Technical Note PR-TN-2015/00194

Issued: 9/2015

Developing Realistic Virtual Patients for Simulation-Based Training of Health Coaches

Sascha Slisser
Philips Research Europe

Unclassified
Author’s address:  Sascha Slisser
  saschaslisser@gmail.com
Title: Developing Realistic Virtual Patients for Simulation-Based Training of Health Coaches

Author(s): Sascha Slisser

Reviewer(s): Aart v. Halteren

Technical Note: PR-TN-2015/00194

Abstract: Health coaching is a means by which those suffering from chronic illness are challenged to address their problems, and work towards a healthier lifestyle. Health coaching involves a coach who gives personal advice to their patients, and helps with setting and achieving goals to increase the quality of life of the patient. The health coaches use a personalized profile (MAY: More About You Profile) of their patient, and based on this profile, the coach supports and guides the patient in improving their ability of self-care. The MAY profile is created through a questionnaire that the patient fills in. The coaches are expected to use the information in the profile effectively, and change their coaching to fit the personal circumstances of the patient. The health coaches are not adjusted to training with literature and manuals, and role-playing with actual humans can be very costly. Using a simulation based method would be more suited to the style of learning, and is more cost effective.

The goal of this project is to create a system that can simulate patient behavior based on the MAY profile, in order to train the health coaches in applying it effectively. Using the feedback from a virtual patient avatar, coaches will learn how to use the profile effectively. This feedback given is dependent on the MAY properties of the avatar, the current scenario and the decisions made by the user.

The final result of the project is the creation of an avatar within a 3D environment, which can speak through a text-to-speech generator, and emote using facial expressions. This thesis describes the design of a prototype avatar. The systems itself, as well as the difficulties and findings made during this process are described in detail.

Conclusions: In this thesis report the design of a virtual patient avatar is discussed detail. The background domain includes the MAY profile designed by Philips, and motivational interviewing. The goal of the virtual patient avatar is to help health coaches use the MAY profile and motivational interviewing correctly, and effectively. This report focuses on the architectural design of the avatar, the design of how a coaching session between the health coach and avatar flows, and how every concept is implemented. The avatar is created through combining several techniques, such as a knowledge base, 3d virtual environment, 3d avatar model, use of facial animations, and text-to-speech. A combination of software is used; Drools (running on a PC) is used for the logic, and Unity 3d (running an an Android tablet) is used for the visuals; they communicate through network socket. The goal is to create a dynamic dialog system that can be easily adjusted and expanded in order to best train the health coaches. The prototype was presented to several domain experts, who have indicated that the prototype is a useful tool; potential future additions are also discussed. The next step is to implement more improvements, and test the system for use with actual health coaches.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong> Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem Statement</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Approach</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Thesis Structure</td>
<td>2</td>
</tr>
<tr>
<td><strong>2</strong> Background</td>
<td>3</td>
</tr>
<tr>
<td>2.1 MAY Profile</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Motivational Interviewing</td>
<td>3</td>
</tr>
<tr>
<td>2.2.1 Motivational Interviewing in the Avatar System</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Virtual Learning</td>
<td>4</td>
</tr>
<tr>
<td>2.3.1 Simulated Patients</td>
<td>4</td>
</tr>
<tr>
<td>2.3.2 Virtual Patients</td>
<td>5</td>
</tr>
<tr>
<td>2.3.3 Virtual Patient Implementation Choices</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Discussion</td>
<td>6</td>
</tr>
<tr>
<td><strong>3</strong> Requirements, Architecture and Tools</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Capturing System Requirements</td>
<td>7</td>
</tr>
<tr>
<td>3.1.1 Final Requirement List</td>
<td>7</td>
</tr>
<tr>
<td>3.1.2 Main-requirement - Improving of Coach Knowledge</td>
<td>8</td>
</tr>
<tr>
<td>3.1.3 Main-requirement - Simulate Realistic Behavior in Avatar</td>
<td>9</td>
</tr>
<tr>
<td>3.2 System Architecture</td>
<td>9</td>
</tr>
<tr>
<td>3.2.1 Concepts within Coach</td>
<td>10</td>
</tr>
<tr>
<td>3.2.2 Concepts within Avatar</td>
<td>10</td>
</tr>
<tr>
<td>3.2.3 Concepts outside of Avatar/Coach</td>
<td>11</td>
</tr>
<tr>
<td>3.3 Tools</td>
<td>11</td>
</tr>
<tr>
<td>3.3.1 Drools</td>
<td>12</td>
</tr>
<tr>
<td>3.3.1.1 Rete Algorithm</td>
<td>12</td>
</tr>
<tr>
<td>3.3.1.2 Node Sharing</td>
<td>13</td>
</tr>
<tr>
<td>3.3.1.3 Stateless and Stateful</td>
<td>14</td>
</tr>
<tr>
<td>3.3.2 Unity</td>
<td>14</td>
</tr>
<tr>
<td>3.3.3 Connection between Drools and Unity</td>
<td>15</td>
</tr>
<tr>
<td>3.3.4 Faceshift</td>
<td>16</td>
</tr>
<tr>
<td>3.4 Discussion</td>
<td>16</td>
</tr>
<tr>
<td><strong>4</strong> System Design</td>
<td>17</td>
</tr>
<tr>
<td>4.1 Interactive Session Design</td>
<td>17</td>
</tr>
<tr>
<td>4.1.1 Dialog Trees</td>
<td>23</td>
</tr>
<tr>
<td>4.1.2 System and Avatar Reaction</td>
<td>23</td>
</tr>
<tr>
<td>4.1.3 Emotion Value</td>
<td>23</td>
</tr>
<tr>
<td>4.2 Problem Scenario Design</td>
<td>24</td>
</tr>
<tr>
<td>4.2.1 Motivation</td>
<td>24</td>
</tr>
<tr>
<td>4.2.2 Confidence</td>
<td>24</td>
</tr>
</tbody>
</table>

© KONINLIJKE PHILIPS ELECTRONICS NV 2015
## 5 Implementation

5.1 Implementation of Connection between Drools and Unity .......................................... 27
5.2 Implementation of Java and Drools ................................................................................... 28
5.2.1 Java Application Socket ............................................................................................... 28
5.2.2 Knowledge Base ......................................................................................................... 29
5.2.3 Inpuiter and Outputter ............................................................................................... 30
5.2.4 Rules for Controlling Avatar Behavior ....................................................................... 33
5.3 Implementation of Unity .................................................................................................. 37
5.3.1 Unity Application Server-Socket ................................................................................... 38
5.3.2 Objects and Visual Environment ................................................................................ 39
5.3.2.1 GUI ...................................................................................................................... 40
5.3.3 Text To Speech .......................................................................................................... 41
5.3.4 Lip-sync ..................................................................................................................... 42
5.3.5 Implementation of Faceshift in Unity ......................................................................... 42
5.3.5.1 Animations ............................................................................................................ 42
5.3.5.2 Animation Transitions ......................................................................................... 44
5.4 Discussion ....................................................................................................................... 44

## 6 Evaluation

6.1 Areas of Improvement ...................................................................................................... 47
6.1.1 Avatar Behavior ......................................................................................................... 47
6.1.2 Problem Scenarios ...................................................................................................... 47
6.1.3 User Experience and Interface .................................................................................. 47
6.2 Implemented Improvements .......................................................................................... 48
6.3 Discussion ....................................................................................................................... 49

## 7 Conclusion

7.1 Summarization of Concepts to Avatar Realization ........................................................ 51
7.2 Strengths of the System .................................................................................................. 51
7.3 Future Improvements ..................................................................................................... 52

## References

A Summary First Meeting Domain Experts ........................................................................ 57
B Summary Second Meeting Domain Experts .................................................................... 63
C Dialog Trees ..................................................................................................................... 65
C.1 Explanation of Symbols .................................................................................................. 65
C.2 Healthy Diet Dialog Trees ............................................................................................. 67
C.2.1 Motivation - Scenario 1 - Weight Loss ...................................................................... 67
C.2.2 Motivation - Scenario 2 - Blood Pressure ................................................................ 75
C.2.3 Confidence - Scenario 1 - Past Experience ............................................................... 83
C.2.4 Confidence - Scenario 2 - Circumstances ................................................................. 92
C.2.5 Skills - Scenario 1 - Lacking Cooking Skills ........................................................... 101
C.2.6 Skills - Scenario 2 - Lacking Recipe Skills ............................................................... 109
1 Introduction

For patients suffering from chronic illnesses medical care is not the only important aspect in health care. The psycho-social well-being of these patients plays a crucial role in maintaining a high quality of life. Due to the growing demand in health care it is difficult to support these patients every step of the way, making it necessary to give them the tools to be self-sufficient in maintaining their own psycho-social well-being. By using health coaches, it becomes possible to support these patients while they improve their lifestyles independently.

The amount of patients that fall under the care of one coach is quite large; an average number of 50 patients per coach. This large number of patients can obviously lead to problems when trying to remember the personal circumstances of each patient. So the coaches require some support to make the coaching sessions more effective; reducing the time that is necessary to re-familiarize with each patient and achieve consistency between visits. This can be done through creating personal profiles of the patients, which allows for the coaches to have a quick overview of the personal circumstances regarding their patient. One of these profiles is called the More About You profile (MAY) developed by Philips. The goal of this project is to train the coaches to use these profiles effectively, allowing them to give personal care to each patient despite the large number of patients per coach.

1.1 Problem Statement

Domain experts, who often work with the health coaches, have indicated that the coaches have difficulties with familiarizing themselves with a MAY profile during their coaching activities. Giving the coaches a set of training manuals would most likely not prepare them enough for an actual session with a patient, and a more hands-on approach is much better suited to the learning style of the health coaches. Ideally, the coaches would be trained by acting out a coaching session with an actor who is capable of emulating patient behavior. The actor would react appropriately to the choices made by a coach, as well as explain any mistakes being made. This method gives a more stronger experience compared to written text; a study by Norman et al. (1982) has shown that the learning performance between training with actors and training with real patients is the same. Domain experts who have experience with the health coaches, have indicated that the coaches respond better to hands-on experiences rather than training manuals. This means that hands-on experiences allow for information to remain more easily within the memory of the health coach. Unfortunately using actors can be very time consuming, since they would only be able to train one coach at a time. Not to mention the costs that would be associated for hiring and training the actors.

A solution for this problem is to implement a system that allows for hands-on training without actors. Using a virtual patient (avatar) to replace an actor would both fit the learning method of the coach and would also result in lower costs. The avatar would have to show behavior that falls in line with how an actual patient would act, simulating an actual coaching session. The model used for simulation would be based of the concepts used in the MAY profile, allowing the users to familiarize themselves with those concepts. Within this thesis we propose a system that allows for the creation of such an avatar and show a prototype of how it would be realized.

1.2 Approach

The first step in the creation process of the avatar is to define the requirements of the system. This is necessary to set the goals that have to be achieved. The input of domain experts was used to set up a list of requirements that would lead to the creation of an avatar that addresses the problem as described in section 1.1. After all the requirements have been set, a global system architecture was designed. This describes the design document, which is the basis for how the avatar will be realized.
After the system architecture had been finalized, the next step was to find software in which the architecture could be implemented, as well as fulfill the list of requirements.

The following step was to explore the domain further, in order to build a set of knowledge about the MAY profile and the concepts that it used, in particular motivational interviewing. Motivational interviewing is a method of coaching that emphasizes the autonomy of the patient. The interactions between coach and avatar, are based on the guidelines that are used in motivational interviewing.

The next step was combining the requirements, the design document, and the domain knowledge on coaching sessions, and set up a rule base which will control what the system will do. This rule base is then implemented into the software that was chosen. Not only the rule base has to be implemented, the visualization of the avatar is crucial as well. Again, this was also implemented using suitable software.

The next, and final step, is to present the finalized prototype to the same domain experts that helped set up the list of requirements. By validating the prototype, the feedback that is received can be used to update the avatar, as well as set up a list of future improvements that are outside the scope of this project.

In summarization the approach that was developed for and used in this thesis is as follows:

1. Set up the list of requirements.
2. Create a design used as the blueprint of the system.
3. Explore the domain and motivational interviewing.
4. Set up the rules to be used in the rule base.
5. Implement the rules.
6. Implement the visualization of the Avatar.
7. Validate the prototype with the domain experts.

1.3 Thesis Structure

First, in chapter 2 a detailed description will be given on the MAY profile and how it is to be used effectively. Following that, motivational interviewing will be discussed, explaining its, and how it is performed correctly. In addition a more in depth view will be given about virtual environments; the basic concepts that exist within such an environment, as well as an overview on how these environments can be used for educational training. In chapter 3 the system requirements will be discussed. These requirements must be fulfilled for the prototype to work as desired. Furthermore, the system architecture will be shown and the concepts within it will be defined. Several pieces of software used for building the system architecture will be discussed as well. The system design that controls the avatar and system is examined in chapter 4. The behavioral aspects of the avatar in response to the choices made by the coaches is explained. The different problem scenarios and dialog trees are discussed as well. Next is chapter 5 here the details of the implementation given. The link between the behavioral script outputted, and the visualization of the avatar is clarified as well. Chapter 6 will focus on the evaluation of the prototype. And finally, chapter 7 will conclude the thesis paper.
2 Background

This chapter discusses the background domain in detail. Crucial concepts used in the avatar prototype are defined; such as the MAY profile, Motivational Interviewing, and the use of Virtual Learning. These concepts are key in the development of the avatar prototype, and it is important to grasp them properly in order to understand how the avatar was created and functions.

2.1 MAY Profile

NOTE: Section 2.1 contains Philips confidential information, and is not available in the public version of this report.

2.2 Motivational Interviewing

Motivational Interviewing was developed by [Miller (1983)] as an “approach based upon principles of experimental social psychology, applying processes such as attribution, cognitive dissonance, and self-efficacy”. Motivation Interviewing emphasizes individual responsibility and deemphasizes the labeling of patients. Cognitive dissonance is created when the problem behavior is contrasted to the awareness of this behavior’s negative consequences. The goal is to make the patients realize that they have the ability to undertake action to solve their problems, without pushing or blaming them into performing these actions.

As described by [Miller and Rollnick (2012)], “Motivational Interviewing is a process that helps people resolve their ambivalence and move toward healthy change”. Miller defines several concepts that play a role within Motivational Interviewing:

- **Empathy**: Nonjudgmental understanding of the patients perspective.
- **Develop Discrepancy**: Explore the patients values in life, contrasted to their current situation. Through this realization, patients can make case for change.
- **Avoiding Argument**: The coach has to avoid being the one who’s arguments awaken the change in the patient. The change should come of the patient’s own volition.
- **Rolling With Resistance**: Accept the ambivalence of the patient, and motivate them to enter the process of solving their problems.
- **Self-Efficacy**: The coach should encourage the patient’s own sense of the possibility of change.

Motivational Interviewing was initially developed as a coaching method for patients suffering from addiction (alcohol, smoking, drugs etc.). Studies have shown that using Motivational Interviewing with these patients increases treatment involvement, and therefore improves the outcome as can be seen in a study by [Brown and Miller (1993)]. Many other studies have shown positive effects for the use of Motivational Interviewing for substance abusers ([Carroll et al. (2001); Soria et al. (2006); Walters et al. (2009)].

Outside of addiction problems, Motivational Interviewing can also have a broad application in other areas, such as health promotion ([Resnicow et al. (2002)]. Evidence for its useful application in these other areas can be seen in studies done by ([Schwartz et al. (2007); West et al. (2007)] which shows its effectiveness in weight loss, and studies by ([Westra and Dozois (2006); Westra et al. (2009)]) which show positive results in those suffering from anxiety.

This versatility makes Motivation Interviewing a useful tool to use in combination with the MAY profile. For that reason, The MAY profile coaching documentation ([Vinkers et al. (2014)]) heavily integrates many Motivational Interviewing concepts. Naturally, these concepts were then also implemented in the avatar system.
2.2.1 Motivational Interviewing in the Avatar System

For building the avatar system, it was necessary to achieve a basic grasp on the Motivational Interviewing concepts, so that they could be properly applied within the system. This was done by watching a video course series by Cole (2012) focusing on the application of Motivation Interviewing for health promotion. This video series explained the principles of Motivational Interviewing in great detail. The videos also contained several recorded sessions with actual patients as reference material.

