Image acquisition

The vehicle is equipped with a camera to assist in its navigation, thus, requiring it to be fed to the user's platform appropriately. To do so, several functionalities details need to be addressed efficiently. It was selected the most relevant three and these include the frame rate, the resolution and the image range.

4.2.5.1. Frame rate

Frame rate refers to the frequency at which independent still images appear on the screen. A better image motion is the result of a higher frame rate but the processing overhead increases as well, so a compromise must be achieved between the quality of the image and the increased processing overhead required. The minimum frame rate defined must be such that allows a clear view of the navigation.

4.2.5.2. Range

How far can the camera capture images without being distorted or unseen by the user. The range must be such that allows the user to see the obstacles when the car is heading to them and provide enough time to change the direction.

4.2.5.3. Resolution

The amount of detail that the camera can capture. It is measured in pixels. The quality of the acquired image is proportional to the number of pixels but a increased resolution requires a greater data transfer and processing overhead, thus, a compromise must be achieved. The minimum resolution must be such that provides the least amount of information required for the user.

4.2.6. Communication

For the implementation of the communication, several stages must be considered: Reliability, redundancy and communication range.

4.2.6.1. Reliability

A communication is reliable if it guarantees measures to deliver the data conveyed in the communication link. As reliability imposes these measures, it also increases memory footprint, which must be considered depending on the case. For the devised product, an user command must be acknowledged to be processed, otherwise, the user must be informed; on the other hand, loosing frames from the video feed is not so critical — user can still observe conveniently the eld of vision if the frame rate is within acceptable boundaries.

4.2.6.2. Redundancy

The communication protocols are not flawless and the car relies on them to be controlled. If the communication is lost, the car cannot be controlled. A possible solution for this issue is using redundancy in the communication protocols (e.g Wi-Fi and GPRS), so if one protocol fails, the car will still be controlled using the other.

4.2.6.3. Range

The communication protocols have a limited range of operation, and, as such, regarding the environment on which the car is used the range can be changed. The range established the maximum distance allowed between user and system for communication purposes.

4.2.7. Responsiveness

The movement of the car will be determined by the tilt movement of the smartphone. Sensibility refers to the responsiveness of the car on the minimum smartphone tilt movement. The sensibility must be in an range of values in which small unintentional movements will not be enough to change the state of the car and it does not take big smartphone tilts for the car to move.

4.2.8. Closed loop error

The speed, direction and safe distance to avoid collisions must be continuously monitored to ensure proper vehicle operation. The closed loop error must then be checked mainly in three situations as a response to an user command:

- speed: the user issued an command with a given mean speed, which should be compared with the steady-state mean speed of the vehicle.
- direction: the user issued an command with a given direction, which should be compared to the vehicle direction.
- safe distance to avoid collisions: the user issued an command with a given direction and speed which can cause it to crash. The local control must influence, to prevent collision, and the final distance to the obstacles must be assessed and compared to the defined one.

4.2.9. Summary

Table 4.1 lists the foreseen product specifications.

Table 4.1.: Specifications

	Values	Explanation
Autonomy	4 h	Time interval between battery fully charged and safely discharged
Speed Range	0.1 to 1 m/s	Speed at which the car can operate
Frame Rate	60 fps	Frequency at which independent still images appear on the screen
Camera Range	20 m	How far can the camera capture images without loosing resolution
Camera resolution	480p	Amount of detail that the camera can capture
Communication Range	50 m	Maximum distance between the car and the smarphone without losing connection
speed Error	5 %	Maximum difference between desired and real speed
Direction Error	5%	Maximum difference between desired and real direction
Distance Error	5 %	Maximum difference between desired and real distance to the obstacle
Dimensions	20x12x5 cm	Dimensions of the car
Weight	0.5 kg	Weight of the car

4.3. Initial design

Following an analysis of the product's family tree (remote controlled cars), the state of the art and the QFD matrix in fig. 4.3, an initial design of the product itself can be produced (fig. 4.4). The selected approach was top-down, in the sense that the requirements and specifications were addressed and that resulted in a general diagram of the product concept. Some macro-level decisions were made in this stage to narrow the problem's solutions pool, as follows:

- The car itself should be battery-powered, as it is a free-moving object that is intended to work in environments where trailing cables could interfere with its regular movement.
- The device used to control the car should ideally be one already owned by the user, with an integrated screen (e.g. smartphone), as it would make it more affordable and have a more straightforward interface.

- The protocols for communication between the controlling device and the Rover should be chosen from within the pool of those readily available to smartphones (e.g. Wi-Fi, GPRS, Bluetooth) to keep the price of the overall product down and make it as practical as possible.
- The control and communication unit for the Rover should be divided into two modules: one which can measure and process sensor inputs and control the actuators in real-time, as well as communicate with the user-operated device through a low-latency connection. And another one which can interface directly with the camera module and manage data transmission to the user at the applicational level of the TCP/IP protocol stack, with enough throughput for the specified video resolution and framerate. The latter should also exchange sensor reading values and commands with the user-controlled device, introducing redundancy to the controls and thus allowing for more reliable operation.

