Robotic Sensorimotor System Testing Platform (RSSTP)

Final Report

Version 0.1

🦾

|  |  |
| --- | --- |
| Name | Student number |
| Filip Georgiev | 2208085 |
| Khalil Chakal |  |
| Mourad Boulfrad | 2207592 |
| Roshan Fernando |  |
| Shabbir Hasan | 2208301 |

Executive abstract

Table of Contents

[Executive abstract 2](#_Toc117676349)

[Acronyms and Definitions 4](#_Toc117676350)

[1 Introduction 4](#_Toc117676351)

[2 Customer 4](#_Toc117676352)

[3 Project description 5](#_Toc117676353)

[3.1 Project Background 5](#_Toc117676354)

[3.2 Project Goal 5](#_Toc117676355)

[3.3 Project requirements 5](#_Toc117676356)

[4 Realised requirements 6](#_Toc117676357)

[4.1 Software structure 6](#_Toc117676358)

[4.2 Internal design 7](#_Toc117676359)

[4.3 External design 7](#_Toc117676360)

[4.4 Chosen programming language 7](#_Toc117676361)

[5 Used software development method 7](#_Toc117676362)

[6 Testing 8](#_Toc117676363)

[6.1 Process 8](#_Toc117676364)

[6.2 Unit tests 9](#_Toc117676365)

[6.3 Acceptance testing 9](#_Toc117676366)

[7 Project timeline 10](#_Toc117676367)

[7.1 Milestones 10](#_Toc117676368)

[7.2 Phases 10](#_Toc117676369)

[7.2.1 Phase I 10](#_Toc117676370)

[7.2.2 Phase II 10](#_Toc117676371)

[7.2.3 Phase III 10](#_Toc117676372)

[7.2.4 Phase III 10](#_Toc117676373)

[7.2.5 Phase IV 11](#_Toc117676374)

[7.2.6 Phase V 11](#_Toc117676375)

[7.2.7 Phase VI 11](#_Toc117676376)

[8 Feedback to the course organizers 12](#_Toc117676377)

[9 References 12](#_Toc117676378)

[10 Appendix 13](#_Toc117676379)

[10.1 Original project pitch 13](#_Toc117676380)

[10.2 Software structure 14](#_Toc117676381)

Acronyms and Definitions

|  |  |
| --- | --- |
| Agent | The test subject (e.g., a robot arm) |
| Gym | Alias for testing platform |
| MVP | Minimum Viable Product |
| PE | Perception Engineering |
| RSSTP | Robotic Sensorimotor Software Testing Platform |
| TDD | Test-Driven Development |
| UBICOMP | Center for Ubiquitous Computing |
| UML | Unified Modeling Language |
| VR | Virtual Reality |
|  |  |

# Introduction

How does a person learn to traverse their surroundings? How does an animal? When all senses of a living being (e.g., sight, hearing, smell, etc.) are combined, they create a mesh of information, which can be used by said being to obtain a feeling about their environment and consequentially navigate through it. However, what happens when an agent is stripped of most of their senses and is only left with minimal sensory feedback? What is the best algorithm, which can be used to learn one’s environment when there is little to no information to utilize?

In the Perception Engineering (PE) research group of The Center for Ubiquitous Computing (UBICOMP) research unit in the University of Oulu, possible answers for the last question stated above will be explored. It is planned for research to be done, so that optimal algorithms, which e.g., teach a robotic hand to move in an environment with different obstacles, can be found. Before this can be done, however, a testing platform, in which these algorithms can be evaluated and compared needs to be created.

In this document, the design and creation of such a platform, named Robotic Sensorimotor System Testing Platform (RSSTP), are outlined. The terms testing platform and gym will be used interchangeably. In the following chapters more information about the customer, the project itself and the realised requirements will be given. This will be followed by an explanation of the software development and testing methods used during the project. Afterwards the project timeline will be discussed, and the document will be concluded with the feedback given to the Software Project course organizers. Any additional (and necessary information) regarding this project will be found in the [*Appendix*](#_Appendix) chapter.

# Customer

The customer for this project is the UBICOMP research unit and more specifically the PE group, which focuses on research related to Virtual Reality (VR) and robotics-related problems, similar to the one described in this document, as for example sensing, sensor fusion, planning, learning and control [1].

The people who will act as representatives for the customer and in so as supervisors and contact persons for this project are Dr. Vadim Weinstein ([vadim.weinstein@oulu.fi](mailto:vadim.weinstein@oulu.fi)) and Dr. Kale Timperi ([kalle.timperi@oulu.fi](mailto:kalle.timperi@oulu.fi)).