Based on these video the concepts of Motivational Interviewing were implemented in the avatar system. Following is a list of guidelines based on the video series that is used as the foundation of Motivational Interviewing within the system.

Positive Guideline:
1. Ask open questions; allow the patient an opportunity to come up with their own answers.
2. Give autonomy to the patient; make them feel independent and motivate them to think for themselves.
3. Let the patient decide the pace of a session; make them choose when to go forward.
4. Let the patient make a plan for improvement; this will make it more likely for them to follow the plan through to the end.
5. Ask permission when giving advice, changing the subject of the conversation; the patient should feel in control of the session.
6. Reflect and summarize on what the patient has said in a session; the coach should show that they have listened and understood the things that the patient has said.

Negative Guideline:
1. Righting Reflex: the coach should not push his own ideals onto the patient; they should not tell the patient what to do, and how to feel.
2. Force the patient to change against their will, or when they are not ready.
3. Give info/advice or change the topic without the permission of the patient. This also includes changing the subject when the previous has not been finished properly.
4. Be judgmental; blaming the patient for not changing their lifestyle.
5. Show a lack of interest, understanding and compassion.

2.3 Virtual Learning

The coaches have to familiarize themselves with both the MAY profile, and Motivational Interviewing. In this case the goal is to train coaches using a virtual patient avatar. This is a teaching method within a virtual space, using a virtual representation of the patient. The concept of virtual patients is based on Simulated Patients, which are further described in section 2.3.1.

2.3.1 Simulated Patients

A simulated patient is an actor trained to behave like a patient; the simulate the patient’s behavior, symptoms, and problems. Barrows (1968) describes several benefits of training health practitioners using simulated patients:

1. The student can practice techniques without feeling embarrassment in front of a patient, tiring the patient, or accidentally aggravating their illness.
2. The simulation can be played out until results are satisfactory.
3. All necessary aspects of the disease and symptoms can be discussed freely.
4. Patients are not treated as test subjects.
5. A problem can be presented to the student in which all the findings are predetermined.
6. The same problems can be presented multiple times in order to gauge its teaching effectiveness.
7. The simulated patient can report on the skills of the student.

The use of simulated patients is not an ideal solution; the patient actors need to be trained, are costly, and availability is limited. A solution to these issues are the use of a virtual simulated patient (avatar); it has the same benefits as described previously, while being cheaper in use and always available. Of course, there also cons; actions and behavior that come naturally to a human are very difficult to simulate in a machine, and it requires combining many different techniques and technology to make it possible. And so, the implementation of a virtual patient that can show the correct behavior, is much more complicated to do, compared to training an actor.
2.3.2 Virtual Patients

This section discusses several studies and papers involving virtual patients; they were found by searching for the keyword "virtual patient". From the list, several studies were chosen based on whether communication was possible between the student and virtual patient, whether they described all the functions of the virtual patient, showed some details of the implementation, and tested its capabilities (realism, student learning effect). Papers that discuss the general use of virtual patients were not chosen, since they often do not focus on the details of the virtual patients themselves.

A crucial aspect of virtual patients functionality, is whether the results between virtual training, or actor training differs. A study by Triola et al. (2006) shows that students trained using a virtual patient showed similar results to those trained using simulated patient actors. This virtual avatar by Triola et al. (2006) makes use of 75 potential questions that the students can ask the virtual patient; an accompanying video clip of the virtual patient will play when a question is asked. Students are allowed to ask any question in the order they want, but the responses are determined beforehand due to use of prerecorded video.

One more example of studies involving a virtual patient was done by Bergin and Fors (2003). In this virtual patient system the student can use free-text driven questions to inquire about the virtual patient’s medical history; the patient responses are again done using prerecorded video clips. The student also has access to the virtual patient’s medical files (X-ray, CT scans, lab test etc.). Based on the information available, the student can make a diagnosis of the patient. Afterwards, the system will give feedback on this diagnosis. The results of this study showed that 80% of the students found the experience with the virtual avatar realistic. Bergin and Fors (2003) states that realism is important for the learning process, meaning that the results are advantageous.

A similar study was done by Salminen et al. (2014) on the use of virtual patients for training primary care students. This virtual patient makes use of text based communication from student to patient; students are allowed to use both open-ended questions and free-text answers. The system is capable of showing pre-recorded video clips as the response of the virtual patient. The performance of the virtual patient was judged by the students, and the results showed that they accepted its authenticity and found it a useful tool to use in combination with the theoretical aspects of the education, indicating that it is a good way to fill in gaps of knowledge.

Another virtual patient is Justine by Kenny et al. (2008). This virtual patient avatar simulates a young female suffering from Post Traumatic Stress Disorder (PTSD). The system makes use of speech recognition, 3d animation and a set of possible questions mapped to responses. Domain expertise was used to map a possible 459 questions to 116 responses; these questions and responses were categorized into eight subjects of PTSD (Trauma, Re-experience, Other etc.). The student asks a question, and the system will map this to a category and a response from Justine; the responses are again, predetermined. The students experiences showed that they considered a useful tool, provided that the questions were categorized in such a way that the proper answer was given. The test results indicated that due to the speech recognition many questions were categorized as “other”, and so the avatar would explain that they did not understand the question. The students received this response many times, which was a frustrating experience. Another paper written by Parsons et al. (2008) describes Justin, a virtual patient similar to Justine. Justin is a male patient with a conduct disorder. This virtual patient also makes use of speech recognition and pre-recorded speech files. The avatar is capable of showing non-verbal body movements as well as lip-sync. The questions from the student are first converted to text, using speech-to-text, and then mapped to a predetermined question, which will then trigger the predetermined avatar response.

A study by Johnsen et al. (2005) describes a patient avatar named DIANA to help students train their medical communication skills. The virtual patient is can respond to questions using predefined answers which are vocalized using a text-to-speech engine. The avatar response script is written by a domain expert, such as a doctor. The student can communicate with the patient using speech recognition, and mapping the contents of the question to the set of predefined patient avatar reactions. Students also have detectors which can measure where the student is looking, as well as where they are pointing. The avatar is also capable of showing non-verbal behavior through dynamic facial expressions and gestures. In addition to the virtual patient, there is also a virtual expert named VIC present in the simulation. This virtual expert represents another doctor that guides the student. The virtual expert will give real-time performance feedback to the student. The study tested the performance of DIANA, using the experiences of the students. The performance of DIANA was given a high mark by the students, and they indicated that especially the feedback from the virtual expert added onto the experience.

A paper by Walters et al. (2009) describes a virtual patient for training health-care providers in performing medical interviews in a second language. The Virtual Language Patient (VLP) makes use of pre-recorded videos of an actor, to show the reactions of the virtual patient. The student communicates with the virtual agent through...
text-to-speech; they read the predetermined questions out loud in a second language. The virtual patient will react appropriately to the question, as well as indicate if the pronunciation of the question was incorrect. A feedback system will also display which words were spoken incorrectly.

2.3.3 Virtual Patient Implementation Choices

The virtual patient systems described in 2.3.2 use the following methods for implementation:

- Asking Questions: All systems described allow the student to ask questions through predetermined questions, free text, or free speech recognition. Due to technical limitations, speech recognition may fail to work as intended. Some systems also make use of detectors to measure pointing and head movements.
- Avatar Responses: The described systems allow the virtual patient to respond to the questions of the user. This is done by using predetermined responses through video or 3d modeling. The responses are mapped to the questions of the student directly, or through categorization. They are communicated to the student using pre-recorded audio and text-to-speech. In some avatars animations of facial expressions or body movements are used to convey non-verbal behavior.
- Feedback: Explaining to the student what went wrong or right will give a better learning experience.
- Patient Profile: In some virtual patient systems, patient medical information is available for use by the student.

Like Bergin and Fors (2003) stated, a realistic experience is important; the more realistic the behavior of the virtual patient, the better the response of the students will be. And from the student experiences as seen in the study by Kenny et al. (2008), the sense of realism is very much damaged whenever the virtual patient does not respond in a way that is appropriate towards the actions of the student. An important goal when creating a virtual patient avatar is making sure that behavior falls in line with how an actual patient would react.

Taking in account both the characteristics of a simulated patient as described in section 2.3.1, as well as the methods of implementation described in section 2.3.2, it becomes clear how a virtual patient avatar should function, and how it can benefit the learning process.

2.4 Discussion

This chapter discusses the background domain of the avatar system. The MAY profile is a tool developed by Philips, to be used by health coaches. It represents the profile of the patient, including the personal situation and personality. The MAY profile is constructed using a questionnaire filled in by the patient themselves. The MAY profile summary contains all the data of the MAY profile in a simple overview; the MAY profile documentation contains detailed information on all the different MAY profile concepts, and it is effectively the training manual for correct use of the MAY profile.

Motivational Interviewing is a method of coaching that gives autonomy to the patients. It is a crucial aspect of the MAY profile documentation, meaning that coaches should follow the principles of Motivational Interviewing in order to use the MAY profile effectively.

Finally, Virtual Learning of health practitioners is done through using a virtual patient avatar. The concept of using virtual patients is based on simulated patients, which uses actors instead. Theoretically, virtual patients have the same benefits as simulated patients, except using virtual learning is much cheaper, and scenarios can be repeated endlessly. The crucial aspect of the virtual patient is its realism; if it can show fitting behavior, then the student will not learn properly. For the virtual patient to be just as effective as a simulated patient, realism is key. This realism is achieved by implementing the virtual patient in such a way the response of the avatar fits the question from the student. And as described in section 2.3.1, the implementation of the virtual patient avatar is complicated, and requires the combination of many techniques. This report describes in detail how such an avatar can be implemented.
3 Requirements, Architecture and Tools

In this chapter the system requirements, architecture and tools are discussed in detail. First the capturing of the system requirements is explained, including the different types of requirement desires for the multiple users that would benefit from the avatar system. Following is the final requirement list which is used as the guidelines for the architecture and design of the system. The architecture and how the concepts within interact with each other is defined. In the tools section several softwares for realizing the architecture are explained and discussed in detail. These include Drools (business logic intergeneration platform), Unity (3d engine), and Faceshift (facial capture).

3.1 Capturing System Requirements

For the prototype to work as intended, there is a list of requirements that it has to fulfill. There are multiple groups that have different demands, and the goal is to create an avatar that can satisfy all of these.

Coach: Wants to help their patients effectively, and so wishes to train and improve their skills. They do not feel comfortable with using manuals and training literature, and would prefer a more hands-on approach.

Patient: Wants to receive treatment from coaches that are capable and can apply their skills in the correct way, allowing the patient to work towards a plan to improve their quality of life. A coach should listen to what the patient says, and make them feel at ease.

Coach Manager: Wants to train many coaches quick and effectively, while also improving their skills. The amount of patients per coach is large, so effective use of tools like the MAY profile and Motivational Interviewing are crucial in helping as many patients as possible. Since a hands-on approach is more suited to the learning style of the coaches, role-playing with actors would be a desirable but expensive option. An avatar would therefore be a cheaper alternative, while still achieving similar results.

Philips: As the creators of the MAY profile product, their goal is to have users apply this project as was intended.

First, some basic concepts of the prototype were defined based on previous knowledge and the MAY Profile documentation by Vinkers et al. [2014]. Then these concepts were presented to experts on the domain. Based on the input of those experts, adjustments and a list of requirements was made. See appendix A for a summary of the first meeting with the experts. This list is divided into main-requirements, which represent the overall goals of the project, and the sub-requirements, which on the other hand, represent the smaller goals necessary for achieving the overall goal.

3.1.1 Final Requirement List

Main Requirements:
- Improve coach knowledge on how to approach patients with self-care interventions using the MAY profile and motivational interviewing.
- Simulate realistic patient behavior within an avatar, based on a MAY profile.

Sub-requirements:
- Present problem scenarios and choices in which a coach has to apply their knowledge. The choices have to be a representation of what a coach could actually say during a session. This is to give the coaches examples on how to start a conversation.
• Coach has access to a MAY profile summary of the avatar that fits the personal circumstances of the avatar.
• Storing information on the choices the coach makes, the MAY profiles and the problem scenarios.
• The avatar has to behave realistically based on the assessment of the input from the coach:
  – Speak sentences that fit the choice of the coach and personal circumstances as described in the MAY profile.
  – Show emotional facial expressions that fit the choice of the coach and personal circumstances as described in the MAY profile.
• Communication between coach and avatar:
  – Coach has to be able to communicate their choices to the avatar.
  – Avatar has to communicate to the coach through words.
  – Avatar has to communicate to the coach through facial expressions.
• A way for the coach to gauge the overall mood of a session.
• Explain to the coach why the avatar behaved in a certain way in case of a negative assessment.

3.1.2 Main-requirement - Improving of Coach Knowledge

It is not possible to look within the mind of a human and see its inner processes, in other words, the coach is a black box in the system. For that reason, many of the concepts relating to the coach will have to be considered as assumptions on how the human mind works and learns.

The coach has certain knowledge on how to approach a coaching session with a patient stored within their memory. The goal of the system is to improve this set of knowledge, by giving positive or negative feedback through communication with an avatar. The reactions of the avatar are positive or negative, based on the choices the coach makes when confronted with a problem. The desired result is to create a learning process within the coach that updates his/her knowledge using a feedback loop.

Figure 3.1 shows a basic diagram of the learning process. Knowledge is applied by the coach leading to a performance which is evaluated by the avatar, resulting in feedback which the coach then uses to update his or her knowledge.
3.1.3 Main-requirement - Simulate Realistic Behavior in Avatar

Unlike the human coach, the processes in the avatar are completely simulated, so there is no need to make assumptions about its internal processing. The main goal of the avatar is to simulate realistic behavior and reactions to the choices made by the coach. This behavior is based on the MAY profiles; for example, an avatar with a sensitive phenotype should have a positive reaction to an authoritative approach of the coach. Then there is also a need to simulate the cause for the problem that the patient avatar has. For example, a certain scenario, such as a past experience, which is causing the problem in their lives. These problem scenarios are not a part of the MAY profile; the scenarios will be discussed further in chapter 4.

3.2 System Architecture

On a global level, the system contains four elements, the human coach, the MAY profile summary, the virtual environment including the 3d representation of the avatar, and the GUI. The coach is presented with a scenario and choices within the GUI, and has access to a MAY profile of the avatar.

Figure 3.2 shows a global diagram of the system, containing the concepts and relations/interactions between these concepts.

The process flow as shown in figure 3.2:
1. The coach is given access to a MAY profile summary which describes the virtual patient’s personal characteristics concisely.
2. The coach is presented with problem scenarios in which decisions have to be made.
3. The coach has to choose which intervention category to discuss with the patient, as well as the cause behind the problem (motivation, confidence, skill):
   a) A choice for what problem point to address (Intervention Category).
   b) A choice in how to approach the problem point (Personalized Intervention).
4. The coach uses a combination of his/her knowledge on motivational interviewing, the MAY profile, and the information given by the avatar during the session, to make a decision and pick a choice.

5. The choices are communicated to the avatar.

6. The avatar knowledge base assesses the choices as positive or negative by using its knowledge base.

7. Realistic behavior sentences in the form of text is created, and the emotion value is updated. The emotion value is used to gauge the overall mood of a session, it is further discussed in section 4.1.3.

8. The avatar creates a reaction using speech (text-to-speech) and facial expressions.

9. The reaction is communicated to the coach.

10. Based on the content of the avatar reaction, the learning process of the coach begins:

    a) A positive reaction will indicate that a correct choice was made (positive feedback loop).
    b) A negative reaction will indicate that an incorrect choice was made (negative feedback loop).
    c) If feedback is negative, an explanation as to why the avatar reaction was negative will be given to the coach.

11. The learning process will update the knowledge of the coach.

### 3.2.1 Concepts within Coach

**Knowledge Set** The assumption is that knowledge on the MAY profile and motivational interviewing exists within the coach. By testing this knowledge it can be proven to be sufficient or insufficient. The assumption is that when knowledge is sufficient, a positive feedback will further solidify that piece of knowledge within the coach. If knowledge is insufficient then negative feedback is used to point out the lack of knowledge, and an explanation will be given to close the knowledge gap. As the coaching session progresses, the patient will reveal more information about themselves and their circumstances. This information should be added to the knowledge set of the coach. If the coach does not remember the information given to them, they will receive negative feedback.