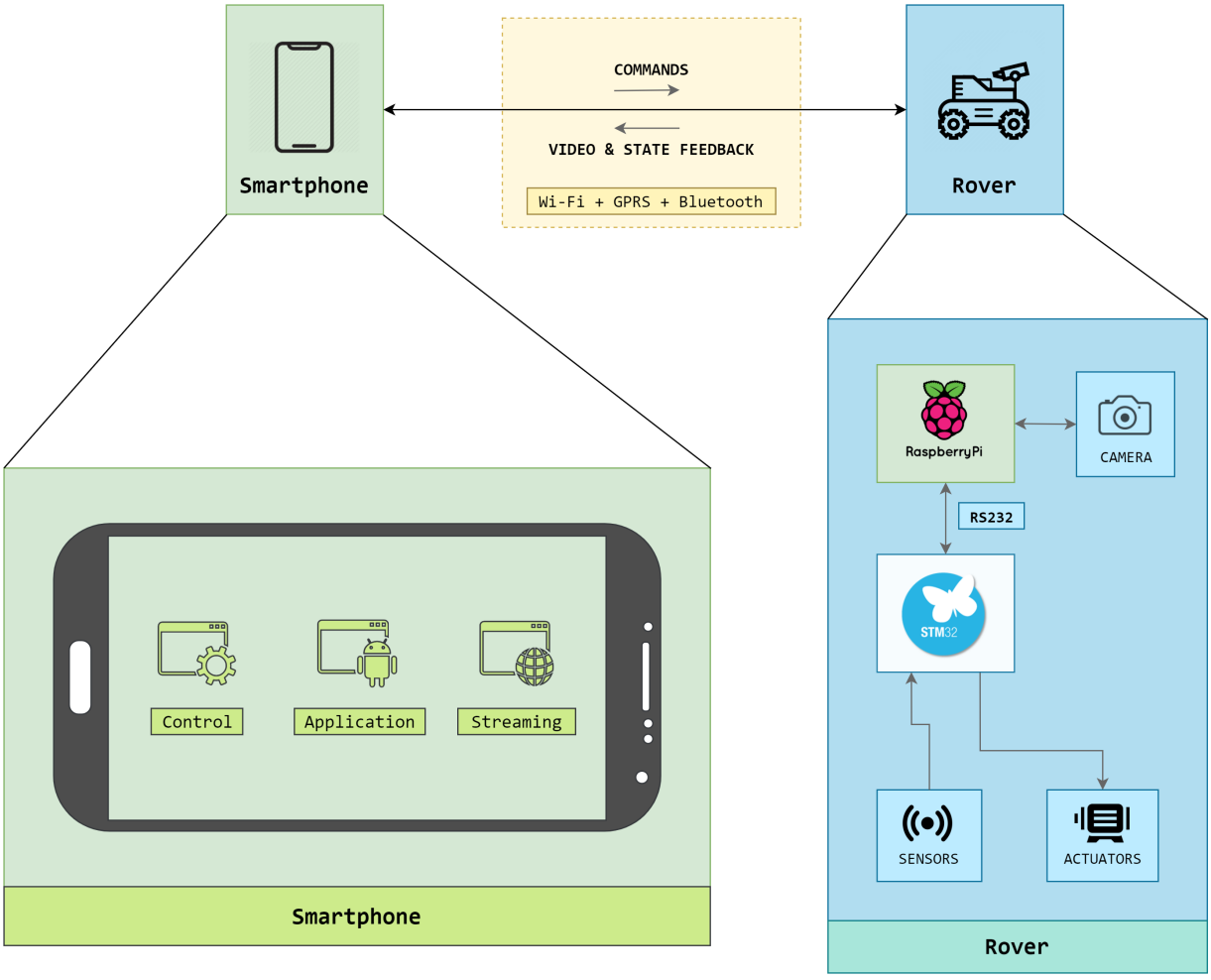


Figure 4.4.: Initial design: Block diagram view

Thus, summarising, the initial design yields the system illustrated in fig. 4.4, comprised of:

- **Raspberry Pi**: Interfaces with the camera directly, streaming that information to the smartphone. It also receives user commands and sends sensorial information it receives from the STM32 module back to the Smartphone, all through redundant Wi-Fi and GPRS connections;
- **STM32**: Receives user commands and sends back sensorial information through a Bluetooth connection, information it also uses to control the actuators;
- **Actuators**: DC Motors that control the movement of the Rover and headlights for nocturnal or low light conditions;
- **Sensors**: Odometric sensors that support the detection of obstacles and luminosity sensors;
- **Camera**: Device connected to the Raspberry Pi that allows the live stream of the car´s surrounding environment;
- **Smartphone**: Grant visual feedback from the live feed of the camera also allowing the user to control the movement of the vehicle intuitively;

For simulation purposes and in conformance with the extraordinary conditions imposed by the recently enacted confinement measures, the need rose to create a virtual environment to simulate the various subsystems of the Rover. This solution allows for integrated testing without the need to deploy it to the hardware, as illustrated in fig. 4.5.

The first of said subsystems is the Physical Environment Virtual Subsystem, which simulates the Rover and the physical environment, also receiving actuator inputs from the Navigation Virtual Subsystem, for which it generates the sensor readings. The latter also exchanges that information, as well as the status of the Rover and the commands from the user with the Smartphone module through a Bluetooth connection, which separates it from the non-simulated environment.