# Project description

## Project Background

In the scientific world the ideas behind cognition and thinking are not well-defined. However, there are two main schools of thought – one is called cognitivism, while the other enactivism. The former is the belief that action comes as a result of thought. This means that one must think about an action first, before executing it. The latter describes the idea that cognition is a result of action. This means that a person knows innately how to do something and thought only happens afterwards, as a consequence of said action. More precisely, enactivism is the belief that living beings learn to think through the act of doing.

The supervisors of this project have decided to investigate enactivism from the lens of computer science and mathematics. They wish to create a mathematical representation of cognition by doing research in the field of robotics. To achieve this, they are trying to determine the best possible way to teach a robot to learn about its surroundings with minimal sensory feedback (e.g., without the use of visual sensors). The idea is for the robot to do movements, through which it can learn about its environment.

The nature of this task is theoretical, so there are no hard constraints set on the type of robot or type of environment, in which learning will happen. Furthermore, the task is seen as an unsupervised problem and hence algorithms which will try to create a model in real-time based on the minimal data received will be created. However, before these algorithms can be investigated and created, a testing platform in which learning can be simulated and its results can be observed and evaluated needs to be designed and implemented. The RSSTP project was established based on these requirements and its original description can be seen in [Appendix 1](#_Original_project_pitch).

## Project Goal

Originally, the RSSTP project was planned to last one year. However, as the course in which it is currently being implemented (Software project) is only one semester, the goal has been set to be time-appropriate in the form of a minimum viable product (MVP). This MVP must include the design and creation of the testing platform itself and the functionality of simulating how a given algorithm affects the learning capabilities of one robotic arm, which has two joints and minimal sensory feedback. Additional project requirements are set in the section below.

## Project requirements

* The MVP platform must inform the user of the results, but this does not need to be done with a visual simulation. The evaluation of the results will be for the user to interpret, although in the future this might become part of the testing platform.
* The chosen technological stack must have the ability to interface with Python.
* The environment in which learning will happen must contain obstacles, which will hinder the agent’s movement.
* The agent should be allowed to receive minimal information as for example whether it reached its destination or not, but it should not store exact data about its surroundings (e.g., obstacle locations).
* The supplied software structure seen in [Appendix 2](#_Software_structure) should be used to create the platform.

# Realised requirements

## Chosen programming language

The first step in this project was choosing the technology stack. As the gym was supposed to be interfaceable with the Python language and there were no other constraints (e.g., computation speed) it was concluded that Python would be used to create the testing platform. This would also ensure quick development time.

## Design

The document, which can be found in [Appendix 2](#_Software_structure), was supplied by the supervisors to the students. In it, pseudocode for the software structure of the testing platform for the robot arm can be found. It is split into two parts – external and internal. The external part covers the movements of the robot arm in the simulated environment, while the internal part represents a “brain in a jar”, which does not have any knowledge of the outside world. The description of variables and functions mentioned below can also be found there.

### External design

A simplified Unified Modeling Language (UML) representation of the External class can be seen in Figure 1.

Table

Description automatically generated

Figure External class

Then, in Figure 2 the state machine of the external part is defined. The external part takes care of two functions – it moves the arm, and it sends out information to different listeners. After starting the gym and the arm is in its home position (**Home Position state**), it can move out of it with the *update* function. The *update* function will then determine in the **Update state** whether the actual arm should move (**Moving state**) or not (**Idle state**). In the cases where an obstacle has been hit no movement is seen. It is important to note that the *update* function is a “twin” of the *transition* function of the internal class. This means that whenever this internal function is called, the external *update* is also called. In the cases where the update is stuck in an infinite loop, a reset function which can be called by the gym user can be called. Information-wise, the *get\_sensory\_data* function will return either a 0 or a 1, depending on whether it has reached a desired position set by the user. This function is global as it is a wrapper around the *get\_S* function which keeps track of the current position and sensory distribution of the arm.

Diagram

Description automatically generated

Figure External state diagram

### Internal design

The UML representation of the Internal class can be seen in Figure 3.

Table

Description automatically generated

Figure Internal class

As mentioned earlier, the internal class acts as a “brain in a jar”. This means that it does not have any knowledge of its surroundings and the only way to represent its environment is in its own transition matrix. An example can be seen in Figure 4. This matrix keeps track of different states in which the “brain” can be. These states are ambiguous, but one example is a specific movement that can be done. To achieve this, the supplied algorithm makes use of the internal class as its own “brain”. It keeps track of the transition matrix and manipulates it with the different functions in the internal class, based on the information the algorithm gets from the external class. This way it can create new states or update old ones, so that it can find the optimal transition matrix, which will move the arm from its home position to the desired position. The description of the different functions in this class can be seen in internal part of [Appendix 2](#_Software_structure). Finally, after the matrix has been selected, the *transition* function can be called, so that the current state in said matrix can be changed and effectively an attempt to change the real position of the arm can be made in the External part.