**Decision Making** In decision making, the knowledge of the coach, information from the MAY profile, as well as the problem scenario and possible choices come together to create a process in which the coach has to pick the choices they believe fits the patient and scenario the best.

### 3.2.2 Concepts within Avatar

**Assessment of Input** The choice made by the coach is assessed positively/negatively based on the information taken from the knowledge base within the avatar. Choices that go against the principles of motivational interviewing or choices that do not fit with the personal characteristics of the patient will be assessed negatively.

**Knowledge Base** A knowledge base is a technology which is used to store complex structured and unstructured information in a computer system. The knowledge base within this system contains information on the patient profiles, motivational interviewing and the multiple problem scenarios that exist.

**Feedback** Extra feedback is given on top of the avatar reactions. After receiving a negative feedback from the avatar, the coach realizes that the choices made were not correct, yet they might not understand why. An important step in the learning process is to explain why a certain choice is incorrect.

**Behavior Process** The behavior of the avatar is controlled by a rule base, which applies the results from the input assessment, and the information from the knowledge base, to simulate realistic behavior. Output is represented through behavior, which consists of lines of text and an emotion value. This value indicates the overall performance during a session. Choosing the correct or incorrect choices will increase or decrease the emotion value respectively. The emotion value is used to change the facial expressions of the avatar, as well as adjust the environment, such as changing the colors on the screen to indicate the current overall mood.

**Communication** The communication from avatar to coach is done through speech and facial expressions. Speech is created by transforming the behavioral text using a text-to-speech (TTS) generator. Using a TTS generator allows for more freedom compared to pre-recorded lines. A study by [Dickerson et al. (2006)](Dickerson et al. (2006)) has shown that synthesized
speech can be as effective as real speech when using an avatar to teach. Although, this study has also shown that in some cases more emotional weight is required, such as when training students in-depth on how they have to formulate questions. In order to compensate for this lack of emotion in synthesized speech, this avatar makes use of non-verbal behavior that can give emotional weight to what is being said. In other words, emotions of the avatar are communicated by altering the facial expression of the avatar. In addition Lip-sync is used to make the experience more immersive; the lip-sync is created dynamically based on the audio levels of the generated TTS.

3.2.3 Concepts outside of Avatar/Coach

**MAY Profile** The MAY profile is used by the coach as an information source and guideline for what points to discuss with the avatar, as well as how the avatar is to be approached. The avatar characteristics should fall in line with the MAY profile that is presented to the coach. Due to the time-limit the only relevant MAY profile concepts are the 'Phenotype’ and ‘Ability or Self-Care’. See section 2.1 for further clarification of these concepts.

**System GUI** The system GUI is used for presenting the coach with the possible choices they can pick. The choice is a direct representation of what is conveyed to the avatar. A description of the choices, which is also shown in the GUI, informs the coach on what type of choice they’re making. These descriptions are based on the MAY profile documentation (reference MAY documentation), see section 2.1 for further information. As discussed before, feedback is given to the user. This is done through a textual notice within the GUI, when the coach requests so.

3.3 Tools

A combination of several pieces of software is used to create the avatar system. Most of the programming was done in Java, within the Eclipse integrated development environment (IDE), combined with the Java based Drools language (see section 3.3.1). Furthermore, the 3d game engine Unity was used extensively as well. And to create facial animations for the avatar, the software Faceshift was used and applied in the Unity engine. Java and Drools are used for the reasoning aspects of the avatar, while Unity takes care of the visualization and GUI.

![Figure 3.3: Architecture of the Software](image)

The following concepts fall under the Drools part of the system:

- Assessment of input of the coach.
- Knowledge base including the information of the MAY Profile of the Avatar, and the choices that the coach has made.
- Creation of Feedback.
- Creation of lines of text representing what the Avatar is supposed to say.
- Adjusting the emotion value.
- Deciding which facial expression has to be shown.

The following concepts fall under the Unity part of the system:

- Communication from Avatar to coach through TTS of the lines of text.

---

1 Eclipse Version: Luna Service Release 1a (4.4.1)
- Communication from Avatar to coach through playing facial expression animations.
- Generate facial expression animations based on real-life facial expression, using Faceshift software within Unity.
- Creating lip-sync that fits the generated audio file.
- Show the feedback.
- Change the environment based on the emotion value.
- Communication from coach to Avatar using the GUI.

3.3.1 Drools

Drools is used for the implementation of the knowledge and rule base that controls the avatar. The programming of the Drools rules was done using a plugin for Eclipse. Drools is a Business Logic integration Platform (BLiP) and it is written in Java. It is an open source project supported by JBoss and Red Hat, Inc. It is licensed under the Apache License, Version 2.0. The specifics of Drools as discussed in this section, are based on information taken from the "Drools Developer’s Guide" by Balis (2009). A typical example of a rule within Drools can be seen in listing 3.1.

```
1 rule "Example Rule: Print notice if user is male and under minimum age + Set access to false"
2   when // conditions
3     Age ( $m: minimumAge )
4       $person: Person ( $a : age < $m, gender == 0)
5   then // consequences
6     System.out.println("User is male. User's age is"+$a
7         + ". User is underage");
8     modify ($person) {SetAccess(false)};
9 end
```

Listing 3.1: Drools Example Code

This example rule checks if the user is under a certain age and if they are male (where male == 0, female == 1), prints a notice, and calls the SetAccess, modifying the user access false. The variables age, gender and the method SetAccess are part of the Person class. The variable minimumAge is part of the Age class. These objects are inserted into the knowledge base as facts, where they can then be used in the rule base. When a fact is updated or modified, the rules that are relevant to the change will be checked again.

Drools is specifically designed to prevent 'spaghetti code'; a large collection of nested if-then-else statements. By keeping the rules "seperated", it becomes much easier to change or add rules if necessary, without breaking the programming. The avatar has many rules that control its behavior; if this was to be implemented using if-then-else statements, the amount of nested code and conditions would quickly become very difficult to read and edit. For this very reason, the use of a rule engine is preferred. Drools was chosen because it is one of the more well known rule engines, as well as the fact that it is Java based. Drools also makes use of the Rete algorithm, making it much faster.

3.3.1.1 Rete Algorithm

Since Drools 2.0, it has been using the Rete algorithm to boost its performance. The Rete algorithm was developed by Forgy (1974, 1979, 1982) as an efficient method for comparing a large collection of patterns to a large collection of objects. A naive algorithm would check every single fact against every single rule in the knowledge base, and fire the rule when the conditions are fulfilled. After that, the process would start from the first rule again and loop. This can be a slow process when the amount of rules is very large. The Drools Rete algorithm, on the other hand, uses a network of nodes to reach the rule that has to be triggered (rete is the Latin word for network). The root node is the starting point and the leaf node triggers the rule. The nodes are tied to the conditions of a rule, and the path from the root to the leaf defines the conditions that have to be satisfied. Using this node method makes the Rete algorithm much more faster and efficient compared to a naive alternative.

---

2 Drools version 6.1.
3 http://www.apache.org/licenses/LICENSE-2.0.html
Figure 3.4 shows the Rete network of the example rule generated by Drools. Drools makes use of several different nodes to create a network.

- **ReteNode**: The default entry point of the network. When a fact is inserted in knowledge base it enters the knowledge base through this node.
- **EntryPoint**: An entry point, can be re-named.
- **ObjectTypeNode**: Inserted facts are filtered in this node; only certain objects are allowed through. In figure 3.4 only Person and Age are not filtered out.
- **AlphaNode**: The AlphaNode is the first level of matching; it is responsible for evaluating constraints on single facts. In figure 3.4 this can be seen in the constraint gender == 0. Each constraint creates a new AlphaNode.
- **LeftInputAdapterNode**: The second level of matching takes place in the beta nodes; the LeftInputAdapterNode acts as the entry point for the beta nodes. It creates a tuple out of a single fact.
- **TerminalNode**: This is the leaf node of the network. Each TerminalNode is associated with a specific rule. When the TerminalNode is reached, that rule will be placed on the execution Agenda. The Drools Agenda is used for managing the order in which rules are executed.
- **JoinNode**: The JoinNode is a type of beta node. This node evaluates constrains on one or multiple facts. The node joins two inputs together; the left (tuples) and the right (facts) input. So in the case of figure 3.4 the $m$ is the left input tuple (created by the LeftInputAdapter), while the age is the right input fact. If the constraints on the input tuple and fact are satisfied, the TerminalNode will be reached. So that means that “age < $m$” has to be satisfied within the JoinNode.

### 3.3.1.2 Node Sharing

Node sharing is used to keep the size of the Rete network as small as possible to improve performance. When a new rule is created, its nodes are placed inside the network. For example, an ObjectTypeNode has to be created.
for the objects that are used within a rule. If then another rule is created, which uses the same objects, no extra ObjectTypeNodes have to be made, since the nodes that were created for the first rule can be reused. This holds true for the AlphaTypeNodes and beta nodes as well. But in order to use node sharing effectively the placements of the constraints in a rule are crucial; the order of the conditions between rules should remain similar.

The rules of the avatar have many constraints that can be reused, and the order of many rules are similar as well. This means that node sharing can be used effectively within the avatar system to improve performance speed, while using sharing to keep the memory use low.

3.3.1.3 Stateless and Stateful

Drools implements two different forms of sessions for executing rules, stateless and stateful. A stateless session discards all the facts in the knowledge base after the current session is over. If a new session is created, all the facts have to be entered once more. A stateful session keeps the facts that are in the knowledge base for the following session. When the new session starts, the current facts still exist. A stateful session is used when it is necessary to have multiple sessions with the same facts. When the facts no longer need to be remembered, the session needs to be discarded.

For this prototype avatar, the stateful session method is used. In the current version there is only one knowledge session per dialog session with the avatar, after which the facts are discarded and the application ends. For the sake of potential future implementations, where multiple sessions using the same facts take place, the choice was made to use stateful sessions.

3.3.2 Unity

Unity is used to create the visual aspects of the avatar, including the GUI. Unity is a 3d development engine used to create 3d and 2d games, as well as other interactive experience (e.g. training simulations). The Unity engine is capable of implementing many aspects of 3d environments, including the GUI, 3d models, lighting, texturing, animations and many more. Unity makes use of both a visual interface as well as the MonoDevelop IDE for scripts. Unity is compatible with scripts written in JavaScript or C#. This avatar project mostly uses scripts written in C#. Unity is capable of building to a large variation of platforms, such as Windows, a webplayer, iOS, Android and more. For this avatar prototype Unity builds an Android Package (.apk) to be run on an Android Tablet.

Figure 3.5: Unity GUI

---

4Unity3d version 5.1.
5Unity can build to: Webplayer, PC, Mac, Linux, iOS, Android, Blackberry, Windows Store, Windows Phone 8, WebGL (preview), Xbox 360, Xbox One, PS3, PS Vita, PS4, Samsung TV
Figure 3.5 shows the Unity GUI. Unity works by creating scenes, and objects to place in the scene. In this example the objects Sphere and Cube, were placed into the scene. The close-up seen in figure 3.6 shows the Hierarchy View on the left, which displays the objects in the scene.

![Hierarchy View](image)

Figure 3.6: Object Hierarchy Close-up

The Game View on the middle left, shows what the user will see when the application is running; the Scene View on the middle right, is for editing the scene and placing the objects. The Inspector View on the right, shows the details of an object. A script can be placed on an object to control its behavior. For example, figure 3.7 shows the two C# scripts CubeScript and SphereScript placed on Cube and Sphere respectively.

![CubeScript and SphereScript](image)

Figure 3.7: CubeScript on Cube object and SphereScript on Sphere object

Scripting uses two basic methods Start() and Update. The Start() method is used for initializing before the application starts, whereas Update() is called every frame; it is effectively a loop. When it is necessary for one script to access the script placed on another object, a connection between the objects has to be made. This done by placing the object with the script that has to be accessed, onto the object that wants the access. This process is shown in figure 3.8, where the CubeScript has access to the SphereScript placed on the Sphere object.

![CubeScript has access to SphereScript](image)

Figure 3.8: CubeScript has access to SphereScript

The details of the implementation of the avatar prototype in Unity, will be further examined in section 5.3.

### 3.3.3 Connection between Drools and Unity

The visualization by Unity is shown on the tablet, while the rules that control the avatar run on a separate system. This means that updating the rules can be done on one level without the need to rebuild the .apk, e.g. multiple tablets can run the updated rules while still using the same version of the Android application. Since there is no need to reinstall, it is much easier to change the behavior of the avatar. Another potential benefit is that reasoning over the rules and the visual processing is shared over multiple systems; reducing the load on an individual level. The details of the implementation of the connection between Drools and Unity, will be further explained in section 5.1.
3.3.4 Faceshift

Faceshift is a facial capture software for creating realistic facial animations for 3D models. The software makes use of a special depth camera, capable of capturing 3D image data. The software works by recording the facial expressions of a human, and creates an animation file that can allow a 3D model to perform the same facial expressions. Faceshift was used for the avatar prototype, to create the facial expressions that are shown when the coach makes a choice. These expressions are negative or positive based on the type of feedback required. The first step in applying Faceshift is a training session, in which the software learns the facial expressions of an individual user. After the initial step, it becomes possible to record animation sessions. These sessions are then imported in Unity and placed on an object, in this case the 3D model of the human avatar\(^6\). The details of how Faceshift was applied within Unity, and the specifics of the facial animations are further discussed in section 5.3.5.

3.4 Discussion

In this chapter the focus is on explaining the system requirements, architecture, and the tools used for building:

- The explanation on how the final requirements list as shown in section 3.1.1 was created. The list was created by:
  - Defining different user-cases to establish the different desires.
  - Discussing with domain experts.
- Further defining the two main requirements
- Showing the system architecture, and explaining the concepts used.
- Describe which concepts are to be implemented using each of the tools.
- A brief explanation on Drools, and it’s benefits in regard to the Rete algorithm.
- A brief explanation on Unity, and how it will connect with Drools and make use of the Faceshift software.

In regards to any potential improvements, it may be possible to implement both the logic and visual aspects of the system in such a way that they can be run on a single machine, ruling out any need for communication. This has the downside that when the logic is updated, all the machines running this must be updated as well. With the system’s current design, it is possible to run the logic on one machine, controlling multiple Unity applications running on different tablets at the same time. It is also possible that other types of software are more suited for building the architecture. The choice to use Java-based Drools, and Unity 3D, was because of previous experience with Java, the fact that Unity is free software, as well as the large amount of information available for both Java/Drools and Unity.

\(^6\)The avatar 3D model used in this project was received for research purposes from CAMeRA@VU (Center for Advanced Media Research Amsterdam). The Faceshift recordings were also made at CAMeRA@VU.
4 System Design

In this chapter the process and design of a coaching session will be explained in more detail. These design choices were based on the MAY Profile documentation from Vinkers et al. (2014), as well as Motivational Interviewing educational videos by Cole (2012). Several diagrams are shown to illustrate the flow of a session, giving a detailed view on the design. Following that is an explanation on the dialog trees that function as the script of a session, the details on how the system and avatar will react to the choices of the coach, and more information on the Emotion Value concept used for gauging the mood of a session. Finally, the multiple problem scenario’s which represent the topics of the session are examined.

4.1 Interactive Session Design

Figure 4.1: Global Overview of a Session

Figure 4.1 shows the global overview of the structure of one session. The session starts by having the coach choose which Intervention Category and Personalized Intervention Method they want to discuss with the avatar. As long as the coach choices do not fit the what is written in the MAY Profile of the avatar, the session can not advance to the first phase.

In the beginning of the session the coach is required to identify the situation and problems of the avatar through the use of the MAY profile and communication with the avatar. Then the first reflection phase takes place, in which the coach summarizes and reflects on what the avatar has revealed on their situation. In this phase the coach is tested on whether they have paid attention to what the avatar has said.

Early in the session, the avatar give a random number on a scale of 1 to 10 that represents how motivated/confident/skilled they feel about the Intervention Category, the higher the number the more motivated/confident/skilled they feel. After the end of the first reflection phases, the avatar will once again give a number on a scale of 1 to 10
top represent their feelings; if the current emotion value is higher than neutral, the mood if the session is good and the number of the scale will have increased. This gives the coach the sense that the session until that point has gone well; it is a clear indication of gauging improvement and represents a change within the avatar, which is based on the concept of change-talk (see section 2.2 for more details).