The Remote Vision Virtual Subsystem interfaces with the computer's Web Camera, which is meant to simulate the onboard camera of the Rover. Moreover, it can also communicate through Wi-Fi and GPRS, thus ensuring that communication is kept despite any failure from the Navigation Virtual Subsystem, as well as having shared access to the sensor reading values for monitoring of certain key parameters, depending on their refresh rate.

The RS232 connection between both controlling subsystems ensures proper synchronism and cooperation between them.

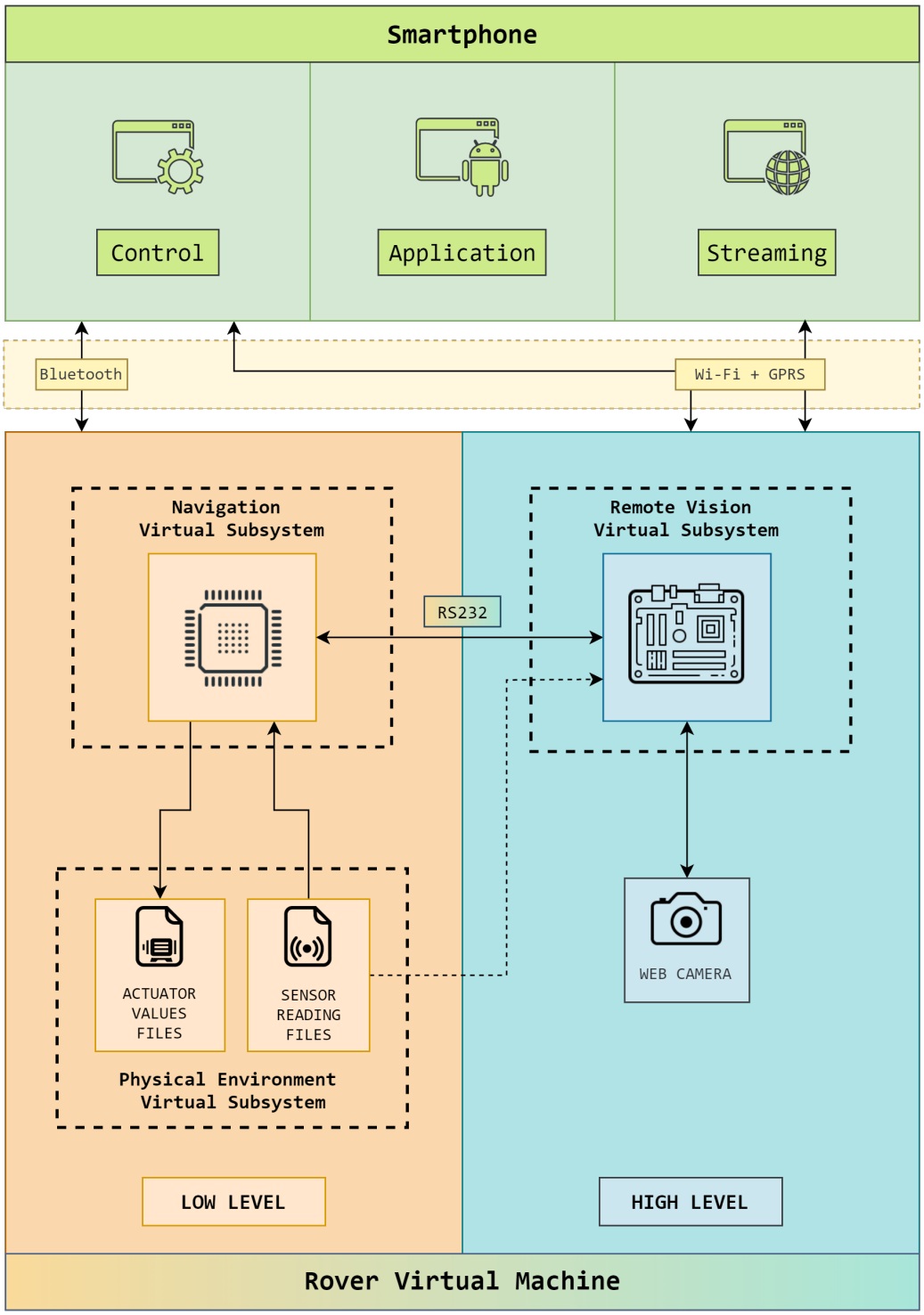


Figure 4.5.: Initial design: Virtual environment block diagram view

4.4. Planning

In fig. 4.6 is illustrated the Gantt chart for the project, containing the tasks' descriptions. It should be noted that the project tasks of Analysis, Design, Implementation and Tests are performed in two distinct iterations as corresponding to the Waterfall project methodology.

Due to unpredictable circumstances, limiting the mobility of team staff and goods, the implementation stage will not be done at full extent, but rather at a simulation stage. Thus, to overcome these constraints, the project focus is shifted to the simulation stage, where an extensive framework as to built to model the system operation, test it, and providing valuable feedback for the dependents modules. As an example, the modules previously connected just by an RS232 link, must now include upstream a web module (TCP/IP) — the data is now effectively sent through the internet, and must be unpacked and delivery serially as expected if only the RS232 link was used.

The tasks are described as follows:

- Project Kick-off: in the project kick-off, the group is formed and the tutor is chosen. A brainstorming about conceivable devices takes place, whose viability is then assessed, resulting in the product concept definition (Milestone 0).

