ETSA02-ADM-INS

Project Instructions for the LU Rumble

Version 0.5

Prepared by Markus Borg
Dept. of Computer Science, Lund University

February 5, 2018

Contents

Revision History

Name	Date	Change description	Version
Markus Borg	2017-12-07	Initial draft.	0.1
Markus Borg	2017-12-27	Complete draft of intro, engineering, monetizing, and strategizing. Sent for external review.	0.2
Markus Borg	2017-12-31	First draft of sprints.	0.3
Markus Borg	2018-01-02	First complete draft.	0.4
Markus Borg	2018-02-05	Updated after internal review.	0.5

1 Introduction

This document describes the project in ETSA02 Introduction to Software Engineering – Methodology. Together with other students, you will work together to engineer a robot for competition in LU Rumble – a local instance of team battles in Robocode, a programming game in which robots compete in battles without human intervention by executing their programs.

In the remainder of this document, we use the term "group" to refer to the students working together during the project. While the student constellations would better be described as teams, i.e., groups of people working together toward a common goal, we reserve the term "team" to robots competing together in Robocode.

This document is organized as follows: Section 1.1 presents the learning goals of the project in relation to the ETSA02 formal course description.

1.1 Learning goals

The project primarily aims to increase your ability to develop high quality software systems using established software engineering best practices. Moreover, basic software business concepts will be introduced as each project group will offer their product on a highly competitive market.

The project complements the theoretical concepts introduced during the lectures by taking a practical approach to the presentation of fundamental software engineering concepts such as specification, version control, testing, sprints, and releases. By the end of the course, you will have acquired new skills with essential components of the contemporary software engineering tool chain: the Java programming language, the Eclipse integrated development environment, the JUnit testing framework, the git configuration management system, GitHub cloud-based project hosting, the Maven build system, and the Checkstyle, PMD, and FindBugs automated quality assurance tools.

On top of your skills with traditional software engineering concepts and the tool chain, you will address aspects of market-driven software engineering. Each project group will practice making critical business and engineering decisions in a controlled fashion. The course introduces several important activities in software business, including analyzing the software ecosystem, finding a niche market, marketing a product, competitive pricing strategies, and customer negotiations.

More formally, the project is primarily designed to meet the practical requirements of the ETSA02 course description, i.e., the "Competencies and skills". However, the project will also reinforce the theoretical concepts listed under "Knowledge and understanding" and it contributes to the "Judgment and approach" section.

- Competencies and skills
  - Be able to develop a project plan, requirements specification and test plan for a small project.
  - Be able to review a project plan, requirements specification and test plan for a small project.
  - Be able to formulate text for project documentation.
- Knowledge and understanding
  - Be able to define basic terms and definitions in software engineering.
  - Be able to describe the most common software development processes.
  - Be able to explain the most important steps in requirements engineering
  - Be able to explain how testing is conducted.
  - Be able to describe what a software architecture design is.
  - Be able to describe the most omportant steps in project planning and project tracking.
  - Be able to describe how an organization plan and manage a series of projects.
- Judgment and approach
  - Understand the complexity involved in developing software systems
  - Have an understanding of the professional role of the engineer

1.2 Robocode – Build the best, destroy the rest!

The goal of Robocode is to implement the behavior of a robot to compete against other robots in a battle arena. The contestants have no direct influence on the game, instead they write the AI of the robot telling it how to behave and react on events occurring in the battle arena. Battles are running in real-time and on-screen, see Figure 1.1. Robocode battles, referred to as "rumbles", are either in the form of duels between individual robots, free-for-all battles with multiple robots, or battles between robot teams.

From http://robocode.sourceforge.net/docs/ReadMe.html: Robocode offers a complete development environment, and comes with its own installer, built-in robot editor and Java compiler. Robocode only pre-requires that a JVM (Java Virtual Machine) to exist already on the system where Robocode is going to be installed. Hence, everything a robot developer needs to get started is provided with the main Robocode distribution file (robocode-xxx-setup.jar). Robocode also supports developing robots using external IDEs like Eclipse.

Robocode is an Open Source project, which means that all sources are open to everybody. In addition, Robocode is provided under the terms of EPL (Eclipse Public License). The Robocode game was originally started by Mathew A. Nelson as a personal endeavor in late 2000 and became a professional one when he brought it to IBM

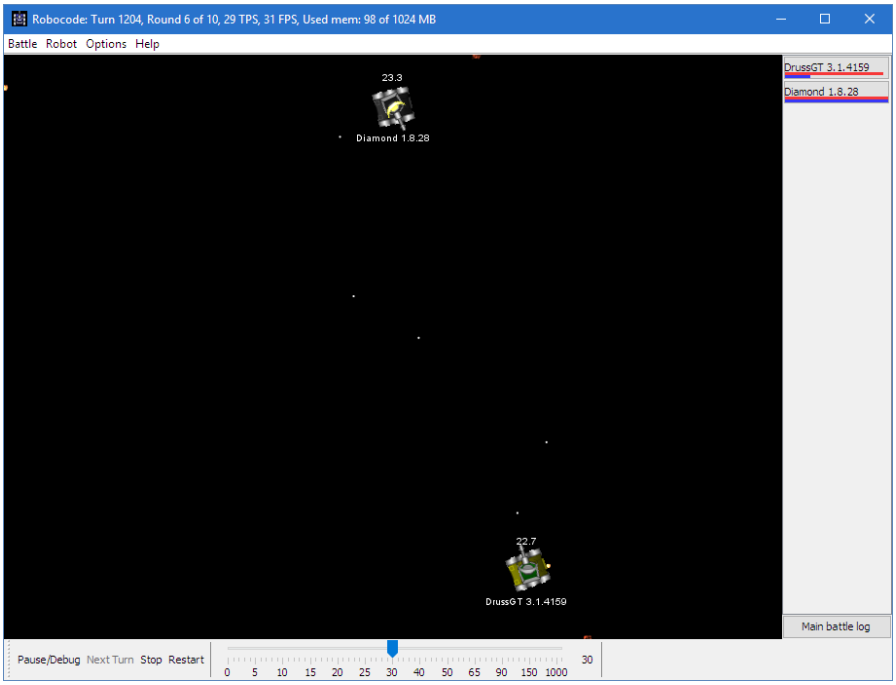

Figure 1.1: Screenshot from the Robocode battle arena. (Image credit: robowiki.net user MultiplyByZer0)

in July 2001. In the beginning of 2005, Mathew convinced IBM to release Robocode as Open Source on SourceForge. Eventually, Flemming N. Larsen took over the Robocode project at SourceForge as administrator and developer in July 2006 to continue the original Robocode game – which now is hosted on GitHub.