Then the second part of the session. Within these phases the coach discusses potential plans of improvement for the next session. This second set also contains a special phase, in which the coach asks the avatar to make a food diary for the next session. This special phase takes place in all possible scenarios, regardless of the Personalized Intervention Method. Following is the second reflection phase, here the coach summarizes the plan that the avatar has made for the next session. Again, this is to test if the coach has paid attention.

The final part of a session at the end of the second reflection, the coach gives a final piece of advice to the avatar. The correct advice is decided by the **Phenotype** of the avatar. The coach is presented with four different advice options, each representing a phenotype.

The content and order of the phases of a session are based on the instructions taken from the MAY Profile documentation by Vinkers et al. (2014). It is necessary to follow a certain structure of a session; for example, it would feel out of place to discuss potential plans for next week at the beginning of session. This is why it is also important to teach the coaches on how to structure a session.

Figure 4.2: Choosing Intervention Category

Figure 4.2 shows the start of a session; choosing the subject of discussion; the **Intervention Category**. There are seven different categories, but in this current prototype, only **Healthy Diet** is implemented.
Figure 4.3: Reaction to Chosen Intervention Category and Choosing Personalized Intervention Method

Figure 4.3 shows the reaction the chosen Intervention Category. If the choice is incorrect the coach will have to re-choose. The figure also shows the second part of the start of a session; choosing the cause of the patient’s problem; the Personalized Intervention Method. Again, if the choice is incorrect the coach will have to re-choose.

Figure 4.4: Phase Architecture

Figure 4.4 shows the progress from one phase to the next. The same information is received from the avatar in two different ways; through the Positive Path or Negative Path. This setup is necessary to prevent a split later on. The coach has to have the same knowledge regardless of the path that was followed. If this is not the case, it would
be unfair to test the coach on the information later, if their was a split in knowledge. For further information on how the paths differ see section 4.1.1.

Figure 4.5: Present Possible Choices to the Coach

Figure 4.5 shows how the system uses a decision tree to determine which choices to present to the coach. The system uses a dialog tree as a script (see section 4.1.1); based on the dialog tree, the system knows exactly which choices to present to the coach, and what type of avatar reaction needs to be shown. The following concepts are used to decide which choices can be shown to the coach:

- The Intervention Category of the avatar.
- The Personalized Intervention Method of the avatar.
- The current problem scenario (see section 4.2 for more information).
- The current dialog phase.

Using these concepts, the system can pinpoint which choices to present at the beginning of a phase, which means that the coach has made no input yet in this phase. After a choice has been made for a phase, then a different decision tree is used, pictured in figure 4.6.
Figure 4.6: Present Avatar Reaction to the Coach

Figure 4.6 shows how the system uses a decision tree to determine what reaction the avatar and the system has to communicate to the coach, as well as the set of choices that have to be presented. The “…” in the figure, represents that the tree sides mirror each other, this is to prevent the image from becoming too large. The following concepts are used to decide which avatar reaction and next set of choices can be shown to the coach:

- The Intervention Category of the avatar.
- The Personalized Intervention Method of the avatar.
- The current problem scenario.
- The current dialog phase.
- The choice state that the coach has inputted. Can be positive or negative. Each phase has a set of possible positive/negative choices that can be chosen; for example, a choice that follows the principles of motivational interviewing has a positive choice state.
- The choice number that the coach has inputted. It is used to identify which positive/negative choice was made, since a single phase can have multiple positive/negative choices. For example, if the choice state is negative and the number is two, then the second negative choice was made.
- If the choice involves other properties such as phenotype, or comfort with technology. If so, then a different decision tree is used from that point (see figure 4.7).

At the end of the tree, both the fitting avatar reaction is given together with the next set of possible choices. This decision tree is only used after the coach has made an input. As mentioned before, at the beginning of a phase (the coach has not inputted yet) the decision tree as seen in figure 4.5 is used.
If the coach choice involves MAY Profile properties such as Phenotype or Comfort with Technology, then a different type of decision tree is used from that point on to check if the coach choice fits that particular property. Rather than using the choice state and choice number as shown in figure 4.5, the alternate decision tree determines the avatar reaction.

![Decision Tree Diagram](image)

Figure 4.7: Present Avatar Reaction to the Coach when Involving the Phenotype and Comfort with Technology Properties

Figure 4.7 shows the alternate decision tree that is used when the choice involves extra MAY properties such as the **phenotype** or **comfort with technology**. This decision tree is a continuation of the tree from figure 4.6. Within this tree, the system simply checks if the choice made by the coach is the same as the properties on the MAY Profile of the avatar. For example, at the end of each session the coach gives advice based on the phenotype. If the advice that is given is fitting for an analyst, and the avatar is an analyst, then a positive avatar reaction will be communicated to the coach. In this particular case, the previously chosen **choice state** and **choice number** (see figure 4.6), are no longer relevant, as the reaction to the chosen phenotype property takes priority.

![Flow Diagram](image)

Figure 4.8: Global View of the Flow of Input and Output

Figure 4.8 shows the global view of how the input and output of the system is controlled in 5 steps. The system
receives input from the coach, and based on that input, the avatar reaction, feedback, and next set of choices is decided. The facial expression and change in emotion value, is done automatically based on the choice state, which can be negative or positive. The automatic change in expression and emotion value can be overwritten, and set by hand if necessary. For example, as explained previously in figure 4.7, in some cases the choice state becomes irrelevant, since the other properties (phenotype/comfort with technology) take priority. In those cases, it is necessary to overwrite the automatic change based on the choice state.

After all these values are set, the information is sent to the Unity application, where it communicated to the coach. The system then waits on the next input of the coach, and then proceeds to sent the next input back to Drools in the Java application, after which the step 1 starts again.

4.1.1 Dialog Trees

A set of dialog trees are created that represent the script of the sessions containing all the possible path’s that can be followed by the coach. The dialog trees start from phase 1 of the session, which takes place after the Intervention Category and Personalized Intervention Method have been chosen by the coach. All the dialog trees can be found in appendix C together with a detailed explanation of the symbols.

As can be seen in figure 4.4 from section 4.1, it is possible to go through a dialog phase by following the Positive Path or Negative Path. To follow that Positive Path, the choices made have to adhere to the principles of Motivational Interviewing; this means picking choices that give autonomy to the patient. By making choices that use the righting reflex, or blame the patient, the Negative Path is followed. A positive or negative choice will be followed by a fitting reaction by the avatar, which will be shown through speech and expression; textual feedback if negative; and a change in the emotion value.

4.1.2 System and Avatar Reaction

When a coach makes a choice the system and avatar responds in a fitting manner. The response typically consists of the following aspects:

1. A line of text spoken by the avatar.
2. A facial expression made by the avatar. These can be either positive or negative, and can have various variations. For example, two versions of a positive expression are a smile or nodding yes. There are default expressions for normal reactions and for reactions during reflection. The default expression can be overwritten if necessary, as shown in figure 4.8.
3. The emotion value is calculated based on the whether the choice that was made is positive or negative. The uses and calculations of the emotion value are further discussed in section 4.1.3.
4. Feedback is given within the system GUI when the choice which was made is negative. The feedback explains in detail why the choice is wrong, and what the correct approach is.
5. The next set of possible choices that the coach can pick are presented within the system GUI.
6. A description of the current phase of the session is presented within the system GUI. This is to inform the coach about the context of the choices they can make. The description is based on the instructions from the MAY documentation (reference MAY documentation), see section 2.1 for further information.

4.1.3 Emotion Value

The Emotion Value is used to gauge the overall mood of a session. After each choice made by the coach, the avatar gives an immediate reaction that represents the feedback on that particular choice. This feedback short-term and only applies to the last choice that was made. In order to measure the performance over time, the Emotion Value was established. This value increases when a positive choice is made, and decreases if the choice is negative. The higher the value, the better the mood of the avatar. As described earlier, at the beginning of a session, the patient will give a score on how motivated/confident/skilled they are. Later on in the session the score will be given again to gauge improvement during the session. If the mood is good then the score will increase, if not, then the score will remain the same. Whether the mood is positive is determined by the Emotion Value crossing a certain threshold. The Emotion Value is also used to adjust the environment in the GUI, as well as the idle expressions of the avatar. For further details on how the emotion value is implemented to affect the GUI and expressions, see section 5.3.

The emotion value is automatically calculated based on the negative or positive state of a choice. The automatic calculation can be overwritten as shown in figure 4.8; this means that the positive calculation can be used even if the choice identification is negative. For further details in the calculations of the emotion value, see section 5.2.
4.2 Problem Scenario Design

A problem scenario is presented to the coach in order to test their knowledge and capabilities for solving the particular problem together with the avatar. The scenario is, in other words, the subject of discussion during a coaching session. The choices, the description of the choices, and the behavior of the avatar, are adjusted based on the current problem scenario.

As mentioned in section ??, the avatar prototype only implements the intervention category healthy diet. This means that the avatar always has a low ability for self-care for their dietary practices; so sessions will always involve the subject of diet. It is, of course, possible to implement other scenarios regarding the other intervention categories later.

The best way of approaching the problem is determined by its cause, which is defined by the personalized intervention methods: motivation, confidence and skills. The healthy diet scenario is divided into three sub-scenarios based on these personalized intervention methods. Each sub-scenario is then again divided into two different scenarios, which have the same personalized intervention method, but have several other variations. In total, the current avatar prototype has 6 scenarios. Having different scenarios is crucial for the learning process, as it allows the coach to apply their newly acquired knowledge in a different situation. These scenarios test the basic skills on the MAY profile and Motivational Interviewing of the coach; it is possible to add scenarios that are more in-depth, and challenge the coaches on more advanced skills. The scenarios were written based on the information taken from the Motivational Interviewing educational video series by Cole (2012). Each scenario has several dialog trees which, as explained in section ??, function as the script. Following, is an explanation of each scenario.

4.2.1 Motivation

The patient lacks motivation in trying to improve their lifestyle. It is necessary to find out what the patient values as important in their life, and help them understand how improving their lifestyle can allow them to achieve those values.

Scenario 1 In this scenario the patient values their body weight as a value that is important to them; they want to feel better about themselves and improve their body image. The goal of this scenario is to make the patient realized how adjusting their diet can allow to satisfy the values that are important to them.

Scenario 2 In this scenario the patient values a low blood pressure as a value that is important to them; they want to feel safer and better about their health. The goal of this scenario is to make the patient realized how adjusting their diet can allow to satisfy the values that are important to them.

4.2.2 Confidence

The patient lacks confidence in trying to improve their lifestyle. It is necessary to find out the reason for this lack of confidence; inquire about the past experiences and current circumstances of the patient. When the cause is identified, the next step is to try and take it away.

Scenario 1 In this scenario the patient lacks the confidence to improve their diet because of a past experience. It is necessary to address the cause behind their lack of confidence, which in this scenario is because of a diet that has failed in the past. The goal is to guide the patient into focusing on the future, rather than lingering on the past.

Scenario 2 In this scenario the patient lacks the confidence to improve their diet because of their current environment. It is necessary to address the cause behind their lack of confidence, which in this scenario is because of a family that is not willing to adjust and support the diet of the patient. The goal is to guide the patient into trying to convince their family to be more supportive, or find a way to keep to the diet despite the circumstances.

4.2.3 Skills

The patient lacks the skill to improve their lifestyle. It is necessary to identify in what way the patient feels unskilled, and the cause behind the lack of skill. The following step is to address a potential plan to improve the skills of the
patient by discussing training.

Scenario 1  In this scenario the patient lacks cooking skills in general; this is caused by a lack of experience with cooking. It is necessary to guide the patient into tacking a cooking class or follow an on-line tutorial. The most fitting training method is dependent on the Comfort with Technology property of the patient.

Scenario 2  In this scenario the patient lacks knowledge on how to cook healthy, but does know how to cook; this is caused by a lack of knowledge on healthy recipes. It is necessary to guide the patient into tacking a cooking class or follow an on-line tutorial. The most fitting training method is dependent on the Comfort with Technology property of the avatar.

4.3 Discussion

In this chapter the following issues were addressed:

- Giving several diagrams to explain the flow of a session, from beginning to end.
  - Choosing an Intervention Category.
  - Choosing an Personalized Intervention Method.
  - Explaining the concept of phases based on MAY documentation, where the same information can be retrieved through both the negative or positive path.
  - How the system knows what the present to the coach at each given time based on the action of the coach.
  - How the system handles the loop flow of input and output.
- Explain the use of dialog trees as a script.
- Describe in which ways the system and avatar react to the choices of the coach.
- Define the uses of the emotion value in the system for the feedback on the overall mood of a session.
- Illustrate how problem scenarios are used to create context, and allow the coach to apply their skills.

In regards to potential improvements, more problem scenarios have to be created to fully test the scope of the knowledge that a coach has. Currently, the only scenarios are pertaining a healthy diet; the other Intervention Categories need to be implemented as well. The uses of the two paths to attain the same information was designed to prevent too much branching out; a point of improvement would be to create more paths while keeping the branching contained to reasonable levels. The use of dialog trees as a script makes the session linear, more variables can be used to create a more free experience. Combining this with more generic dialog trees that can easily be reused, makes implementing easier; it is necessary to strike a good balance between re-usability and branching.
5 Implementation

In this chapter the details of the implementation are shown, including several code excerpts, as well as the realization of the concepts discussed in the previous sections. As explained in section 3.3.3, the communication between Drools and Unity is crucial in the implementation of the system. First, this connection will be examined in detail. Following that is the details of the Java application side, including setting up the knowledge base with Drools, as well as some examples of rules implementing the dialog trees used for creating avatar behavior. The methods used in the Java application to input and output data from Unity are focused on as well. Furthermore, the Unity application side is highlighted, including how it receives and processes the data received from the Java application. These Unity sections discuss how the visual environment is designed, along with the GUI. The methods used to create the text-to-speech audio is examined, along with the automatic lip-sync. And finally, the implementation of the Faceshift software within Unity is highlighted, focusing on how the facial animations are created, what type of animations there in total, and how Unity controls which animations are played at a given time.

5.1 Implementation of Connection between Drools and Unity

The connection between Drools and Unity is done through creating a server and a socket. The server socket is created within the Unity application, and it waits until it receives a connection request from a socket. While the server is waiting, the Java application is started. The Java application creates a socket and tries to connect with the server. When the connection has been established, the data exchange can begin, and the first Drools rules start firing.

In order to create an avatar reaction as described in section 4.1.2, the following data is sent from the Java application to the Unity application:

- Text: A string of text that represents what the avatar is going to say.
- Feedback: A string of text that represents the feedback given to the coach.
- Expression State: A string that contains the word "positive" or "negative". This represents what type of expression the avatar should show.
- Expression ID: An integer that represents which expression should be shown. It is used in combination with the state, for example: negative 1 is shaking no.
- Emotion: A double that represents the emotion value. The double is converted to a string and sent the Unity application; Unity then converts the string to a float. This is done to send the correct value through the socket.
- Choice Content: A string that represents the content of the choice that is presented to the coach.
- Choice ID: An integer that is used as an identifier of a choice. An ID consists out of a value representing the positive or negative state of a choice, where 1 is negative and 2 is positive. The second value represents the number of the choice. For example, a choice could have the ID "1,2", which means that it’s the second negative choice within a particular dialog tree.
- Description: A string of text that represents the description of the current phase of the session. The contents of the description is based on the MAY Profile documentation by Vinkers et al. (2014).

The following data is sent from the Unity application to the Java application:

- Choice ID: An integer that is used as an identifier of a choice. It tells the Java application which choice the coach made within the Unity application. Using this ID the system knows how to proceed through the dialog tree.

Since the data that the Unity application receives has no particular meaning (a string could be interpreted as feedback/description/choice content), it is also necessary to notify what sort of data the Unity application will be receiving.
next. This notification is done through commands in the form of strings. The Unity application will check the content of the string, and prepare to receive and process the information that follows. Each piece of information necessary to control the avatar within Unity, requires a command. It’s not necessary to send commands from the Unity application to the Java application since it will only receive Choice ID, and therefore it does not need to know what type of information it is receiving.