- State of the Art: in this stage, the working principle of the device is studied based on similar products and the system components and its characteristics are identified.

- Analysis: In the first stage — Analysis 1 — contains the analysis results of the state of the art. It should yield the specifications document, containing the requisites and restrictions to the project/product, on a quantifiable basis as required to initiate the design; for example, the vehicle's desired speed should be, at maximum, $2\ m/s$. The second stage — Analysis 2 — contains the analysis of the first iteration of the development cycle.

- Design: it is done in two segments: modules design — where the modules are designed; integration design — where the interconnections between modules is designed. It can be subdivided into conceptual design and solution design.
  - In the conceptual design, several problem solutions are identified, quantifying its relevance for the project through a measuring scale, inserted into an evaluation matrix, for example, QFD.
  - In the solution design, the selected solution is developed. It must include the solution modelling, e.g.:
    * Control system: analytically and using simulation;
    * Transducer design: circuit design and simulation;

    * Power system: power supply, motors actuation and respective circuitry design and simulation;
    * Communications middleware: communication protocols evaluation and selection;
    * Software layers: for all required modules, and considering its interconnections, at distinct levels:
        · front end layer: user interface software, providing a easy and convenient way for the user to control and manage the system.
        · framework layer: software required to emulate/simulate and test the required system behaviour, providing seamless interfaces for the dependents modules
        · back end layer: software running behind the scenes, handling user commands received, system monitoring and control.

- Implementation: product implementation which is done by modular integration. In the first stage, the implementation is done in a prototyping environment — the assisting framework developed, yielding version alpha; in the second stage it must include the coding on the final target modules, yielding prototype beta.

- Tests: modular tests and integrated tests are performed. Tests are generally considered as those performed over any physical component or prototype. Here, it is used as a broader term, to reflect the tests conducted into the system and the several prototypes.

- Functional Verification/Validation: System verification may be performed to validate overall function, but not for quantifiable measurement, due to the latencies involved. Regarding validation, specially for an external agent, thus, it should be limited to user interface validation.

- Delivery: — project closure encompassing:
    1. Final prototype
    2. Support documentation: how to replicate, instruction manual.
    3. Final report
    4. Public presentation

4.5. Foreseen specifications tests

The functional testing is generally regarded as those performed over any physical component or prototype. Here, however, a broader sense is used, to reflect the tests conducted into the system and the several prototypes, under the abnormal present circumstances. Moreover, as indicated in the design, the current development strategy encompasses the virtualization of all hardware components, enclosed in a single virtual environment.

Thus, it does not make sense to perform hardware related tests such as velocity measurements, autonomy, safety, etc. As such, the focus is shifted towards software and control verification, encompassing the following tests: functionality, image acquisition, communication, and control algorithms correctness.

The tests are divided into verification and validation tests.

4.5.1. Verification tests

The verification tests are tests performed internally by the design team to check the compliance of the foreseen specifications. These tests are done after the prototype alpha is concluded.

4.5.1.1. Functionality

The remotely operated vehicle is composed of several modules distributed along several different platforms, some of which distanced from each other. In doing so, the proposed sets of functionalities should be tested in the integrated system, by tracking and analysing the user commands issued along the way until it finally reaches the vehicle (in the virtual environment), assessing if it is correctly processed. For example, if the user issues the vehicle to move to a given place (via smartphone interaction), the message sent to the vehicle must be signalled in each endpoint hit, and the vehicle should move to that place, symbolically detected by the modification of its virtual coordinates.

4.5.1.2. Image acquisition

The vehicle is equipped with remote vision to assist the user in its navigation, thus, requiring the following variables to be tested: frame rate, range, and resolution. In the current scenario, the virtual environment should provide access to a integrated camera, being fairly common nowadays in every modular component, thus, enabling easy testing.

4.5.1.2.1. Frame rate

To test frame rate, the user screen must be updated with the number of frames received from the camera per second and checked against the defined boundaries.

4.5.1.2.2. Range

To test camera's range, an object must be captured at increasing distances, until the object resolution fades or is unclear.

4.5.1.2.3. Resolution

The minimum resolution should be tested as providing the least amount of information required for the user, while minimizing data transfer and processing overhead.

4.5.1.3. Communication

The communication tests are performed in compliance to the specifications provided in Section 4.2.6, namely reliability and redundancy.

4.5.1.3.1. Reliability

To test communication reliability, the most critical communication link is chosen, namely the wireless communication between user's platform (smartphone) and vehicle's platform (virtual environment). Then, the communication link and protocol selected are tested by monitoring the ratio of dropped packets versus total packets, using an appropriate tool on both directions for transmission and reception.

4.5.1.3.2. Redundancy

To test communication redundancy, one communication channel should be turned off, verifying if the communication between nodes is still possible through another communication channel. For example, in the communication between host (user's platform) and remote (virtual environment) guest systems, the priority communication is performed between host and high-level subsystem. However, if this is turned off, the host must also be able to communicate with the remote system via low-level subsystem.

4.5.1.4. Correctness of the control algorithms

As previously mentioned, the speed and position must be continuously monitored to ensure proper vehicle operation. Under the current circumstances, where the sensors and actuators are virtualized, the control

loops can be externally stimulated through input files containing the relevant data. Then, the behaviour of the system can be analysed and verified for some cornercase situations, assessing the control algorithms correctness.