1.3 Project overview

Each project group will develop a robot using established software engineering practices. Furthermore, each group will compose a robot team to compete in a "LU Rumble" at the final lecture in the course. However, no project group is allowed to field their own robot – instead robots developed by other groups must be purchased on an open market (in truth, a somewhat regulated market). Consequently, each project group has two primary goals: 1) maximizing profit by selling a successfully engineered robot on the market and 2) winning the LU Rumble by composing a competitive Robocode team. Project groups pursue the two goals by completing three main activities that we refer to as *strategizing*, *engineering*, and *monetizing*, respectively.

Figure 3.1 shows the four main phases of the project. First, during the course inception, the course infrastructure and the project tasks are introduced. More importantly, project groups consisting of six students (preferably!) are established. Second, the backbone of the course follows: three development sprints mixing engineering, monetizing,

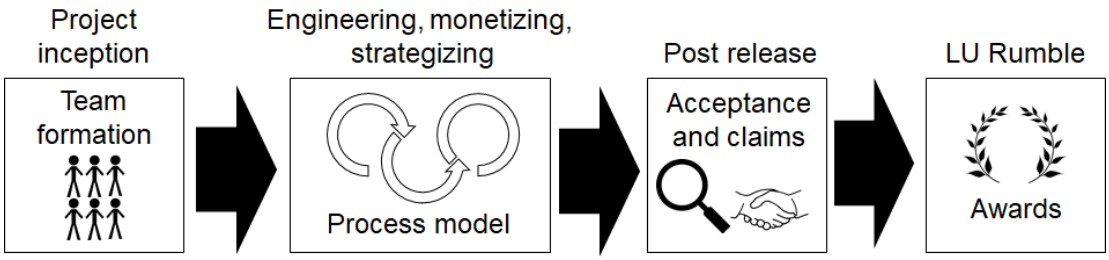

Figure 1.2: The four phases of the project. Expect also a post mortem phase, concluding with the take-home exam.

and strategizing. Third, purchasers of robots perform acceptance testing to ensure that the delivered robot fulfills the expectations – otherwise purchasers file business claims to require money back. Fourth, in the last phase of the project, the LU Rumble takes place followed by an awards ceremony to recognize the winning robot team – and the most profitable project groups.

Figure 1.3 shows the context of each individual group during the project. Organizationally, your group has three divisions: 1) engineering, 2) purchasing, and 3) sales. Your group will assign the following roles: 1) project manager, 2) requirements engineer, 3) development lead, 4) test lead, 5) domain expert, and 6) sales engineer. Each role is described in detail in Section 5.1.

Project groups do not exist in isolation, instead they act in a software ecosystem [REF]. During engineering of the robot, each project group will be part of two supplier-customer relationships. First, your sales engineer will be responsible for marketing the robot under development. The domain expert of another project group will sign a contract of sale, establishing a formal relationship. Your sales engineer will be the primary communication point for the purchasing group's domain expert, i.e., a supplier-customer relationship involving feedback, feature requests, and negotiations. Second, analogously, your domain expert will act as the customer in a supplier-customer relationship with another project group. Note that, in some cases, project groups might buy from and sell to the same other group – this is not regulated in the ecosystem.

On top of the two supplier-customer relationships, your project group will also communicate with the regulatory body (represented by your project supervisor). Your project group will hand in various signed contracts to the regulatory body, turn to it if to get support during supplier-customer negotiations, and to report the results from your acceptance testing. The regulatory body might also contact you with regulatory changes during the course of the project. Finally, you will practice one-way communication with the Robot Market, i.e., the open market where you will sell your Robot. All project groups will complement their robot teams with additional robots during the last week of the project – and your sales engineer will be responsible for maximizing sales of your robot.

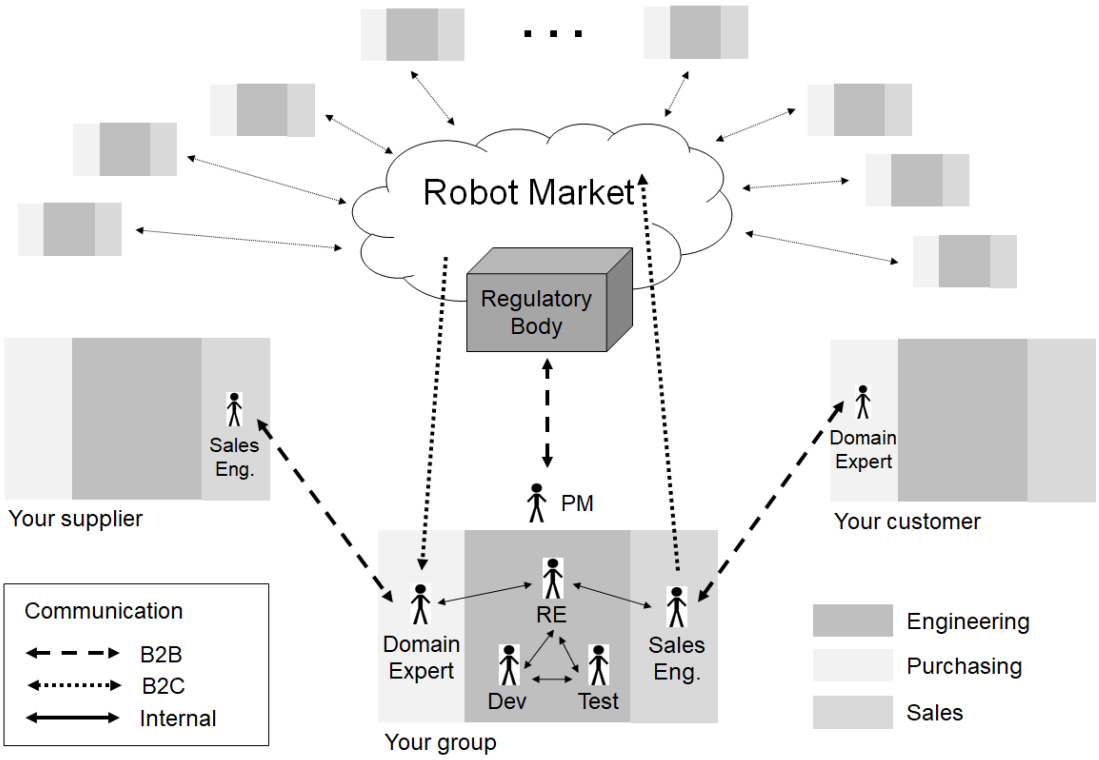

Figure 1.3: The project context. Your project group will interact with your supplier's sales engineer, your customer's domain expert, the open Robot Market, and the regulatory body.

