Collaboration analysis in multi-player based simulations

Bruno Filipe da Cruz Carreira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Rui Filipe Fernandes Prada
Prof. Manuel Fernando Cabido Peres Lopes

Examination Committee

Chairperson: Prof. Nuno João Neves Mamede
Supervisor: Prof. Rui Filipe Fernandes Prada
Member of the Committee: Prof. Pedro Alexandre Simões dos Santos

January 2021
Acknowledgments

Teamwork makes the dream work.

This famous phrase, first attributed to John Maxwell, an American clergyman, is a perfect description for how this journey went.

When I first started this dissertation, I had no idea how this would end. Will I be able to finish it within the deadline? What if I get a cool job offer before finishing this? What if I can’t finish it at all? These questions filled my mind for months until I realized one small thing. I wasn’t alone.

Although no one could do “my” research, code “my” simulations or write “my” document, there were several people that helped me during the past few months. To those people, this is my thank you.

Thank you to my family, my mom Graça, my dad Celso and to someone I will never say out loud, my sister Beatriz. They’re my roots, the ones that no matter whether I succeeded or miserably failed, always supported me.

Thank you to my girlfriend Patrícia, someone who accompanied my through thick and thin, for all the support, the friendly shoulder and the much needed “though love” which without I would never have finished this dissertation.

Thank you to my dearest friends, Alberto Ramos, André Carvalho, Rodrigo Bernardo and Luís Sousa, who accompanied me during this whole process and even while having their own dissertations and projects to work on, always had the time to help me, advise me and assure me that I was never alone, that the obstacles that I was facing where the same ones they were facing as well.

Thank you to my dissertation supervisors, Prof. Rui, Prof. Manuel and also iv4XR project colleague Pedro Fernandes, who for countless hours provided me with in-depth knowledge and had tremendous patience with me, my work and my (very often silly) questions.

To each and every one of you, my most honest and sincerest Thank you.
Abstract

This work aims at helping developers of interactive software to test collaboration inducing scenarios. When creating a training simulation for team building, developers must make sure that their scenarios promote collaboration but also, don’t force it, meaning a scenario must allow users to behave freely, otherwise did they really collaborated or were just forced to? This creates a difficulty, how can developers test their scenarios on their capability of allowing different behaviours? Our approach is based on using two different automated agents behavioural traces, one specifying the scenario’s Design Goal, collaboration, and the other an example of a non-Design Goal, acting individually. After training said agents and by comparing the agents optimal behaviour when solving each scenario to the two policies, we can determine if the scenarios allow to differentiate between the Design Goal and the non-Design Goal. With this approach we are also able to order the scenarios from easiest to differentiate to hardest. Our approach was tested in two different environments, in a custom built simulator and in the iv4XR game Lab Recruits.

Keywords

Multi-player based simulation; scenario testing; collaboration detection; automated agents behavioural traces, automated testing.
Resumo

Este trabalho consiste numa abordagem para ajudar criadores de software interativo a testar cenários indutivos de colaboração. Quando se desenvolve uma simulação de treino para promover trabalho de equipa, os criadores devem ter em atenção não só se os seus cenários promovem colaboração mas também se não a forçam, ou seja, um cenário deve permitir que os utilizadores/jogadores achem livremente, caso contrário como é que se pode ter a certeza se eles realmente colaboraram ou se apenas foram forçados a agir dessa forma? Ista questão cria um problema, como é que os criadores podem testar os seus cenários relativo à sua capacidade de permitirem e distinguirem diferentes comportamentos? A nossa abordagem é baseada em utilizar comportamentos de dois tipos diferentes de agentes automáticos, um a especificar o Comportamento Ideal, agir colaborativamente, e o outro um exemplo de um Comportamento Não Ideal, agir individualmente. Após o treino dos agentes automáticos e comparando o comportamento nas suas resoluções ótimas em cada cenário com as duas políticas poderemos determinar se os cenários permitiram distinguir entre o Comportamento Ideal e o Comportamento Não Ideal. Com esta abordagem podemos também ordenar os cenários pela sua facilidade nessa distinção. A nossa abordagem é testada em dois ambientes diferentes, um simulador personalizado construído para este trabalho e também o jogo Lab Recruits feito no âmbito do projeto iv4XR.

Palavras Chave

Simulação multi-jogadores; análise de cenários; detecção de colaboração; análise de comportamento de agentes automáticos, testes automatizados.
List of Figures

2.1 Reinforcement Learning Architecture .................................................. 5
4.1 How to test a scenario ................................................................. 14
4.2 Squary - Shappy: Map matrix example ............................................. 15
4.3 Squary - Shappy: Visual map example .............................................. 15
4.4 Centralized architecture ................................................................. 16
4.5 Individual architecture ................................................................. 17
4.6 MDP with Q-Learning ............................................................... 17
4.7 iv4XR Framework to Unity architecture ......................................... 20
4.8 Lab Recruits: Map matrix example .................................................. 21
4.9 Lab Recruits: Visual map example .................................................. 21
4.10 MDP + Dyna-Q ................................................................. 24
5.1 Squary - Shappy: Scenario 1 “Advantage red” ..................................... 29
5.2 Squary - Shappy: Scenario 2 “Perfect division” ................................... 30
5.3 Squary - Shappy: Scenario 3 “Center focus” ....................................... 30
5.4 Results: Squary-Shappy Centralized agents optimal behaviour ............. 31
5.5 Results: Squary-Shappy Individual agents optimal behaviour ............... 32
5.6 Results: SS likelihood difference ..................................................... 33
5.7 Lab Recruits: Scenario 1 “One man’s choice” ..................................... 33
5.8 Lab Recruits: Scenario 2 “Even distribution” ...................................... 34
5.9 Lab Recruits: Scenario 3 “Random maze” ......................................... 35
5.10 Results: Lab Recruits Centralized agents optimal behaviour ............... 36
5.11 Results: Lab Recruits Individual agents optimal behaviour ............... 36
5.12 Results: Lab Recruits likelihood difference ....................................... 37
5.13 Scenario 1: Users question 1 answers ............................................. 38
5.14 Scenario 1: Users question 2 answers ............................................. 38
5.15 Scenario 1: Users question 3 answers ................................................. 39
5.16 Scenario 1: Users question 1 answers and policy likelihood values .......... 40
5.17 Users: Question 1 answers and sequence of buttons - Part 1 .................. 41
5.18 Users: Question 1 answers and sequence of buttons - Part 2 .................. 41
5.19 Users: Question 1 answers and sequence of buttons - Part 3 .................. 41
5.20 Scenario 1 Centralized and Individual agents optimal sequence of buttons ..... 41
5.21 Users: Scenario 1 sequence of buttons and policy similarity - Part 1 .......... 42
5.22 Users: Scenario 1 sequence of buttons and policy similarity - Part 2 .......... 42
5.23 Users: Scenario 1 sequence of buttons and policy similarity - Part 3 .......... 42
5.24 Scenario 2: Users question 1 answers ................................................. 43
5.25 Scenario 2: Users question 2 answers ................................................. 43
5.26 Scenario 2: Users question 3 answers ................................................. 44
5.27 Scenario 2: Users question 1 answers and policy likelihood values .......... 44
5.28 Scenario 2 Centralized and Individual agents optimal sequence of buttons ..... 45
5.29 Users: Scenario 2 Question 1 answers and sequence of buttons - Part 1 ...... 45
5.30 Users: Scenario 2 Question 1 answers and sequence of buttons - Part 2 ...... 46
5.31 Users: Scenario 2 Question 1 answers and sequence of buttons - Part 3 ...... 46
5.32 Scenario 3: Users question 1 answers ................................................. 47
5.33 Scenario 3: Users question 2 answers ................................................. 47
5.34 Scenario 3: Users question 3 answers ................................................. 47
5.35 Scenario 3: Users question 1 answers and policy likelihood values .......... 48
5.36 Scenario 3 Centralized and Individual agents optimal sequence of buttons ..... 49
5.37 Users: Scenario 3 sequence of buttons and policy similarity - Part 1 .......... 49
5.38 Users: Scenario 3 sequence of buttons and policy similarity - Part 2 .......... 49
5.39 Users: Scenario 3 sequence of buttons and policy similarity - Part 3 .......... 49

7.1 Questionnaire - Lab Recruits Main Menu with open Instructions tab ............ 58
7.2 Playtesting - Lab Recruits Training Scenario Instructions ........................ 58
7.3 Playtesting - Lab Recruits Scenario1 gameplay ..................................... 59
7.4 Playtesting - Questionnaire introductory page ..................................... 60
7.5 Playtesting - Questionnaire instructions page ..................................... 61
7.6 Playtesting - Questionnaire Scenario1 questions .................................. 62
7.7 Playtesting - Questionnaire scenario ordering question .......................... 63
7.8 Lab Recruits Scenario 1 with game components identified ........................ 63
7.9 Lab Recruits Scenario 2 with game components identified ........................ 64
7.10 Lab Recruits Scenario 3 with game components identified ........................ 64
List of Algorithms

4.1 Squary-Shappy agents individual reward system ........................................ 19
4.2 Lab Recruits agents reward system ......................................................... 23
### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DG</td>
<td>Design Goal</td>
</tr>
<tr>
<td>iv4XR</td>
<td>Intelligent Verification/Validation for Extended Reality Based Systems</td>
</tr>
<tr>
<td>MDP</td>
<td>Markov Decision Process</td>
</tr>
<tr>
<td>RL</td>
<td>Reinforcement Learning</td>
</tr>
<tr>
<td>SMM</td>
<td>Shared Mental Model</td>
</tr>
<tr>
<td>SS</td>
<td>Squary - Shappy</td>
</tr>
<tr>
<td>UX</td>
<td>User experience</td>
</tr>
</tbody>
</table>
Introduction

Contents

1.1 Context and motivation .................................................. 2
1.2 Dissertation objectives .................................................. 3
1.3 Document organization .................................................. 3
1.1 Context and motivation

Starting from components like checking if users enjoy the soundtrack chosen for a movie or understanding if the image picked for an advertisement passes the intend message, to aspects such as understanding how a cellphone’s case feels while using it or evaluating if Non Playable Characters have the correct interactions in video games, testing has always been a focus point. Like other areas, companies that aim at developing interactive software must take into consideration the different components that can bring different contributions to their products as well as the problems that come with said contributions, like their dynamic and subjective concepts as well as their financially expensive and time demanding testing procedures.

One example of this, concretely in line with our work, is in the use of serious games for team training simulation. When a company is focused on using training simulations to improve team collaboration, several questions arise. Can the developers conclude that there was teamwork by simply observing if the team reaches the objective? What if the individual members didn’t work as a team and they actually just acted by themselves but luckily reached the objective? How about if the scenario allowed them to have a choice between collaborating and not? Did the scenario simply forced them to work together? If so, can they really say that they learned to collaborate?

Answering these questions can be somewhat of a difficult task, especially during the developing phase of scenarios where there are constant changes in the maps, objectives, among others. The result of this difficulty is a development bottleneck since the most common testing procedures are based on hiring testers and manually testing each scenario and component. Knowing that a perfect evaluation of user experience and collaboration in a designed scenario, is all but easy, we believe that the use automated agents can bring preliminary results to help reduce some of the overall cost and time of the testing process.

With this in mind we want to develop a method that helps designers of interactive software to test their scenarios and see if they can distinguish different behaviours. For this we will consider different scenarios in two different environments:

- Squary-Shappy - Self-developed game/simulator for this work.
- Lab Recruits \(^1\) - AlGym and game developed in the scope of the Intelligent Verification/Validation for Extended Reality Based Systems Intelligent Verification/Validation for Extended Reality Based Systems (iv4XR) project\(^2\), EU H2020-ICT-2018-3 856716.

\(^1\)https://github.com/iv4xr-project/labrecruits
\(^2\)www.iv4xr-project.eu/
1.2 Dissertation objectives

Emerging from the iv4XR project, our main goal is to use automated agent policies to evaluate designed scenarios on their capability of allowing and distinguishing collaborative and non-collaborative behaviours.