The Java Application uses the following commands:

- **serverInputter**: Let the Unity application know that the coach is required to give input, and send this input to the Java application.
- **outputFeedback**: Let the Unity application know that a string containing the feedback will be sent next.
- **outputDescription**: Let the Unity application know that a string containing the description will be sent next.
- **outputEmotion**: Let the Unity application know that a string containing the emotion value will be sent next. This value has to be converted from a string to a float.
- **outputSpeech**: Let the Unity application know that a string containing the speech of the avatar will be sent next. This string is converted with a TTS generator.
- **outputState**: Let the Unity application know that a string containing the state of the avatar expression will be sent next. This can be positive or negative.
- **outputExpressionID**: Let the Unity application know that an integer containing the ID of the expression of the avatar will be sent next.
- **outputChoiceContent**: Let the Unity application know that a string containing the content of a choice will be sent next.
- **outputChoiceID**: Let the Unity application know that an integer containing the ID of a choice will be sent next.
- **endSession**: Let the Unity application know that the session has ended, and that the application can be closed.

It is not necessary to send commands from the Unity Application to Java application, since only one type of data is sent through that connection, the choice ID.

### 5.2 Implementation of Java and Drools

The Unity application performs the following processes:

- Creating a socket that connects to the Unity application server-socket using an IP address.
- Setting up a stream printer and stream reader for sending and receiving to the Unity application.
- Creating several different "inputters" and "outputters" used for managing what is sent and received over the socket, as well as when.
- Loading the knowledge base, and insert the necessary data taken from the MAY profile.
- Receive the choice ID from the Unity application and insert it into the knowledge base using the correct "inputter".
- Decide which rules have to be fired based on the newly inserted choice ID.
- Have a set of rules that follows the script from the dialog trees (see appendix C).
- Use the rules to set the speech, expression state, expression ID, feedback, description, choice content, and choice ID.
- Calculate the emotion value.
- Send the necessary data to the Unity application using the correct "outputter".

All the sample codes in section 5.2 are written in Java. If the code contains a rule, then it is written in Drools.

#### 5.2.1 Java Application Socket

The first step of the Java application is to create a socket and connect to the server. The socket has an IP number and a port; the IP number has to be identical to the IP of the machine running the Unity application; the port has to be identical to the port used by the server. Input and print streams are created to receive and send to the Unity application. The code for creating the socket can be seen in listing 5.1.
To send a command from the Java application, a method called `giveCommand` is used. This sends a string to the Unity application using the PrintStream. The code can be seen in listing 5.2.

```java
public void giveCommand(String command) throws Exception {
    toUnity.println(command);
}
```

Listing 5.2: Give Command Method

### 5.2.2 Knowledge Base

The first step in creating the knowledge base for the avatar, is loading a stateful knowledge session, and creating a container in which the objects can be inserted. This step can be seen in listing 5.3.

```java
// Load up the knowledge base
KieServices kieServices = KieServices.Factory.get();
KieContainer kContainer = kieServices.getClasspathContainer();
KieSession ksession = kContainer.newKieSession("ksession-rules-stateful");
```

Listing 5.3: Load Knowledge Session

The following step is the creation of all the objects that are to be inserted in the knowledge base. The following objects are inserted:

- **Profile**: Contains and sets the MAY Profile variables for the **Phenotype**, **Intervention Category**, **Personalized Intervention Method** and **Comfort with Technology**. It is also capable of generating a random value to represent the scale the avatar gives on how motivated/confident/skilled they feel. The variable that decides which of scenarios is played, is also stored in the Profile object.
- **EmotionValue**: Contains and sets the emotion value variable. It also has methods for calculating the emotion value.
- **DialogPhase**: Contains and sets the variables that indicate which dialog phase the session is currently in, as well as if the current phase is a reflection phase.
- **CoachChoice**: Contains and sets the variable for the **Choice ID** and the chosen **Phenotype**. The Choice ID consists out of the variables, *negorpos* (negative/positive) and *choice* (choice number).
- **CoachInputCat**: Contains and sets the variable for the chosen **Intervention Category**. Also has methods that stores the name of the chosen **Intervention Category** to be used in avatar speech and feedback.
- **CoachInputMethod**: Contains and sets the variable for the chosen **Personalized Intervention Method**. Also has methods that stores the name of the chosen **Personalized Intervention Method** to be used in avatar speech and feedback.
- **Inputter**: Contains the variables that indicate which Inputter has to be used.
- **Outputter**: Contains the variables that indicate which Outputter has to be used. Also contains several methods for outputting data to the Unity application.
- **Server**: Contains several methods for controlling the connection between the Java and Unity application.

Before the object insertion takes place, it is necessary to define values for the variables. Most these values are set to zero, since the session has not begun yet. Certain values, like the MAY Profile, are defined before the session starts,
therefore these are not set to zero. The Emotion Values is also not zero; it is set to a neutral value that is half of the maximum value. After all the values have been set, the objects are inserted and the rules are fire, as can be seen in listing 5.4.

```java
ksession.insert( profile);
ksession.insert( emotion);
ksession.insert( phase);
ksession.insert( choice);
ksession.insert( coachInputCat);
ksession.insert( coachInputMet);
ksession.insert( inputter);
ksession.insert( outputter);
ksession.insert( server);
ksession.fireAllRules();
```

Listing 5.4: Insert Objects into Knowledge Base

### 5.2.3 Inputter and Outputter

As was explained in figure 4.8 from section 4.1 the system is in a constant loop of receiving input, setting the avatar behavior, and then sending output. Depending on how the input has to be processed, and which types of output have to be sent, the system fires different types of rules. These rules are called inputters and outputters, and they vary in functionality. For all inputters, the system should be done outputting the data to Unity before attempting to input into the knowledge base. For all outputters, it is necessary that the input has been processed, and that the data is ready to send, this means that concepts such as the emotion value has to have been calculated, and the Expression State and ID have to be set before outputting; this is shown in figure 4.8 where step 1 to 3 have to be completed before the output can be sent. The system will only output if it knows both step 2 and 3 have been executed; it knows that it is ready when both an outputter is true, and the emotion/expression setter is true. When an outputter is true, the system knows step 2 is completed. The emotion/expression setter is used to set the emotion value, and the expression that the avatar should show. When the emotion/expression setter is true, then the system knows step 3 is completed, and these values have been set.

At the end of step 2 the system will set an outputter to true, and the emotion and expression setter to false; this will trigger the emotion/expression setter, which will execute step 3 and put emotion/expression setter to true. This results in the execution of step 4, which will put the outputter to false. Step 3 can be skipped by not setting the emotion/expression setter to false at the end of step 2; this means that emotion/expression setter is already true, so step 3 will not be triggered, rather step 4 will be executed immediately. Following is a definition of all the inputters and outputters, as well as some example codes.

**Inputter 1**  
Inputter 1 sends the command serverInputter, and then receives the Choice ID containing the negative/positive value of the choice, and the choice number, referring to which node was chosen from the dialog tree. These values are then inserted in the knowledge base as part of the CoachChoice. Listing 5.5 shows the code to create Inputter 1.
Listing 5.5: Drools: Inputter 1

For receiving the input from the Unity application, as shown in listing 5.5, the method from listing 5.6 is used.

Listing 5.6: Input Method

**Inputter 2**  Inputter 2 sends the command `serverInputter` and receives two values. One of these values is then inserted as the *Personalized Intervention Method*; part of the CoachInputMethod. For example, 1 is motivation, 2 is confidence, and so on. The second value that is received is discarded; it is a dummy value with no use. This is simply because the system reuses the same methods as *Inputter 1*, as it was easier to reuse them and discard the dummy value, rather than creating more methods in both Java and Unity application.

**Inputter 3**  Inputter 3 is very similar to *Inputter 2*. It receives two values and inserts one of them as the chosen *Phenotype*; part of the CoachChoice. Again, 1 represents analyst, 2 is fighter, and so on. It also discards the other dummy value.

**Inputter 4**  Inputter 4 is very similar to *Inputter 2* and *Inputter 3*. It receives two values and inserts one of them as the chosen *Intervention Category*; part of the CoachInputCat. Again, 1 represents healthy diet, 2 is physical activity, and so on. It also discards the other dummy value.

**Outputter 1**  Outputter 1 sends the data to the Unity application, and then fires the *Inputter 1* rule. This outputter rule is used when the system has to output, and then has to receive the input next. This outputter sends the following data: Text (avatar speech), Feedback, Expression State, Expression ID, Emotion, Choice Content, Choice ID, and Description to the Unity application. Listing 5.7 shows the code to create Outputter 1.
As shown in listing 5.7, several different methods are called to send multiple types of data to the Unity application. Listing 5.8 shows the code for sending to feedback.

As shown in listing 5.7, several different methods are called to send multiple types of data to the Unity application. Listing 5.8 shows the code for sending to feedback.

Outputter 2
Outputter 2 sends the data to the Unity application, but does not fire the Inputter 1 rule at the end. Outputter 2 is used at the end of a phase where new input is not yet needed, in other words, it is used when the end a dialog tree (see appendix C) is reached. It also sets the Choice ID to zero, which indicates that a new phase is about to begin. This outputter sends the following data: Text, Feedback, Expression State, Expression ID, and Emotion to the Unity application.

Outputter 3
Outputter 3 is very similar to Outputter 2; it sends the data and does not fire the Inputter 1 rule at the end. But unlike Outputter 2, this outputter is not used at the end of a phase, and therefore the Choice ID is not
set to zero. **Outputter 3** is used when **Inputter 2, Inputter 3, or Inputter 4** has to be used next; this is why Inputter 1 is not automatically called at the end. This outputter sends the following data: Text, Feedback, Expression State, Expression ID, Emotion, Choice Content, Choice ID, and Description to the Unity application.

**Outputter 4** Outputter 4 is used at the beginning of phase, when only the next set of choices have to be presented to the coach. At the beginning of a phase the **Choice ID** is zero, in other words, it is the start of a new dialog tree (see appendix C). At this point, the coach has made no choices yet in the phase, so there is no need for the avatar to react, and so less data has to be sent. This outputter sends the following data: Choice Content, Choice ID, and Description.

### 5.2.4 Rules for Controlling Avatar Behavior

Since the total amount of rules is large, and many of them are very similar, it is unnecessary to address every rule. Therefore, this section focuses on three different types of rules as an example for how the multitude of rules function. The following example rules are highlighted: a rule for starting a new phase; a rule for choosing the wrong answer; a rule for choosing the right answer and ending the phase. The example rules are part of the Healthy Diet Motivation Scenario 1, which is about a patient feeling unmotivated about their diet, although they would like to lose weight for body image reasons (see section 4.2.1 for more information). In this case the focus is on phase two "Determine the patients thoughts about Healthy Diet", highlighted in figure 5.1. This dialog tree has two negative choices and one positive choice; it is possible to go to the next phase by either choosing [1,1] or [2,1] (see appendix C for more information on how to read the dialog tree symbols).

![Figure 5.1: Healthy Diet Motivation Scenario 1: Phase 2](image)

Listing 5.9 shows the Drools code for a beginning of a phase. It sends the three choices and the description, as
seen in figure [5.1] to the Unity application. The rule checks if the system is currently not inputting or outputting data. It also checks the Intervention Category, Personalized Intervention Method, Scenario, and Phase; this is how if the system is at a beginning of a phase, and which reaction should be shown after a choice was made. The current values are [0,0], so the phase is at its beginning.

```
rule "Diet Motivation Scenario 1 Phase 2 (0,0)"
when
  // Check if outputting is done
  $outputter: Outputter() outputter == false, outputter2 == false, outputter3 == false, outputter4 == false)
  // Check if inputting is done
  Inputter() inputter == false, inputter2 == false, inputter3 == false, inputter4 == false)
  // Intervention Category is healthy diet
  Profile(interventionCat == 1,
  // Personalized Intervention Method is motivation
  interventionMethod == 1,
  // Scenario is 1
  scenario == 1)
  // Current phase is 2
  DialogPhase(phase == 2)
  // No negative/positive choice made yet
  CoachChoice($n : negorpos == 0,
  // No choice number yet
  $c : choice == 0)
then
  // Set the description
  $outputter.setDescription("Determine the patient’s thoughts about healthy diet");
  // Set choice content + choice ID
  $outputter.setChoicesContent("A healthy diet is essential for maintaining a healthy weight.", 1, 1);
  $outputter.setChoicesContent("Could you give me an example of why you think a healthy diet is important? ", 2, 1);
  $outputter.setChoicesContent("A healthy diet is essential for lowering blood pressure.", 1, 2);
  // Use outputter 4
  modify($outputter) {setOutputter4(true)};
```

Listing 5.9: Diet Motivation Scenario 1 Phase 2: Beginning of a Phase

Listing [5.10] shows the rule that fires when a particular negative choice was made. If "A Healthy Diet is essential for lowering blood pressure" (see figure [5.1]), is chosen by the coach, the choice ID [1,2] will be inserted in the knowledge base, resulting in the firing of this rule. The avatar will speak, the feedback will be shown, and the new
set of choices will be presented in line with dialog tree. The emotion value and expression are set to the default.

```java
rule "Diet Motivation Scenario 1 Phase 2 (1,2)"
when
   // Check if outputting is done
   $outputter: Outputter(
      outputter == false,
      outputter2 == false,
      outputter3 == false,
      outputter4 == false)
   // Check if inputting is done
   Inputter(
      inputter == false,
      inputter2 == false,
      inputter3 == false,
      inputter4 == false)
   // Intervention Category is healthy diet
   Profile ( interventionCat == 1,
   // Personalized Intervention Method is motivation
   interventionMethod == 1,
   // Scenario is 1
   scenario == 1,
   // Current phase is 2
   DialogPhase (phase == 2)
   // Negative choice
   CoachChoice ($n : negorpos == 1,
   // Choice number is 2
   $c : choice == 2)
then
   // Set avatar speech
   $outputter . setText("I don't really have a problem with my blood pressure.");
   // Set feedback
   $outputter . setFeedback("Try not to assume what the patient’s thoughts and feelings are. In this case the assumption was wrong, since the patient has no issues with their blood pressure. ");
   // Set choice content + choice ID
   $outputter . setChoicesContent("Could you give me an example of why you think a healthy diet is important? ", 2, 1);
   $outputter . setChoicesContent("A Healthy Diet is essential for maintaining a healthy weight. ", 1, 1);
   // Use outputter 1
   modify($outputter) {
      setOutputter(true),
      setChangeExpr(true), //Use default expression
      setChangeEmo(true)}; //Use default emotion change
end
```

Listing 5.10: Diet Motivation Scenario 1 Phase 2: Choosing a Wrong Answer

Listing 5.11 shows the rule which fires when the coach chooses a particular positive choice, and the end of the phase is reached. The coach chose "Could you give me an example of why you think a Healthy Diet is important?".
and the avatar reacts according to how the dialog tree dictates. Since the choice was positive, no feedback is given, and the default emotion and expression are used. Finally, since this is the end of phase two, the current phase is set to three.

```java
rule "Diet Motivation Scenario 1 Phase 2 (2,1)"
when
  //Check if outputting is done
  $outputter: Outputter(
    outputter == false,
    outputter2 == false,
    outputter3 == false,
    outputter4 == false)
  //Check if inputting is done
  inputter == false,
  inputter2 == false,
  inputter3 == false,
  inputter4 == false)
  //Intervention Category is healthy diet
  Profile(interventionCat == 1,
    //Personalized Intervention Method is motivation
    interventionMethod == 1,
    //Scenario is 1
    scenario == 1)
  //Current phase is 2
  DialogPhase(phase == 2)
  //Positive choice
  CoachChoice($n : negorpos == 2,
    //Choice number is 1
    $c : choice == 1)
then
  //Set avatar speech
  $outputter.setText("It would help me with losing weight.");
  //Set new phase
  modify($dialogphase) {setPhase(3)}
  //Use outputter 2
  modify($outputter) {
    setOutputter2(true),
    setChangeExpr(true), //Use default expression
    setChangeEmo(true)}; //Use default emotion change
end
```

Listing 5.11: Diet Motivation Scenario 1 Phase 2: End of a Phase

Listing 5.12 shows the rule that sets the default emotion based on the negative/positive value of the Choice ID; the rules for setting the default expression are very similar. The system must be preparing to output to the Unity application, therefore one of the four outputters has to be active. The new emotion value is then set by calling the changeEmotion() method, shown in listing 5.13. The method requires both the current emotion value, as well as the negative/positive value from the Choice ID. The change variable will then be subtracted or added on to the current emotion value; this is based on the negorpos, where negorpos == 1 will subtract, and negorpos == 2 will add. The method will also not allow the new emotion value to exceed the minimum or maximum. The variables change, minimum, and maximum, can be adjusted fit the requirements.
rule "Change default emotionvalue based on CoachChoice (negorpos)"
when
  // Choice has to be postive/negative
  CoachChoice($n : negorpos > 0)
  // Check if system is currently ready to use one of the outputters
  $outputter : Outputter({
    outputter == true ||
    outputter2 == true ||
    outputter3 == true ||
    outputter4 == true,
    changeEmo == true} // Default emotion change should be used
  $emotionvalue: EmotionValue($e : emotion)
then
  // Call emotion change method
  $emotionvalue.changeEmotion($e, $n);
  // Changing emotion is now done, so set to false
  modify {$outputter} {setChangeEmo(false)};
end

Listing 5.12: Changing the Default Emotion Value Automatically

public void changeEmotion(double emotion, int negapos) {
  // Add/subtract the change based on negative or postive
  if (negapos == 1){
    this.emotion = emotion - change;
  }
  if (negapos == 2){
    this.emotion = emotion + change;
  }
  // Make sure that the value does not exceed min/max
  if (this.emotion > maximum){
    this.emotion = maximum;
  }
  if (this.emotion < minimum){
    this.emotion = minimum;
  }
}

Listing 5.13: Change Emotion Value Method

5.3 Implementation of Unity

The Unity application performs the following processes:

- Creating a TCP listener and making the server-socket through which the connection with the Java application is started.
- Setting up a stream reader and steam writer for receiving and sending to the Java application.
- Receive the command string, and then process the information that is received after the command.
- Send the choice ID of the choices made by the coach to the Unity application.
- Visualize the environment through 3d models, such as the avatar and the location it is placed in (i.e. living room).
- Create the GUI.
- Convert the speech string to and audio file using TTS.
- Create lip-sync for the avatar fitting the TTS audio.
- Play expression animations of the avatar.
• Use the emotion value to influence the environment. This is done by changing the colors of a lamp placed in
the "living room" of the avatar.