4.5.2. Validation tests

The validation tests should be performed by the client using the product's manual, so it is expected that a user without prior experience with the product should be able to use it correctly and safely. On the current circumstances, validation tests should be limited to user interface validation.

For this purpose, an external agent will be provided with the software application and the respective installation and usage manuals, and the feedback will be collected and processed to further improve the product.

	Nome
1	Group constitution and Tutor choosing
2	Brainstorming about conceivable devices
3	Devices' viability assessment
4	Product concept definition
5	Milestone 0: Product concept Doc
6	State of the Art
7	Analysis 1
8	Specifications Document
9	Analysis 2
10	Design 1
11	Conceptual Design
12	Solution Design
13	Design 2
14	Modules design
15	Unit design
16	Implementation 1
17	Modules implementation
18	Unit implementation
19	Milestone 1: Phase 1 Doc
20	Implementation 2
21	Modules implementation
22	Unit implementation
23	Prototype Alpha
24	Tests 1
25	Modules tests
26	Unit tests
27	Tests 2
28	Modules test
29	Unit test
30	Verification & Validation
31	Delivery
32	Milestone 2a: Final product design - Report
33	Milestone 2b: Final product design - Presentation

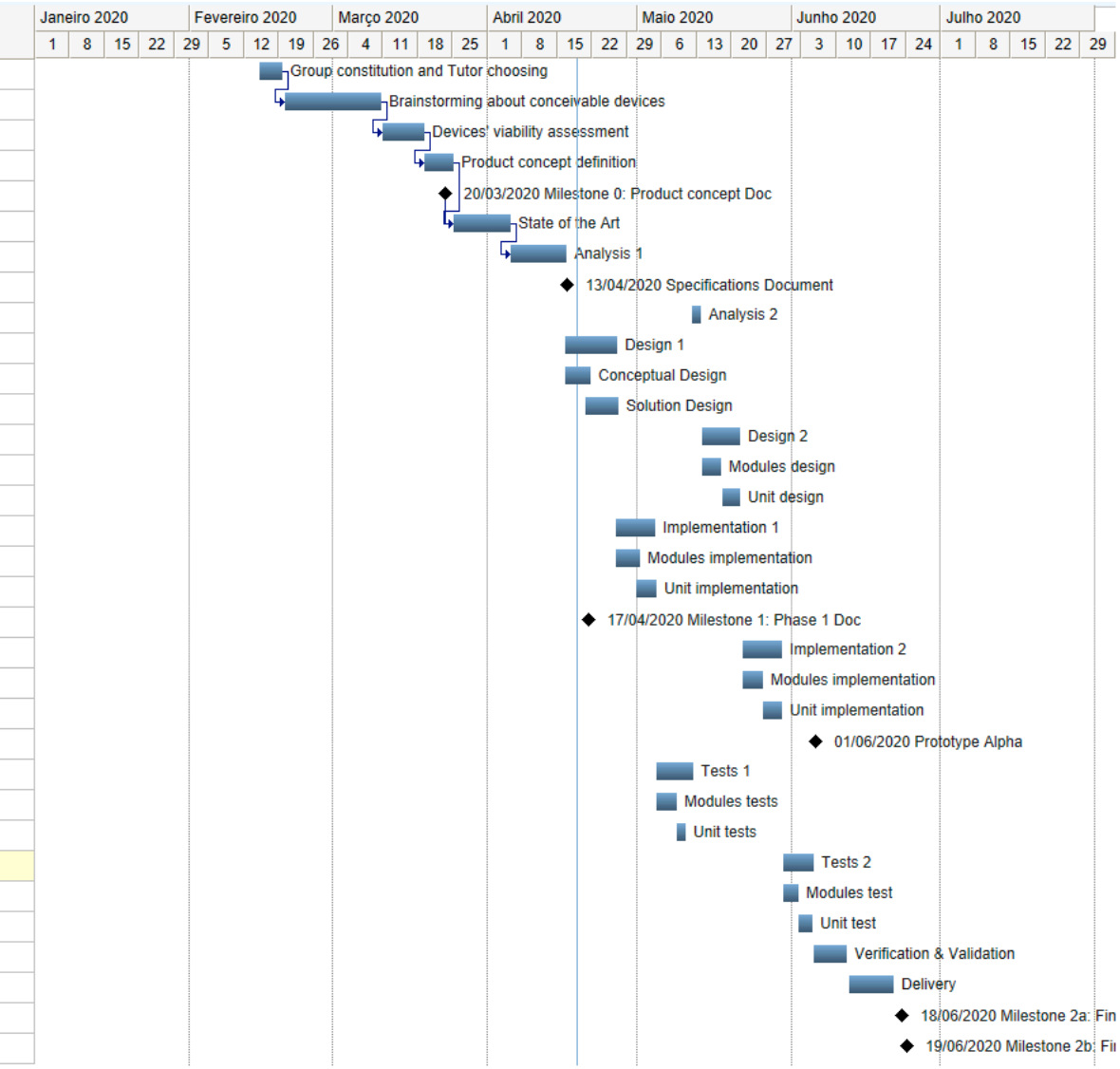


Figure 4.6.: Project planning — Gantt diagram

5. Design

The multi-material part fabrication is a complex topic and most current commercially available systems have been designed for mono-material part fabrication [6] and are unprepared for multi-material processing due to the lack of flexibility and processing capability.