We defined several intermediate objectives that when completed, we know will allow us to reach our main goal:

1. Determine the type of agents to simulate the needed behaviours for this work and define them in terms of Reinforcement Learning parameters (states, actions, among others).
2. Create our own custom simulator to allow us to perform initial tests on our approach.
3. Apply our approach in a shared version of the iv4XR game and AIGym - Lab Recruits.
4. Set up a practical experience with real-life users to corroborate our agents behaviour and results.
5. Analyze the results gathered by both the automatic agents and users.

It is our belief that by completing these objectives this approach can be used as a starting point for future work in the iv4XR project as well as other related research.

1.3 Document organization

This thesis is structured as follows:

• Chapter 2 gives a clear explanation of how a simple Reinforcement Learning training is performed.

• Chapter 3 is divided into three sections. Section 3.2 presents work related to the use of automated agents to test game scenarios. Section 3.3 offers an explanation of how collaboration is achieved and how we can recreate it in automated agents.

• Chapter 4 explains how our approach works. Section 4.1 defines how to analyse scenarios. Sections 4.2 and 4.3 detail how our approach was implemented in both testing environments. Section 4.4 defines how we compared behaviours.

• Chapter 5 presents the results of this work.

• Chapter 6 compiles the core conclusions of this work, its limitations and provides context for future work.
2

Background
In this chapter we present a simple explanation of what Reinforcement Learning (RL) is and how it works in order to provide better context for this Dissertation’s work.

Reinforcement Learning is a popular domain of machine learning focused on the training of automatic software agents for the purpose of teaching them how to act, navigate and complete a multitude of tasks in a specific software. The basic premise of RL is the use of trial and error as well as feedback from actions and past experiences to find a sequence of decisions/actions that completes the intended task. A basic RL problem is composed of a number of parameters:

- **Environment** - The world in which agents operate and interact with.
- **Agent** - The entity that acts upon an environment and makes decisions based on rewards.
- **State** - A set of variables that describe the current situation.
- **Actions** - All possible ways of interacting with the environment.
- **Reward** - The positive or negative feedback that an agent receives after taking an action in a particular state.
- **Policy** - The strategy that an agent updates and uses to complete an objective. The policy maps the actions that the agent takes based on the state.

Let’s consider the popular video game Pac-Man as a RL example. The goal of the agent (Pac-Man) is to eat all the food in a grid-based world, the environment. The states can be defined by a set of positions composed by the agent, the food and the enemy ghosts. Pac-Man can perform actions such as *Move Left, Move Right, Move Up* and *Move Down* and gets a reward every time he consumes food (positive reward) or gets killed by a ghost (negative reward). The agent will save this information (current state, action made and reward received) in its policy so that in the future he can make a better decision on which action to make. After the agent plays the scenario during a number games, in RL referred as training episodes, depending on the scenario complexity, possible actions, among others, he will eventually learn the policy that maps the state-action sequence needed to win that level. Figure 2.1 illustrates this process of viewing the current state, performing an action, receiving feedback and saving all this information.

![Figure 2.1: Reinforcement Learning Architecture](image)
3 Related Work

## Contents

3.1 Software testing .......................................................... 7
3.2 Automated playtesting with Agents .................................. 8
3.3 Collaborative behaviour .................................................. 9
3.4 User experience analysis ................................................. 11
The approach taken for this work derives from areas such as Software testing, Automated playtesting with agents, Collaborative behaviour and User Experience analysis. Regarding Collaborative behaviour, since our objective is to test scenarios on their ability to allow and distinguish collaborative and non-collaborative behaviour, we are particularly interested in understanding how collaboration can be identified in interactions between humans and structured between agents.

In this chapter we present a condensed description of work previously made in the mentioned areas related to our approach and that was also used as inspiration.

### 3.1 Software testing

Software testing ([1], [2]) is a process to evaluate the functionality of a software application and it aims at finding if the developed software meets the specified requirements and design goals in order to produce a quality product. Regarding the type of tests that can be made, there are three major levels:

- **Unit Testing** - A test performed on the smallest piece of testable software to check if said module is properly built.

- **Integration testing** - Executed to test the connectivity or data transfer between two, or sometimes more, unit tested modules.

- **System testing** - A complete end-to-end test to verify complete system specification requirements.

All these previous types of tests can be used in two major approaches, *White Box Testing* and *Black Box Testing*. White Box testing is defined as an approach that tests the software’s internal structure and primarily focuses on analysing the flow of inputs and outputs through the application. By doing so, it is capable of testing all the individual code paths in a module, where all the logical decisions are made, check possible loops and so on. On the other hand, Black Box testing aims at testing the functionality of an application without the need of going into the implementation detail level. Mostly thought of as the end-user perspective, it provides cover to a system functionality and specifies if it meets the initial requirements. A third category *Grey Box Testing* has been gaining prominence since it serves as a middle ground for the two previous ones. In sum, software testing is characterised as the activity of determining logic properties of a particular system and evaluating if said system meets the specified goals.

Our work derives from standard software testing since we are aiming at performing Black Box tests at a system level to determine if the system (scenario) can meet its initial requirements (distinguish collaborative and non-collaborative behaviours). The main difference in our work comes from the fact that our design goals can not be specified only in terms of logic properties of the software but since we’re aiming at distinguishing behaviours, they also need to take into account human factors.
3.2 Automated playtesting with Agents

Aiming at decreasing the time taken and the manual overhead that playtesting causes has led to the use of automatic playtesting. The main objective behind automated playtesting consists in the use of autonomous agents that will test the software and gather data only previously obtained by real-life users. Correlated to our work, automated playtesting with agents is commonly used in the gaming industry where automatically testing maps and specific scenarios is one of the main focuses since it could help the content creation process during a game’s development.

Silva et al. [3] expanded on their previous work [4] and demonstrated how the use of different intelligent agents allowed them to evaluate the *Ticket To Ride* game. Their work consisted on simulating common user strategies, gathered from the game’s community forums, with game related heuristic-based automated agents and, with said agents, analyse several map variations on their impact in different strategies and number of players. Their work proved to be successful since they were able to show that it is possible to both characterize the desirability of various parts of the maps, the relative strengths of strategies in the map variants and with different number of total players. They were also able to determine the differences between maps in terms of what strategies work best as an unexpected result, they also identified two failure cases in the game, where agents found game states that were not covered by the game rules.

Holmgard et al. [5] presented a method that consisted on using player modeling to create automated agents as game-personas and enabled them to automatically playtest and evaluate content in the game *MiniDungeons 2*. After several experiences they observed that use of automated persona-like agents, in games with similar complexity to the *MiniDungeons 2*, proved to be helpful when evaluating the game levels, more significantly, identifying which maps were better suited for each type of players. With a similar method and objectives, Mugrai et al. [6] presented their work on the use of different persona-based automated agents to playtest various levels of the game *Match-3* in order to analyse the impact that level designs have on the different playstyles and strategies. Supported by a user study, their experiences were successful in showing that the use of different player perspectives and strategies allows to properly evaluate what can be perceived as the difficulty of new level designs for said playstyles in the *Match-3* game.

From understanding which strategy is better for a certain map to determining how the design of levels affect different playstyles, the work mentioned in this Section provides an indication that analysing game maps and scenarios can be done using specifications of design goals (strategies, game objectives, among others) in automated agents, which is in-line with this works objective of using automatic agents as a way of evaluating game-like scenarios on their capability of distinguishing collaborative and non-collaborative behaviours (design goals).

---

1Personas are fictional characters created to represent a user type that might use a specific strategy and playstyle.
3.3 Collaborative behaviour

To understand how to differentiate and recreate collaborative and non-collaborative behaviours for our automated agents, we must first understand how collaboration works and is defined in human-human interactions to then define how to replicate it between agents.

3.3.1 Collaboration between humans

Most of, if not all, living things have used collaboration in several aspects of their lives since they know it improves individual production [7] and the human species has been collaborating since the early stages of our existence until the modern day, whether it’s hunting animals in a prehistoric age or raising a modern building [8], working as a team towards a shared goal has been a common behaviour for our species.

Although every situation is unique and every group interacts in a different manner, there are several collaboration aspects that are considered universal such as, clear and open communication [9], consensus about goals and methods for completing tasks [10], clear definitions on each contributor’s role [11], placing group goals above individual satisfactions [12], among others. The successful implementation of these elements through communication and coordination is the key for teams/groups to work effectively in a collaborative manner. An example of this is presented in [13] where Bagshaw et al. displayed their work regarding collaboration in international research projects and how to promote it. The authors started by highlighting some of the roadblocks they encountered from experience and related literature, such as Individual promoting self-interests, Ambiguity and uncertainty regarding the project definition and focus, among others. These roadblocks can be difficult to overcome in any group and even more in an international team, where people come from different backgrounds, speak different languages and have different educational systems. In order to counteract the previously mentioned obstacles, the authors presented a list of several strategies that they used such as: All team members must jointly establish clear goals and objectives; Access to information should be provided from and to all members; Team members should be open to new information and be willing to integrate different perspectives and ideas into their individual logic, therefore adapting to new situations. These strategies, and a list of others, were proven to be successful in elevating the level of collaboration inside the team.

We’ve shown capable of pointing out different elements of collaboration such as clear definitions on each member’s role or placing group goals above individual satisfactions, but can we explain how teams connect all these strategies and define a structure for collaborative behaviour?

Stephen et al. [14] researching high performing teams suggested that experienced teams develop a shared mental model in order effectively coordinate and predict each others movements while also improving their communication. Shared Mental Model (SMM) are described as knowledge structures
for group members to similarly understand the team’s objectives, peers individual needs and everyone’s responsibilities. The use of this structure allows for all members of a team to have the shared understanding of a problem and to have a clear understanding of the most effective way to approach the problem, meaning the techniques and resources the team should utilize as well as everyone’s role in them. An example of a shared mental model can easily be explained in a professional team sports environment where the members of a team must rely on their shared understating of a game situation and play according to their knowledge of what is the best strategy for the team.

The use of shared mental models allows for all members of a team to visualize a problem at hand with the same mindset and anticipate future actions and states. No matter the complexity or simplicity of a task, the use of shared mental models has been identified as a collaborative indicator in human-human interactions [15].

3.3.2 Collaboration between agents

Creating collaborative agents can be a difficult task. How can we have artificial agents working as team? Putting group goals above their own? How can we create agents that genuinely work as a team and not just happened to have the same individuals goals?

Different methods have been used but a common technique on helping agents collaborating is to create a system similar to what human teams develop, a shared mental model. The SMM method has been widely researched for intelligent agents ([16], [17]) and proven to help agents act collaboratively. Similarly to humans, a mental model means an internal representation of a situation and the shared knowledge of the best actions for the group. More specifically for automated agents, a SMM defines a cognitive structure for common information access and that connects and extends the notion of individual goals/needs to a team context [18].

The use of an SMM is proven to be an effective method to help agents collaborating, but since this work will focus on distinguishing collaborative behaviour, we want to create inherently collaborative agents. One way to do that in to use a centralized method. A centralized approach aims at providing a complete scheme of the general current state [19] to all agents, simulating intrinsic communication, as well guaranteeing that by executing coordinated joint actions, the best action for both agents (collaboration) is always chosen.

The work presented in this Section showed us that based on interactions between humans we can define a structure, SMM, to help agents act collaboratively. Since we want to have inherently collaborative agents we can also utilize the centralized method that possesses homogeneous properties. These structures/methods served as the base inspiration for or automated agents architectures, further explained in Chapter 4.
3.4 User experience analysis

We want to verify if our automated agents can simulate human-like behaviour and also to determine if users corroborate our scenario order on their capability of distinguishing behaviours. This will be helpful when concluding if our approach can, or not, bring meaningful contributions. In order for this to happen, we want to understand what is the best way to have context behind the data retrieved from users. When it comes to User experience (UX), some of the most utilized methods to provided developers with appropriate context have been video recordings, physiological reactions such as eye tracking, heart rate and of course, questionnaires.

User questionnaires are made to provide users with a personal input on, mostly, subjective system properties and topics. Vermeeren et al. [20] detail that in a questionnaire, the questions are categorized in mainly two different groups, Predefined Measures, also known as Closed Questions, and Open Evaluation. Predefined measures are questions usually created towards the evaluation of specific components, meaning that they rely on a quantitative scale and the answers gathered are restricted by the options given. Questions from the Open Evaluation category rely on a more qualitative approach since they allow for testers to freely describe their experience. These type of questions do allow for a more in-depth knowledge of the users state of mind but often require longer periods of time to analyse.