All the sample codes in section 5.3 are written in C# scripts, which are placed on objects in the game scene.

5.3.1 Unity Application Server-Socket

The server-socket, and stream reader/writer is created using the the code shown in listing 5.14.

```
1 IPHostEntry ipHostInfo = Dns.Resolve(Dns.GetHostName());
2 IPAddress ipAddress = ipHostInfo.AddressList[0];
3 IPEndPoint localEndPoint = new IPEndPoint(ipAddress, port);
4 //Create and start the Tcp Listener
5 TcpListener tcpListener = new TcpListener(port);
6 tcpListener.Start();
7 //Create the server socket when accepting the connection from the Java socket
8 Socket serverSocket = tcpListener.AcceptSocket();
9 networkStream = new NetworkStream(serverSocket);
10 //Set the stream reader to "fromJava"
11 fromJava = new StreamReader(networkStream);
12 //Set the stream writer to "toJava"
13 toJava = new StreamWriter(networkStream);
```

Listing 5.14: Creating the Unity Server-Socket

On every frame, the Unity application will check if a command has been received. The command is stored as a
variable string, and if/then rules decide which method has to be performed. For example, listing 5.15 shows the code
for what happens when the command outputFeedback is received.

```
1 //Receive the command by reading the string from the "fromJava"
2 command = fromJava.ReadLine();
3 if (command.Equals("outputFeedback")){
4     //Call the outputFeedback method () that shows the feedback in the GUI ) from the feedback C# script
5     feedback.outputFeedback(fromJava);
6     //After the command has been processed, make the command string null
7     command = null;
8 }
```

Listing 5.15: Receiving a Feedback Command
5.3.2 Objects and Visual Environment

![Figure 5.2: Visual Environment in the Editor (Non-Game View)](image)

The avatar is placed in a living room environment, as can be seen in figure 5.2. The design of the visual environment was not a focus of this project, so there is much room for improving the sense of realism. The same holds true for the avatar itself, which was not specifically built for this project. In order to create a more realistic situation, the avatar would have to look older. Again, in regards to the visual aspects of the avatar, there is also much room for improvements.

The environment consists of several objects, both visible and invisible. Following is a list of all the objects, including the name of the scripts that are placed onto them. For more information on how Unity objects, and scripts work, see section 3.3.2.

- **Main Camera**: Used for setting the game view through which the coach observes the application.
- **Directional Light**: Used for creating lighting in the scene.
- **Avatar**: 3D model that represents the avatar; the **Jaw** is a part of the model. The **AvatarAnimation** script is placed on the whole avatar. This script sends relevant values to the Unity Animator (for more information about the animations see section 5.3.3.1). The **Jaw** has the **SimpleSync** script placed on it, which is used for lip-sync (for more information see section 5.3.4). The **Jaw** also has the **TextToSpeech** script placed on it, and this script plays the generated TTS audio on the **Jaw**, which is necessary for the lip-sync script to function properly (for more information on TTS see section 5.3.3).
- **Lamp**: 3D model that represents a lamp. The **Lamp Light** is the light bulb part of the model. The **LightController** script is placed on the **Lamp Light**. This script controls the colors of the lamp.
- **Server**: Invisible object on which the **Server** script is placed. This script controls the creation of the socket and processes the commands.
- **GUI**: Invisible object on which the **Visuals** script is placed. This script controls all aspects of the GUI, from the interactivity to the style.
- **Description**: Invisible object on which the **Description** script is placed. This script contains the methods for processing the description data received from the Java application.
- **Feedback**: Invisible object on which the **Feedback** script is placed. This script contains the methods for processing the feedback data received from the Java application.
- **Emotion**: Invisible object on which the **Emotion** script is placed. This script contains the methods for processing the emotion value data received from the Java application.
- **Choices**: Invisible object on which the **Choices** script is placed. This script contains the methods for processing the Choice Content and Choice ID data received from the Java application.
• **FaceSpeech**: Invisible object on which the **FaceSpeech** script is placed. This script contains the methods for processing the speech and expression data received from the Java application, as well as generating the speech audio files.

• **Inputter**: Invisible object on which the **Inputter** script is placed. This script contains the methods for sending a Choice ID to the Java application.

• **Other**: The remaining objects are visible 3D models of the walls, floor, couch, pillows, and wall painting. These objects have no scripts assigned to them.

The standing lamp in the background is used for conveying the mood of the avatar using the emotion value. In the current parameter settings, the emotion value is between 0 and 100; a higher number indicates a more positive mood. The emotion value starts out at neutral, which is the half of the maximum (in this case 50). At the neutral value the lamp shines white, and has no color. At above neutral, the lamp shines with a green color; below neutral, the lamp glows red. The intensity of the color increases based on the emotion value; the higher the value, the brighter the green will be; the lower the value, the brighter the red will be. The color intensity is controlled by RGB values with a range of 0 to 1. The percentage difference of the current emotion value compared to the distance between neutral and minimum or maximum, is then reflected on the red or green values of RGB. For example, when the emotion value is 75, the green intensity has to be calculated. In this case, the relevant distance is between neutral and maximum, which is a distance of 50; and the distance of neutral to the emotion value, which is 25. This leads to a percentage difference of 25/50 = 0.5; resulting in a green value of 0.5.

### 5.3.2.1 GUI

![Figure 5.3: Graphical User Interface (Game View)](image)

Figure 5.3 shows a screen-shot of the interface as the coach would see it; this is called the **Game View**. The GUI consists out of several buttons and labels for the choices, feedback, and description. The choices are presented to the coach as buttons, that when pressed, call the method to send the Choice ID to the Java application. The choice buttons are placed on the bottom of the screen, and the amount of buttons are automatically generated based on the number of choices that are available. The description of the choices is shown in a label on the top of the screen and the font is blue so it can be easily distinguished from the choices. Whenever a negative choice is made the feedback button will show on the top of the screen, as can be seen in figure 5.4. When this button is pressed, the feedback will be shown in a label like in figure 5.5.
5.3.3 Text To Speech

For the TTS generation, an Android TTS Plugin\footnote{The Unity Android TTS plugin can be found on the asset store} is used. This plugin allows for the use of the internal TTS generator on an Android device. This means that TTS using the plugin cannot be used within the Unity editor, as this runs on a PC. When not running on an Android device, the avatar will make use of the TTS capabilities from the Google Translate site\footnote{When using Google Translate, the TTS is limited to 100 characters}. 

\footnotesize{© KONINKLIJKE PHILIPS ELECTRONICS NV 2015}
The Android TTS is capable of streaming the audio file immediately after it is generated. For the avatar system it’s necessary to play the audio at a certain time, when the avatar is supposed to react. Because of this, streaming is difficult in regards to the timing. To solve this issue, the TTS manager writes a WAV file to the internal memory of the Android device, so it can be played later when the timing is right. Even by generating a file which can be played at any time, the correct timing is still an issue. The generation of the WAV file requires a certain amount of time, and the other processes in the Unity application will keep running separately. This means that by the time the avatar requests to play the audio file, it has not been generated yet, which skips playing the audio file entirely. To solve this problem and extra dummy audio file has to be created. The TTS generator works using a queue system, which means that an audio file will only be generated when the previous file has finished. The second dummy audio file is for that reason, used to check whether the audio file containing the actual speech of the avatar is complete. The Unity application checks on every frame if the audio file has to be opened, and if the dummy file exists. If both conditions are met, the file with speech will be played. After the audio is done playing, both the speech audio file and the dummy file are deleted from the internal memory.

5.3.4 Lip-sync

In order to create a realistic feeling, it is necessary that the avatar’s mouth, moves in line with the audio file that was generated using TTS. An important aspect that this avatar system adheres to, is flexibility. This is why the speech is generated on the fly, rather than recorded beforehand. Of course, this also means that lip-sync becomes an issue, as it very difficult to create realistic lip-sync automatically. But creating lip-sync by hand, to match pre-recorded audio, would go against the concept of flexibility. In this case, it is necessary to sacrifice realism to attain flexibility. For that reason, the choice was made to create lip-sync automatically with no use of phonemic mouth movements; the lips do not fit with the words being said, rather the mouth simply opens and closes. For the automatic lip-sync of the avatar, the SimpleSync Lite plugin is used. The SimpleSync script is placed on the object that has to move, which is the jaw of the avatar. The audio file is also placed on the jaw. The script checks the volume of any audio that plays on the object, in other words, when the WAV file plays while placed on the jaw, the jaw will move up and down according to the volume level. This is a very simple form of lip-sync were the mouth only opens and closes. It is not realistic, but it fulfills the purpose.

5.3.5 Implementation of Faceshift in Unity

Faceshift was used to record real-life facial expressions which were placed on the avatar. Several different types of emotions were recorded, including: positive reactions, negative reactions and idle expression that are neutral, positive or negative. The idle animations are longer, and play when the coach is making a choice while the avatar is not talking. Around 20 animations were recorded in total, of which 11 fitting animations were chosen. Positive expressions are defined by the avatar looking at the coach, making affirmative gestures and smiling. Negative expressions are represented by disinterest, looking away, and negative gestures. These expression characteristics were based on the information given by the domain experts.

5.3.5.1 Animations

The Faceshift plugin uses the files that were created with the Faceshift software, and generates an animation file for the 3D model of the avatar. This animation is placed within the Animator of the Unity editor, which is used to decide what animation to play at certain times. Figure 5.6 shows the overview of the different animations within the Animator, as well as the possible transitions between them.

---

3The Unity SimpleSync Lite plugin can be found on the asset store.
4The Unity faceshift plugin can be found on the asset store.
Descriptions and conditions of the 11 animations are as follows:

- **neutral_long**: Idle neutral expression. Plays and loops this expression when the emotion value is between 20-80 and the avatar is not talking.
- **smile_long2**: Idle positive expression. Plays and loops this expression when the emotion value is greater than 80 and the avatar is not talking.
- **look_away_eyes_long**: Idle negative expression. Plays and loops this expression when the emotion value is less than 20 and the avatar is not talking.
- **yes**: A short animation where the avatar smiles and nods yes. Plays when the avatar is talking, the expression state is positive and the expression ID is 1. This animation is used during the reflection phases, where the avatar will confirm or deny the summarizations and reflections of the coach.
- **smile2**: The avatar smiles. Plays when the avatar is talking, the expression state is positive and the expression ID is 2. The standard animation that plays when an avatar reaction is positive, excluding the reflection phases, or when it is overwritten.
- **open_smile**: The avatar smiles broadly. Plays when the avatar is talking, the expression is positive and the expression ID is 3. Can overwrite the standard positive animation in situations where a more stronger positive expression is required.
- **smile_short**: A short animation of the avatar smiling. Plays when the avatar is talking, the expression is positive and the expression ID is 4. Can overwrite the standard positive animation in situations where a short positive animation is required.
- **no2**: A short animation where the avatar shakes no. Plays when the avatar is talking, the expression state is negative and the expression ID is 1. This animation is used during the reflection phases, where the avatar will confirm or deny the summarizations and reflections of the coach.
- **look_away_eyes**: The avatar moves the head and eyes to the side, looking away from the coach. Plays when the avatar is talking, the expression state is negative and the expression ID is 2. The standard animation that plays when an avatar reaction is negative, excluding the reflection phases, or when it is overwritten.
- **angry2**: The avatar grimaces and look angry. Plays when the avatar is talking, the expression is negative and the expression ID is 3. Can overwrite the standard negative animation in situations where a more stronger negative expression is required.
- **uninterested_short**: A short animation of the avatar looking disinterested. Plays when the avatar is talking, the expression is negative and the expression ID is 4. Can overwrite the standard negative animation in situations where a short negative animation is required.
5.3.5.2 Animation Transitions

The animations transition from idle to reaction when the avatar starts talking. After the avatar is done talking, the animation transitions back from reaction to idle. Which reaction animation to transition to, is dependent on the expression state and ID, and which idle animation to return to is based on the current emotion value. It’s not possible to transfer from one idle animation to another; the same holds true for reaction animations. An example of such a transition in the Unity editor is shown in figure 5.7.

Figure 5.7: Animation Transition

In this particular example the transition is from `look_away_eyes` to `neutral_long`. As explained in section 5.3.5.1, in order to play the neutral idle animation, the avatar must not be talking, and the emotion value has to be between 80-20. This is set in the Unity editor under the transition conditions, as can be seen in figure 5.7. The `talking` condition has to be false and `emoValue` has to be greater than 20 and less than 80. The `emoValue` is received by the socket connection with the Java application. The `talking` boolean is dependent on the generated TTS audio file. While the file is playing, the boolean is true; while the file is not playing the boolean is set to false.

Another important aspect of animation transitions, is the transition duration. In figure 5.7 this duration is represented by the blue section on the time-line. It sets how long it takes for one animation to flow in to the next. A short duration will lead to erratic changes in movement, whereas a longer duration will blend the animations together, leading to a more smoother transition. On the other hand, when the transition duration is too long, the animations will blend too much, and the motions lose their characteristics. For this avatar system, the transition duration from idle to reaction animations are shorter, while the duration from reaction to idle is longer. Using these setting, the animation blending is done smoothly, while still playing the animation properly.

5.4 Discussion

In this chapter the focus was on explaining the different stages of implementation:

- Showing how previously explained concepts are realized in the implementation.
The connection between the Java and Unity application, including how communication takes places.

The Java side of the system:
- Creating the socket.
- Creating and setting the data in the Knowledge Base.
- Giving sample code that shows how a dialog tree is implemented.
- Explaining how the different Inputters and Outputters work.

The Unity side of the system:
- Creating the server socket and explaining how Unity understands how to process the received data.
- The visual environment, including the standing lamp that reacts to the emotion value.
- The design of the GUI.
- The creation of the TTS audio, including the lip-sync.
- The integration of Faceshift animations in Unity.

In regards to any potential improvements, the design of the behavior rules could be more streamlined so that they can be easily used as templates when creating additional scenarios. For this system, the choice was made to use several different types of outputters and inputters to control the flow of communication between Java and Unity. This choice was made because at different times, Unity requires different types of information; for example, at the beginning of a phase the Unity application does not need info about the facial expression. Another choice would be to send all data at all times, this would make it so that less outputters are required, meaning the creation of the rules can be more streamlined. In other areas, the rules might also be made more generic, with the goal of making implementation of many scenarios easier.