5.1. Navigation Virtual Subsystem

5.1.1. Control

5.2. Physical Environment Virtual Subsystem

5.3. Remote Vision Virtual Subsystem

5.4. Smartphone

6. Implementation

The multi-material part fabrication is a complex topic and most current commercially available systems have been designed for mono-material part fabrication [6] and are unprepared for multi-material processing due to the lack of flexibility and processing capability.

6.1. Navigation Virtual Subsystem

6.1.1. Control

6.2. Physical Environment Virtual Subsystem

6.3. Remote Vision Virtual Subsystem

6.4. Smartphone

7. Testing

The multi-material part fabrication is a complex topic and most current commercially available systems have been designed for mono-material part fabrication [6] and are unprepared for multi-material processing due to the lack of flexibility and processing capability.

7.1. Unit testing

7.1.1. Navigation Virtual Subsystem

7.1.1.1. Control

7.1.2. Physical Environment Virtual Subsystem

7.1.3. Remote Vision Virtual Subsystem

7.1.4. Smartphone

7.2. Integrated-testing

8. Verification and Validation

The multi-material part fabrication is a complex topic and most current commercially available systems have been designed for mono-material part fabrication [6] and are unprepared for multi-material processing due to the lack of flexibility and processing capability.

8.1. Verification

8.2. Validation

9. Conclusion

In this chapter the conclusions and prospect for future work are presented.

Bibliography

[1] Fumio Harashima, Masayoshi Tomizuka, and Toshio Fukuda. Mechatronics-" what is it, why, and how?" an editorial. IEEE/ASME Transactions on Mechatronics, 1(1):1–4, 1996.

[2] Jürgen Gausemeier and Stefan Moehringer. New guideline vdi 2206-a flexible procedure model for the design of mechatronic systems. In DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm, 2003.

[3] Ian Sommerville. Software process models. ACM computing surveys (CSUR), 28(1):269–271, 1996.

[4] Michael A Cusumano and Stanley A Smith. Beyond the waterfall: Software development at microsoft. 1995.

[5] Bernd Bruegge and Allen H Dutoit. Object-Oriented Software Engineering Using UML, Patterns and Java-(Required), volume 2004. Prentice Hall, 2004.

[6] Terry Wohlers. Wohlers report 2011: Additive manufacturing and 3d printing, state of the Industry, wohlers associates, ft. Collins, CO, 2011.

Appendices

A. Use cases (Detailed description)

In this section, the main use cases are described extensively. As many uses cases are similar, only changing the parameter to which they refer to, the Lighting subsystem was used as an example for the Manage<Parameter> use case, where <Parameter> is Temperature, DoorBell, DateAndTime and SystemNotifications.

Although several notifications from relevant events are to be notified by the Master machine to the User, for brevity purposes only the most relevant one is showcased here, namely NotifyManufEnd.

Table A.1.: Use case LoadGeometryFile

Use case name	**LoadGeometryFile**
Participating actors	Initiated by the User
Flow of events	1. The User selects the load geometry file option. 2. The User selects the geometry file to load from a list. 3. If the file is successfully loaded, the filename is displayed, and the file previewed (include PreviewGeometryFile use case). 4. Otherwise, an error message is displayed to the user.
Entry conditions	The User has started the MMSLS machine control software and has previously generated a valid geometry file.
Exit conditions	The file is successfully loaded, previewed and with the filename displayed or an error message is displayed to the User.
Quality requirements	Feedback must be given to the user within 2 seconds; if files are “heavy”, display an ongoing processing.

Table A.2.: Use case PreviewGeometryFile

Use case name	**PreviewGeometryFile**
Participating actors	Initiated by the User
Flow of events	1. The User loads the geometry file. 2. The geometry file is displayed on the canvas.
Entry conditions	A valid geometry file is loaded into the application.
Exit conditions	The geometry file is displayed on canvas.
Quality requirements	The preview should be resizable to accommodate the various component’s dimensions.

Table A.3.: Use case AssignColorsToLaserParams

Use case name	**AssignColorsToLaserParams**
Participating actors	Initiated by the User
Flow of events	1. The User selects a layer. 2. The User associates a material’s color to its laser’s processing color counterpart. 3. The User can edit the laser’s processing color attributes.
Entry conditions	The file is successfully loaded into the application and the preview is editable.
Exit conditions	The materials’ colors has been assigned to valid laser’s processing color with the respective processing parameters.
Quality requirements	Allow multiple line selection.

Table A.4.: Use case LoadManufFile

Use case name	**LoadManufFile**
Participating actors	Initiated by the User
Flow of events	1. The User selects the load manufacturing file option. 2. The User selects the manufacturing file to load from a list. 3. If the file is successfully loaded, the filename is displayed. 4. Otherwise, an error message is displayed to The User.
Entry conditions	The User has started the MMSLS machine control software and has previously generated a valid manufacturing file.
Exit conditions	The file is successfully loaded and with the filename displayed or an error message is displayed to the User.
Quality requirements	Feedback must be given to the user within 2 seconds; if files are 'heavy', display an ongoing processing.

Table A.5.: Use case ConnectToMach

Use case name	**ConnectToMach**
Participating actors	Initiated by the User
Flow of events	1. The User selects the appropriate connection to the machine from a list. 2. The User selects the Connect to Machine option. 3. If the connection is successful, the connection status is updated to Connected and machine operation options are enabled. 4. Otherwise, an error message is displayed to The User.
Entry conditions	The User has started the MMSLS machine control software and there is physical connection between the Master system and the Multi-Material Sintering (MMS) machine.
Exit conditions	The connection is established between Master system and the MMS machine or an error message is displayed to the user.

Table A.6.: Use case DisconnectToMach

Use case name	**DisconnectToMach**
Participating actors	Initiated by the User
Flow of events	1. The User selects the Disconnect to Machine option. 2. If the disconnection is successful, the connection status is updated to Disconnected and machine operation options are disabled. 3. Otherwise, an error message is displayed to The User.
Entry conditions	A successful connection between the Master system and the MMS machine is established.
Exit conditions	The connection is closed and the machine operations options are disabled or an error messaged is displayed to the user

Table A.7.: Use case ManualResetMach

Use case name	**ManualResetMach**
Participating actors	Initiated by the User
Flow of events	1. The User selects the axis to reset, the direction and the reset distance. 2. When the reset is satisfactory, The User acknowledges this fact by selecting the option Reset end. 3. The option to Start Manufacturing is now enabled.
Entry conditions	A successful connection between the Master system and the MMS machine is established.
Exit conditions	The reset is satisfactory (User has selected option Reset End) and the Start Manufacturing is enabled.
Quality requirements	Provide feedback to the User of the manual reset operations (ascent/descent of the axis)

Table A.8.: Use case StartManuf

Use case name	**StartManuf**
Participating actors	Initiated by the user
Flow of events	1. The User selects the Start manufacturing option. 2. If successful, the machine initiates the manufacturing process the machine status is updated to Run.
Entry conditions	1. A successful connection between the Master system and the MMS machine is established. 2. Reset is finished. 3. Valid geometry and manufacturing file have been loaded.
Exit conditions	• Success: The MMS machine status is updated to Run, the Start manufacturing option is disabled, and the options Pause manufacturing and Stop manufacturing are enabled. • Fail: An error message is displayed to the user.
Quality requirements	Update the relevant processing information to the User (include use case UpdateInfo).

Table A.9.: Use case PauseManuf

Use case name	**PauseManuf**
Participating actors	Initiated by the user
Flow of events	1. The User selects the Pause manufacturing option. 2. If successful, the manufacturing process is paused.
Entry conditions	The manufacturing has started (startManuf was triggered), but has not yet finished.
Exit conditions	• Success: The manufacturing process is paused, and the MMS machine status is updated to Idle. The Pause manufacturing option is disabled and the Start manufacturing option is re-enabled. • Fail: An error message is displayed to the user.

Table A.10.: Use case StopManuf

Use case name	**StopManuf**
Participating actors	Initiated by the User
Flow of events	1. The User selects the Stop manufacturing option. 2. If successful, the manufacturing process is stopped.
Entry conditions	The manufacturing has started (startManuf was triggered), but has not yet finished.
Exit conditions	• Success: The manufacturing process is stopped, and the MMS machine status is updated to Stopped. The Stop manufacturing option is disabled and the Start manufacturing option is re-enabled. • Fail: An error message is displayed to the user.

Table A.11.: Use case NotifyManufEnd

Use case name	**NotifyManufEnd**
Participating actors	Initiated by MMS-Mach; User participates
Flow of events	1. When manufacturing is completed, the Master system notifies The User about this fact.
Entry conditions	The manufacturing has started (startManuf was triggered), and is not paused or stopped.
Exit conditions	The manufacturing process stops and the User is notified about this fact.

Table A.12.: Use case VisualizeManuf

Use case name	**VisualizeManuf**
Participating actors	Initiated by MMS-Mach or Laser; User participates
Flow of events	1. Relevant manufacturing information is sent by the MMS-Mach or the Laser to the Master system — on a time-basis or triggered by a relevant event like the manufacturing completion of one layer — that is updated to the User.
Entry conditions	The manufacturing has started (startManuf was triggered)
Exit conditions	The manufacturing is stopped or has ended.
Quality requirements	• Time-basis The relevant manufacturing information must be updated with 1 second refresh rate. • Event The information associated with the triggering event must be updated immediately.

B. Sequence Diagrams

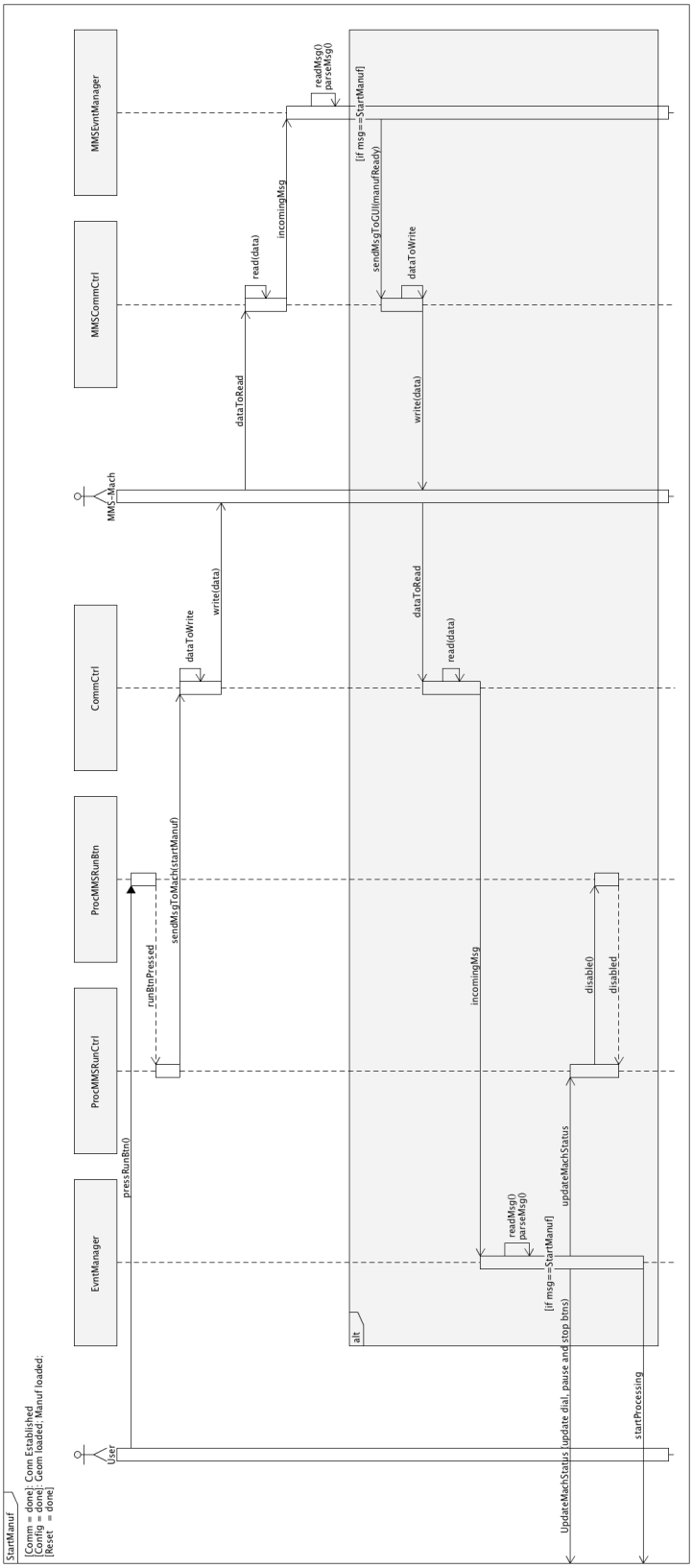


Figure B.1.: Sequence diagram for the StartManuf use case (Part 1)

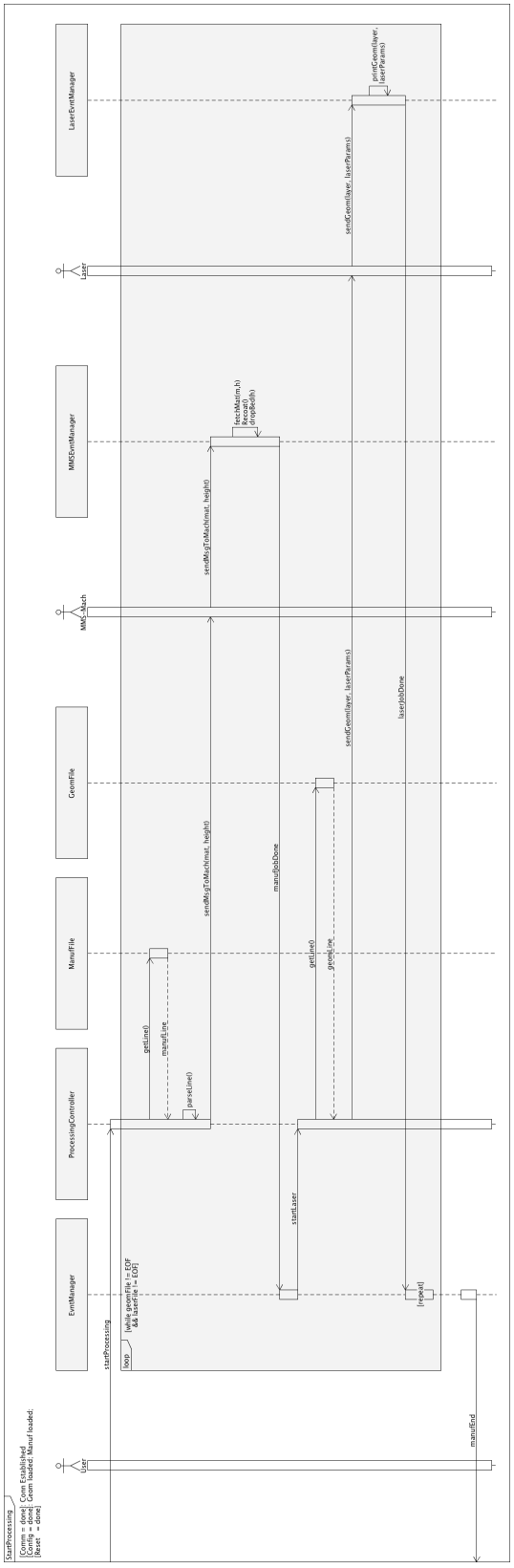


Figure B.2.: Sequence diagram for the StartManuf use case (Part 2)