Our user playtesting experience will be made online with no direct access to the users, meaning that for us to obtain context on how the users behaved while playing we have to create a user questionnaire. Our questionnaire will include questions from both the Predefined Measures and the Open Evaluation categories in order to corroborate our results regarding the scenario order and get context on why the users behaved the way they did in each scenario.
Methodology

Contents

4.1 Scenario analysis ................................................. 13
4.2 Squary-Shappy .................................................. 14
4.3 Lab Recruits ..................................................... 20
4.4 How to identify behaviours? ................................. 26
In this chapter we will describe our approach, used to create, detect and analyse behaviour in each scenario. We give an explanation of the two environments used to test our solution, detailing the map designing process, the different architectures used to create the automatic agents, the algorithms implemented to train the agents as well the optimization methods for said training. We will also delineate a description of the real-life users playtesting experiment with the Lab Recruits game made in order to corroborate the results from the automated agents.

4.1 Scenario analysis

In a software testing method the main premise is to check if the software reaches its main purpose, the Design Goal (DG). For a game developer the DG can be to create an engaging experience, for a developer of an intelligent tutoring system a DG could be to ensure that the students acquire the knowledge components and for a developer of a serious game for team training simulation, the DG may be that the users work as a team. Although this is strongly related to regular software testing, the main difference is that in our case the Design Goal can not be specified only in terms of logic properties of the software but also need to take into account human factors.

4.1.1 How to define the Design Goals?

Design Goals can be defined in many different ways such as examples of the intended behaviour, a reward/cost function that induces the behaviour, a set of rules and many others. Here are some examples:

- Behavioural traces - The DG can be directly specified via designer provided demonstrations of the desired behaviour as well as different behaviours that are to be tested. For instance if the DG is to train users on how to work as a team, the designer might provide traces of a group of people working as a team and traces of people not working as a team.

- Consequential descriptions - Instead of focusing directly on the behaviour, the DG can be a specification of the consequences the behaviour must have. Those consequences can be external or internal to the user. For instance, if the DG is to provide an engaging simulation experience, the designer might provide a description of the envisioned emotional state of the user.

- Goal descriptions - Another alternative way is to provide the goal of the behaviour and not the behaviour itself. In a reinforcement learning perspective, while the behavioural descriptions provide the policy, the goal description will provide the reward.

In this work, our Design Goal is to have agents/real-life users playing in a collaborative manner. For this we want to check if it’s possible to distinguish collaborative and non-collaborative behaviours in different scenarios by comparing behavioural traces.
4.1.2 How to test a scenario?

We will now understand how we can test if a given scenario achieves a desired Design Goal.

Considering that some DG are very abstract there are numerous situations where we cannot guarantee if a DG is achieved, but, in most cases, we can determinate if a DG is not achieved. A clear example is if our DG is to understand if a user can learn basic arithmetic's. If we present a user with \((2 \times 2 + 2)\) and he responds 6, although is the correct answer, we are still not sure if the user understood that \(\times\) has priority in relation to +. Meaning that for any of the different ways to specify the Design Goals we must get a set of behaviours that achieve the desired goals \(\pi_g\) and another set that does not verify the design goals \(\pi_{\bar{g}}\).

Figure 4.1 explains this difference. On the left side we have a system that given only a desired behaviour is able to decide if given scenarios permit to fulfill the DG, but, like we’ve seen, this may not be the best option. On the right side we consider that by having the definition of the desired behaviour and a definition of an undesired behaviour allows the system to then decide if not only the DG is fulfilled but also if the non-Design Goal \((\neg DG)\) is not.

![Figure 4.1: How to test a scenario](image)

Given this difference, our approach is based on testing each scenario on the DG and on a specification of \(\neg DG\)\(^1\) to determine if we can distinguish them and order the scenarios on their ability to differentiate both behaviours.

4.2 Squary-Shappy

We started by creating our own simulation gym, a Python based program called Squary - Shappy (SS) that, due to its simplicity, served as the initial environment to create the automatic agents and implement our approach. The Squary-Shappy environment simulates a 2D object collecting game where agents roam around the map trying to eat as much food as they can. Both the characters and the food are positioned around the map by the designers (in this work means ourselves) and the food objects do not

\(^1\)Although there are clearly many different amounts of possible \(\neg DG\), for specific problems, such as ours, the main effects can be easily identified.
regenerate, meaning the food is limited by the number initially deployed. The maps are also all designed in a closed room format, meaning the agents cannot move out of the established bounds.

Although our focus of this work is in testing our approach in the Lab Recruits game, the creation and use of this custom simulator allowed us to first test our approach in small and simple scenarios and to understand if the initial results were positive. One other motivation for this was to demonstrate that our approach could be used in two different environments/games.

### 4.2.1 Map Designing

Our initial work in map designing consisted of creating 1-Dimensional maps to test our agents but since 1-Dimensional maps are somewhat restrictive we quickly moved on to 2-Dimensional maps. The maps in the SS environment were designed in individual `.txt` files which allowed us to quickly create new scenarios and alter existing ones, since that, at their core, they were a simple matrix using a basic symbol format:

- `'.'` (dot) - Represents empty locations where the agents could move to.
- `'1'` - Represents walls, objects that agents can not pass through, visually identified by a black square.
- `'2'` - Represents the objects (food) that agents would aim to collect, providing them with a positive reward, visually identified by a green square.
- `'3'` - Represents the location of the first agent, visually identified by a blue square.
- `'4'` - Represents the location of the second agent, visually identified by a red square.

![Figure 4.2: Squary - Shappy: Map matrix example](image1)

![Figure 4.3: Squary - Shappy: Visual map example](image2)

This method of designing the maps as matrices also brought advantages when it came to the agents training, since any change/action/movement performed in the simulation could be made by simply changing values in matrix, a process that has very low processing cost. In order to create the visual aspect
of the simulation we used the Pygame \(^2\) module which includes a number of easy to use libraries for computer graphics and game designing in Python.

4.2.2 Agents

To automatically create the previously mentioned desired, collaborative, and undesired, non-collaborative, behaviours we started by developing two different agent architectures and training them using RL.

4.2.2.A Architecture

Using the SMM referred in Section 3.3 as inspiration, we choose to use the Centralized architecture since our goal is not to teach agents to collaborate, but instead to create inherently collaborative and coordinated agents to then use their behaviour to evaluate the scenarios. In order to define our Desired Behaviour, we implemented as an agent architecture a centralized method, illustrated in Figure 4.4. Our Centralized architecture means that all agents are a part of a shared mind that takes into account the combination of the effort made by every single agent (joint effort), makes the agents act in combined actions (joint actions), rewards the agents as a unit and not individuals (joint rewards) and fundamentally makes them function as an optimal collaborative group/team.

![Centralized architecture](https://www.pygame.org/)

On the other hand, and like we mentioned in Section 4.1, we can have a multitude of possible undesired behaviours. For this work, and according to the scenarios created, we believe the most general undesired behaviour can be created using an Individual architecture, depicted in Figure 4.5. This architecture follows a completely opposite thought-process to the centralized one. The individual agents have no capability of knowing information regarding the other characters in the environment and therefore their only focus is themselves. Each of the agents have their own individual mind that only observes their personal state and all of the decision making, and consequently acting, is made for their personal interest.

\(^2\)https://www.pygame.org/
4.2.2.B Training

In order to train agents with the aforementioned architectures we chose to use Reinforcement Learning, more specifically a Markov Decision Process (MDP) with Q-Learning.

We choose to use an MDP since it’s predicated on the Markov Property “The future is independent of the past given the present” which implies that in a RL problem, the next state \( N+1 \) only depends on the current state \( N \), which directly relates to our scenarios since the next best action to perform is always based on the current state.

\[
Q(s, a) = Q(s, a) + \alpha \times [r + \gamma \times \max (Q(s', a')) - Q(s, a)]
\] (4.1)

Figure 4.6 which allows for a more visual understanding of the MDP with Q-Learning algorithm, explains that during the training, agents observe their current state and, from the possible actions, chose one to execute. After performing the selected action, the agents they will then receive a reward (positive or negative) from the environment and observe the resulting state. This information will then be used to update the Q-Table, the agents “brain”, by means of the Q-Learning formula:
Where \( Q(s,a) \) represents the Q-value of the performed action in the current state, \( \alpha \) represents the learning rate, \( r \) represents the reward, \( \gamma \) is discount factor, a real value in the range \([0,1]\) and is used to balance the relevance of the immediate reward with respect to future actions. \( \max(Q(s', a')) \) is the maximum Q-value for any action of the next state.

The scenarios were formulated into a RL problem by defining the MDP parameters as the following.

**States:** Like we previously mentioned, in these maps the objective is eat as much food as possible. Since the agents are only able to move one step at a time, the food has a static position and is limited, we decided to describe the state of the environment using the positions of the agents (combined position for both agents in the centralized type and only their own position in the individual agents) as well as the position of the food objects that still exist. Every time a food object was eaten, that position was removed from the state.

- **Individual:** \{PosAgent, PosFood1, PosFood2, PosFood3\}
  - Example agent1 state: \{\(<2, 2>, <1, 1>, <4, 8>, <6, 1>\)\}
  - Example agent2 state: \{\(<6, 3>, <1, 1>, <4, 8>, <6, 1>\)\}
- **Centralized:** \{PosAgent1, PosAgent2, PosFood1, PosFood2, PosFood3\}
  - Example: \{\(<2, 2>, <6, 3>, <1, 1>, <4, 8>, <6, 1>\)\}

**Actions:** In the Squary-Shappy simulator, the agents don’t possess any type of pathfinding method and since the simulation is made in a simple 2D matrix, the agents can only perform low-level actions. It is also relevant to mention that in this environment, a character eats an object by simply occupying its position, no additional move was required. For the Individual agents they had the following action space:

- **NOTHING** = Stay in the same place.
- **UP** = Take one step up.
- **DOWN** = Take one step down.
- **LEFT** = Take one step left.
- **RIGHT** = Take one step right.

For the Centralized type agents, since their actions were made in pairs, they had the following action space:

- \([\text{NOTHING, NOTHING}]\) = Both agents stay in the same place.
- \([\text{NOTHING, UP}]\) = First agent stays in the same place, the second one moves one step up.
- ... remaining permutations...
• \([\text{RIGHT, RIGHT}] = \) Both agents take one step right.

**Rewards:** For the rewards we took a simple approach. Every time an agent consumed a food object they were positively rewarded with a value of +100. If the agent didn’t move they didn’t receive any reward, positive or negative, meaning the reward resulting in 0. One other condition was that for every step an agent took they would lose energy, meaning they would get a small punishment of \(-1\). The reward system for the Centralized agents functioned as a collective, meaning the actual centralized agent, the shared mind controlling two characters, received the combined amount, whereas for the Individual type agents, they were attributed their own rewards without being affected by other agents. This reward system is exampled on Algorithm 4.1.

**Algorithm 4.1:** Squary-Shappy agents individual reward system

```
if Agent eats food then
    reward = +100;
else if Agent does nothing then
    reward = 0;
else
    reward = -1;
end
```

Using these MDP parameters we trained our agents and learned two policies. The Centralized policy, representing the DG, counted with two agents working with the centralized architecture whereas the \(\neg \text{DG} \) was represented by the Individual policy, containing two agents with the individual architecture. Since the scenarios were considerably small and easy to solve, both types of agents, centralized and individual, performed 100,000 training episodes. This training episodes ended when the agents either completed the objective, eating all of the food objects, or performed the maximum number of allowed actions, 64.

The constant Q-learning values were defined as the following. The learning rate \(\alpha\) was set at 0.1 and the discount factor \(\gamma\) was set at 0.9. The agents exploration-exploitation trade-off regarding their decision making was based on an value \(\epsilon \in [0, 1]\). The probability of making random decisions was defined by the \(\epsilon\). This value was initially set as 1 and decreased by a ratio of \(\frac{1}{\text{totalEpisodes}}\). this means that in the first episode, the agents had a completely random decision making (maximum exploration) and during the course of the training those decisions started to be made less randomly and more accordingly to their policy (past experiences), until the final episode where the agents had a random decision probability of \(\approx 0\), which means maximum exploitation.
4.3 Lab Recruits

Upon creating the Squary-Shappy simulator and implementing our approach in it, we moved on to a more high-end environment, the Lab Recruits game designed by the University of Utrecht for the iv4XR project. It is worthy to notice that since this game is being developed for the iv4XR project, it is constantly being updated with new features, therefore, the version used for this Dissertation was, at the time, the most recent, but due to the project continuous progress, the latest version can have several differences from the one we used.

In this game, players are positioned inside a room, or combination of rooms, where the objective is to click the target green button(s). For this to happen, the characters have to roam around the map and click red buttons to open doors and access new rooms to explore. Contrary to the Squary-Shappy food objects, these buttons do not disappear and could be clicked unlimited times, turning them ON/OFF to respectively open/close doors.

For the Lab Recruits game, the iv4XR framework ran all its tests and simulations in a Unity environment. Every move/action/change done in the map had its information passed to the Unity application through a socket, applied using the Unity game physics engine and relayed back to the Java application through the same socket.

![iv4XR Framework to Unity architecture](image)

**Figure 4.7:** iv4XR Framework to Unity architecture

Although this method, depicted in Figure 4.7, provides more accurate data regarding to map positioning, physics, among other parameters, the amount of time needed to train agents in real-time would be too high. For this reason, and since for this work and in our scenarios the Unity physics could be considered meaningless, we decided to create a model of the map that could be simulated in a 2D matrix, similar to the Squary-Shappy environment, to help speed up the agents training process and still provide the wanted policies.

4.3.1 Map Designing

In the Lab Recruits game, the map designing followed a very similar approach to the one we took in the Squary-Shappy scenarios, meaning that we used a simple symbol format, previously defined by the University of Utrecht, so that could easily build/alter scenarios in a .csv file. Although the game itself was a Unity executable, the iv4Xr framework possessed a method to create the maps as 2D matrices using discrete units, which allowed for an accurate visualization when creating them. Figure 4.8 depicts
an example of a map matrix for Lab Recruits while Figure 4.9 depicts the correspondent visual example. It also worthy to notice that the Lab Recruits Unity executable inverts the map when turning it visual.

\[
\begin{array}{cccccccccccc}
\text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} \\
\text{w} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{a0} & \text{f} & \text{f} & \text{w} \\
\text{w} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{w} \\
\text{w} & \text{f} & \text{f} & \text{w} & \text{w} & \text{f} & \text{f} & \text{w} & \text{w} & \text{w} & \text{w} \\
\text{w} & \text{b1} & \text{f} & \text{w} & \text{f} & \text{f} & \text{f} & \text{w} & \text{f} & \text{b3} & \text{w} \\
\text{w} & \text{f} & \text{f} & \text{w} & \text{w} & \text{f} & \text{f} & \text{w} & \text{f} & \text{f} & \text{w} \\
\text{w} & \text{w} & \text{w} & \text{w} & \text{f} & \text{f} & \text{w} & \text{w} & \text{d2} & \text{w} & \text{w} \\
\text{w} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{f} & \text{w} \\
\text{w} & \text{w} & \text{d1} & \text{w} & \text{f} & \text{f} & \text{w} & \text{w} & \text{w} & \text{f} & \text{f} & \text{w} \\
\text{w} & \text{f} & \text{f} & \text{w} & \text{f} & \text{f} & \text{w} & \text{f} & \text{f} & \text{f} & \text{w} \\
\text{w} & \text{b2} & \text{f} & \text{w} & \text{f} & \text{f} & \text{w} & \text{a1} & \text{f} & \text{f} & \text{f} & \text{w} \\
\text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w} & \text{w}
\end{array}
\]

Figure 4.8: Lab Recruits: Map matrix example

Figure 4.9: Lab Recruits: Visual map example

### 4.3.2 Agents

For this environment, we used the same agent architectures as in the Squary-Shappy simulator. We again developed a Centralized architecture for our desired behaviour where agents are a controlled by a shared mind who takes into account the joint effort, makes joint actions and gets joint rewards, making the agents behave with only the whole teams best interest in mind. For the undesired behaviour we implemented the Individual architecture where agents have their own individual decision process and only act taking into account their own state and personal benefit.

#### 4.3.2.A Training

The development and training of the agents was done inside the Java iv4XR framework and we once again used an MDP with Q-Learning since it provided simplicity in the implementation and it had also demonstrated good initial results in the Squary-Shappy scenarios. One other possibility regarding where to apply our approach was to create a Python module \(^3\) but since an MDP with Q-Learning is a fairly easy algorithm to implement and the framework already had some built in methods related to the Lab Recruits game, the difficulty of pairing a Python module to the iv4XR Java framework was out-weighed by the intricacies of working with Machine Learning in Java.

Like previously explained, instead of training the agents using the game itself, we developed a 2D matrix model to represent and simulate the environment of the Lab Recruits game. One problem re-

\(^3\)Python is a more used commonly language for Machine Learning due to the large number of libraries, documentation and its simple coding language syntax.
Regarding the use of a model to represent the game was that since the model was made using discrete units, one step for the characters in this model would be equal to a number of smaller steps made in the Unity application. Although pondered, this problem was not deemed a major obstacle for our work.

In order set up our environment scenarios as a RL problem for the agents training, we defined the MDP parameters as the following.

**States:** In the Lab Recruits games, the agents interact with buttons to open and close doors until reaching the target button. Since the state of the doors (open or closed) are the only visual feedback that real life players have, we choose to use this as part of the agents MDP state. The doors internal state can be described as:

- 0 = Door is closed.
- 1 = Door is open.

Appending this to the agents position, and with the already mentioned differences between the centralized and the individuals agents, resulted in the following states:

- Individual: \{PosAgent, IntStateDoor1, IntStateDoor2\}
  - Example agent1 state: \{<3, 0, 8>, [0, 1]\}
  - Example agent2 state: \{<10, 0, 3>, [0, 1]\}

- Centralized: \{PosAgent1, PosAgent2, IntStateDoor1, IntStateDoor2\}
  - Example: \{<3, 0, 8>, <10, 0, 3>, [0, 1]\}

**Actions:** Although the iv4XR framework had access to the built-in pathfinding system of the Unity application combined with a set high-level actions, since the users controls were composed of low-level actions we decided to once again use the basic low-level actions but added one specific action related to the Lab Recruits game, pressing on a game button. The individual agents agents could perform the following actions.

- Individual:
  - NOTHING = Stay in the same place.
  - UP = Take one step up.
  - DOWN = Take one step down.
  - LEFT = Take one step left.
  - RIGHT = Take one step right.
  - PRESS = If the character is on top of a button, press it.
In the centralized architecture and to have the agents act coordinatively, the actions were again in pairs with all possible permutations.

- Centralized:
  - [NOTHING, NOTHING] = Both agents stay in the same place.
  - [NOTHING, UP] = One agent stay in the same place and the other moves one step up.
  - ...remaining permutations...
  - [PRESS, PRESS] = Both agents try to press buttons.

Rewards: Agents will get positively rewarded, with a value of +100, for completing an objective (turning ON a target green button). If the agents choose to stay still and do nothing, the reward attributed will be 0. For any other action such as moving in any direction, pressing a red button or clicking an already clicked green button, a negative reward will be applied with a value of -1 simulating the energy spent on that action. This system is simply explained on Algorithm 4.2.

```
Algorithm 4.2: Lab Recruits agents reward system

if Agent turns ON TargetButton and isFirstTime then
    reward = +100;
else if Agents does nothing then
    reward = 0;
else
    reward = -1;
end
```

Contrary to the agents training made in Squary-Shappy, the Lab Recruits game presented more complex scenarios with different sizes and shapes. Due to this fact and also considering the larger action space for both the individual and the centralized agents, we did not defined a defined maximum number of training episodes, but instead we evaluated the agents performance (number of steps until reaching the objective) and ended the training once that value converged. Each training episode ended under one of two conditions, the agents could either successfully complete the scenario objectives by clicking all the green buttons, or execute a maximum number of actions. This maximum number of actions was defined for each map by calculating the number of total centralized actions by the size of the map.

4.3.3 Optimizations: Dyna-Q

One difficulty that we encountered in the Lab Recruits game was the time taken to train the agents, even using a 2D matrix model representation of the game. This could attributed to the size of the maps, the complexity of the scenarios and the large number of combinations between all states and actions. In
In order to try and speed up the process, we implemented an optimization method for the agents training called Dyna-Q.

The Dyna-Q algorithm aims at collecting the agents past experiences and using them to further update the policy without the need for the agents to physically perform an action in the “real” world. Upon each MDP cycle, meaning after each Q-table update, the agents save their experience by updating a Transition table. The Transition table contains the information that if an agent is in a state $s$ and performs action $a$, then the resulting state will be $s'$ and will receive reward $r'$. This means that all the agents past experiences are kept inside that Transition Table. After each training episode, the agents enter a state of “hallucination”4 where they use the information inside the table to virtually replay $N$ random actions and update the Q table with the resulting values.

![Figure 4.10: MDP + Dyna-Q](image)

Figure 4.10 depicts the Dyna-Q algorithm. After a regular training episode where the agents constantly save data regarding not previously known experiences (constantly mapping state $\rightarrow$ action $\rightarrow$ reward $\rightarrow$ new state) in the Transition Table, they enter the state of hallucination and for a defined number of times, in our case it was the same as the maximum number of steps in each training episode, the agents replay random movements by selecting a random state and action that they already experienced, followed by searching in the Transition Table for the corresponding reward and resulting state to then update the Q-Table. This means that the agents are using their “memory”, from past experiences, to reproduce real values and use them to update the Q-Tables.

---

4Hallucination - Commonly used to describe a Dyna-Q cycle since no actions are done in the real world and the agents only use information in their “memory”, the Transition table.
4.3.4 User Playtesting

Although the purpose of our work is to help developers test scenarios that promote collaborative behaviours using automated agents and without the need of real-life users, in order to understand if the created agent policies could be deemed as “human-like behaviour”, we decided to have a playtesting experience with users to corroborate the agents behaviour and our results.

Aiming at reaching a higher number of players, the playtesting experience was made entirely online. Users were asked to play the Lab Recruits game and each answer a small questionnaire, depicted in Figures 7.4, 7.5, 7.6 and 7.7. The Lab Recruits game was created with the purpose of research in the scope of the iv4XR project and, at the time of this work, the game was still being developed and constantly updated. For these reasons and also with the purpose of allowing less game knowledge users to participate in the experience, some modifications to the Lab Recruits game were made:

- We created a Main Menu for a more user-friendly interface where we displayed information needed for the game such as general instructions and keyboard controls, Figure 7.1.

- We designed an initial training scenario where players received a summed version of the games controls, key bindings and other important information, Figure 7.2.

- We developed a two-player system with two keybindings and cameras, one for each character, Figure 7.3. In the version initially used we could only move one player at a time.

- Since the Lab Recruits was mainly designed for researchers, each map had to be individually and manually uploaded every time it was necessary. In order to eliminate that manual process, we set the game up to automatically switch scenarios after the users completed each scenario’s objective, clicking on the green button(s).

- Other minor details were added to give a more enjoyable experience to user such as fading screens between transitions, key reminders before starting a new scenario and the option to pause mid-game and review initial instructions.

To try and simulate the same playing conditions as our automated agents, users, playing pairs, were given the opportunity to get familiar with the game controls and mechanics of opening doors, accessing new rooms, by playing in a Training Scenario. Users were allowed to play each scenario presented in Subsection 4.3.1 a total of three times to simulate the agents training scenarios, better explained in Chapter 5. We also advised them to not communicate with each other, since the automated agents didn’t had explicit communication. And finally we asked them to try and complete the scenarios objectives as fast as possible.

The data gathered from the users playtesting were their behavioural traces, meaning that at every frame and using the same centralized state format from our MDP implementation, Section 4.3, we
recorded every action made the users and linked them the game state, resulting in the traces being a sequence of state-action elements. Here we present an example:

\[ (3, 0, 7), (9, 0, 7), [1, 0], (\text{Left, Nothing}) \]

Where \((3, 0, 7)\) refers to the first character's position, \((9, 0, 7)\) to the second character's position, \([1, 0]\) means that door1 was opened and door2 was closed and \((\text{Left, Nothing})\) represented the actions made by both characters, in this example character1 moved left and character2 did nothing.

Regarding the questionnaire and in order to minimize a possible opinion persuasion between users, we asked them to answer a set of questions, each in an individual section. The questions asked were selected to help us understand the internal decision process from the users, to verify our results when evaluating the scenarios and also to confirm if our agents could be considered to have a human-like behaviour. They were repeated for each one of the scenarios and were related to behaviour users had while playing, if and what type of communication they had during the game, if they tried to finish the scenario as fast as possible or roamed around the map. These questions are further explained in Section 5.3.

After completing all the runs in each of the scenarios, users received a randomly generated ID in-game to insert on the questionnaire. This random ID was the only identification we had to connect the questionnaire answers to the playtesting traces. No other type identification from the users was requested or retained.

### 4.4 How to identify behaviours?

Like we previously explained, our main objective is to help developers of collaborative inducing software to test their scenarios by comparing behavioural traces of a DG and a ¬DG to understand if the scenarios allow to distinguish between collaborative and non-collaborative behaviour. In order to achieve that, we had to create a method of comparing any behavioural trace with the Centralized policy (DG) and the Individual policy (¬DG).

Many forms of comparison could be made. Imagining that we have an example behavioural trace to compare to both policies, the most direct comparison was to simply count at every state if the action made was the best action (highest Q-value) possible according to each policy. But what if the actions made were not the best possible actions at each game-state, but the second best? Or what if in particular states there wasn’t a big difference, regarding the Q-value, between the best and worst action in the policy? How could we account for those types of scenarios?

We had to define a voting system to make sure every action was properly accounted for. We decided
to go with the softmax function using the policies action Q-values. The softmax function, commonly used in Machine Learning, serves as a function to calculate the likelihood of a given input, state-action pair, belonging to the target classes, Centralized and Individual. One relevant property of this function is the output values since they’re in the range [0, 1] which is useful because we can avoid a binary classification and allows for easy interpretation. The softmax function goes as following:

\[
L(s,a) = \frac{e^{Q(s,a)}}{\sum_{k \in \text{actions}} e^{Q(s,a_k)}}
\]  

(4.2)

Where \((s,a)\) represents a given state-action pair, \(Q(s,a)\) is the Q-value of a policy for said state-action pair and \(Q(s,a_k)\) is the Q-value of an state-action pair where \(a_k\) is a specific action belonging to the centralized action space.

The softmax classification for this work goes as follows:

1. The system will receive as an input a behavioural trace, either from the automated agents or from real-life users.

2. For a state-action pair from said behavioral trace, the system will compute the softmax probability value of that state-action being more similar to each policy.

3. The previous step will be repeated for every state-action pairs in the behavioural trace.

4. After analysing all state-action pairs, the system will then output the summed and normalized similarity values for both policies.

By calculating the highest likelihood this voting process will allow us to identify if any given behavioural trace followed the DG or the ¬DG. Since the system outputs a normalized similarity value for both policies we will be able to calculate the difference of said values which will be used to order the scenarios by their capability of allowing and distinguishing collaborative and non-collaborative behaviours, from best (highest difference) to worst.
Results

Contents

5.1 Squary-Shappy .................................................. 29
5.2 Lab Recruits .................................................... 33
5.3 Lab Recruits: Users playtesting .............................. 37
5.4 Discussion ..................................................... 50
In this chapter, we present the results gathered from our experiments where we created three different scenarios in both environments, the Squary-Shappy simulation and the Lab Recruits game, and calculated the likelihood of the optimal behaviour of the two agent architectures belonging to their own policies. With this comparison we were able to infer on the scenarios ability to allow and distinguish the two behaviours and with which difference. We also put forward the results from our user playtesting experience where we used the responses from the questionnaires answered and tried to compare the users behavioural traces to verify if our agents policies can be considered representations of human-like behaviours.

5.1 Squary-Shappy

We began our experiments using our own custom simulator, Squary - Shappy. Like we previously explained, in this environment the objective is for two agents to roam around the map and eat the all the existing food.

5.1.1 Scenarios description

We created three different maps with the purpose of evaluating designs that may induce the characters to play in different ways. The maps created for this work and for this environment were all created in a squared shape and with minimal obstacles to restricts the movement. They were the following:

![Figure 5.1: Squary - Shappy: Scenario 1 “Advantage red”](image)

The Squary-Shappy Scenario 1 titled "Advantage red", depicted in Figure 5.1, was designed to simulate a map where both the characters and the uneven number of different food objects to eat are initially positioned in a way that give a small advantage to the red character since it can easily reach two of the three total food objects before the blue character, but in order to do so it will also have to travel a longer
distance where as the blue agent can collect two food objects with a lesser number of total steps. This map also has a small movement obstacle inside the room.

Figure 5.2: Squary - Shappy: Scenario 2 “Perfect division”

Scenario 2 title “Perfect division”, which is illustrated in Figure 5.2, was created to test a map that simulates a completely even and perfect division. The scenario is equally divided, an even number of food objects is provided which are also spread evenly in each corner. The characters initial position gives none of the agents the advantage since both have the same distance to travel in order to eat all four food objects. Both agents also have the quickest path to two different food objects.

Figure 5.3: Squary - Shappy: Scenario 3 “Center focus”

Titled “Center focus”, Scenario 3, depicted in Figure 5.3, was made to, similarly to Scenario 2, test a map with an even design since a total of four food objects were also created but the main difference was the positioning. All four food objects were placed in the center of the map and the characters in opposite corners. With this initial deployment both agents have to travel the same distance to reach the food objects. One other twist is the order of food objects which the agents choose to eat, one “bad” move and they can end up eating only one of the food objects.
5.1.2 Automated agents experiences

In order to understand if the scenarios could differentiate the DG and the ¬DG, meaning playing with a collaborative and non-collaborative behaviour, we used the Centralized and Individual agents optimal behaviours that resulted by solving each of the three scenarios and using the voting system, explained in Section 4.4, calculated the likelihood of those behaviours belonging to each of the two policies, Centralized and Individual. These optimal behavioural traces are referent to the agents behaviour while following each policy optimal action\(^1\) in each game state. They can be recorded by simply observing the sequence of state-action pairs that the agents make during a game run. Here is an example of a said state-action pair:

\[
< (2, 5), (3, 8), [3, 4], [2, 6], [6, 8], (\text{Down}, \text{Left}) >
\]

Where \((2, 5)\) refers to the first agent’s position, \((3, 8)\) to the second agent’s position, \([3, 4], [2, 6]\) and \([6, 8]\) are referring to the positions of the food objects still not eaten and \((\text{Down}, \text{Left})\) represents the actions made by both agents, in this example agent1 moved down and agent2 moved left.

The results gathered were the following.

\[
\begin{array}{l}
\text{SS Centralized agents optimal behaviour} \\
\hline
\text{Scenario 1} \quad \text{"Advantage red"} & 0,5 & 0,5 \\
\text{Scenario 2} \quad \text{"Perfect division"} & 0,5 & 0,5 \\
\text{Scenario 3} \quad \text{"Center focus"} & 0,497 & 0,503 \\
\hline
\end{array}
\]

![SS Centralized agents optimal behaviour](image)

**Figure 5.4:** Results: Squary-Shappy Centralized agents optimal behaviour

Figure 5.4 depicts the results for the centralized agents optimal behaviour. The green columns are representative of the likelihood values of the centralized agents optimal behaviour belonging to the Centralized policy, whereas the orange columns represents the value of likelihood of the same behaviour belonging to the Individual policy. We can observe that for both Scenario 1 “Advantage red”

\(^1\)An optimal action is the action that always maximizes the reward of the current state.
and Scenario 2 “Perfect division” both policies received the same likelihood value (0.5), meaning that the system could not differentiate if the centralized agents optimal behaviour was similar to the Centralized or the Individual policy. For Scenario 3 “Center Focus” we had an unexpected result since the system attributed a higher likelihood value (0.503) to the Individual policy and a lower value (0.497) to the Centralized policy, meaning that, according to the system the centralized agents optimal behaviour, although with a small difference, was considered more similar to the Individual policy.

The results for the individual agents optimal behaviour can be seen in Figure 5.5. The green columns are again representative of the likelihood values that the individual agents optimal behaviour belongs to the Centralized policy while the orange columns represent the likelihood that the same behaviour is more similar to the Individual policy. For this behaviour we can observe that the system accurately identified them as more similar to the Individual policy in all scenarios. We can also detect that the closer likelihood values, meaning that it was the hardest to distinguish, are from Scenario 2 (0.499 to the Centralized policy and 0.501 to the Individual policy) were both policies generated an almost equal behaviour and. These lower values were expected since Scenario 2 “Perfect division” was designed to be the hardest map to differentiate the two behaviours.

Figure 5.6 shows us the likelihood differences\(^2\) in all scenarios. From this figure we extrapolated the order of the scenarios that allow and better distinguish collaborative and non-collaborative behaviours. With the largest likelihood difference value of 0.032 Scenario 3 “Center focused” is considered the best scenario to differentiate the two behaviours, followed by Scenario 1 “Advantage red” with 0.024 and finally Scenario 2 “Perfect division” with 0.002.

\(^2\)These were calculated by summing the difference between the likelihood values of the centralized agents optimal behaviour and the likelihood values of the individual agents optimal behaviour belonging to the Centralized and Individual policy. Example: For Scenario 1, \((0.503-0.497) + (0.519-0.481) = 0.024\)
5.2 Lab Recruits

Moving to the Lab Recruits game where the characters had to click on a sequence of buttons to open new rooms and reach the green target buttons, we again present the maps designed for this environment, the results from the automated agents as well as the results from our user playtesting experience.

5.2.1 Scenarios description

Aiming at having maps that provide interesting scenarios to explore and analyse, the three maps created for the Lab Recruits experiments were designed to have higher complexity than the ones in the Squary-Shappy since the maps had larger size, had several movement obstacles such as corridors and doors as well as due to agents having one more individual action to account for. We now present the said maps.
Scenario 1 title “One man’s choice” and illustrated in Figure 5.7, was created to evaluate a scenario where the map was completely unbalanced designed. The map is composed by one green button (the objective), three red buttons and three doors. By observing the map from a top-down view we can see that one of the characters, the one on the right side of the image, has no way of leaving its initial room, while the other character, on the left side of the image, has total freedom regarding its options. He can either choose to quickly open the trapped characters door and allow him to reach the target button, or choose a more time and distance consuming set of actions of opening two different doors in order for himself to reach the objective.

![Figure 5.8: Lab Recruits: Scenario 2 “Even distribution”](image)

For Scenario 2 with the title “Even distribution”, depicted in Figure 5.8, we wanted to again test a map that was evenly designed and didn’t offered any advantage to either of the characters. This map was created with a total of two green buttons, meaning both buttons have to be clicked in order to complete the scenario, two red buttons and two doors. This map also gives both characters the option to either click their closest red button to reach their closest green button, or travel a larger distance through an open corridor to reach the other. While designing this map, we aimed at having a map with a similar thought process to the Squary-Shappy Scenario 2 “Perfect division”.

Figure 5.9 illustrates Scenario 3 titled “Random maze”. The map is composed of one green button, two red buttons and two doors. This scenario was created to represent a simple maze adding an extra layer of complexity to the characters movement. The characters are initially positioned in opposite sides of the map and since the sequence of buttons to click in order to reach the objective is easily identified and gives no apparent advantage to either character, we consider this as a design were playing either collaboratively or individually can lead to an extremely similar behaviour.
5.2.2 Automated agents experiences

Similar to Subsection 5.1.2, we wanted understand if the scenarios could differentiate the DG, collaborative behaviour, and the \( \neg DG \), non-collaborative behaviour. We again used the centralized and individual agents optimal behaviour when solving the three mentioned scenarios and using the voting system explained in Section 4.4 calculated the likelihood of those behaviours belonging to the Centralized and Individual policy. These optimal behavioural traces were obtained with the same format as for the real-life users, meaning they are composed by a sequence of state-action pairs equal to the one explained in Section 4.3.4, and were achieved by recording each agent type solving a scenario a following the optimal actions according to their policy in each individual state. We now present the results gathered.

Figure 5.10 shows the results regarding the centralized agents optimal behaviour. The green columns are again relative to the likelihood of the centralized agents optimal policy belonging to the Centralized policy, while the orange columns indicates the likelihood of the same behaviour being more representative of the Individual policy. Regarding Scenario 1 “One man’s choice” and Scenario 3 “Random maze”, we observed that the system accurately identified that the centralized agents optimal behaviour, for these scenarios, are more similar to the Centralized policy, with Scenario 3 having a major difference between both likelihood values. For Scenario 2 “Even distribution” we noticed that the likelihood values are closer to each other, as expected, but the system identified the centralized agents optimal behaviour as belonging to the Individual policy.

Regarding the results from the individual agents optimal behaviour, Figure 5.11, with the same representations for the green and orange columns, shows that the system was able, for all scenarios, to accurately indicate that the individual agents optimal behaviour was more likely to belong to the Indi-
By again creating a figure to display the likelihood differences on all scenarios to order them by their ability to allow and distinguish collaborative and non-collaborative behaviours, we obtained the results depicted in Figure 5.12. This figure shows a considerable difference between the scenarios, especially regarding Scenario 3 “Random maze” with a likelihood difference of 0.57, and, by coincidence, gives us the exact same scenario order as in the Squary-Shappy environment regarding the scenarios capability to distinguish collaborative and non-collaborative behaviour. The order, from best to worst, starts with Scenario 3 “Random maze”, followed by Scenario 1 “One man’s choice” with 0.032 and lastly Scenario 2 “Even distribution” with 0.008.
Like we previously mentioned, we made a playtesting experience where we asked users to play each of the Lab Recruits scenarios three times and to individually answer a questionnaire regarding their behaviour during the game. This experience was made to corroborate the results gathered from our approach regarding the order of each scenario can better distinguish collaborative and non-collaborative behaviours while also aimed at collecting the users behavioural traces to determine if their behaviours were more likely to belong to the Centralized or Individual policies. Comparing these likelihood values to the users answers on each scenario’s question 1 we hope to determine if our agent architectures can be used to accurately simulate human-like behaviour. We receive 13 playtesting experiences and questionnaires responses, totaling 26 individual participants.

The questionnaire consisted on three questions for each scenario:

1. When playing Scenario X, did you feel that you played as a team, played in an individually manner or neither? - The answers to this question, when combined to the likelihood values of our comparison system, allowed us determine if our policies were simulating human-like behaviour.

2. Did you had any communication while playing Scenario X? If so, what was it about? - To verify if one of the main playing conditions of no communication was followed.

3. When playing Scenario X, describe your approach. Were you just trying to finish it as fast as possible? Were you exploring the map? - To understand if the users behavioural trace were made with the same objective as the agents, meaning to finish as fast as possible.

In the end we also asked users to order the scenarios, from best to worst, on which they felt promoted collaborative behaviour the most.
5.3.1 Scenario 1

Splitting our evaluation of the playtesting results by scenario, for Scenario 1 titled “One man’s choice” and depicted in Figure 5.7, we present the user answers to each of the questionnaire questions as well as the likelihood values of each user-pair behavioural trace belonging to each policy.

<table>
<thead>
<tr>
<th>Users ID</th>
<th>Player 1 - Question 1</th>
<th>Player 2 - Question 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>6c1of1ey8o</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>1s9ub0ud5e</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>4x8ug2os9u</td>
<td>Played individually</td>
<td>Played individually</td>
</tr>
<tr>
<td>3d3if7ec6e</td>
<td>Played as a team</td>
<td>Played individually</td>
</tr>
<tr>
<td>3x8ux7iq6e</td>
<td>Played individually</td>
<td>Played as a team</td>
</tr>
<tr>
<td>9r7iy8ax1i</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>3x8eg9eq1u</td>
<td>Played individually</td>
<td>Played individually</td>
</tr>
<tr>
<td>263uv1us2e</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>0s9ol1aw2u</td>
<td>Played as a team</td>
<td>Played individually</td>
</tr>
<tr>
<td>1r5uf20n9e</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>2b1oti1en9i</td>
<td>Played individually</td>
<td>Played individually</td>
</tr>
<tr>
<td>1b8e6b7e</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>8d3x3uy9o</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
</tbody>
</table>

Figure 5.13: Scenario 1: Users question 1 answers

Figure 5.13 shows us the unprocessed answers from users in question 1. By calculating the percentages of the possible answers, based on the total of twenty-six individuals, we determined that, for this scenario, 69% of the users responded that they played as a team while the remaining 31% answered that they played individually. One interesting observation came from the user-pairs with IDs “3d3if7ec6e” and “0s9ol1aw2u” where both players 1 responded that he played as a team and both players 2 answered it saying they played individually. This result was again repeated in the user-pair with ID “3x8ux7iq6e” where Player 1 claimed it played individually and Player 2 says it played in team manner, meaning that out of thirteen user-pairs there were 54% that agreed that they played a team, 23% agreed that they played individually and the 23% had mixed answers regarding their behaviours (one user played individually and the other as a team).

<table>
<thead>
<tr>
<th>Users ID</th>
<th>Player 1 - Question 2</th>
<th>Player 2 - Question 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>6c1of1ey8o</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>1s9ub0ud5e</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4x8ug2os9u</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3d3if7ec6e</td>
<td>Get to the button!</td>
<td>Yes, to get to the button</td>
</tr>
<tr>
<td>3x8ux7iq6e</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>9r7iy8ax1i</td>
<td>Yes, about who is opening the door first</td>
<td>No</td>
</tr>
<tr>
<td>3x8eg9eq1u</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>263uv1us2e</td>
<td>We didn't</td>
<td>No</td>
</tr>
<tr>
<td>0s9ol1aw2u</td>
<td>Yes, about who needs to open the door first</td>
<td>No</td>
</tr>
<tr>
<td>1r5uf20n9e</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2b1oti1en9i</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>1b8e6b7e</td>
<td>sim, colaboração na abertura de portas</td>
<td>sim, na abertura de portas</td>
</tr>
<tr>
<td>8d3x3uy9o</td>
<td>sim, abertura de portas</td>
<td>sim, abertura de portas</td>
</tr>
</tbody>
</table>

Figure 5.14: Scenario 1: Users question 2 answers
For question 2, Figure 5.14 we calculated that 69% of the twenty-six total users said they didn’t communicate and the 31% said they did. We did however had mixed responses from the user-pairs with IDs “9r7iy8ax1i” and “0s9ol1aw2u” where both players 1 said that they indeed communicated and both players 2 responded that they didn’t which raised some concerns. By combining the answers given by users in this question resulted in 62%, out of the thirteen user-pairs, agreeing that they didn’t had communication, 23% user-pairs agreeing that they did indeed communicated and the remaining 15% had conflicting answers.

<table>
<thead>
<tr>
<th>Users ID</th>
<th>Player 1 - Question 3</th>
<th>Player 2 - Question 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>6c1of1ey8o</td>
<td>Finish as fast as possible</td>
<td>Finish as fast as possible</td>
</tr>
<tr>
<td>199ub0d5e</td>
<td>As fast as possible</td>
<td>As fast as possible</td>
</tr>
<tr>
<td>4x8ug2es9u</td>
<td>Just figure it out and tried to finish as fast as possible</td>
<td>Just figure it out and tried to finish as fast as possible</td>
</tr>
<tr>
<td>3d3f7ec65e</td>
<td>Just finish</td>
<td>Just finish</td>
</tr>
<tr>
<td>3s8u7ed9e</td>
<td>Exploring the map</td>
<td>Fast as possible</td>
</tr>
<tr>
<td>97jy8ex11</td>
<td>Finishing as fast as possible</td>
<td>Finishing as fast as possible</td>
</tr>
<tr>
<td>9s8q9dq1u</td>
<td>Exploring the map</td>
<td>Exploring the map</td>
</tr>
<tr>
<td>2e3xv1ux2e</td>
<td>Finish it as fast as possible</td>
<td>I was trying to finish fast as possible</td>
</tr>
<tr>
<td>0x9ol1aw2u</td>
<td>Finishing as fast as possible</td>
<td>Finish as fast possible</td>
</tr>
<tr>
<td>1f5u2ex9a</td>
<td>Finishing as fast as possible</td>
<td>Fast as possible</td>
</tr>
<tr>
<td>2b10y1m9i</td>
<td>A explorar o mapa</td>
<td>Finish fast as possible, not exploring the map</td>
</tr>
<tr>
<td>1h80w6d7e</td>
<td>rápido som explorar</td>
<td>rápido som explorar</td>
</tr>
<tr>
<td>8d3x3uy9o</td>
<td>Rapidez</td>
<td>Rapidez</td>
</tr>
</tbody>
</table>

Figure 5.15: Scenario 1: Users question 3 answers

Regarding question 3, Figure 5.15, a majority of 85% of users said that they tried to finish the scenario as fast as possible while the remaining 15% acknowledged that they roamed around the map.

We know want to understand if the two architectures used, Centralized and Individual, can accurately represent human-like collaborative and non-collaborative behaviour. By joining the answers given in question 1, Figure 5.13 and calculating the likelihood values of each user-pair behavioural trace belonging to each policy, using the same voting system as in our automated agents experiences, we got Figure 5.16.

In this figure we combined the users Scenario 1 question 1 answers and the likelihood values of their behavioural traces belonging to each policy, with the green column representing the likelihood of belonging to the Centralized policy and the orange to the Individual policy. We labeled three IDs with “Mixed behaviour” since their users had different responses whereas the rest of the user-pairs were identified by either “Played as a team” or “Played Individually” since both players answered the same play style while playing Scenario 1. By analysing the results, we observe that out of seven total user-pairs that answered that they played as a team, our likelihood system was only able to accurately identify three as belonging to the Centralized policy. Regarding the three user-pairs that acted individually, the system labeled only only one of them as more likely to belong to the Individual policy. We also identified that the user-pairs that had a mixed responses regarding their type of play were all categorized by the system as having a behavioural trace more likely to belong to the Centralized policy.
The previously mentioned results, depicted in Figure 5.16, showed us that our comparison system had ≈ 43% accuracy when connecting the Centralized policy to supposed collaborative behaviours and accuracy of ≈ 33% when identifying as more similar to the Individual policy the supposed non-collaborative behaviours. But using only this information, can we already conclude if the Centralized and Individual architectures do or do not properly represent the DG, collaboration, and the ¬DG, non-collaboration? What if the problem was that we used a wrong comparison system to attribute the likelihood values to each policy? Or what if the users had different understandings of what collaboration means? In order to try and understand if these questions had relevance, we undertook a second and different comparison between the users behaviours and both policies.

Considering that in these scenarios we can condense any behavioural trace by eliminating the path to each game button and only take into account the sequence of red and green buttons clicked, we analysed each users sequences of buttons clicked and manually compared them to each policy optimal sequence of buttons.

By analysing the game runs for each user-pairs and also the optimal sequence for each agent type we observed the following sequence of game buttons $^3$.

Figures 5.17, 5.18 and 5.19 show us the full sequence of game buttons clicked by each user-pair along with their combined responses in question 1.

$^3$The data regarding the user-pairs was divided in three figures to allow a better visualization. Additionally, for visual identifica-

<table>
<thead>
<tr>
<th>User-Pair</th>
<th>Centralized policy likelihood</th>
<th>Individual policy likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>6c1of1ey80</td>
<td>0.383</td>
<td>0.617</td>
</tr>
<tr>
<td>1s9ub00ud5e</td>
<td>0.484</td>
<td>0.516</td>
</tr>
<tr>
<td>4x8uq2os9u</td>
<td>0.503</td>
<td>0.497</td>
</tr>
<tr>
<td>3d3if7ec6e</td>
<td>0.503</td>
<td>0.497</td>
</tr>
<tr>
<td>3x8ux7iq6e</td>
<td>0.504</td>
<td>0.496</td>
</tr>
<tr>
<td>9r7iy8ax1i</td>
<td>0.503</td>
<td>0.497</td>
</tr>
<tr>
<td>3s8ig9i1q1u</td>
<td>0.503</td>
<td>0.497</td>
</tr>
<tr>
<td>2d3uv1us2e</td>
<td>0.505</td>
<td>0.495</td>
</tr>
<tr>
<td>0s9ol1aw2u</td>
<td>0.507</td>
<td>0.493</td>
</tr>
<tr>
<td>1r5uf2on9a</td>
<td>0.503</td>
<td>0.497</td>
</tr>
<tr>
<td>2b1oy1in9i</td>
<td>0.307</td>
<td>0.693</td>
</tr>
<tr>
<td>1h8ew6ib7e</td>
<td>0.387</td>
<td>0.613</td>
</tr>
<tr>
<td>6d3ix3uy9o</td>
<td>0.422</td>
<td>0.577</td>
</tr>
</tbody>
</table>
In order to evaluate the users sequences to understand if they had the same sequence of buttons but different question 1 answers we “cleaned” said sequences from the following redundancy heuristics:

- One character clicking on the same button an uneven number of times in a row (e.g. three) is equivalent to clicking only one time.
- One character clicking on the same button an even number of times in a row (e.g. two) is equivalent to not clicking at all.

Figure 5.20 shows the optimal sequence of buttons for both agent types. Agents with the centralized architecture solved the scenario by character1 pressing button2 and character2 pressing button4. For the agents with the Individual policy, the scenario was solved by character1 pressing button1, button3 and button4.

In order to evaluate the users sequences to understand if they had the same sequence of buttons but different question 1 answers we “cleaned” said sequences from the following redundancy heuristics:

- One character clicking on the same button an uneven number of times in a row (e.g. three) is equivalent to clicking only one time.
- One character clicking on the same button an even number of times in a row (e.g. two) is equivalent to not clicking at all.

Figure 5.20: Scenario 1 Centralized and Individual agents optimal sequence of buttons
Figures 5.21, 5.22 and 5.23 show us the treated sequence of game buttons clicked by each user-pair along with their combined responses in question 1. By using Figure 5.20 as reference, we identified with green colour the elements from the same sequence as the Centralized agents optimal button sequence and with orange the elements from the same sequence as the Individual agents optimal button sequence. With this color separation we were able to identify which agent architecture optimal sequence each user-pair button sequence was more similar to. With this manual comparison of the buttons sequence we were able to observe that most user-pairs, nine out of thirteen, had the exact same button sequence as the Centralized agents and also that, from these nine user-pairs, five of them answered that they either played individually or had mixed answers. We also noticed that the user-pairs with ID “6c1of1ey8o” and “6d3xi3uy9o” both had join answers of playing as a team when in fact they had a more similar button sequence to the Individual agents type and we can also notice that character1 was the only one “working”, meaning clicking buttons to solve the scenario.
5.3.2 Scenario 2

We now present the users responses regarding the questionnaire with its correspondent analysis as well as the likelihood comparisons between the users behavioural traces and each of the two policies for Scenario 2 titled “Even distribution” and depicted in Figure 5.8.

<table>
<thead>
<tr>
<th>Users ID</th>
<th>Player 1 - Question 1</th>
<th>Player 2 - Question 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>6c10f1ey8o</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>1s9ub0ud5e</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>4x8u920s5u</td>
<td>Played individually</td>
<td>Played individually</td>
</tr>
<tr>
<td>3d3f7ec6e</td>
<td>Played individually</td>
<td>Played individually</td>
</tr>
<tr>
<td>3x8u7q9e</td>
<td>Played individually</td>
<td>Played individually</td>
</tr>
<tr>
<td>9r7y8ax1i</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>3s8p9q9u</td>
<td>Played individually</td>
<td>Played individually</td>
</tr>
<tr>
<td>2d3u1us2e</td>
<td>Played individually</td>
<td>Played as a team</td>
</tr>
<tr>
<td>0s9o1ua2u</td>
<td>Played individually</td>
<td>Played as a team</td>
</tr>
<tr>
<td>1r5u06q9e</td>
<td>Played individually</td>
<td>Played individually</td>
</tr>
<tr>
<td>2b1oy1sn9</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>1b8o6b7e</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
<tr>
<td>8d3x3uy9o</td>
<td>Played as a team</td>
<td>Played as a team</td>
</tr>
</tbody>
</table>

Figure 5.24: Scenario 2: Users question 1 answers

Scenario 2, like we previously mentioned, had the most split and evenly distributed design of all three Lab Recruits Scenarios, which was also corroborated by the results from our automated agents experiences with the lowest difference in the likelihood values. In Figure 5.24, regarding question 1 of “When playing Scenario 2, did you feel that you played as a team, played in an individually manner or neither?” we can see that the users had an almost equally balanced responses. 54% of the users responded that they played as a team where as the remaining 46% of the answers said that they played individually. This again goes along to what we previously mentioned that, due to the design of this scenario, it is hard to differentiate the two behaviours. By combining the answers given by each user we can observe that out of thirteen user-pairs 46% had matching answers regarding playing as a team, 39% agreed that they both played individually and the remaining 15% of the user-pairs had mixed behaviours.

<table>
<thead>
<tr>
<th>Users ID</th>
<th>Player 1 - Question 2</th>
<th>Player 2 - Question 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>6c10f1ey8o</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>1s9ub0ud5e</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4x8u920s5u</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3d3f7ec6e</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3x8u7q9e</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>9r7y8ax1i</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3s8p9q9u</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2d3u1us2e</td>
<td>We didn’t</td>
<td>No</td>
</tr>
<tr>
<td>0s9o1ua2u</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>1r5u06q9e</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2b1oy1sn9</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>1b8o6b7e</td>
<td>sim, colaboração na abertura de portas</td>
<td>sim, na abertura de portas</td>
</tr>
<tr>
<td>8d3x3uy9o</td>
<td>sim, abertura de portas</td>
<td>sim, abertura de portas</td>
</tr>
</tbody>
</table>

Figure 5.25: Scenario 2: Users question 2 answers
By analysing Figure 5.25 we determined that 85% of the total users said that they didn’t communicated while the remaining 15% did but, in this scenario no user-pairs had conflicting answers. With these low percentages of communication we believe that it shows that for this scenario a more implicit type of collaboration, meaning no explicit communication and planning, was used by the users that answered that they played has a team.

![Figure 5.26: Scenario 2: Users question 3 answers](image)

Regarding question 3, Figure 5.26 shows that a high value, 92%, of the users said they tried to finish as fast as possible while the remaining 8% answered that they roamed around the map.

![Figure 5.27: Scenario 2: Users question 1 answers and policy likelihood values](image)
Figure 5.16 depicts the combined answers by users in the first question for Scenario 2 “Even distribution” combined with the likelihood values of the user behavioural traces belonging to each policy calculated with the system explained in Subsection 4.4 and used in our automated agents experiences. The green columns again represent the likelihood of a behavioural trace belonging to the Centralized policy and the orange columns to the Individual policy. By observing the Figure we can immediately notice that the differences between the likelihood values are extremely low, which was expected due to our results from the automated agents experiences, Subsection 5.2.2, and can even identify that four user-pairs behavioural traces had an equal likelihood of belonging to both polices. By comparing the higher likelihood values to the answers given by each user we determined that no supposed collaborative behaviours were more likely to belong to the Centralized policy and also that only two supposed individual behaviours were more likely to belong to the Individual policy. This can however be explained since all likelihood difference values are extremely low, which was somewhat expected due to our results from the automated agents and confirms that Scenario 2 has a lower capability of distinguishing behaviours.

By again analysing the user-pairs sequence of buttons for Scenario 2 and also for each agent type we obtained the following sequences.

<table>
<thead>
<tr>
<th>Centralized agents optimal sequence of buttons</th>
<th>Individual agents optimal sequence of buttons</th>
</tr>
</thead>
<tbody>
<tr>
<td>character1 -&gt; button1</td>
<td>character1 -&gt; button1</td>
</tr>
<tr>
<td>character2 -&gt; button2</td>
<td>character2 -&gt; button2</td>
</tr>
<tr>
<td>character1 -&gt; button3</td>
<td>character1 -&gt; button3</td>
</tr>
<tr>
<td>character2 -&gt; button4</td>
<td>character2 -&gt; button4</td>
</tr>
</tbody>
</table>

**Figure 5.28:** Scenario 2 Centralized and Individual agents optimal sequence of buttons

Figure 5.20 shows the optimal sequence of buttons for each agent type, which was identical. Both agent types solved the scenario with character1 pressing button1, followed by character2 pressing button2, character1 pressing button3 and lastly character2 pressing button4.

<table>
<thead>
<tr>
<th>Users ID</th>
<th>Question 1 answers</th>
<th>Sequence of buttons</th>
</tr>
</thead>
<tbody>
<tr>
<td>6stof1ey8o</td>
<td>Played as a team.</td>
<td>character1 -&gt; button1, character1 -&gt; button3, character2 -&gt; button2, character2 -&gt; button4</td>
</tr>
<tr>
<td>1rs9ubj15se</td>
<td>Played as a team.</td>
<td>character2 -&gt; button2, character1 -&gt; button3, character2 -&gt; button4</td>
</tr>
<tr>
<td>4x8uy20s9hu</td>
<td>Played individually</td>
<td>character2 -&gt; button2, character1 -&gt; button3, character2 -&gt; button4</td>
</tr>
<tr>
<td>3d3f7se9se</td>
<td>Played individually</td>
<td>character1 -&gt; button1, character1 -&gt; button3, character2 -&gt; button2, character2 -&gt; button4</td>
</tr>
<tr>
<td>3s5ax7jqbe</td>
<td>Played individually</td>
<td>character1 -&gt; button1, character1 -&gt; button3, character2 -&gt; button2, character2 -&gt; button4</td>
</tr>
</tbody>
</table>

**Figure 5.29:** Users: Scenario 2 Question 1 answers and sequence of buttons - Part 1

Figures 5.29, 5.30 and 5.31 show us the treated sequences, with regards to the two previously mentioned redundancy heuristic, of game buttons clicked by each user-pair along with their combined responses in question 1.

---

4For visual identification of the game components such as “button2” or “character1” in Scenario 2, check Figure 7.9 of the Appendix.
Observing the users sequence of buttons we noticed that they consist on exactly the same characters clicking on the same buttons but in different orders. One remark we can make regarding these different orders is that they can be deemed insignificant by simply analysing the Scenario 2 map design and the distance each character has to the buttons clicked. From these results and since the both automated agents button sequences are also identical we can once again confirm that in this scenario, due to its design, it is difficult to distinguish behaviours.

5.3.3 Scenario 3

Titled “Random maze” and depicted in Figure 5.9, Scenario 3 had the better results in our automated agents experiences regarding distinguish behaviours, since it showed to have a much higher likelihood difference value when compared to the other two scenarios. We now present the answers given by each users in the questionnaire regarding Scenario 3 as well as the comparisons between the two policies, Centralized and Individual, and the users behavioural traces.

Analysing the results for the first question regarding Scenario 3 shown in Figure 5.34, we observed that out of the total twenty-six individual users, 77% of them responded to have played with a team behaviour while the remaining 23% responded that they played individually. We do however have an interesting remark to make since no user-pairs had any disagreement regarding the way they played, making it a first in all scenarios and that goes according to the results from our experiences with the automated agents, were we determined that Scenario 3 was the easiest from where to distinguish behaviours.

For the second question in Scenario 3 we asked users if they any any type of communicating while
playing. Figure 5.33 depicts those answers. 77% of the users said that they did not communicate while the remaining 23% did. One observation was that the user-pairs with ID “9r7iy8ax11” again had different responses regarding if they communicated, meaning that 70% of the thirteen user-pairs agreed that there was no communication, 23% did answer that there was communication and 8% had mixed answers.

For the last question in Scenario 3, the results shown in Figure 5.34 indicate that again 77% of the
users said that they played to finish as fast as possible whereas the other 23% of the users roamed around the map while playing.

For Scenario 3 “Random Maze” we once again combined the answers given by users in the first question with the likelihood values of each user behavioural trace belonging to each policy, calculated with the system explained in Subsection 4.4, in Figure 5.35 where the green columns again represent the likelihood of a behavioural trace belonging to the Centralized policy and the orange columns to the Individual policy. Contrary to the results from our experiences with the automated agents where this scenario proved to be the easiest to differentiate the two behaviours, collaborative and non-collaborative, in this playtesting experience, our system wasn’t able to attribute a higher Centralized likelihood value to any behavioural trace, meaning that when connecting the Centralized policy to the supposed collaborative behaviours and the Individual policy to the supposed non-collaborative behaviours in this scenario the system had an accuracy of $\approx 23\%$. These results raised concerns regarding the likelihood system when comparing the traces to the Centralized policy and reinforced the question regarding if likelihood comparison system used is the most adequate.

By once again performing the manual comparison of the sequence of buttons clicked between the users and the two agent types in Scenario 3, we got the following results.

---

5For visual identification of the game components such as “button2” or “character1” in Scenario 3, check Figure 7.10 of the Appendix.
Figure 5.36: Scenario 3 Centralized and Individual agents optimal sequence of buttons

Figure 5.36 shows the optimal sequence of buttons for both the centralized and individual agents. Agents with the centralized architecture solved the scenario by character1 pressing button1 followed by also pressing button2 and ultimately character2 pressing button3. For the Individual agents the scenario was solved by character1 pressing button1 and character2 pressing button2 and also pressing button3.

<table>
<thead>
<tr>
<th>Centralized agents optimal sequence of buttons</th>
<th>Individual agents optimal sequence of buttons</th>
</tr>
</thead>
<tbody>
<tr>
<td>character1 -&gt; button1</td>
<td>character1 -&gt; button1</td>
</tr>
<tr>
<td>character1 -&gt; button2</td>
<td>character2 -&gt; button2</td>
</tr>
<tr>
<td>character2 -&gt; button3</td>
<td>character1 -&gt; button3</td>
</tr>
</tbody>
</table>

Figure 5.37: Users: Scenario 3 sequence of buttons and policy similarity - Part 1

Figure 5.38: Users: Scenario 3 sequence of buttons and policy similarity - Part 2

Figure 5.39: Users: Scenario 3 sequence of buttons and policy similarity - Part 3

Figures 5.37, 5.38 and 5.39 shows us the treated sequence of game buttons clicked by each user-pair along with their combined responses in question 1. By using Figure 5.36 as reference, we again identified with green the elements from the same sequence as the Centralized agents optimal button sequence and with orange the elements from the same sequence as the Individual agents optimal button sequence. By observing the figures regarding the users button sequence we can identify that contrary to the results from Figure 5.35 we can identify that ≈ 54% of the thirteen user-pairs had more similarity to the Centralized policy. One other observation is that all of the five user-pairs that we could not assign (N/A) a more similar policy to, all had the same behaviour, which once again leads us to question that likelihood comparison system may not be the most adequate for this type of comparison.
5.3.4 Scenario order

For the last question of the questionnaire we asked each individual user to order the scenarios by their ability of promoting collaborative behaviour. By analysing their answers we determined in this question users answered that Scenario 3 promoted the most, followed by Scenario 1 and lastly Scenario 2. This subjective users order corroborates the scenario order achieved in our automated agents experiences, Subsection 5.2.2, which provides an indication that this approach can accurately test and order scenarios with a similar complexity and dynamic to the ones we used in this work, by their capability of allowing and differentiating collaborative and non-collaborative behaviours, in scenarios with a similar complexity and dynamic to the ones we tested.

5.4 Discussion

By analysing the results presented in the previous sections of this chapter regarding our automated agents, we identified some interesting points. First, when analysing the probability values for the agents optimal behaviour in both environments, we can notice that the centralized agents optimal behaviours in Scenario 3 of Squary-Shappy and Scenario 2 of Lab Recruits were given a higher probability of belonging to the Individual policy, which although it was with a small difference, it wasn’t expected and started to give us the initial sense that the comparison system in Section 4.4 may not be the most adequate for this approach. Secondly, as expected from their designs, the system calculated a lower difference between the probability values of each policy in the both scenarios 2 of the environments. Lastly, by comparing the likelihood differences in the Squary-Shappy environment and in the Lab Recruits game, we verified that the values in the Lab Recruits game are higher, which directly connects to a higher ability to distinguish both behaviours. This difference in values for both environments can be attributed to the larger map size and complexity as well as the increased action space in the Lab Recruits game.

Moving to the results from our user playtesting experience with the Lab Recruits game where we aimed at verifying if our agents could be deemed as representatives of a human-like behaviour for the Lab Recruits game. From the results in Figures 5.16, 5.27 and 5.35 we determined that our system could not accurately associate the Centralized policy to the behavioural traces where users claimed to be playing as team and the Individual policy to users claiming to have played individually. There can be several possible contributions for this. The first one is regarding the playtesting experience. We receive a total of 13 responses which is considerably small sample size and since the experience was made open for anyone willing to play, online and anonymously, we could be dealing with a diverse group of people regarding their age, computer dexterity, gaming background, among others, which can impact the results. One other factor in relation to the playtesting experience was that the automated agents and some of the users, based on their responses to the questions 2 and 3 in each scenario, had different play
styles, an example of this was that agents were focused on finish as fast as possible and some of the users responded that they took their time while roaming the map. Another contribution for these results can be attributed to the likelihood comparison system explained in Section 4.4. This comparison system, that showed some unexpected results in the automated agents experiences, had bad accuracy levels in the users experiments. The prime example of this was in Figure 5.35 where not one behavioural trace was deemed as more likely to belong to the Centralized policy. Both these possible factors motivated us into performing one other comparison. This form of manual comparing the sequence of buttons clicked, although rudimentary, demonstrated that users may have different definitions of what it means to work collaboratively as well as, referring to Scenario 3, most users actually had a sequence more similar to the Centralized policy, strengthening the our believe that other forms of automatic comparison should be tested in order to accurately determine if the Centralized and Individual policies represent human-like behaviour.

A positive result, like explained in Subsection 5.3.4, was the corroboration of the scenario order calculated by our system. This means that even though we couldn’t determine if our automated agents have the best representation of human-like behaviour, our system was successful in ordering the scenarios by their capability of allowing and distinguishing collaborative and non-collaborative behaviour.
6

Conclusions

Contents

6.1 Limitations .................................................. 53
6.2 Future work ................................................... 53
Aiming at evaluating designed scenarios on their capability of allowing and distinguishing collaborative and non-collaborative behaviours, we developed two types of automated agents to represent the Design Goal, centralized agents, and the non-Design Goal, individual agents, in different scenarios from the Squary-Shappy simulator and the Lab Recruits game. Upon training said agents we used a classifier to compare each agents type optimal behaviour when solving the scenarios to both policies in order to give a probability value of which policy a behavioural trace was more similar to. This comparison allowed us to observe in each scenario if we could differentiate the two behaviours and also order the scenarios by their ability of distinguishing collaborative and non-collaborative behaviours, starting at the highest average probability difference to the lowest. For the Lab Recruits game we also had a playtesting experience with real-life users where, although we could not accurately connect each users behavioural trace to our Centralized and Individual policies, we did manage to corroborate our scenario order using the users subjective evaluation from the questionnaires answered.

Although this work proved to have some limitations, the results gathered give us an indication that this approach of using definitions of a Design Goal and a non-Design Goal as behavioural traces from automated agents provides positive results and allows to order the scenarios by their ability to allow and distinguish different behaviours. We believe that this is evidence that this approach can provide developers with initial data regarding their scenarios during the development process.

6.1 Limitations

Like previously mentioned we encountered several limitations in this work.

The first one, and most referred in the chapter 5 is regarding the comparison between the behavioural traces and the two policies and we once again iterate that it is our believe that other methods of automatic comparison should be made. One other limitation is regarding the binary choices given to users in the questionnaire regarding whether they played as a team or individually. We now identify that since the approach provides a non-binary results (the likelihood values can vary) we should have given the same opportunity for the users, meaning that we could have used a Likert scale to accommodate for different collaboration "intensities".

6.2 Future work

Taking into account all the work developed for this dissertation there are several possibilities for future work.

Although the behaviours from our simple RL agents and architectures proved to be limited, they still provided some positive results when analysing the scenarios and we believe that a more accurate defi-
nition of the Design Goal and the non-Design Goal can be achieved using other methods. An example of this is a combination of Reinforcement Learning with Imitation Learning where a group of experts can provide general traces of the $DG$ and the $\neg DG$ to more accurately represent human-like behaviour. Another interesting possibility is the combination of this approach to the analysis of agents and users internal state. This could allow developers to not only determine if their scenarios are being developed according to the design goals but also allow them to have an indication of which areas/components of the scenarios are affecting the agents and users decision making\(^1\).

To finalize, we would like to state that is also our believe that this approach is not limited to collaboration and could be used to analyse other types of behaviours as long as definitions of the DG and the $\neg DG$ can be properly provided and the resulting RL behavioural traces and polices accurately compared.

\(^1\)A research related to the users internal state is currently being made in the scope of the iv4XR project using emotional modelling.
Bibliography


Appendix
Figure 7.1: Questionnaire - Lab Recruits Main Menu with open Instructions tab

Figure 7.2: Playtesting - Lab Recruits Training Scenario Instructions
Figure 7.3: Playtesting - Lab Recruits Scenario1 gameplay
Collaboration analysis in a multi-player based simulation

My name is Bruno Carreira. I am a MSc student at Instituto Superior Técnico, and I need your help to complete my dissertation "Collaboration analysis in a multi-player based simulation" supervised by prof. Rui Prada and prof. Manuel Lopes.

We are interested in collecting information from real-life users to identify collaborative and non-collaborative behaviour in a game.

If you agree to participate, you will be playing locally with another person (it is a multi-player game). We will ask you to play four different scenarios (1 training map + 3 testing maps). After all the scenarios, we will ask each of you to complete a small questionnaire.

We will not collect any personal information, meaning that your participation is anonymous and you will not be identified at any stage. We will store some of the games data, such as the buttons you press and each character's position during the game and also your answers to the questionnaire.

You are free to stop your participation at any time by simply exiting the Lab Recruits game and closing this questionnaire.

Thank you for agreeing to participate in this playtesting experiment. Your answers are very important to us.

Please note that by proceeding to the questionnaire, you are giving your consent.

If you have any questions, do not hesitate to contact us.
E-mail: bruno.carreira2@isdtmail.com
Phone: 917395969

Google Forms

Figure 7.4: Playtesting - Questionnaire introductory page
Collaboration analysis in a multi-player based simulation - Lab Recruits game

You can download the Lab Recruits (for Windows ONLY) game from the following link: https://mega.nz/file/04xS05L1#8xRnKx6l-ctj0Z3b5p1yp@z4oi/0J3pyqTJgLe9R-91x8DybuK

Upon downloading, you just need to extract the .ZIP folder and you can run it!

INSTRUCTIONS:

In each map, the players will each control one character and can press on multiple buttons. In order to press a button the character must be on top of it.

Player 1 controls the character on the left side of the screen. Uses the WASD keys to move and the F key to press buttons.

Player 2 controls the character on the right side of the screen. Uses the Arrow keys to move and the Enter key to press buttons.

There are several red buttons and doors spread around the map. Each red button opens/closes only one door, but each door can be controlled by multiple red buttons.

The objective in all scenarios is to click on all the greens buttons as fast as you can. Each map may contain either one or several green buttons.

In order for you to get familiar with the maps, you will have three tries at each map!

ATTENTION: Avoid having any sort of communication while playing the scenarios.

The game must be played while connected to the internet, otherwise it won’t work!

After finishing all the scenarios in the Lab Recruits, please fill the following individual links:

Place your game ID here. *

If you closed the game before copying the ID, you can search for the ID in the file "Logs" inside the games directory.

Your answer

Figure 7.5: Playtesting - Questionnaire instructions page
Figure 7.6: Playtesting - Questionnaire Scenario1 questions
Figure 7.7: Playtesting - Questionnaire scenario ordering question

Figure 7.8: Lab Recruits Scenario 1 with game components identified
Figure 7.9: Lab Recruits Scenario 2 with game components identified

Figure 7.10: Lab Recruits Scenario 3 with game components identified