1.4 Project grading

To be written...

The project will be graded as 3, 4, 5 or UG. TODO: Group grading and individual grades.

The project supervisors' overall grading guidelines include the following:

- The final release meets the final requirements specification. This is partly indicated by the customer's successful acceptance testing.
- The delivered robot is successful and non-trivial.
- All the robots behaviors are covered by the requirements specification.
- The test specification describes a comprehensive verification of the robot. Automated test cases and test reports are provided.
- The project group adhered to the process model and met all deadlines.
- All deliverables are of high quality. Appropriate technical language and a consistent look-and-feel across documents.

2 The project from a bird's eye view

The project consists of three main activities, interconnected as presented in Figure 2.1. This section presents an overview of the three activities.

2.1 Engineering the robot

Each project group will develop a robot (see Figure 2.2) according to established engineering principles. These principles include product prototyping, integrated requirements engineering, and automated testing. The process model is organized in three development sprints, further described in Section 3.2. The engineering process is dominated by three core activities: *specification*, *construction*, and *verification*. In line with agile development methodologies, all sprints consist of a mix of these three activities. Still, the groups' focus during the project will progress from specification, through construction, to verification – also reflected by the order of the lectures.

2.1.1 Specification

Requirements engineering, i.e., a systematic approach to developing a product with an end-user perspective, will be an integrated activity throughout the project. The requirements will be captured in a Software Requirements Specification (SRS) adhering to a template structure provided by the Institute of Electrical and Electronics Engineers (IEEE). IEEE is the world's largest technical professional organization, and a natural resource for software engineers – possibly contented by the Association of Computing Machinery (ACM). Both organizations support engineers in development of software products adhering to state-of-practice approaches and provide ethical guidelines.

The IEEE SRS template provides a standard document structure: 1) an introduction section, 2) an overall description of the product under development, and 3) a list of product features further broken down into detailed requirements. Furthermore, the template supports engineers to specify quality requirements such as performance, maintainability, and memory footprint. Project groups are recommended to use a combination of traditional requirements of "shall" format and step-wise use case descriptions. While also user stories are covered in the lectures, it uses a narrative style that is less appropriate for development of autonomous robots.

The SRS connects all roles in the project group. While the requirements engineer is responsible for the SRS, the sales engineer is responsible for developing a marketing strategy based on its content. The SRS will be public on the Robot Market, i.e., when other project groups consider purchasing your robot, they will critically review your SRS. The domain expert, having the most chance to predict what features will be valued on

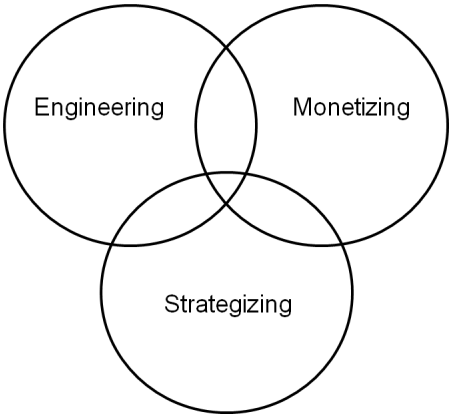

Figure 2.1: Project components.

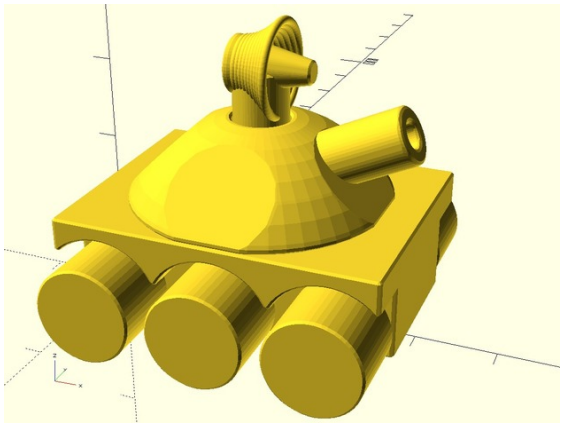

Figure 2.2: Model of the Robocode robot (© Klaus Knopper under CC BY-SA 3.0).

the Robot Market, will provide important input to the SRS – effectively acting as a proxy-customer until the supplier-customer relationship has been established. Finally, the development lead is responsible for implementing the SRS in source code and the test lead is responsible for verifying that the developed product fulfills the SRS.

There are several quality attributes that a high-quality SRS should comply with. First, the SRS should be complete, i.e., if properly implemented, the developed product shall be a competitive robot. Just as importantly, the robot shall not offer any features that are not properly captured in the SRS – the customer is not allowed any surprises. Second, the SRS should constitute a coherent document that is easy to follow. Furthermore, each individual requirement should be:

Complete All information needed to implement source code to fulfill the requirement shall be specified. Developers must not be forced to read between the lines.

Consistent There are no contradictions between individual requirements.

Feasible The requirement can be practically implemented by the developers given the available resources. Also, the requirement will bring value to the end user or another stakeholder.

Modifiable An SRS is rarely static, thus all requirements must be able to change. Each requirement should be clearly identifiable.

Unambiguous A requirement shall only be interpretable in one way, i.e., no subjectivity can exist. Quality requirements should be quantified, not described using vague adjectives.

Design free Requirements shall specify what the system must or must not do, but not how the software will ensure the requirement is met – leave that to the construction phase.

Testable Requirements shall be possible to verifiable, preferably by testing, otherwise inspection.

As a result from the specification activities, each project group shall produce the following deliverables, further described in Section 3.2:

Sprint 1 SRS v0.5 – Main features captured

Sprint 2 SRS v0.9 – Detailed requirements specified

Sprint 3 SRS v1.0 – A complete Robot specification

2.1.2 Construction

The first two computer labs will help project groups set up Robocode development in Eclipse and source code control in git. Moreover, the second computer lab will initiate automated unit testing, an activity that overlaps construction and verification.

An early construction activity is software design. In this project, the design is limited to object-oriented design – the robot development is too restricted to address any considerations on the software architecture level and there is also no user interaction design. Each project group will present the Object-Oriented Design (OOD) using a Unified Modeling Language (UML) class diagram. As the project in inspired by agile development methodologies, there will be no big design activity upfront. Instead, the OOD will evolve as the construction phase proceeds. The development lead is responsible to complement the final source code delivery with a class diagram to describe the OOD after-the-fact. Any project group that recognizes a value in using class diagrams in their robot marketing are of course welcome to do so. Furthermore, the final source code deliverable shall be complemented by Javadoc documentation.

As most practical construction work will be done outside of scheduled sessions, how the backbone of the programming will be conducted must be decided by the individual project groups. Still, for this project, we recommend groups to follow a handful of well-known agile development practices, originally proposed as part of Extreme Programming (XP) [REF]. Below we list five that we believe are particularly suitable for the process model, along with their Wikipedia definitions:

Pair programming All code is produced by two people programming on one workstation. One programmer has control over the workstation and is thinking mostly about the coding in detail. The other programmer is more focused on the big picture, and is continually reviewing the code that is being produced by the first programmer. Programmers trade roles after minute to hour periods. The pairs are not fixed; programmers switch partners frequently, so that everyone knows what everyone is doing, and everybody remains familiar with the whole system, even the parts outside their skill set. This way, pair programming also can enhance team-wide communication and goes hand-in-hand with the concept of Collective Code Ownership.

Continuous integration The development team should always be working on the latest version of the software. Since different team members may have versions saved locally with various changes and improvements, they should try to upload their current version to the code repository every few hours, or when a significant break presents itself. Continuous integration will avoid delays later on in the project cycle, caused by integration problems, aka. “big bang integration”.

Code refactoring Restructuring existing code without changing its external behavior. Refactoring improves quality attributes of the software, e.g., code readability, performance, maintainability. Refactoring is often initiated when a so called “code smell” is identified, i.e., structures in the code that indicate violation of fundamental design principles and negatively impact design quality.

Test-driven development Create unit tests before the eventual code is written. This approach is intended to stimulate the programmer to think about conditions in

which his or her code could fail. Test driven development proceeds by quickly cycling through the following steps: 1) write unit tests, 2) run tests, watch them fail, 3) write code, 4) run tests, watch them pass, and 5) refactor the code. Adhering to test-driven development leads to a large set of test cases that can be automatically executed, increasing developers' confidence when e.g. refactoring code.

Collective code ownership Everyone is responsible for all the code; therefore, everybody is allowed to change any part of the code. Collective code ownership is not only an organizational policy but also an attitude. Pair programming, especially overlapping pair rotation, contributes to this practice: by working in different pairs, programmers better understand the system context and contribute to more areas of the code base. Collective code ownership may accelerate development because a developer who spots an error can fix it immediately, which can reduce bugs overall. However, programmers may also introduce bugs when changing code that they do not understand well. Sufficiently well-defined unit tests should mitigate this problem: if unforeseen dependencies create errors, then when unit tests are run, they will show failures.

The construction phase involves prototyping to quickly create a working robot. Prototyping supports specification activities by enabling early executable software – truly beneficial in contexts where the requirements are very unstable. Already at the end of sprint 1, around the LU Robot Fair, the project group shall deliver a first prototype of the robot. The prototype will help the purchaser to understand your product vision, and enable better feedback. The second deliverable shall be a Minimum Viable Product (MVP) inspired by lean software development practices [REF], i.e., a product with just enough features to satisfy early customers and to stimulate feedback for future robot evolution throughout Sprint 3.

As a result from the construction activities, each project group shall produce the following deliverables. Each release shall be prepared as two jar-files: 1) one jar-file for customers and 2) one jar-file with embedded source code for the regulatory body. As previously stated, the final release to the regulatory body shall be complemented by a class diagram and Javadoc documentation, as further described in Section 3.2:

Sprint 1 Robot Alpha release (Robot v0.5) – A prototype

Sprint 2 Robot Beta release (Robot v0.9) – An MVP

Sprint 3 Robot Final release (Robot v1.0) – The final robot

2.1.3 Verification

The first two computer exercises will introduce working with the JUnit framework for automated testing. Project groups will work with unit testing as a natural companion to the programming during development. While JUnit is designed for automated execution of unit test cases, groups will also use JUnit for system testing of both functional and non-functional (quality) aspects of the robot under development. Finally, groups will

learn to use measure code coverage and use static code analysis tools to support software quality assurance.

The test lead is responsible for delivering three growing development artifacts during the projects, as well as a final test report at the final release. First, a suite of automated unit test cases shall evolve during the project. At sprint 1, an initial version shall simply be executable, but at the final release all source code shall be exercised by unit tests. Second, a suite of automated system test cases shall be developed – an initial version at sprint 2, and the complete suite with the final release. Third, a test specification that describes the project group's test and quality activities. At sprint 3, all requirements in the SRS shall be verified either by automated test cases or targeted by a convincing argumentation that the requirement has been verified through inspection.

Verification activities are naturally supported by tools. Apart from the JUnit framework, the project groups will use Eclipse plugins to measure code coverage and a set of static analysis tools to automatically analyze the source code from different quality perspectives. TODO: describe the tools to use.

The final robot release shall be accompanied with a test report, signed by the test lead, showing that all test cases have been executed – and all corresponding test verdicts. The customer will surely expect that all test cases passed, i.e., that you provide evidence that the entire SRS has been fulfilled by the final robot. In addition to the test verdicts, the test report shall contain appendices with extracted report from the code coverage tool as well as the static code analysis tools.

As a result from the verification activities, each project group shall produce the following deliverables, further described in Section 3.2:

Sprint 1 Test specification v0.5 – Initial unit test suite

Sprint 2 Test specification v0.9 – Initial system test suite

Sprint 3 Test specification v1.0 and a signed test report

2.2 Monetizing the robot

One of the primary goals of the project is to maximize the profit of the group, i.e., monetizing the robot. The business aspects of the project involves a combination of bespoke software engineering and software engineering for an open market. All project groups will offer their robot on a highly competitive market – consisting of the other groups, each with a €100 budget to invest in a robot team for the LU Rumble.

The project groups will market their Robot already at a robot fair already after Sprint 1. At this early point in time, only an Alpha release of the robot will be available and an initial version of the SRS – still, the sales engineer is responsible for developing a convincing marketing concept. All groups will get the chance to pitch their robot through a two minute video at the robot fair, and also present it online on the Robot Market, complemented by the most recent SRS.

After the robot fair, all project groups will submit a "purchase array" to the regulatory body. The purchase array consists of one element per group, and each element

	Robot A	Robot B	Robot C	Robot D	Robot E	Robot F	Robot G
Group A	N/A	€10	€10	**€60**	€20	€10	€10
Group B	€10	N/A	**€30**	€22	€35	€30	€15
Group C	**€10**	€10	N/A	€30	€22	€25	€10
Group D	€10	**€15**	€29	N/A	€32	€31	€10
Group E	€10	€10	€30	€23	N/A	€12	**€30**
Group F	€10	€10	€28	€17	**€40**	N/A	€30
Group G	€10	€10	€10	€27	€10	**€50**	N/A

Figure 2.3: Example of seven purchase arrays submitted to the regulatory body.

should contain an integer representing how much your group is willing to pay for the corresponding robot. You are not allowed to make an offer for your own robot, and you are not allowed to offer less than €10 for any other robot. The purchase array should be signed by the domain expert and delivered as a physical copy to the regulatory body through the mailbox in "grå skåpet".

Figure 2.3 shows an example of purchase arrays submitted to the regulatory body. The purchase arrays are never disclosed, the regulatory body matches project groups into supplier-customer relationships. In the example, cells with bold text show winning bids. Group D, highlighted with a white row, was particularly interested in Robots E, F, and C – but unfortunately, other project groups offered more. Group D instead got to buy Robot B for €15. Note that the highest bid for a robot can be a tie, such as for Robot A and G. In the case of Robot A, no project group offered more than the minimum €10, which resulted in a (possibly unexpected) bargain buy.

When all project groups have provided their purchase arrays, the regulatory body will commence its work to establish supplier-customer relationships. For each robot, the highest bidder turns into a customer of the group engineering the corresponding robot. If there is a tie, the customer relationship will be assigned randomly. In the end, all project groups will 1) pre-order a robot from a supplier, and 2) have a customer that offered at least €10 to pre-order a robot (the price of the cheapest basic bot, see TODO). Note that the decision of the regulatory body cannot be appealed.

Once robots have been pre-ordered, 25% of their costs will be transferred from the customer to the supplier. This amount will not be refunded in any case – it represents the risks of purchasing a robot from an external organization. The remainder of the cost will be transferred at the delivery of the final robot. After the final delivery, the domain expert of the purchasing project group conducts acceptance testing of the robot. If the acceptance testing shows that delivered robot does not fully comply with the SRS, the project manager of the purchasing group is entitled to file a business claim to the regulatory body, further described in Section 4.1.

Each project group will also offer the robot under development on the Robot Market. After Sprint 3, all groups will seek to complement the pre-ordered (by now hopefully delivered) robot with additional robots to build a competitive robot team for the LU

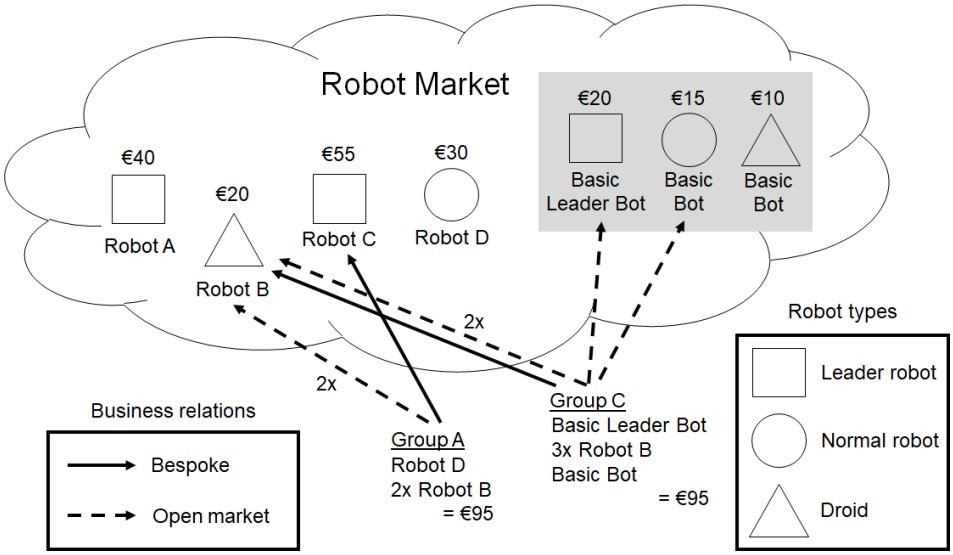

Figure 2.4: Example of two project groups purchasing robots for their teams.

Rumble. Throughout Sprints 2 and 3, the sales engineer is responsible for marketing the robot on Robot Market. Each group will also set a price for its robot – a group must for at least ask as much as what the bespoke customer has paid. Before the LU Rumble, the domain expert of each project group will sign purchase orders and deliver to the regulatory body through "grå skåpet".

Figure 2.4 shows an example of how two project groups purchase robots. After the robot fair, Group A strikes a deal with Group C. As Robot C was quite costly, Group A complements it with two cheaper droids from Robot B to create a robot team. On the other hand, Robot B resulted from bespoke development for Group C – and they liked the result enough to buy two more at a later stage. Furthermore, Group C complements the cheap droids by two basic bots: a Basic Leader Bot and a Basic Bot. Both Groups A and C spend €95 of their budgets.

2.3 Strategizing to win the LU Rumble

The LU Rumble is a customized instance of a RoboCode team rumble. Each project group will compose a team of up to five robots that will compete together against other robot teams. Each robot team must have one leader robot, which can be complemented by normal robots or droids.

As in any rumble, the goal is to reduce the energy of enemy robots to 0. Team robots will compete on a battlefield until only one team is left. A standard robot starts with 100 energy and is equipped with a radar and a gun turret. Each time the gun is fired, the robot loses energy according to the payload. If the bullet does not hit any target, the energy is lost. If the bullet hits another robot, energy is transferred from the hit robot to the firing robot.

Apart from the standard robots, there are two special types of robots. The leader robot has an additional 100 energy compared the normal robot. Droids have an additional 20 energy, but are not equipped with radars.

At the end of Sprint 1, your project group will have established a relationship with a robot supplier by pre-ordering either a leader robot, a normal robot, or a droid. The supplier will provide you with a final robot at the end of Sprint 3, but your group will have a chance to influence the evolution of the robot during Sprints 2 and 3. The domain expert in your group is responsible for developing a strategy around the robot you have pre-ordered – to guide your feedback, you will receive two prototypes for evaluation: an Alpha release after Sprint 1, and a Beta release after Sprint 2.

At the end of Sprint 3, you will complement the robot you have ordered with other robots from the Robot Market. The goal of your strategizing is to identify and purchase additional robots that can successfully cooperate with the robot from your supplier. It is important to keep an eye on the Robot Market during Sprints 2 and 3, both to find the best robots to purchase, and to understand what type of features you should provide through your own robot under development. Note that three robots are always available at fixed prices: 1) BasicLeaderBot €20, 2) BasicBot €15, and 3) BasicDroid €10.

Each robot team must adhere to two strict constraints:

1. Each team must consist of between one and five robots, including one leader robot.
2. Each team must not exceed a total cost of more than €100 worth of robots.

3 Project phases

3.1 Project inception

The first week of the project is referred to as the project inception. The project inception phase is dominated by project group development, i.e., forming groups, assigning roles, and aligning goals and expectations.

According to a well-known model of group development, proposed by Tuckman in 1965 [REF], four distinct phases are all necessary and inevitable for a group of people to face up to challenges, collaborate successfully, and to deliver results. Being aware of the four phases might accelerate the less productive phases and mitigate group conflicts:

1. **Forming** The group meets and learns about the project and the required activities. In the beginning, individuals tend to behave quite independently – all trying to comprehend what is about to happen. Individuals might be motivated and on their best behavior, but typically the understanding and expectations vary considerably.

   In the project, groups will be established already during the break in the first lecture, i.e., before lunch the very first Monday. The individual group members are expected to read the project instructions carefully as early as possible.

2. **Storming** After a while, some disagreements and personality clashes are likely to emerge within the group. People are people, and people are different. Tolerance of each group member should be emphasized, otherwise the group will fail or rely on individual "hero efforts". This phase can become destructive to the group, and some teams will never develop past this stage. On the other hand, settling disagreements within the team can make members stronger, more versatile, and able to work more effectively together. Note that some groups never enter this stage at all, i.e., transitions directly from forming to norming are possible.

   In the project, the supervisors and the regulatory body will be particularly responsive to questions and clarifications during the first week. Also, the group communication will be kick-started by two physical meetings soon after the formation, i.e., exercise sessions on Wednesday and Thursday the first week.

3. **Norming** Resolved disagreements and personality clashes lead to a spirit of co-operation – this typically happens when the group becomes aware of the external competition and share a common goal. In this stage, all group members take the responsibility and agree to work for the success of the common goal. The group members accept others as they are and make an effort to move on.

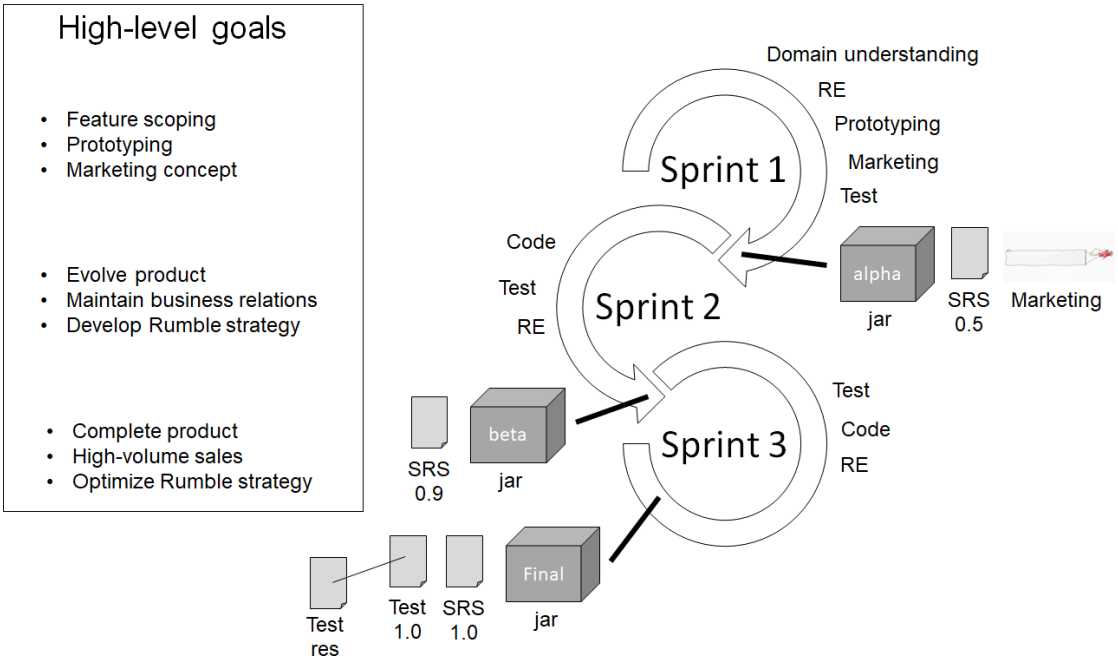

Figure 3.1: The ETSA02 process model, organized in three sprints.

In the project, the first week concludes with assigning roles within the group, see Section 5.1. All members of the group will also sign a collaboration contract, specifying the roles, and the intentions to collaborate successfully. The contract will be submitted to the regulatory body through "grå skåpet".

4. **Performing** When the group norms and roles have been established, focus is shifted to achieving common goals. However, groups might revert to earlier circumstances in certain circumstances, e.g., unexpected events, emerging personal conflicts, or changing leadership. In the project, the goal is of course to remain in the performing stage until the LU Rumble!

3.2 Process model

The process model describing the product development encompasses three sprints, each with a separate set of deliverables.

3.2.1 Sprint 1

The first sprint overlaps with the project inception as the groups will start working on robot development already at the first exercise sessions in the first week. The high-level goals of Sprint 1 are to successfully do: 1) feature scoping, 2) robot prototyping, and 3) developing a marketing concept. This means that the project group will be focused on

engineering and monetizing during Sprint 1. Some strategizing will also occur, as the domain expert will purchase a robot from a supplier after the Robot Fair.

The first steps of the requirements engineering involves analyzing what type of robot would be in high demand by the other project groups. This type of market analysis is no easy task to conduct, and surely the requirements engineer will discuss intensively with the domain expert and the sales engineer. Engineering a software product means making a series of successful technology and business decisions, and already during Sprint 1 the project group needs to set the direction for the engineering department. High-level requirements are referred to as features, and the task of deciding which features to implement in the product is called **feature scoping**. At the end of Sprint 1, SRS v0.5 should specify the features the group intends to implement in their robot. The features should be promising enough to generate good offers by other project groups, but do not promise too much (aka. "over scoping") – if you do not deliver a robot that fulfils the specification in the final release, your customer will most likely file business claims to the regulatory body! On the other hand, the SRS v0.5 is not carved in stone. There will be negotiations during Sprints 2 and 3, and if both the customer and the regulatory body accept your change requests, you can both modify and remove features later – and adding new features is of course always possible.

A central part of the engineering in Sprint 1 is the **robot prototyping**, an activity that serves two main purposes. First, it helps your project group experiment with different candidate features for the robot under development. There is no better way to learn the Robocode domain than by practically implementing different approaches and evaluating them. Second, the prototyping will lead to an Alpha version of the Robot. The customer will obtain the Alpha release, which will enable much richer feedback than an SRS alone. To ensure that testing is part of the development process from the start, Sprint 1 will introduce unit testing in the second week.

The **marketing concept** is the strategy that an organization implements to satisfy customers needs, increase sales, maximize profit, and beat the competition. The customers' focus and values are the routes to achieve sales and maximize profits. In the project, the marketing concept is a customer-centered "sense and responds" philosophy. Your project group's job is not to find the right customers for your robot, but rather to find the right robot for your customers – first for the long-lasting supplier-customer relationship, and later through the Robot Market. At the end of Sprint 1, the course will organize a Robot Fair. Each group will get a 2 minute slot to pitch their product through a video. All project groups will also have a webpage on a shared Google Drive folder.

Sprint 1 shall result in the following deliverables to the following stakeholders. The responsible role is shown in parentheses.

- SRS v0.5 to the customer and the regulatory body (Requirements engineer)
- Robot Alpha release to the customer (Development lead)
- Test specification v0.5 to the regulatory body(Test lead)
- 3 min sales pitch and website for Robot Market (Sales engineer)

- A purchase array to the regulatory body (Domain expert)

To support Sprint 1, the following sessions are scheduled:

- Lecture 1: Robocode, Requirements engineering
- Lab 1: Robot development in Eclipse
- Exercise 1: Requirements engineering
- Lecture 2: Construction, Testing
- Lab 2: Automated robot testing
- Exercise 2:

3.2.2 Sprint 2

In Sprint 2, the project group should be up to speed both regarding engineering and monetizing. Now the monetizing will decrease and the strategizing will increase in its place. The high-level goals of Sprint 2 is to: 1) evolve the robot, 2) maintain the customer relationship, and 3) develop a strategy for the LU Rumble.

In Sprint 2, you will have an established customer who has received your Robot Alpha release. As that project group will be keen to win the LU Rumble, you will likely get feedback that they believe will guide your development to a successful robot. During the remainder of the project, maintaining a healthy **customer relationship** is critical. Your sales engineer is the main point of contact with the customer.

Based on your customer's feedback, your own vision – and possibly some negotiation in between – you will **evolve the robot** during Sprint 2. The features will be broken down into detailed requirements, and the test specification will be elaborated to also cover system testing. At the end of Sprint 2, you will deliver Robot Beta release to your customer, constituting an MVP, i.e., a minimal product that still brings some value. To be in a good position for the next sprint, the project group should try to let the SRS, source code, test specification, and marketing concept co-evolve – otherwise they might diverge considerably, increasing the risk of a dissatisfied customer, wasted effort within your group, and reduced sales on the Robot Market. Make sure that any changes you make to the SRS are traceable, i.e., make sure there is a "paper trail", in case the regulatory body needs to settle business claims from your customer at a later stage. Significant changes to the SRS must be done in agreement with the customer and documented. If the supplier organization does not store a chain of evidence, the regulatory body is likely to side with the customer in legal contentions.

Finally, you will develop a first draft of your **Rumble strategy**. The cornerstone of your robot team will be the robot you pre-ordered. Your domain expert will be responsible for communication with your supplier's sales engineer. Just like your project group should be responsive to your customer, your supplier should respond to your requests.

Sprint 2 shall result in the following deliverables to the following stakeholders. The responsible role is shown in parentheses.

- SRS v0.9 to the customer and the regulatory body (Requirements engineer)
- Robot Beta release to the customer (Development lead)
- Test specification v0.9 to the regulatory body (Test lead)

To support Sprint 2, the following sessions are scheduled:

- Lecture 3:
-
-

3.2.3 Sprint 3

In Sprint 3, the project group shall conclude and tie up all activities. The engineering should lead to a successful robot, the monetizing should ensure that it is well-received on the market, and the strategizing shall finalize the robot team shall be competitive in the upcoming LU Rumble. The high-level goals of Sprint 3 are to: 1) complete the product, 2) generate high-volume sales, and to 3) optimize the Rumble strategy.

The engineering department will be busy implementing the remaining requirements and putting the finishing touches to the robot. This means **completing the product** and preparing the Robot Final release. When the Final Robot is released, it is important that the SRS, the test specification, and the marketing concept are all aligned. Your customer will perform acceptance testing of your robot – rest assured that they will scrutinize your deliverables to identify any discrepancies. The customer will be greatly motivated to assure that the Robot Final release complies with the SRS – their entire robot team for the LU Rumble might rely on it! If the customer is not satisfied, the regulatory body is likely to receive a business claim based on their acceptance testing. However, your project group will provide the regulatory body with evidence that everything in the SRS has been verified by providing a complete test report based on Test specification v1.0.

The sales department will keep active communications with the customer, to make sure the final engineering efforts adhere to their expectations. Close communication with the customer will reduce the risks of incoming business claims. However, the sales engineer will also focus considerable effort on the robot offering on Robot Market – trying to make the robot as attractive as possible on the open market, i.e., the sales engineer will maintain the Robot webpage with the intention to **generate high-volume sales**.

The purchasing department will keep a close eye on the engineering efforts of the supplier. After all, this is the supplier's last chance to deliver the robot that has been specified in the SRS. At the same time, your domain expert will analyze the Robot Market to make a final decision of your robot team composition, i.e., **optimizing the Rumble strategy**.

Sprint 3 shall result in the following deliverables to the following stakeholders. The responsible role is shown in parentheses.

- SRS v1.0 to the customer and the regulatory body (Requirements engineer)
- Robot Final release to the customer and the regulatory body (Development lead)
- Test specification v1.0 to the regulatory body and the customer (Test lead)
- Test report to the regulatory body and the customer (Test lead)
- A robot team composition to the regulatory body (Domain expert)

To support Sprint 3, the following sessions are scheduled:

- Lecture 4:
-
-

3.3 Post release

The phase following the process model is referred to as post mortem. In this phase, your project group will perform acceptance testing of your supplier's Robot Final release. If the acceptance testing identifies flaws, you will file business claims to the regulatory body – possibly deducting from the total payment. Furthermore, after settling the business claims, the domain expert will submit the final purchase orders to the regulatory body, i.e., the robot team composition for the LU Rumble will be finalized.

The post mortem shall result in the following deliverables to the following stakeholders. The responsible role is shown in parentheses.

- Acceptance test report (Domain expert)
- (Potentially) Business claims (Project manager)
- Purchase orders to the regulatory body (Domain expert)
- A robot team composition to the regulatory body (Domain expert)

3.4 LU Rumble

The final phase consists of one single event: the LU Rumble. The LU Rumble will take place during the final lecture of the course on May 21, 2018.

Prior to the event, the regulatory body will run a qualification session, inspired by the format used in Formula one racing. The purpose of the qualification session is twofold. First, it assures that all robot teams are functional on the Robocode arena. Second, it will let a number of robot teams get "pole positions", i.e., seeds for the LU Rumble main

tournament. The qualification session entails a large number of simulations between all teams. Note that while the qualification session will indicate some robot teams as favourites, they might be good on average, but have weaknesses against certain team strategies. No matter the results of the qualification session, all robot teams can still win the LU Rumble.

The main tournament, i.e., the playoff, will follow a cup pairing according to single elimination knockout rules. The four robot teams performing the best during the qualification session will be seeded into the tournament bracket. All remaining robot teams will enter the bracket at random. Each match in the LU Rumble will be played in best-of-five format, and displayed live at reasonable speed for the entertainment of the spectators.

Immediately after the LU Rumble, three awards will be given:

- The Strategizer Award goes to the winner of the LU Rumble.
- The Monetizer Award goes to the most profitable project group.
- The Engineering Award is a people's choice with voting during the LU Rumble (or a jury in the regulatory body?).

4 Inter-group communication

To be written...

4.1 The regulatory body

To be written...

Formal document templates:

- Contract of sale between supplier and customer
- Purchase order for Robot Market
- Business claims

4.2 Maintaining supplier-customer relations

To be written...

What to do and what not to...

Guidelines for suppliers:

- The customer is always right
- Do not sacrifice the chance to succeed on the open market
- Be open to new ideas from the customer.
- Embrace change – it is inevitable anyway.
- Negotiate nicely.

Guidelines for customers:

- Accept what you've pre-booked.
- Do not provide feedback that deviates from the original vision of the robot.
- Be open to alternative solutions proposed by the supplier.
- Negotiate nicely.

4.3 Group conflicts

To be written...

5 Detailed instructions

This section presents some aspects of the course in detail: 1) the roles you should assign within the project group, 2) expectations on the deliverables, and 3) the course infrastructure.

5.1 Role assignment

During the course, all students are expected to contribute to all activities. The purpose of the roles is not to isolate work tasks that could be individually completed. Software engineering is a collaborative effort, and completing work tasks in isolation is not part of a scalable methodology. Instead, we stress the value of "collective ownership" during the project, not only for the source code as described in Section 2.1.2, but for all artifacts produced.

The purpose of assigning roles in the project is to comply with the ETSA02 course description regarding "judgment and approach", i.e., understanding that software engineering is a complex endeavor involving several different roles, and that working as a software engineer might result in a variety of roles in an organization. Second, all roles with be responsible for some deliverables. The regulatory body will use the group member with the corresponding role as the primary point of contact in case any deliverable is missing or critically incomplete.

As the project progresses through phases and sprints, different roles with have different expectations. The project group will track the variation through weekly time reporting, submitted by the project manager to the regulatory body. The reported figures should be complemented by a qualitative analysis of the causes, and the project manager is responsible for balancing the workload to make sure the contributions are comparable, see 5.2 for further information. Below are the different roles, ordered alphabetically:

Development lead Design, implementation, source code quality. Assign a group member comfortable with the art and science of programming. – jar_v0.5, jar_v0.9, jar_v1.0

Domain expert Mastering Robocode, LU Rumble strategy, supplier communication, negotiations, acceptance testing. The domain expert needs to be willing to learn about the game aspects of Robocode, thus pick your gamer. – Signed contract (as customer), acceptance test report, team jar-file

Project manager Coordination, time reporting, communication with the regulatory body. Identify a group member with leadership skills willing to shoulder over-

all responsibilities. – Weekly time reports with reflections, business claims (based on failed acceptance test)

Requirements engineer End-user perspective, feature scoping, detailed requirements. A requirements engineer bridges engineering and business, identify a generalist. – SRS v0.5, SRS v0.9, SRS v1.0

Sales engineer Marketing, customer communication, negotiations. Identify your natural salesman, a person with excellent communication skills. – Marketing concept, signed contract (as supplier)

Test manager Test strategy, unit testing, system testing. Suitable for an individual with an eye for details. – Unit tests, Test spec v1.0 (incl. test code), test results

5.2 Time reporting and reflections

Report time per role on a weekly basis. Reflect on the figures after each sprint. Plan changes if needed to better balance the work load.

5.3 Deliverables

To be written...

The following deliverables will be produced during the project, the list below presents the expectations, and the roles responsible for the delivery.

Marketing concepts Maximum two minute video showcasing the robot you offer to other project groups.

SRS

Source code Only the regulatory body should receive the source code of the robot.

Releases Package two releases of the robot. First, a jar-file without source code for your customer. Second, a jar-file with embedded source code for the regulatory body.

Test code

Test spec

Test res

Acceptance test

LU Rumble team

5.4 Course infrastructure

To be written...

The following infrastructure is central during the project:

SRS template

Slack workspace

Contract templates

Collaboration contract

Contract of sales

Time reporting

Robot team composition sheet

6 Week by week instructions

To be written...

Week 12
Lecture 1: Group formation, Requirements engineering
Lab 1: Introduction to Robocode development
Exercise 1: Robot feature scoping
Week 13
Lecture 2: Construction, Testing I, Business
Lab 2: Automated unit testing
Exercise 2: Robot marketing
Week 14
Exam week
Week 15
Exam week
Deadline Friday April 13: Alpha release and marketing concept
Week 16
Lecture 3: Robot Fair, Testing II
Lab 3: Automated system testing
Exercise 3: Detailed requirements
Deadline Monday April 16: Purchase array
Week 17
Lecture 4: Design
Exercise 4: Test specification
Deadline Monday April 27: Beta release
Week 18
No scheduled sessions
Week 19
Lecture 5: Processes
Deadline Friday May 11: Final release
Week 20
Lecture 6: Backup, Old exam
Deadline Wednesday May 16: Acceptance testing and business claims
Deadline Friday May 18: Purchase orders and robot team composition
Week 21
Lecture 7: LU Rumble
Week 22

Exam May 22-23