Visually the system can be improved as well. The design of the environment and avatar can be made more realistic in regards to the setting of a coaching session. More facial animations can be added, as well as making the facial animations more realistic. One crucial part is better lip-sync, although this may be difficult to improve considering the current technology. An alternative TTS system that sounds less robotic can also improve the realism of the avatar.
6 Evaluation

This chapter discusses the evaluation of the avatar system. The avatar system as described in chapter 3, chapter 4 and chapter 5 was presented to several experts on the MAY profile and motivational interviewing. The goal of these interviews were to follow-up on the first meetings held at the beginning of the project, as well as a means of receiving criticisms on the results of avatar prototype. The feedback interviews were held with four domain experts at separate occasions. An example session was presented, and the experts gave their thoughts and opinions. Further details of the first meeting can be seen in appendix A and the second meeting in appendix B.

6.1 Areas of Improvement

The criticisms and suggestions for improvements by the experts can be divided into three areas:

- Avatar behavior
- Problem scenarios
- User experience and interface

In the following sections, the different areas of improvement will be discussed in more detail.

6.1.1 Avatar Behavior

The avatar as it behaves in the prototype always gives a satisfactory answer, even if a negative choice was made by the coach. In a realistic situation the patient does not always have the right things to say, they might not give an answer at all. It is also necessary for the coaches to learn how to deal with situations where the patient is unclear, so it is important that the avatar can reflect this patient behavior to challenge the coach’s skills. This means that the avatar should be capable of showing fuzzy behavior, allowing the coach an opportunity to practice these skills. An example of fuzzy behavior is, when a coach asks for reasoning when the patient gives a score on how motivated/confident/skilled they feel, but the patient themselves do not know the reason, therefore, they cannot answer properly. Creating fuzzy behavior within an avatar can be challenging, and it would require the addition of variables that control the randomness of fuzzy behavior. It might be possible to work with a chance based system, which occasionally changes the “expected answer” into a “non answer”.

6.1.2 Problem Scenarios

The avatar system as it functions now, can be used to train the basic concepts of the MAY profile as well as motivational interviewing. The current problem scenarios are not in-depth enough to train more advanced skills. It is necessary that domain experts, who are familiar with the type of advanced skills the coaches need to learn, write the problem scenarios. These experts know which skills are more difficult to attain, and can adjust the problem scenarios is such a way that the coaches can train and improve these skills.

The current problem scenarios also contain several mistakes regarding the correct application of motivational interviewing. Again, an expert of motivational interviewing is required to create problem scenarios that teach the coaches the correct skills.

6.1.3 User Experience and Interface

In the current demo, the coach interaction with the avatar is done through a set of predetermined choices presented on a screen, and chosen using a touch-screen interface. This method of communication is unrealistic in terms of
experience a coach has during an actual session. Instead of presenting predetermined choices, it would be better to have more input from the coaches themselves, allowing them to formulate their own choices. This might be implemented using keywords to convey to the knowledge base what the coach wants to communicate. For example, certain words that are negatively associated with motivational interviewing (e.g. righting reflex words), can be noticed by the avatar and so a proper reaction can be created. On top of that, rather than using a touchscreen it would be more realistic to use some form of speech recognition, allowing the coach to simply speak to the avatar, giving a more immersive experience.

Currently, the order of the phases in a session is also predetermined. Since the coaches have to learn how to make a session flow properly, it is a good exercise to let the coach choose what phase to go to next. If the phase order that the coach chooses is vastly different from the MAY profile documentation, feedback can be given explaining what the correct phase order would be. This addition would be fairly easy to implement in the current avatar demo.

As for the visual interface, in the prototype demo the feedback is shown at the top of the screen when the coach requests the feedback to be shown. This means it is very much possible for coaches to skip the feedback. Since the feedback is very crucial in helping understand what went wrong, it may be better to give mandatory feedback. There should be a clear indication that a wrong choice was made. Feedback should also not be stretched out over the entire screen, since this makes reading it unpleasant.

Another point of improvement on the visual interface is to place the description of the current phase at a location closer to the choice buttons. This way coaches are more likely to read the description and fully understand the choice they are making.

6.2 Implemented Improvements

After receiving feedback from the domain experts it was possible within the time constraints to implement some of the smaller changes in regards to the interface. Feedback was changed to be mandatory, showing up in the middle of the screen whenever a wrong choice is made. The feedback is shown in a smaller text-box to improve readability, and the coach has to press an acknowledge button to indicate that they have read the feedback. The background color changes, and the animations pause to clearly communicate to the coach that a negative choice was made. The improvements for the feedback can be seen in figure 6.1. The location of the phase description was changed to be placed at the top of the choice buttons, the font color remains the same to indicate that it is different from the choices. The improvement for the description can be seen in figure 6.2.
6.3 Discussion

This chapter discusses the evaluation of the avatar system. The evaluation of the avatar was based on feedback received from domain experts on the MAY profile and motivational interviewing. Some of the more easier to implement feedback was applied to the system. These minor changes are done within the user interface, which is relatively simple to do. More significant changes to the actual system inner workings would require more time. In summarization the following changes would improve the system:

- The coach does not currently have enough input in order to simulate a realistic coaching session. Instead of using a touchscreen, voice recognition could lead to a more fitting experience. Of course, when using voice recognition it would not be possible to have predefined choices, and so keywords could be used to check if what coach says is positive or negative. On the other hand, the problem arises that the system might not recognize or misinterpret what the coach is saying. It is also possible that the session becomes incoherent.
- Coaches should be able to choose the order of phases. This way they also learn the session flow.
- Scenarios need to be more in-depth and written by domain experts on the MAY profile and motivational interviewing.

For some of the more in-depth domain issues, such as the problem scenarios or the avatar behavior, domain experts would be required to implement the changes in the correct way. The visual interface could also be improved by having a visual expert design the GUI is such a way that the experience for the coaches is the best possible. Outside of the GUI, the visual graphics can also be improved, for example, by having a better 3d avatar model.
7 Conclusion

This thesis describes the design of an avatar patient that can train health coaches in the use of the MAY profile and motivational interviewing. The avatar consists of a dialog system where coaches can interact with a 3d model of a patient using a tablet. Communication from coach to avatar takes place using predefined choices and a touchscreen. The avatar will react to the choices using dynamically generated text-to-speech, facial expression, and lip-sync. A session with the patients avatar consists out several phases based on the MAY documentation, and the script of a phase is represented through a dialog tree.

This chapter concludes the thesis, giving a summarization of all the different parts that work together in order the realize the avatar prototype. The strength of the system are discussed afterwards, as well as potential improvements for the future.

7.1 Summarization of Concepts to Avatar Realization

The following concepts were used to realize this avatar prototype:

- The MAY profile; a psychological profile of the patient. The avatar’s behavior is dependent on the MAY profile. The coach has access to the MAY system of the avatar. The ultimate goal is to familiarize the coach with the use of the MAY profile.
- Motivational Interviewing; a method of coaching that gives autonomy to the patient. One of the goals is teaching the coach to use to principles of motivational interviewing. The behavior of the coach will change based on whether the coach applies motivational interviewing properly.
- System Architecture designed for this avatar system (figure 3.2); an architecture of the avatar system that shows how concepts interact with each other.
- Knowledge Base; a method for a computer to store information. Used for storing the data of the avatar’s MAY profile as well as the choices that the coach makes throughout the session.
- Rule Base built for this avatar system; a set of rules to control the behavior of a system. Used for controlling the coaching session, behavior of the avatar, and visual interface. The rules trigger based on the choices that the coach makes.
- Visualization of the patient avatar; the interface that is presented to the coach. Unity, a 3d engine, was used to create the interface. The interface also includes a 3d model of the avatar, the vocalization of text through Text-To-Speech and animations for lip-sync and facial expressions.
- Communication between machines; The knowledge/rule runs on a computer, whereas the visualization of the 3d engine takes place on an Android tablet. Communication between machine takes place through the creation of TCP sockets on a network.
- Emotion Value; A value created for this avatar system to measure the mood of the session. The emotion value is a representation of a session over time. The emotion value influences both the behavior of the avatar, as well as the visual environment.
- Feedback; given to the coach when they make a negative choice. Feedback is crucial for the coaches to understand the reason for why a choice might not follow the principles of the MAY profile or motivational interviewing.

7.2 Strengths of the System

The strength of the avatar system, is its dynamic dialog system; flexible and easy to adjust or expand because the different concepts of which the avatar consists, function together dynamically. Through the use of a knowledge
base, it becomes trivial to add new MAY profile concepts. In the current demo not every MAY profile concept was implemented, but it is certainly easily possible to implement more. By adding new information to the knowledge base, it also becomes possible for the rule base to apply this knowledge. With the framework as designed in the current prototype, many of the rules are similar to each other. This means that it is simple to add more rules to expand on a coaching session. Extra dialog phases, or more possible choices can be implemented easily. Of course, there is still a large amount of rules, since sessions are not completely generic. This makes rule adjustments confusing for those who do not understand what every rule does. Changing one rule might also make it necessary for slight adjustments to another; for example, adding an extra phase between two existing phases, means that the phase count of the previous and following phased have to be adjusted. This means there is still areas of improvement on the ease of usability of the rules. This is especially crucial if the coach/MAY profile/Motivational Interviewing domain experts are required to use the rules.

Another aspect that is dynamic, is the on the fly generation of speech and lip-sync. This means that it is easy to add more and varied avatar reactions, with no need to record extra audio or creating lip-sync, as this is done automatically by the Unity application. The expressions are separate animation files that can be called in combination with any line of avatar dialog. A facial expression is therefore not tied to the speech of the avatar, allowing for a large amount of combinations. For example, the same sentence can be said using a happy or angry expression depending on which animation file plays. It is also possible to extend the set of animations greatly, and for example, create a sets of animations for different situations that can be played randomly. This will give the avatar more variation when talking.

Because of the use of Faceshift, it is simple to create a large set of facial expressions. Creating animations by hand is very time consuming, and requires expertise in the area of 3d animation. With Faceshift, only the face of the actor is necessary, and it becomes possible for a non-animator to create a large set of animations to be used by the avatar.

The use of the Drools knowledge/rule base system is more efficient compared to a naive method. Due to the use of the Rete algorithm, and the creation of the network of nodes, the speed in which the rules are able to fire is faster. For a small application such as the current demo, this increase in speed is not noticeable. Potential future improvements could expand the amount of rules greatly, in which the avatar would benefit from the use of the Rete node network. In regards to the creation of the actual rules, Drools allows for preventing a large amount of nested if-then statements (spaghetti code). This makes it easier to change or add new rules, without getting lost in nested statements.

### 7.3 Future Improvements

Many of the potential future improvements based on the expert feedback were already described in chapter 6. In regards to actual system design itself, such as the architecture, session flow, and the implementation choices that were made, there may still be areas of improvement. Following is a list of changes/future implementations that can improve the use of the system.

- Scenarios need to be easy to create for those that write the scenarios. This can be done by having a rule template, which allows for easy implementation of scenarios, even if a writer does not have the detailed knowledge on how the framework exactly works. Ideally, a rule creation tool should be developed that allows for the easy creation of rules within a user friendly GUI.

- Rules written in Drools can be made more generic and reusable. Currently the amount of rules per scenario is very large, it is very necessary to split the rules into multiple files in order to make it manageable. By writing the rules in a different way, it may be possible to reduce the amount of rules.

- The responses of the avatar are mostly predetermined. This makes it necessary for the scenario writer to write the entire scenario from start to finish. By making the avatar more intelligent, it could create responses by itself, reducing the work of the scenario writer. This may proof to be difficult to implement, since creating a conversation with an artificial intelligence on subjects such as personal issues can be difficult to do. Current existing talk-bots cannot hold a consistent conversation over longer periods of time. It may be more achievable to combine AI generated conversation together with predetermined responses.

- The rules should be as separated from each other as possible; adjusting one rule should not require the adjustment of another. Currently, adjusting/adding rules requires changing other rules, ideally the rules have to be rewritten in such a way that this is not necessary.
• Lip-sync can be improved. The current technique which uses volume levels to rotate the jaw of the 3d model, results into a very basic and unrealistic automatic lip-sync. Ideally, rather than using volume levels, it would be better to analyze the audio file on what types of phonemes were used. Then based on this analysis, the lips can be animated more realistically. With current technology this may still be impossible to do on-the-fly, but with future developments realistic lip-sync might be possible dynamically, and automatically.

• Text-to-speech can be improved. The current system makes use of Android’s build in TTS, which does not sound very human-like. By implementing a more emotional sounding TTS, the dialog can fit better with the facial expression, leading to a more realistic experience for the health coaches.

Finally, the system needs to be tested using actual real life coaches. Then based on the feedback from the coaches, the system could be improved even further.
References


A Summary First Meeting Domain Experts

The first meeting was held in January 2015, two weeks after the start of the project. The design ideas at that point were presented to 3 domain experts in separate meetings. The presentation consisted out of showing 6 slides representing a basic interaction between coach and the virtual patient avatar. These slides can be seen in figure A.1 to A.6. Early designs of figure 3.1 and 3.2 were presented as well.

Figure A.1: Present Coach with a Problem Patient (Analyst Phenotype)
Figure A.2: Present Choices to the Coach

Figure A.3: Coach Chooses Analyst Choice
Figure A.4: Patient Responds Positively

Figure A.5: Coach Chooses Sensitive Choice
In addition to the slides, some basic examples of problem scenarios were also presented. This was to receive feedback on whether these types of scenarios are considered realistic in coaching sessions with a patient. Only Healthy Diet problem scenarios were presented. The example can be seen in text box A.

**Healthy Diet**

**Motivation:**
- Scenario 1: Patient wishes to lose weight, but does not feel motivated.
- Scenario 2: Patient wishes to lower blood pressure, but does not feel motivated.

**Confidence:**
- Scenario 1: Patient has tried to diet in the past but failed, this has lowered their confidence.
- Scenario 2: Patient is not supported by family members, this has lowered their confidence.

**Skills:**
- Scenario 1: Patient has very little experience with cooking, this has made feel unskilled.
- Scenario 2: Patient has very little knowledge on healthy recipes, this has made them feel unskilled.

Appendix Text 1: Scenario Example

After presenting the design concepts, feedback was received. The following list contains the important points that were used to improve the avatar system design:

- There is too much focus on phenotypes. Focus more on the subject of the discussion, in this case healthy diet.
- Two stages of discussion. Choose the subject (ex. healthy diet), then choose the method (motivation, confidence, skills); first "what", then "how".
- Explain to the coach what went wrong after a negative response.
- Showing avatar dialog as text is not realistic.
- What is said is more important than a facial expression. Speech will override the expression, if speech and expression show a discrepancy.
• Coach must be given concrete choices during the session with the avatar, so that they can later use them in real life situations.
• Negative avatar reactions should be apathetic, and not overly emotional. Unless there is a major conflict between coach and patient.
• Positive avatar reactions should be enthusiastic.
B Summary Second Meeting Domain Experts

The second meeting was held at June 2015, nearing the end of the project. The finished avatar prototype was presented to four domain experts in separate meetings. A single session with the virtual patient avatar running live on a tablet was shown together with a part of the corresponding dialog tree. The specific MAY profile characteristics for the demo were Healthy Diet - Skills. The following list of feedback was received at the meeting:

- Structure of a session should also be chosen by the coach in order to teach them what the correct structure is.
- Description of a phase is not visible enough.
- Feedback needs to be acknowledged.
- Feedback needs to be more visible.
- Feedback needs to be rewritten at times (ex. make less harsh).
- Dialog trees are not always correct (need to be rewritten by an expert).
- The avatar must be more apathetic occasionally (more variation is avatar responses).
- Avatar needs to respond with more unclear/fuzzy responses, since this is more in line with how an actual patient behaves.
- Playback of audio is not very important, so does not need to be added.
- Facial expressions need to be more expressive.
- Dialog must fit the facial expression.
- System needs to allow more input of the coach.
- System needs to be tested on real users.
- System can be useful.
- System is more for training basic knowledge, more specialized problem scenarios are necessary to train in-depth skills.
- It would be useful for coaches to go back in time during a session. This way they can redo sections that might find difficult or do not understand.
C Dialog Trees

C.1 Explanation of Symbols

Figure C.1: Phase symbol

Figure C.1 indicates the name of the current phase of the dialog tree. Also indicates when the next phase begins.

Figure C.2: Scenario symbol

Figure C.2 indicates the name of the current scenario.

Figure C.3: Choice symbol

Figure C.3 indicates the possible choices the coach can choose from. The content of the choice will start with a label with two numbers. For example $(x,y)$, where $x$ indicates whether the choice is positive (1) or negative (2), and $y$ indicates the number of the choice. So the label $(2,3)$ would indicate the third positive choice within the current phase. The label can also indicate a choice related to a phenotype. For example, [Analyst Approach] means that content of the choice is geared toward analysts. The coaches will not be able to see the labels in the application.

Figure C.4: Positive symbol

Figure C.4 indicates a positive response from the avatar. The content will be spoken by the avatar using a text-to-speech engine.
Figure C.5: Negative symbol

Figure C.5 indicates a negative response from the avatar. The content will be spoken by the avatar using a text-to-speech engine.

Figure C.6: Feedback symbol

Figure C.6 indicates the feedback that the system gives to the coach when the coach makes a negative choice. This feedback will be displayed on screen as a blurb of text.

Figure C.7: Fixed arrow symbol

Figure C.7 indicates a forced and fixed reaction by the system, the coach has no control. For example, when a negative choice is made a fixed negative reaction will be performed by the avatar.

Figure C.8: Choice/Or arrow symbol

Figure C.8 shows the 'choice' and 'or' arrows. The 'choice' arrow indicates a moment in which a coach is presented with multiple choices. The 'or' arrow indicates that one choice can lead to different reactions based on properties of the avatar or earlier choices made by the coach.

Figure C.9: Feedback arrow symbol

Figure C.9 indicates a connection between a negative response from the avatar and a feedback to be displayed on screen.
C.2 Healthy Diet Dialog Trees

C.2.1 Motivation - Scenario 1 - Weight Loss

Figure C.10: Motivation - Scenario 1: Weight Loss - Phase 1
Figure C.11: Motivation - Scenario 1: Weight Loss - Phase 2 and 3
Figure C.12: Motivation - Scenario 1: Weight Loss - Phase 4
Figure C.13: Motivation - Scenario 1: Weight Loss - Phase 5
Figure C.14: Motivation - Scenario 1: Weight Loss - Phase 6
Figure C.15: Motivation - Scenario 1: Weight Loss - Phase 7 and 8
Figure C.16: Motivation - Scenario 1: Weight Loss - Phase 9
Figure C.17: Motivation - Scenario 1: Weight Loss - Phase 10 and 11
C.2.2 Motivation - Scenario 2 - Blood Pressure

Figure C.18: Motivation - Scenario 2: Blood Pressure - Phase 1
Figure C.19: Motivation - Scenario 2: Blood Pressure - Phase 2 and 3
Figure C.20: Motivation - Scenario 2: Blood Pressure - Phase 4
Figure C.21: Motivation - Scenario 2: Blood Pressure - Phase 5
Figure C.22: Motivation - Scenario 2: Blood Pressure - Phase 6
Figure C.23: Motivation - Scenario 2: Blood Pressure - Phase 7 and 8
Figure C.24: Motivation - Scenario 2: Blood Pressure - Phase 9
Figure C.25: Motivation - Scenario 2: Blood Pressure - Phase 10 and 11
C.2.3 Confidence - Scenario 1 - Past Experience

Figure C.26: Confidence - Scenario 1: Past Experience - Phase 1
Figure C.27: Confidence - Scenario 1: Past Experience - Phase 2
Figure C.28: Confidence - Scenario 1: Past Experience - Phase 3

© Koninklijke Philips Electronics NV 2015
Figure C.29: Confidence - Scenario 1: Past Experience - Phase 4

(Phase 4) Determine the circumstances that are most hindering to a healthy diet

(Scenario 1) Past Experience

(1,1) Is it difficult for you to maintain a healthy diet?

Yes it is.

(2,1) Can you explain to me what you think is difficult about maintaining a healthy diet?

My fear of failing again is what makes it difficult for me to feel confident.

(1,2) Don't you think that you should be able to change things about those difficulties?

I don't know. I guess I am a bit too focused on the past. Maybe I should focus on the future more.

(2,2) What do you think you can change about those difficulties?

Maybe I should stop focusing so much on what happened in the past.

Go to the next phase

This is not an open-ended question. By forcing the patient to confirm or deny, the patient is not motivated to think for themselves. It also is judgemental, making the patient feel guilty.

This is not an open-ended question. By forcing the patient to confirm or deny, the patient is not motivated to think for themselves.
Figure C.30: Confidence - Scenario 1: Past Experience - Phase 5
Figure C.31: Confidence - Scenario 1: Past Experience - Phase 6
Figure C.32: Confidence - Scenario 1: Past Experience - Phase 7

(SpecialPhase) Raise patient’s awareness of own unhealthy practices

Condition 1.1: In the next week I want you to make a food diary which keeps track of all the things you’ve eaten that week.

Condition 2.1: Would you like to make a food diary to keep track of all the things you’ve eaten in a week?

Sure, I guess I can do that.

Yes, I can do that.

Try not to force a plan onto the patient. Ask the patient for permission to give advice and let the patient decide on their own if they want to follow the advice.

Go to the next phase.
Figure C.33: Confidence - Scenario 1: Past Experience - Phase 8

(Phase 8) Determine strategies or actions that a patient can do when a difficult situation arises.

(Scenario 1) Past Experience

1. You should consult family when you feel that you are in a difficult situation.
   - Ok, I'll keep that in mind.

2. When you find yourself in a difficult situation, you should remember your past successes.
   - Again, I feel like my past experiences is what holds me back. I want to think about the future.

3. Imagine a situation in which a difficult situation arises, in what way do you imagine yourself to deal with it?
   - I feel like I could talk to my family to give me more confidence.

This is a righting reflex, try not to tell the patient what to do, give them space to think for themselves and let them find out what would work best in their situation. Open-ended questions are preferable.

The patient has said that they feel don't feel confident due to a past experience. When the patient runs into difficulties it would not be helpful to think about the past. The patient will feel that you show a lack of understanding the situation. Make sure to pay close attention to what the patient says. This is also a righting reflex, try not to tell the patient what to do, give them space to think for themselves and let them decide what to rely on in a difficult situation. Open-ended questions are preferable.

Go to the next phase.
Figure C.34: Confidence - Scenario 1: Past Experience - Phase 9 and 10
C.2.4 Confidence - Scenario 2 - Circumstances

(Phase 1) Identify the lack of confidence for X

(1.1) I really think you can do it.

(1.2) You should be more confident. I know you can do it.

(1.3) Why such a low number, why not higher?

(2.1) On a scale of 1 to 10, can you tell me how confident you are about it?

My score would be (number).

I don't know. Most of the time I don't think I can do anything about.

Yeah, I know should be.

Well, I guess at times I do feel like I can do (x). I'm still somewhat confident, that's why I gave it a (number).

This is a righting reflex. By telling patients what to do and how to feel you will push them away. Open-ended questions are preferable, as they require the patient to think about their own values and gives them an opportunity to think for themselves.

This is another righting reflex. It also has a judgemental tone. Remember to ask open-ended questions and motivate the patient to find their own answers. Also try to prevent making judgemental statements that make the patient feel guilty.

This statement implies that the number should be higher, which is judgemental and makes the patient feel guilty. The correct approach would be to ask why the number is high, this way the patient is motivated to think about why they might feel confident.

Go to the next phase

Figure C.35: Confidence - Scenario 2: Circumstances - Phase 1
Figure C.36: Confidence - Scenario 2: Circumstances - Phase 2
Figure C.37: Confidence - Scenario 2: Circumstances - Phase 3

This is a righting reflex. By telling patients what to do and how to feel you will push them away. Open-ended questions are preferable, as they require the patient to think about their own values and gives them an opportunity to think for themselves.

(Phase 3) Determine how the patient deals with failure

(1,1) Even when you fail at the diet, you shouldn’t give up that easily, keep trying.

But that’s the problem. If I fail, I feel like I want to give up.

(2,1) Can you tell me what you feel when you fail your diet?

I feel like I failed myself. It feels like I’m giving up.

(1,2) You should ask family or friends to help you with dealing with the feeling of failure.

Like I said before, my family is not very supportive.

(1,3) You should remember your past experiences of success in order to help you with the feeling of failure.

I suppose I can do that.

(2,2) What do you think you can do to help deal with the feeling of failure?

I know that I’ve been able to achieve difficult things in the past. Remembering that might make me feel better about myself.

This is a righting reflex, try not to tell the patient what to do, give them space to think for themselves and let them find out what would work best. Open-ended questions are preferable.

Go to the next phase
Figure C.38: Confidence - Scenario 2: Circumstances - Phase 4
Figure C.39: Confidence - Scenario 2: Circumstances - Phase 5
Figure C.40: Confidence - Scenario 2: Circumstances - Phase 6

(Phase 6) Determine strategies to make healthy diet easier to implement

(1,1) You should be able to make a small change to your diet.

This is a judgemental statement, making the patient feel forced to do something they might not agree to. Try to motivate the patient to create a change that they feel comfortable with.

(2,1) Can you think of a small change in your diet that you think you are capable of keeping up with?

I would feel confident about eating less snacks between meals. I think I would be capable of doing that. Is that change good enough?

(1,2) Yes, that would be fine. Though I think you could do better.

Try to do affirmations to validate the patient. Recognize a personal strength and lift it up. In this case it would have been better to compliment the personal change the patient has decided on for the next week.

(2,2) That's a very nice goal to work towards.

I see.

Great.

Go to the next phase
(SpecialPhase) Raise patient's awareness of own unhealthy practices

(1,1) In the next week I want you to make a food diary which keeps track of all the things you've eaten that week.

Sure, I guess I can do that.

Try not to force a plan onto the patient. Ask the patient for permission to give advice and let the patient decide on their own if they want to follow the advice.

(2,1) Would you like to make a food diary to keep track of all the things you've eaten in a week?

Yes, I can do that.

Go to the next phase

(Scenario1&2) Past & Circumstance

Figure C.41: Confidence - Scenario 2: Circumstances - Phase 7
Figure C.42: Confidence - Scenario 2: Circumstances - Phase 8
Figure C.43: Confidence - Scenario 2: Circumstances - Phase 9 and 10
C.2.5 Skills - Scenario 1 - Lacking Cooking Skills

Figure C.44: Skills - Scenario 1: Lacking Cooking Skills - Phase 1

This is a righting reflex. By telling patients what to do and how to feel you will push them away. Open-ended questions are preferable, as they require the patient to think about their own values and gives them an opportunity to think for themselves.

(1.1) You should try to improve your skills.

Yeah, I know. I'm really bad at cooking healthy.

(1.2) You should try to improve, it really is important.

Like I said, I know I should improve. But I don't really know how to do that.

(1.3) Why such a low number, why not higher?

I really don't know much about cooking.

(2.1) On a scale of 1 to 10, can you tell me how skillful you feel about X.

My score would be (number).

(2.2) Why such a high number, why not lower?

I think I can still make some very basic dishes, their just not very well made or healthy, that's why I still gave it a (number).

This statement implies that the number should be higher, which is judgemental and makes the patient feel guilty. The correct approach would be to ask why the number is high, this way the patient is motivated to think about why they might feel skilled.

Go to the next phase.
Figure C.45: Skills - Scenario 1: Lacking Cooking Skills - Phase 2
Figure C.46: Skills - Scenario 1: Lacking Cooking Skills - Phase 3
Figure C.47: Skills - Scenario 1: Lacking Cooking Skills - Phase 4

- (Phase 4) Give knowledge about healthy diet skills
  - (Do not ask for permission to give info)
  - (Scenario 1) Lacking cooking skills
  - (2,1) Can I give you some information on how you could improve your healthy diet skills?

- (1,1) You could try going to a cooking class to help you learn the basics of cooking.
  - (1,2) You could try watching online tutorial videos on a tablet or computer to help you learn the basics of cooking.
  - (If technology comfort is high) That's seems like a hassle. I would prefer something easier accessible, but I'll try anyway.

- Okay, that sounds good.
  - (If technology comfort is low) I'm not comfortable with using a computer, but I'll give it a try.

- This particular patient is not comfortable with using technology. Advising a non-technical approach would have been better. Try to remember the preferences of the patient.

- Go to the next phase
Figure C.48: Skills - Scenario 1: Lacking Cooking Skills - Phase 5
(Phase 6) Identify which change the patient is willing to make

(1.1) You should apply your new skills from the classes and try to cook some dishes.

Yeah, I guess I can do that. I could take a class or tutorial in the weekend.

I would like to try to take some cooking classes or tutorial in the weekend when I have time, and maybe try some cooking for myself.

(2.1) What do you think you can change about your diet?

(1.2) In the next week you should be able to create some progress.

I guess I could start with cooking a dish a week. Is that enough?

I would like to start with cooking one dish a week. Would that be enough?

(2.2) What kind of progress would you like to see in the next week?

(1.3) Yes, that would be fine. Though I think you could do better.

(2.3) That's a very nice goal to work towards.

Go to the next phase

This is a frightening reflex. Try not to push your own ideas on the patient. Allow the patient to think for themselves and make a change that they agree with.

This is a judgemental statement. Making the patient feel forced to do something they might not agree to. Try to motivate the patient to create a change that they feel comfortable with.

Try to do affirmations to validate the patient. Recognize a personal strength and lift it up. In this case it would have been better to compliment the personal change the patient has decided on for the next week.

I see.

Great.

Figure C.49: Skills - Scenario 1: Lacking Cooking Skills - Phase 6
Figure C.50: Skills - Scenario 1: Lacking Cooking Skills - Phase 7

(SpecialPhase) Raise patient’s awareness of own unhealthy practices

Choice

(1,1) In the next week I want you to make a food diary which keeps track of all the things you’ve eaten that week.

Sure, I guess I can do that.

Try not to force a plan onto the patient. Ask the patient for permission to give advice and let the patient decide on their own if they want to follow the advice.

Choice

(2,1) Would you like to make a food diary to keep track of all the things you’ve eaten in a week?

Yes, I can do that.

Go to the next phase

Lacking cooking skills & Recipes
Figure C.51: Skills - Scenario 1: Lacking Cooking Skills - Phase 8 and 9
C.2.6 Skills - Scenario 2 - Lacking Recipe Skills

Figure C.52: Skills - Scenario 2: Lacking Recipe Skills - Phase 1
Figure C.53: Skills - Scenario 2: Lacking Recipe Skills - Phase 2
(Phase 3) Identify the lack of knowledge

(1,1) Do you have basic cooking skills?

Hardly, I only have experience with very basic things.

(1,2) Do you not know how to cook something healthy?

Like I said before, it's not that I don't know what food is healthy. The problem is the fact that I don't know how to cook in general.

(2,1) Can you explain to me in what ways you feel that you lack cooking ability?

I just don't know how to cook. I have never really cooked for myself, I usually buy dinner. I barely know the basics of cooking.

Try not to assume what the patient's situation is. In this case the assumption was right, but it prevents the patient from thinking for themselves.

Try not to assume what the patient's situation is. In this case the assumption was wrong, since the patient does know which recipes are healthy. The patient had previously made a similar statement. Making the wrong assumptions based on a previous statement makes the patient feel that there is a lack of understanding.

Go to the next phase

Figure C.54: Skills - Scenario 2: Lacking Recipe Skills - Phase 3
Figure C.55: Skills - Scenario 2: Lacking Recipe Skills - Phase 4
Figure C.56: Skills - Scenario 2: Lacking Recipe Skills - Phase 5
Figure C.57: Skills - Scenario 2: Lacking Recipe Skills - Phase 6
(1,1) In the next week I want you to make a food diary which keeps track of all the things you've eaten that week.

Sure, I guess I can do that.

Try not to force a plan onto the patient. Ask the patient for permission to give advice and let the patient decide on their own if they want to follow the advice.

(2,1) Would you like to make a food diary to keep track of all the things you've eaten in a week?

Yes, I can do that.

Go to the next phase

Figure C.58: Skills - Scenario 2: Lacking Recipe Skills - Phase 7
Figure C.59: Skills - Scenario 2: Lacking Recipe Skills - Phase 8 and 9