A picture containing text, whiteboard

Description automatically generated

Figure Example transition matrix

### Overall design

Combining the external and internal parts, the following flow, seen in Figure 5, is established:

1. Algorithm tries a transition from the internal transition matrix by calling *transition* in the internal and *update* in the external part.
2. The external part attempts to move the arm
3. The algorithm gets information about whether the desired position has been reached or not
4. The algorithm compares its expected functionality to what actually transpired
5. Algorithm updates internal part, if needed, by using the internal logic.
6. Goes back to 1.

Diagram

Description automatically generated

Figure Overall testing platform flow

## Implementation

# Used software development method

The SCRUM methodology, which is part of the Agile philosophy was used in this project. Even though it is required by the Software Project course, another reason for using Scrum was that it would keep the supervisors acquainted with the work done by the students. This was essential for this project due to its theoretical nature. By utilizing SCRUM any misunderstandings of the requirements of the project could be dealt with in a timely manner.

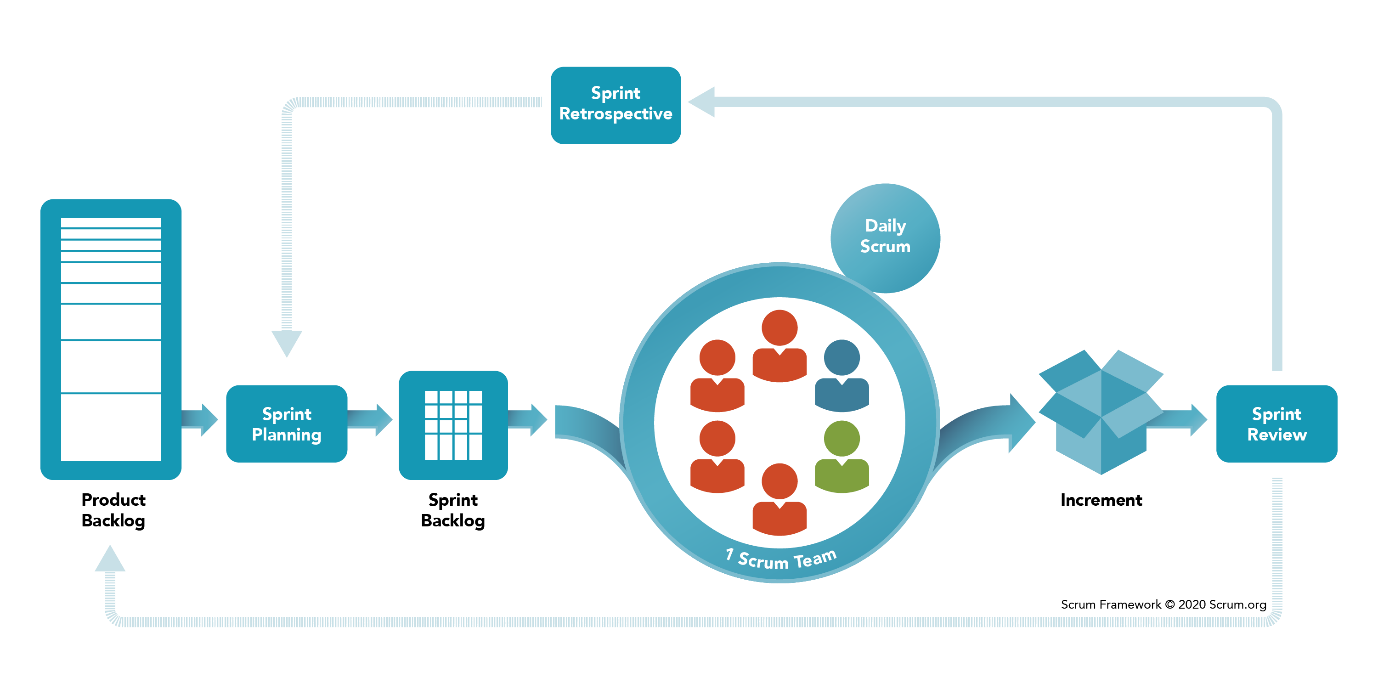


Figure SCRUM Framework. Source: [SCRUM website](https://www.scrum.org/resources/what-is-scrum)

To ensure that all parties are kept up to date weekly hourly meetings on Thursday at 09:00 were scheduled. Furthermore, biweekly sprints were created, which would correspond to the phases outlined in the [Project Timeline](#_Project_timeline) section. The sprint demo, retrospective and planning would be done at the end of each sprint in the same Thursday meeting, which would in this case be scheduled for two hours. An additional meeting would be held by the students only each Tuesday at 18:00, in which they could update each other on their progress and help with issues that have arisen.

The online software tool “Trello” was used to create the SCRUM board and keep track of tasks. There would be six general cards, in which tasks could be put.

1. Backlog card, where all the future tasks would be stored.
2. To Do card, which contains all the tasks that need to be done in the current sprint.
3. Blocking card, which contains tasks that can not be done at the current time, as there is a different task that needs to be done first or some other blocker.
4. Doing card, which contains the tasks currently being done by the students.
5. Review card, which contains the finished tasks that need to be reviewed by the students or supervisors, before they can be considered done.
6. Done card, which contains all the tasks which have been done throughout the sprint. Multiple done cards were used throughout the different sprints.

**TODO: Add picture at the end of the project**

# Testing

## Process

The Test-Driven Development (TDD) process was used during the project. As testing was required by the course and the time allocated to complete the project was limited it seemed like a logical conclusion to invert the normal development cycle and start with tests first. This would mean that the Unit tests for the software would be created first, followed with the software which would have to adhere to the tests. Not only would this ensure that unit tests are created in time, but it would also provide additional benefits. For example, it would force the students to understand their tasks before they have implemented them, as they would need to know what a certain function (or unit) would need to do to be considered successful before they could create a test.

## Unit tests

As python was chosen for the development language, the “pytest” Python testing tool was used to create the testing suite and following tests.

**TODO: …**

## Acceptance testing

Acceptance tests were also done in the form of full platform algorithm runs. This would mean that after the software was developed, algorithms provided by the supervisors would be run through the testing suite and hence its functionality would be tested.

**TODO: overview at the end of the project**

# Project timeline

The timeline of the RSSTP project starts on the 21st of September. On this date, the first meeting with the clients was had. The timeline ends on the 13th of December, when the Final Presentation was had. The project was conducted in 2022.

## Milestones

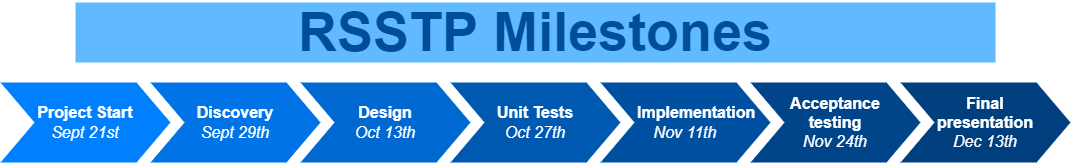


Figure RSSTP Milestones

The Milestones for this project can be seen in Figure 3. Below, an explanation for each one will be given with the respective phases in which it is reached.

## Phases

Each phase of the project contains the work done to reach the next milestone in the timeline. The duration, description and the milestone reached upon completion of the phase are listed below for each phase.

### Phase I

**Duration:** 2022/09/07 – 2022/09/21

**Milestone:** *Project start*

**Description:** Managing to find a project and getting approval from the teacher/TAs was the first task for the project. The milestone was completed after the first client meeting.

### Phase II

**Duration:** 2022/09/21 – 2022/09/29

**Milestone:** *Discovery*

**Description:** The second milestone was completed once the project goal was better understood. This was done by setting up multiple meetings with the clients and discussing what the project entailed.

### Phase III

**Duration:** 2022/09/29 – 2022/10/13

**Milestone:** *Design*

**Description:** Completing the “Design” milestone was achieved when the team designed the software platform with the help of block, class and state diagrams.

### Phase III

**Duration:** 2022/10/13 – 2022/10/27

**Milestone:** *Unit Tests*

**Description:** As the project would be implemented following the TDD method, the unit tests for each software component had to be created. Once this was done, the “Unit Tests” milestone was reached.

### Phase IV

**Duration:** 2022/10/27 – 2022/11/11

**Milestone:** *Implementation*

**Description:** Based on the design and unit tests, the software components of the RSTTP were implemented. The “Implementation” milestone was completed once this was done.

### Phase V

**Duration:** 2022/11/11 – 2022/11/24

**Milestone:** *Acceptance testing*

**Description:** To ensure that the testing platform works as expected, the entire software suite was tested by using it as intended and running different example algorithm through its pipeline. Completion of this milestone meant that the software worked as intended in the requirements.

### Phase VI

**Duration:** 2022/11/24 – 2022/12/13

**Milestone:** *Final presentation*

**Description:** The final step of the project was to create a final presentation on the project. During this phase any leftover tasks/bugs were also resolved and the milestone was finished once the presentation was given.

# Feedback to the course organizers

# References

|  |  |
| --- | --- |
| [1] | "Perception Engineering (PE)," [Online]. Available: https://ubicomp.oulu.fi/research/pe/. [Accessed 24 October 2022]. |

# Appendix

## Original project pitch

Text

Description automatically generated

## Software structure

Text, letter

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated