Using scenario smells to analyse scripted communication scenarios in virtual learning environments

Timo Overbeek

Johan Jeuring

Raja Lala

Technical Report UU-CS-2017-008
June 2017

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

1 Introduction

Face-to-face communication is an important skill in a lot of professions. Doctors need to convey bad news to their patients; teachers inform parents about the progress of their pupils; managers hold salary negotiations with their employees; prison guards need to correctly asses the aggression levels of detainees.

Many educational programs include communication skills in their curriculum objectives. There are many ways these learning objectives can be reached, but practicing these skills with role-playing sessions seems to be a particularly effective method [1]. In these sessions two or more students perform a conversation. They each take on a predefined role, which can either be that of a working professional or that of a client. A third person often watches the conversation, and provides feedback once the conversation is done.

To practice communication skills, a student can use a serious game. Such a game often simulates role-playing sessions, but only one role is performed by a player and the others are performed by the computer. Some communication training software allows a player to write his own statements [10], but most games use scripted conversations [2, 3, 6]. In a scripted conversation a player selects an answer from a limited set of provided answers.

Communication experts possess the knowledge necessary to develop a scripted conversation. To make developing scripted conversations as easy as possible, a communication expert can use a special-purpose editor for developing scenarios. A game can then include the conversations developed by a communication expert. Using a special-purpose editor for developing conversations means that authors do not need programming skills.

Directed acyclic graphs (DAGs) are a popular way to structure a dialog in communication software [7]. In such a DAG every node represents a statement and the edges indicate the flow of the conversation. There are many possible paths that can be taken through a DAG, and every path needs to result in a plausible conversation. Depending on the size and complexity of the dialog, this can make writing the required DAGs a complicated process.

Some games provide feedback to a player in the form of a score [3, 6]. This may make writing a DAG even harder because an author will not only need to make sure that a dialog is always plausible, but also that a scoring correctly represents the player's performance for every player choice. For example, if every NPC node has one preferred player follow-up, and those preferred player statements all have the same score, then the longest path will always result in the highest score. The length of a path does not necessarily tell us anything about its desirability, and is therefore probably not a correct way to measure player performance. A long path that yields a high score might be desirable, but could also be an indication that something is wrong with the scenario.

In computer programming we sometimes talk about code smells [4]. With a code smell we refer to a common symptom of a certain error in a piece of code. This error might be on a much deeper level than the symptom. For example the occurrence of very large classes is often referred to as a code smell. There might be situations where a large class is required, but they are often a symptom of bad class hierarchy. The duplicate code A code smell does not necessarily imply bad code. However, a code smell is often much easier to find than the actual problem, and code smells can therefore be helpful in finding these problems. As discussed before, if the longest path in a scenario yields the highest possible score there might be a problem. The scoring could still be correct, the longest path might also be the most desirable path, but this occurrence often indicates that something in the scenario is wrong. This principle is similar to a code smell, and we will therefore refer to this as a scenario smell.

We hypothesize that identifying scenario smells helps a scenario writer to create better training scenarios. In this study we will attempt to identify different kinds of scenario smells. We will also explore different ways in which we can notify a scenario writer of scenario smells. We will look both at the desirability of these solutions and at their technical feasibility.

This study is organized as follows. In Section 2 we discuss the current state of communication education by looking at the literature that is available on the subject. In Section 3 we explain our hypothesis of the perceived problems and their possible solutions in further detail. In Section 4 we describe various interviews we conducted to verify our hypothesis. In Section 5 we discuss the technical possibilities and limitations of our solutions. In Section 6 we recommend some effective solutions. In Section 7 we conclude our research.

2 Communication Education

2.1 Communication Training Software

In this section we give a brief overview of different serious games that support practicing communication skills. We focus on serious games that simulate a conversation. In particular, we discuss the way a dialog is structured and how decisions are made for an NPC.

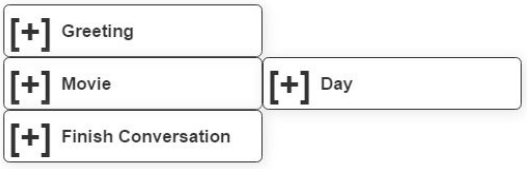
Cláudio et al. [3] describe an application that teaches Pharmaceutical Science student how they should communicate with their patients. The application simulates a conversation between a player and one NPC. The game uses scripted conversations and scenarios can be created in the Dialog Creator Interface. The dialog is represented with a DAG, where every node represents one statement. Statements alternate between player and NPC, and every NPC node connects to exactly three player nodes. Player nodes have a score parameter, which is used to generate a total score at the end of the game.

Bosse et al. [2] propose a way to make NPCs more realistic and less predictable by using a

cognitive model and an aggression parameter. They start from an application much like that of Cláudio et al. [3], where a dialog is represented with a DAG. Like Cláudio et al. they alternate between player and NPC nodes, and every NPC node is connected to a fixed amount of player nodes. They offer four connections, which are matched with the following intentions: letting go, supportive, directive, and call for support. Bosse et al. improve on this initial game, by adding an aggression parameter. The value of this parameter is influenced both by prior statement choices and the player's real-life emotions. Every player node can be connected to multiple NPC nodes and the computer makes a choice between them by looking at the aggression parameter.

Wauters et al. [10] also make use of emotions in their deLearyous application. They use the interpersonal circumplex model [9], which models behavior on two axes; one axis representing submissive versus dominant, and one axis representing cooperative versus antagonistic behavior. Submissive and dominant behavior result in the opposite behavior in a conversation partner. Cooperative and antagonistic behavior result in similar behavior in a conversation partner. DeLearyous leaves the player free to create his/her own statements, and uses a natural language processing (NLP) module to process a player statement. The NLP module classifies where on the circumplex model a statement belongs, and also identifies the keywords in the player's statement. A finite state machine then combines these two to select a NPC statement.

Communicate [6] separates its scenarios into subjects. Each subject is in turn represented by one DAG. Subjects need to be traversed in a predefined sequence. However, subjects can be interleaved, in which case the player is free to choose between statements of the interleaved subjects. This process is illustrated in Fig. 2. The vertical axis represents the sequence of subjects. Subjects on the same horizontal line are interleaved, and can be traversed in any order. All the subjects on one horizontal line need to traversed, before the subjects on the next line become available.



Communicate uses emotions in the animations of the NPCs, but not for determining the flow of the conversation. The dialog flow can be controlled by means of score parameters, which can be defined by the scenario writer. A player node can change score parameters. A player node can also have one or more preconditions. A precondition allows a statement to be presented only if the precondition is satisfied, for example if the scoring parameters are within a certain range.

Leuski and Traum [8] propose to use a virtual human for other purposes than communication training, such as guiding a tour, or perform secretarial services. Like Wauters et al. they use an NLP module for dialogs. They do not use any control flow for dialogs. Instead the statements of the NPC will try to give a correct answer to a player's statement, by using an individual knowledge base. Different NPCs might have different knowledge bases, which results in different behavior.

Gebhard et al. [5] describe Visual Scenemaker, a tool that can be used to create interactive applications with multiple NPCs. Like the application of Leuski and Traum, Visual Scenemaker is not specifically build for communication education, but it can be used for that purpose. Gebhard et al. separate the content, dialog statements, from the logic that describes the dialog flow. They use a directed graph to create a dialog flow, but unlike the other examples the graph does not have to be acyclic.

2.2 Dialog Scenario Language

There are a lot of similarities between the discussed applications. All the applications that use a scripted dialogue make use of a graph to represent the dialog flow. In most cases this graph takes the form of a DAG. Many applications also make use of parameters that can be increased or decreased during the conversation. Scenario writers can use these parameters as a feedback opportunity or as a way to represent emotions. In quite a few applications the parameters also allow the scenario writer to exercise more control over the dialog flow.

Lala et al. [7] compare multiple applications and define a language that can be used to compare different scripted conversation simulations. They define the entire script as a scenario, that can consist of multiple subjects. Every subject in turn consists of a DAG. Nodes in the DAG can have multiple properties, the most important one is the statement, which contains the actual dialog. Other components are also possible, for instance parameter increases, emotion changes, player feedback or preconditions. The authors also define an XML scheme called Utrecht University Dialogue Scenario Language (UUDSL) that can be used to describe the structure of a scenario.

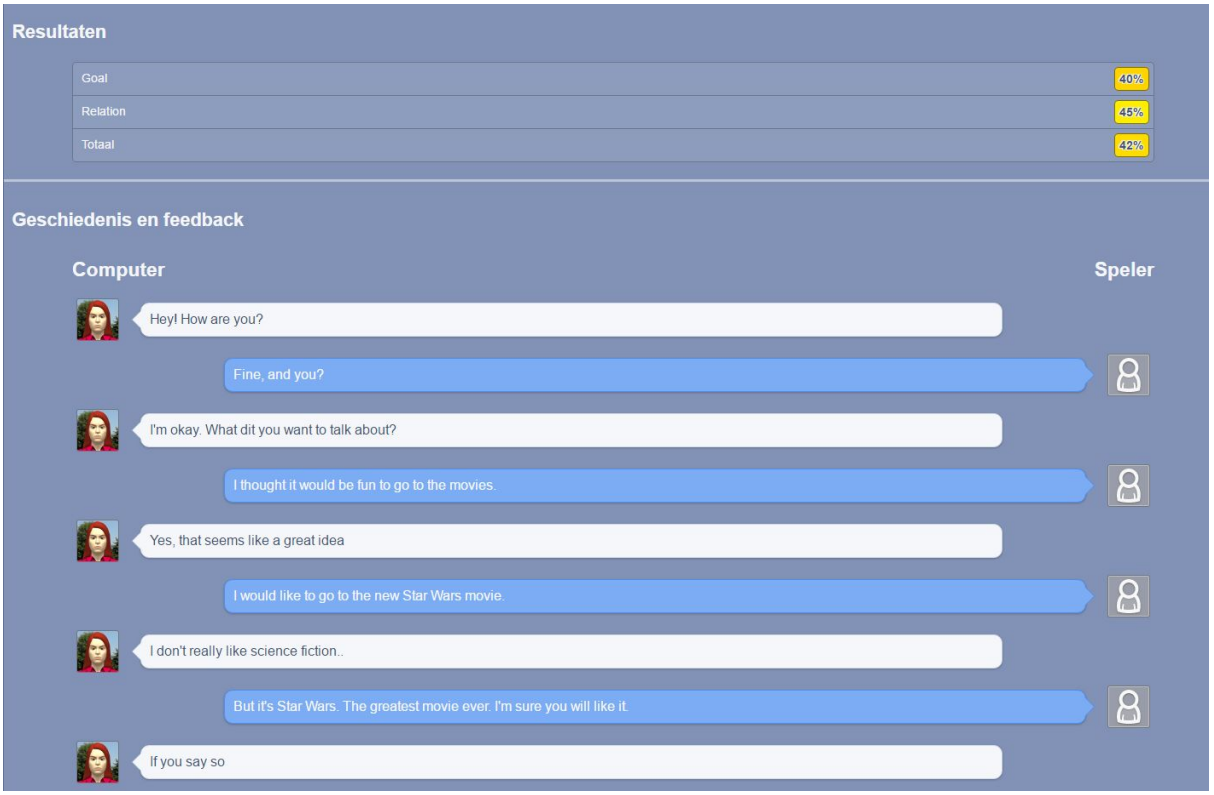
In this study we use UUDSL as much as possible. This makes our research applicable in a wide array of communication training software. We use the Communicate software as benchmark, because we have full access to its code. We now provide an overview of the functions of Communicate for the unfamiliar reader.

2.3 Communicate

Communicate is a web based application that consists of a game and a scenario editor. In the game students have a conversation with a single NPC. The NPC statements are displayed in a text balloon and the possible player statements are displayed at the bottom of the screen.

The game measures the performance of a player with scores. Each score represents a skill or communication goal. For example, in a scenario where a player needs to discuss with a NPC which movie they are going to see, the game can measure the player performance in a *goal* score and a *relation* score. The *goal* score indicates if the player succeeded in selecting the movie he/she wanted. The *relation* goal shows how angry or happy the NPC is with the player.

Choosing certain player statements will increase or decrease the scores. A scenario writer can decide if these score changes are immediately visible or if they remain invisible. Either way, at the end of the game a player sees the final scores, and a total score that is the weighted average of all the scores. In the same screen a player also receives some feedback and can review the entire conversation history.



Scenarios can be created in the scenario editor. When the editor is opened an overview of the different subjects is shown. Figure 2 shows this screen. Subjects on the same horizontal level are interleaved, which means they can be traversed in any order. The vertical levels show the sequence of subjects. A player needs to traverse all the subjects on a horizontal level before he/she can progress to the next horizontal level.

The scenario writer can click on a subject to view the DAG that represents the subject. Figure 5 shows this screen. Red nodes are NPC nodes and blue nodes are player nodes. The green node is the currently selected node and the details of the node are displayed on the right-hand side of the screen. Nodes that have no edges connected to them are nodes that can start the subject. Multiple starting points are possible. Nodes that have no outgoing edges end the subject, and allow a player to select a new starting node from the currently available subjects.

NPC nodes can have three special labels. Firstly, they can be labeled as the end of the conversation. This will end the game even if there are still untraversed subjects. Secondly, they can be labeled as an early end point of the subject. In that case the player can choose to continue with the current subject by choosing one of the connected player nodes. The player can also choose one of the starting nodes of the other available subjects. Choosing the second option will end the current subject. Thirdly, a node can be labeled as a point on which the subject might be switched. This functions in a similar way as the early end to subject option, apart from the fact that the subject is not ended. Instead the connected player statements become available again the next time the player needs to choose a new subject.

A lot of functions in Communicate are controlled with parameters. The entire scenario uses the same set of parameters. The scenario writer can add parameters to and remove parameters from this set. Every parameter has a name and a type. The type can be an integer, a string, a boolean or an enumerate. An Integer has a minimum value, maximum value, and start value. Integers can also have a weight, which marks them as a scoring parameter. The weighted total score is calculated using these weights.

Both scoring parameters and non-scoring parameters can be used to control the dialog flow. Nodes can have preconditions. A node will only be available if its preconditions are true. Preconditions take the following form: [parameter name] [=, ≠, <, >, ≤, ≥] [value]. If there are multiple preconditions a scenario writer can select if all preconditions need to be true or if one precondition needs to true. Preconditions can be grouped together in which case they are interpreted as a single precondition (being either true or false) on the level above their own.

Both player and NPC nodes can have parameter changes attached to them. Parameters can be increased or decreased with a certain value. Parameters can also be set to a specific value. Emotions roughly work the same way. We will therefore largely ignore them, and assume that everything that is possible with parameters can be replicated with emotions.

3 Hypothesis and research question

3.1 Research Question

In our literature study we discussed that many serious games for communication education use DAGs to represent a scripted dialog. This is not surprising, because DAGs are easy to understand for humans and traversing a DAG does not require a lot of computer resources. However, when a DAG becomes larger, it can become difficult to maintain an overview of the entire graph.

To make maintaining an overview easier we would like to keep the visualisation of a node small. This allows us to see a larger part of the graph at the same time. If an individual node has lots of properties, we will need to make a choice between displaying all the information and keeping the node small. A common solution to this problem is to have some information always visible within the node, and have some information only visible when a node is selected. The problem with this solution is that it becomes more difficult to see how some properties are influenced by multiple nodes.

Communication education software often uses parameters that can be increased or decreased by individual nodes. In that case all the traversed nodes have some some influence over the final score. It is therefore important to keep an overview over the relation between different score increases and decreases. The large amount of possible paths through a DAG can make this hard. If some properties are hidden to improve overall visibility, this becomes even harder.

The goal of this study is to make it easier to write scenarios. It is our hypothesis that assigning scores is the most difficult part of creating scenarios. It can be hard to display all the relevant information, and scenario writers have difficulty with combining smaller pieces of information in scenario wide information. We will discuss different ways in which information can be displayed in a DAG. This includes providing combined information, like the value of a parameter over an entire path. We belief that a special focus should be on scenario smells. Scenario smells are possible indications of errors in the DAG. We want to notify the user of these scenario smells, so that the scenario writer can check if there is indeed is an error.

This hypothesis leads to the following research question:

- Can we provide scenario authors with the information they deem important for the improvement of scenario quality?

To answer this question, we want to answer the following sub-questions:

- Is scoring indeed one of the more difficult parts of creating a scenario?
- What information might help with improving the scoring?
- What information can be calculated, both in theory and in practice?

In the rest of the section we explain our hypothesis more. We discuss why we assume that scoring a DAG is difficult and how we think we can help scenario writers. At the end of the section, we introduce our research method. We provide the research results in Section 4 to 7.

3.2 Scenario Smells

It is our hypothesis that the scoring of a scenario is important for the playing experience. The scenario writer creates a scenario with a certain learning goal in mind. This learning goal often dictates which player options are preferred and which are undesirable. The player has implicit expectations about what certain scores mean. It is important that the scores the player gets, correctly represent how the scenario writer values the players performance. Otherwise the player can get frustrated or he/she can wrongly assume that a certain wrong approach is correct. Either way, the learning goals will not be accomplished.

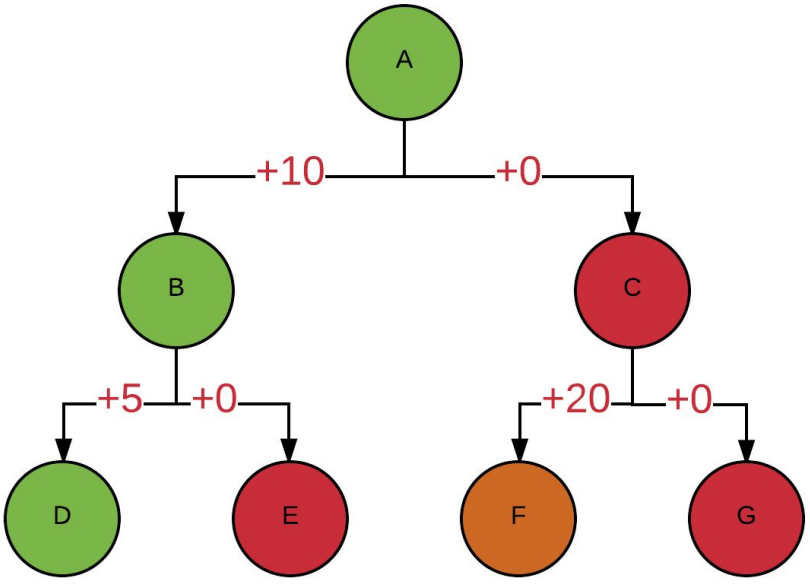
There any many different paths through a scenario, and every path has its own final scores. Most nodes lie on multiple paths. These nodes will therefore contribute to multiple final scores. The scenario writer needs to assign each node a score change that provides the correct final score for every possible path. This can be difficult. We illustrate potential difficulties with two examples.

3.2.1 Example 1

We display the graph of our first example in Figure 6. Based on this example we can assume that the scenario writer favours player option B, D, and F, because those three nodes lead to immediate score increases. By that same measure we can assume that the author does not prefer node C, E, and G. Both node B and D lead to a score increase, so we might conclude that path A-B-D is preferred by the scenario writer.

The path A-C-F leads to a final score of 20, which is the highest possible score. We call this path an optimal path. We expect that a preferred path is also an optimal path. In our example this does not seem to be the case. This can have two explanations. Firstly, the scenario writer might have made a mistake. It could have been the intention to make A-B-D the optimal path, but the scenario writer assigned too high a score increase to node F. Secondly, the assumption about the fact that A-B-D is the preferred path, might be wrong. The author might have intended that option B provides a quick win, while option C makes the player work harder for a larger final reward.

We do not know if the first or second explanation is true. This is an example of a 'scenario smell'. Our assumptions seem to indicate that the scoring might be wrong. However, the scoring might be correct, if our assumptions are wrong. A scenario smell can therefore not be used to prove errors, but it can be used to find them.

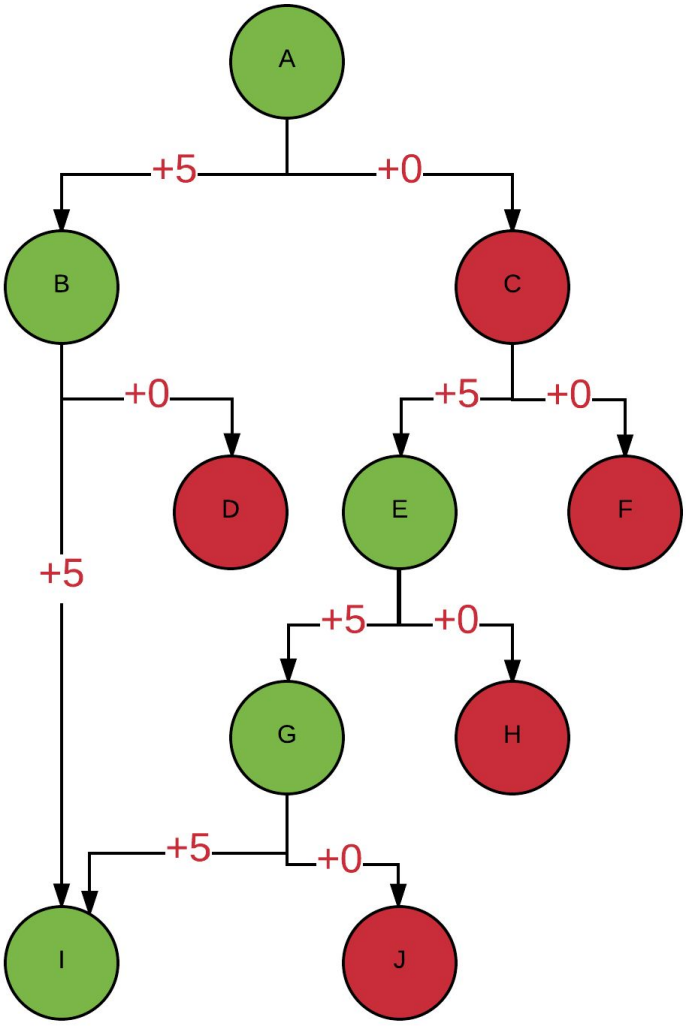


3.2.2 Example 2

Our second example uses the DAG displayed in Figure 7. This example shows a design pattern that is used quite often in scenarios. The player first needs to make a choice between a good option (B) and a bad option (C). If the player choose the wrong option, he/she often receives a chance to make up for their mistake. This is the case in our example. Even if the player chooses B, the wrong option, he/she still has a chance to reach the positive end node I.

The problem with the DAG in Figure 7 is that path A-B-I yields 10 points, while path A-C-E-G-I yields 15 points. In other words, choosing the wrong option C, is the best choice the player can make at node A. An explanation for why this error exist, is that every positive choice adds 5 points to the final score, and every negative choice does not change the score. A-C-E-G-I has a higher score because the path is longer, and the player therefore has more chances to choose a positive node.

Scenario writers indicate that they often accidentally make the longest path the optimal path. Like in example 1, however, we cannot state with certainty that this is the case in figure 7. We can think of examples where the displayed DAG would be correct. In a salary negotiation the player might need to refuse a salary offer, in order to receive a better one down the line. The only thing we can state, is that there is scenario smell if the longest path is also the optimal path. The smell might indicate an error, but is does not have to.



3.3.3 Scenario Smell

Our examples illustrate that it is easy to make mistakes in assigning scores to a DAG. The examples also illustrate that these mistakes can lead to unintended effects in the game. Lastly, the examples illustrate that it is often impossible to say with certainty whether or not a mistake has been made.

We introduced three different definitions that can be used to describe scenarios. Firstly, we have the preferred path, which is the sequence of nodes that, in the scenario writer's view, represents the preferred way to perform a conversation. Secondly, we discussed the optimal path, which represents the sequence of nodes that leads to the highest possible score. Thirdly, we coined the term scenario smell, which refers to a symptom of a common mistake in scenario writing, without guaranteeing that there is a mistake.

Our examples show two different kinds of scenario smells. A scenario smell is based on assumptions. In both cases we start with the assumption that the preferred path and the optimal path should be the same. In the first example we also have another assumption. We assume that if we always select the node with the highest score increase, we will have the preferred path. This is an assumption that does not always hold true. However, if the assumption holds true often enough, it can still provide us valuable feedback. In such cases a scenario writer needs to make sure that the assumption is true for their own scenario, unless there is a compelling reason to ignore the assumption.

The second example shows a common mistake. A scenario writer often accidentally turns the longest path into the optimal path. To avoid this problem we could state that the longest path should never be the optimal path, but that is not necessarily always true. It is better to state that if the longest path is also the optimal path, that can be a sign of bad scenario design. In other words, it is a scenario smell.

Other kinds of scenario smells probably exist. For example, it is a common assumption that the maximal score, as defined by the scenario writer when he defines the parameter, is equal to the highest possible score. If these differ there might be something wrong with a scenario. However, this does not always have to be the case. For instance, in a medical simulation the player might need to warn the patient about possible side effects. This warning can be given at different moments in the conversation, and might even be given multiple times. Only one warning is required, however. In such a case it would make sense to cap the maximum score, to ensure that giving multiple warnings will not have a positive effect. Another possible scenario smell might be that parameters should not have a lot of correlation, because then they measure the same skill. Some scenario smells are more useful than others. In Section 4 we discuss the different kinds of scenario smells and how we assess their usefulness.

3.3 Displaying Information

There are many ways in which creating scenarios can be made easier. We can create scenario editing tools that are easier to use, design better teaching methods to explain the editing tools, create design principles that we have proven to be effective, and let players provide feedback on a scenario while they are gaming. We focus on providing more information to a scenario writer. The question is what information we need to provide, and how we visualize that information.

If we visualise a lot of information there is a risk of information overload. We therefore not only need to decide what information to show, but also when to show it. We categorise the timing of information in three ways. Firstly, some information is always visible. For instance, in the current Communicate editor, the edges connecting nodes are always visible. Secondly, there is information that can be turned on and off. For example, there might be a function that, when turned on, displays the parameter changes for every node. Thirdly, there is information that is only available when specifically asked for. In this last case requesting new information results in the removal of the old information. For instance, in the

Communicate editor, there is a function that can show the editor the parents of a selected node. Only one node can be selected at the same time, and when the editor requests the parents of another node, the old information will disappear.

A simple way to provide more information, is to make more node properties always visible. For instance, we can display the score changes inside the nodes of the DAG. This creates the risk of the earlier discussed information overload. The information overload can be decreased by collecting all the separate score changes in one total score change, or by letting the writer turn the visibility of score changes on and off.

Displaying more information does not necessarily make it easier to asses how the different nodes influence each other. We think that a scenario writer not only needs information about individual nodes, but also about the different paths through the tree. There are an exponential amount of paths, however, so displaying this information might be difficult.

To limit the amount of path information, we might, for instance, only display the optimal paths. This function might be expanded upon, by letting the writer select a node, from which the optimal should be calculated. We can show the least optimal paths instead. Both cases might help scenario writers, because they can check how well the optimal paths and least optimal paths correspond with their preferred path and least preferred path.

Lastly, it is possible to provide statistics about subjects or the entire scenario. The three most useful statistics are probably the minimal value of a parameter, the maximal value of a parameter, and the correlation between different parameters. There are other statistics that might be useful, like the deviation in possible scores over all the paths.

3.4 Research Method

In the previous sections we discussed our hypothesis that small errors in the scoring parameters of a scenario can lead to unexpected and undesired effects. We also hypothesized that finding these errors is difficult, but can be made easier if a scenario editing tool automatically detects so called scenario smells. It is important to verify that this hypothesis is true. We need to verify that scoring is indeed a problem, maybe even the biggest problem, with creating a scenario. We also need to verify that our proposed solutions are seen as helpful by scenario writers.

We verify our hypothesis by holding multiple interviews with current users of a scenario editor. We will ask how they create scenarios, what kind of problems they experience with creating scenarios, and what they think of our proposed improvements. In the same interview, we will also try to get a better picture of the quality of a scenario. Section 4 describes these interviews.

Our goal is to propose improvements for the current authoring tools. An improvement needs to be useful and implementable. The focus of our study will be on finding out what information can be calculated about a DAG. We will look both at the theory of the problem, and at practical implementations. These calculations will help us inform what kind of

improvements can be made. Section 5 discusses the various calculations we want to develop.

The next step is to combine the information about desirability with the information about technical feasibility. We give several recommendations to designers of scenario authoring tools. These recommendations can be implemented in multiple different authoring tools. They should also be a good guideline for creating new authoring tools. We apply our own recommendations to Communicate, to demonstrate how they can be used. Section 6 gives these recommendations.

Lastly, we test if our recommendations satisfy the research question. We do this by returning to our original interviewee. We will demonstrate a prototype and ask them a few question about our recommendations. We will ask them if they intend to use our improvements and how their scenario design will be influenced by said improvements. We realise that an argument can be made that the improvements can only be tested with a pre- and posttest. However, this would mean that our study will only be applicable for a specific scenario authoring tool, and with very specific improvements. This motivated our choice for holding closing interviews instead. In section 7 we explain the closing interviews in more detail and discuss their results. In section 8 we make some recommendations for future research.

4 Interviews

In the previous sections we discussed our hypotheses about the kind of problems that scenario writers experience in creating good scenarios. Before we discuss possible solutions, it is important to find out if this hypotheses hold in the real world. To determine this we decide to interview 6 scenario writers with experience in writing scenarios for Communicate. We first discuss how we decide on the type of questions in the interview. Then, in Section 4.2, we discuss the results of the interviews. Lastly, in Section 4.3, we will summarize the conclusions from performing the interviews. The exact questions of the interview can be found in Appendix I.

4.1 Interview Creation

An interview consists of a mix of open and closed questions. The closed questions allow us to combine different interviews, by collecting the various answers in a single score. The risk of closed questions, is that they might steer the thought process of an interviewee in a certain direction. For instance, we want to know if the interviewee identify certain problems with scenario creation, before we have mentioned them. We therefore include some open questions at the start of the interview.

There are a couple of questions we want to answer with these interviews:

- Is scoring a substantial problem in the creation of scenarios?
- What kind of assumptions do the interviewees have about correct scenarios? If enough people have the same assumptions, would it be helpful if we notify scenario writers when those assumptions are not true for their own scenarios?

- Assuming that the scenario correctly represents the intentions of the scenario writer, and therefore that the preferred path is also an optimal path, what definition of the optimal path do we need to use to calculate the preferred path?
- What kind of information would help a scenario writer in creating better scenarios?
- How does a scenario writer want to receive that information?

The interview is split in 6 different parts. In the first part we collect some general information about the interviewee. This mostly relates to how they use Communicate. In the second part we ask a few open questions about problems they experience while creating scenarios. Until this point, the interviewee has not been notified of our specific research question. The third part consists of closed questions about their assumptions of what makes a scenario correct. In the fourth part we ask them a couple of closed questions about what they consider to be the optimal path. The underlying question here is whether or not there is a mathematical definition for the optimal path, where the optimal path matches with the preferred path in correct scenarios. In the fifth part we propose multiple enhancements to Communicate, and ask our interviewee to review them. Lastly, in the six part, we give our interviewee the chance to speak freely about what they think should be changed in Communicate.

4.2 Interviewee

We briefly discuss the interviewees themselves. To preserve their anonymity we will name them interviewee 1 to 6. All of the interviewees have some experience with Communicate, but their experience differs from only creating a handful of simple scenarios to running a company specializing in creating such scenarios. Most of them have no prior experience with other communication training software, but do have some experience with real-life roleplaying sessions

We interviewed three university teachers (interviewee 1, 3 and 6). All of them teach a course on communication skills at Utrecht University. The educational programs to which they contribute are medicine, pharmacy and veterinary medicine. They all created a few scenarios and used them in their courses. The complexity of the scenarios differ quite a lot. Interviewee 1 is known for creating complex and extensive scenarios. The developers of Communicate often test new features on one of his scenarios, to make sure that a new feature can handle the more complex scenarios. Interviewee 3 is the opposite in that her scenarios are a lot simpler and do not make use of subjects or preconditions. Interviewee 6 holds the middle ground and uses subjects, but not preconditions.

Interviewee 2 is a student who uses Communicate as part of a honors program. She created a few scenarios about general subjects, like a bad news conversation and giving someone feedback. She herself did not use any of the scenarios in a course, but interviewee 6 developed them further, and did use some of them in a course. Interviewee 2 also had no prior experience in creating roleplay scenarios, so we conclude that she probably has the least experience in creating scenarios.

Our fourth and fifth interviewee are employees of DialogueTrainer B.V., a company that specializes in creating Communicate scenarios and that is actively involved in the

development of Communicate. Unlike the other interviewees, they mostly develop scenarios for professionals, such as health-care personnel and managers, who are already working in their respective field. They often work together with players in improving a scenario. They have created more scenarios than the other interviewees (more than a dozen each) and can be considered experts.

4.3 Results

4.3.1 Current Use

At the start of the interview we asked some questions about the current use of Communicate by the interviewee. Some users of Communicate know much more about scenario development than others. For instance, when asked about their use of the *view parents* and *view parameters* functions, multiple users indicated that they were not aware of those functions. Meanwhile, others noted that the *view score* function was overly complicated and difficult to use. Interestingly, the people who are aware of these functions said they used them quite often. Moreover, multiple users who did not use the functions, told us they had troubles with assigning scores. Two users even went so far as printing out their scenarios and calculating the possible end scores by hand. This seems to indicate that creating scenarios could be made a lot easier by providing better instructions and an easier user interface.

We also asked the interviewee how they went about creating scenarios. They all told us that they often create the dialogue first and only add the parameters and emotions later. The interviewees differ in how much they split up these tasks. Some go so far as to write out the entire dialog in a word document before they even start working in Communicate. Others create subparts of the scenario (dialog lines or entire subjects) and then add the parameters.

Quite a few interviewees told us that they first create a perfect dialog path, and then add possible mistakes. One person told us he played his own scenarios many times to check if the dialogue feels natural, and he then makes incremental changes to the dialog to make the dialogue feel more natural. Another person uses a similar principle, but uses test subjects to assess if the flow of the dialogue feels natural.

4.3.2 Valid Scenarios

In the next part of the interview, we asked our interviewee what, in their mind, constituted a valid scenario. To structure the outcome we prepared some statements beforehand and asked them to score the statement with a number between 1 and 5, where a 1 means they completely disagree with the statement, and a 5 means they completely agree with the statement. Interviewees often had difficulties in assigning an exact score to a statement. In a few cases we therefore had to give the interviewee the possibility to give a textual answer. We then interpolated a score from their total answer, which we offered as a suggestion for their final answer. We think that these problems do not have a large impact on the final conclusions because they did not occur too often. The interviewees did not waver between

two extreme answers, but only between small differences, like a score of 1 or 2. The exact scores can be found in table 1.

Question	I	II	III	IV	V	VI	Total
12	1	2	1	1	1	1	1.2
13	4	5	5	4	5	4	4.5
14	3	1	-	3	3	5	3.0
15	5	4	5	5	5	5	4.8
16	5	3	3	5	4	5	4.2
17	5	4	4	5	4	5	4.5
18	2	2	5	3	1	1	2.3
19	2	1	1	1	3	1	1.5
20	2	4	5	1	1	1	2.3
21	3	5	5	1	2	1	2.7
22	4	1	-	-	2	1	2.0
23	1	5	-	1	1	5	2.6
24	1	5	1	5	2	4	3.0
25	1	3	5	2	1	5	2.9
26	2	1	1	1	1	5	1.8

We separate the statements into three groups based on the scores given. First the statements with which the interviewees largely agree (a score of 4 or 5):

13 If the player chooses the best option (the option with the highest score increase) at every node, he / she should have the highest possible score at the end of the scenario.
15 It should be possible to get the maximal score.
16 For every parameter there should be a path that generates the maximal score.
17 Every node (even those with prerequisites) should be reachable in some way.

Then we have the statements with which the interviewees largely disagree (a score of 1 or 2):

12 The longest path should yield the highest score.

19 The longest path should yield the lowest score.
26 Parameters are always positive.

Many interviewees indicated that if the longest path does yield the highest score, it often is a sign of bad scenario design. At the same time they do not think this is necessarily always the case, which might explain why they did not agree with the statement that the longest path should yield the lowest score either.

Lastly, we have the statements with which the inerviewees neither agree nor disagree (a score between 2 and 4):

14 The sequence in which subjects have been dealt with, should not matter for the end score.
18 Parameters should have no correlations with each other.
20 It should be possible to get the minimal score
21 For every parameter, there should be a path that generates the minimal score
22 The player should sometimes sacrifice a scoring opportunity to receive a better opportunity later on.
23 Switching from subject is never a good idea.
24 All the (scoring) parameters should use the same scale (1 – 10, 0 – 100 etcetera).
25 It should be possible to score above 50% on every parameter.

We already discussed how many of the interviewees found it difficult to talk in absolute statements about the validity of a scenario. This can also be seen in the given scores. The statement "*It should be possible to get the maximal score"* scores higher (score: 4.8), then the statement "*For every parameter, there should be a path that generates the maximal score*" (score: 4.2). This seems weird, because the former statement can only be true if the later statement is also true. Interviewee 1 also gives a score of 4 to both the statement that "*always choosing the best option should result in the highest score*" and the statement that "*players should sometimes sacrifice a scoring opportunity to receive a better opportunity down the line*". However, these statements are each other's opposite, and can therefore not be true at the same time.

These discrepancies can partly have arisen because our subjects did not understand the question properly. Quotes like "*I could imagine a situation, where this is true*", seem to indicate, however, that while they broadly agree with certain statements, they want to keep the option to deviate from them. This shows that a scenario writer should be able to choose to ignore the recommendations of a tool. Scenario writers should also be able to make an informed choice about whether or not they want to use any potential validation tool for a specific scenario. It should therefore be clear how such a tool works, and what exactly is measured by the tool.

There are two different ways in which the interviewees look at scoring within scenarios. Some interviewees seem to view the game as an assessment, even though none of them ever used it in this way. They often say that players should be able to reach the minimum and maximum score, and that parameters should not have any correlation with each other.

These are assumptions that are also often used to check the validity of traditional assessments.

Other interviewees disagree though. In their opinion, the reachability of the minimum and maximum scores, tell us very little about the validity of a scenario. They also voiced opinions like: "*scoring isn't that important for my scenarios*" and "*I use scoring mostly to make people aware of what is happening, not to pass a judgement*". On further questioning, they often indicated that they do see how certain statements can be useful for describing a scenario, but they do not see the statements as a way to assess validity. In the words of one interviewee: "*As a scenario writer, you need to understand what you are doing, but correlation can also help you to refine small differences in the effect of their* (read: the player's) *choices.*"

When asked which statements they found difficult to check in their own scenarios we got many different answers. Some answers kept reappearing though. Many interviewees found it difficult to check if the best (read: the preferred) path also yielded the highest score. They also wanted to know what the current maximums of their parameters were.

4.3.3 Optimal Path

We continued the interview by discussing the optimal path. We define an optimal path as the path through the tree that generates the highest possible score. In theory an optimal path should be the same as the preferred path of the scenario writer. The implicit assumption here is that the player should value high scores, and that a high score should therefore be the reward for good behaviour. Communicate displays the total score in a prominent way, and this supports our implicit assumption. Communicate scenario writers often distribute the scenario themself. In the instructions they provide to the students, they can specify other goals than maximising the total score. This could in turn invalidate our assumptions about the preferred and optimal path. By discussing the optimal path with the interviewee we wanted to find out if they think that maximising the total score is indeed the most important goal of players, or if our definition of the optimal path should be changed.

We again asked the interviewee to assess a couple of statements with a number between 1 and 5, where a 1 means they completely disagree and a 5 means they completely agree. Question 35 is a bit different. In general, school grades can be split into two categories, passing grades and non-passing grades. A threshold (usually 5 or 6) is chosen to differentiate between the two categories. If a scenario writer views her scenario as a test for the students, it would make sense if such a threshold also exists for the scoring parameters. For instance, imagine a situation where a player needs to maximize his/her weighted average, but also keep all the individual parameters above a certain threshold. For question 35 we took the definition of the optimal path that was preferred by that individual interviewee, and asked them if they thought that their preferred definition would improve if we added the amendment that the player should also keep all the parameter scores above a certain threshold. The results can be found in Table 2.

Question	I	II	III	IV	V	VI	Total

29	4	4	3	1	4	5	3.5
30	1	2	4	2	4	1	2.3
31	4	4	5	5	5	3	4.3
32	1	1	1	1	2	1	1.2
33	1	1	1	1	3	1	1.3
34	1	1	1	1	1	1	1.0
35	No	No	Yes	Yes	No	Yes	

In summary, we have one definition for the optimal path that is broadly seen as useful:

31 The optimal path is the path with the highest weighted average of all the used parameters

Three definitions, which the interviewees are uncertain about:

29 The optimal path is the path with the highest average over all the parameters (no weights)

30 The optimal path is the path that includes the highest score for any individual parameter

35 The definition of the optimal path would improve if we added the clause that all the parameters should be above a predefined minimum.

And three definitions that are seen as not useful:

32 The optimal path is the path with the minimal number of nodes

33 The optimal path is the path that has the highest score on its lowest parameter

34 The optimal path is the path with the maximal number of nodes

Definition 31 (“The optimal path is the highest weighted average of all the current parameters”) is clearly the definition the interviewees most prefer. This is the same definition that is currently used to calculate the total score in Communicate. The total score is prominently displayed on the feedback screen at the end of every game. It is possible that the interviewees favour definition 31 because of its current prominent use, and that scenario writers of other serious games will favour different definitions. Since we focus our study on Communicate specifically, we will ignore this possibility, and assume that scenario writers favour definition 31.

Many interviewees noted that there is no definition for the optimal path that is applicable in every situation. The total score is the default measure to represent player performance, but other measures might be possible. An example that was given by one of the interviewees was a personality test, where the player is not judged on his/her final score. Instead, the individual scores of parameters would represent the personality of the player. In such a case the definition of the optimal path should be changed, but only for that particular scenario. It would, therefore, be nice if a scenario writer can indicate which definition should be used. If this is not possible, the scenario writer should at least know the definition used.

4.3.4 Proposed Improvements

In the last part of the interview, we asked the interviewees about their opinion of possible improvements to Communicate. We told them that they should feel free to make their answers as long or short as they wanted. Thus we can not condense the answers into absolute numbers. This makes comparing answers hard. We grouped the questions into four categories, which we will discuss in turn. The exact questions can be found in appendix 1. At the start of each section, we also give a short overview of the different questions.

Showing Scores

The first group contains question 37, 38 and 39. These questions are about displaying the score changes directly in the tree. In question 37 we propose to display all the parameter changes within the DAG itself. Question 39 follows the same idea, but displays only the total score change, and not all the individual parameter changes. In question 39 we asked the interviewees their opinion on color coding the nodes on their netto total score changes. The proposed improvements differ mostly on how much information we display at the same time. In this way, we hope to find out if the scenario writers see information overload as a problem.

The reactions of the interviewees ranged from lukewarm to positive. All the interviewees thought that displaying the score change, would improve a scenario, but they questioned how big this improvement would be. The following quote expresses this opinion: “It certainly offers something, but I do not know how much it would actually help me personally”. The interviewees seem to have two primary concerns on the usefulness of the proposed improvements. Will the improvements lead to information overload, and how usefulness is the total score in practice.

Roughly half of the interviewees mentioned the problem of information overload. They did seem to think that this problem can be solved. Many supported the idea that you should be able to customize what changes are visible at any given time. For instance, they proposed an option to select which parameters should be visible at any given time. The color coding system was most popular, with every interviewee indicating that they liked the idea.

The other problem with the improvements proposed in our questions, is the usefulness of the total score. Most interviewees did think that the total score is important, but they said that they usually do not consider it much while creating scenarios. They see it more as something they need to check after the rest of the scenario is created, while the individual parameters are more of a concern during scenario creation. Many interviewees would therefore rather see that the proposed improvements do not show the changes in the total score, but show the changes in individual parameters instead. Despite this preference most interviewees indicated that the changes in total score were still of some use. A few interviewees indicated that only displaying the total scores would have no use for them at all.

Showing Optimal Path

The next group of questions (40, 41 and 42) all relate to showing the optimal path while editing. We started with asking if the interviewees would like to see the optimal path within one subject. In the next question we extended this to the optimal path in the entire tree. In

the last question we proposed an improvement, that shows the optimal path starting from a node selected by the user. Of course, these improvements depend on the definition of the optimal path. This definition is not necessarily the same for every possible scenario, which was something that the interviewees often mentioned.

Despite the problems with the definition of the optimal path, the general reaction to our proposed improvements was very positive. We heard multiple stories of people who were now calculating the optimal paths by hand or who tried to visually arrange the tree in such a way that they could keep track of the optimal path. The interviewees mostly indicated two motivations for this:

First of all, they noted that it was difficult to return to working on a scenario when they had not worked on it for a while. The visual cues helped them remember their original ideas. This problem cannot only be solved by our proposed improvements, but also by adding an option to annotate the optimal path by hand. Although we should note that there is an important difference between the numerical optimal path and the perceived best path, which we call the preferred path. This problem seems to partially exist because scenario writers have trouble quickly interpreting large trees. Improving on the automatic tree arrangement might help alleviate this problem.

The second motivation for showing the optimal path is that the users can see the maximal score this way. Later in the interview, we bring up the option to show the maximum score ourselves. However, some interviewees brought this up themselves before we had mentioned it, which underlines how helpful this feature could be.

Most interviewees prefer to see the optimal for the entire tree, but they understand that this might be very hard to implement. They indicated that showing the optimal path within one subject, instead of in the entire tree, would still be of some help. A few interviewees even told us they prefer to only see the optimal path within one subject because it would give them simpler and clearer feedback.

One thing all the interviewees agree on is that the option proposed in question 42 (a button, which, when clicked, shows the optimal path from a specified node) is the most desirable of the three options because it would allow them to not only see the optimal path, but also the paths that become optimal, once a certain mistake has been made. This was something that was mentioned quite a lot, not only with this question, but also on other questions. Almost all the scenario writers want players to have the option to correct for a mistake. This means that there need to be semi-optimal paths, which allow a few errors, but still yield a high score. So while the scenario writers see the optimal path als useful, they think that ranking all the different paths on desirability would be even more useful.

We asked the interviewees about their opinion on two usability factors. First there is the calculation time, the time that the computer would need to recalculate the provided information, once the scenario writer has made a change in the scenario. Secondly, there is the margin of error. With margin of error we mean both calculations that do not always yield the correct answer and calculations that ignore certain factors of the scenario design, like for

instance preconditions. The interviewees told us that the time needed to calculate any provided information is not that important. They think that a couple minutes of calculation time is still tolerable. They are a lot stricter on the margin of error though. Only one interviewee told us she would be okay if the certainty was less than 100%. Ignoring preconditions might be acceptable, but it would severely limit the usability in their opinion.

Automatic Validation

Question 43, 44 and 45 are about automatic validation of scenarios. In question 43 we discuss validation, where a validator selects a random node, generates the optimal path from that node, and asks a scenario writer to validate this path. The improvements proposed in questions 44 and 45 are a bit simpler, because they require less input from a scenario writer. A scenario writer simply marks some nodes as desirable, in question 44, or undesirable, in question 45. The drawback of this approach is that validation becomes less robust.

At first many interviewees seemed confused about what we meant with these questions, but after some more explanation they all came around and said that they would really like such a function. This emphasizes the fact that any validation tool should be very clear in its functionality and its possible uses. While such a validation tool apparently has some worth, this worth is not necessarily self explanatory. The risk exists that scenario writers will simply ignore the validation option, because they do not understand how or why they should use the option. Clear instruction and a good interface, are of course always important, but they are especially required in this case.

In general, the interviewees liked the option to color code the nodes themselves more than the validating random paths option. This is not completely surprising because these functions seem to align closely with how the interviewee are currently creating scenarios. Almost all interviewees indicated that they usually have a perfect path in mind while creating scenarios. They often even start with creating this optimal path. One interviewee even asked for a color coding function, but without the validation option, before we suggested the option ourselves. He motivated his proposal by stating that color coding would help him keep a better overview while creating scenarios.

The choice between indicating correct nodes or incorrect nodes seems to be a toss-up. Many interviewees seem to prefer marking the nodes they see as the correct or best option, because this aligns better with how they are currently working. At the same time, they often see more use in marking incorrect nodes. A combination between the two options might be preferable.

Interestingly, two interviewees independently proposed the same new idea. They told us they would like it if the process works the other way around. The editor should calculate all the paths and then rank them on their score. This ranking would be displayed using different gradients of green and red. The scenario writer can then quickly assess if the ranking is correct.

In our questions, we introduced the different improvements as calculations that would constantly be refreshed in the background. Later we asked the interviewee if it was a problem if they needed to click on a button to start or refresh the calculation. The interviewees all said that this was not a problem, and quite a few even preferred the second option. We also asked if they would rather see a validation that would encompass the entire scenario or just one subject. Once again there was not one prevailing choice.

Statistics
Lastly, we asked the interviewees if they were interested in seeing general statistics of their scenarios (question 46 and 47). The reactions were mixed. Everybody would very much like to know the maximal score that can be obtained in a scenario. They often even stated this right at the start of the interview as a major problem of the current version of Communicate. Their interest in other kinds of statistics was only lukewarm though. The interviewees had some interest in the minimal score, and a few interviewees were interested in the correlation between parameters, but there was no overwhelming interest.

4.4 Conclusion

The most interesting finding of the interviews is the importance of the optimal path. We define scenario smells by specifying what makes a scenario correct. Quite a few scenario smells relate to the preferred and/or optimal path, and the interviewees seem to mostly agree with the fact that those paths should be the same in a correct scenario.

The importance of the optimal path can be seen in the reactions the interviewees gave on the multiple proposed improvements of communicate. The most preferred improvements all relate to different paths, both optimal and suboptimal, in the dialogue tree. Mathematically, calculating the optimal path, the least optimal path, and a sub-optimal path, are somewhat related. Finding out whether or not it is possible to calculate the optimal path, and if so, how this can be done, therefore also provides us with more information about how we can calculate other paths.

Calculating the maximum score of a parameter is roughly the same problem as calculating the optimal path. It would also be relatively easy for scenario writers to find the maximal score themselves if they knew the optimal path. Combine this with the lackluster interest in such a feature and we propose that statistics should be ignored for now and that the focus should be on calculating the optimal path.

The question whether or not the optimal path is computable is probably not answerable with a strict true or false. It is very well possible that the calculation is possible, but either takes a very long time or will require us to sacrifice some accuracy. We, therefore, wanted to find out which of those two would be the bigger problem for scenario writers.

In general, the interviewees indicate that they find accuracy more important than calculation time. Quite a few interviewees told us that calculation time does not matter at all. It is

questionable if this is indeed the case if a scenario writer is confronted with long calculation times, but for now we can conclude that accuracy is more important than calculation time.

The accuracy requirement can itself be split into two sub-requirements. Does an algorithm provide the correct answer, and does an algorithm take all the information into account? The interviewees were very strict on the correctness requirement, with almost all of them indicating that provided measurements and paths should be correct. The completeness requirement is a lot less strict. The interviewees are willing to accept an algorithm that only works on one subject, and some interviewees are even willing to accept an algorithm that ignores preconditions.

One last observation we have drawn from the interviews is how much the requirements differ from user to user. Even when a proposed improvement was well liked by the interviewees, they often still wanted the option to turn it of. It is important that a user understands what a possible improvement does, and that they can choose to turn it off, or at least ignore its effect. It would be even better if a scenario writer has a degree of control over the functioning of an improvement. For instance, by selecting which definition of optimal path an algorithm should use.

In conclusion, we think it is important to find out how the optimal path through a dialog tree can be calculated. It will be useful to compare different methods and their limitations. This in turn can give us some insight into which of the improvements can be realistically incorporated into Communicate, or what changes need to be made to Communicate to allow for said improvements. Although other information beside the optimal path is desirable, we think that it is best to focus on the optimal path first. The interviews we conducted support this premise, so in the following sections, we take a look at different possible implementations of algorithms to find the optimal path. We will then compare their advantages and disadvantages.

Literature

1. Marianne Berkhof, H. Jolanda van Rijssen, Antonius J.M. Schellart, Johannes R. Anema, and Allard J. van der Beek. Effective training strategies for teaching communication skills to physicians: An overview of systematic reviews. *Patient Education and Counseling*, 84(2):152–162, 2011.
2. Tibor Bosse and Simon Provoost. Integrating conversation trees and cognitive models within an eca for aggression de-escalation training. In *Qingliang Chen, Paolo Torroni, Serena Villata, Jane Hsu, and Andrea Omicini, editors, Proceedings PRIMA 2015: the 18th International Conference on Principles and Practice of Multi-Agent Systems, volume 9387 of LNCS*, pages 650–659, 2015.
3. Ana Paula Cláudio, Maria Beatriz Carmo, Vítor Pinto, Afonso Cavaco and Mara Pereira Guerreiro. Virtual Humans for Training and Assessment of Self-medication Consultation Skills in Pharmacy Students. In *Proceedings ICCSE 2015: the 10th International Conference on Computer Science & Education*, pages 175–180, 2015.
4. Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999.
5. Patrick Gebhard, Gregor Mehlmann, and Michael Kipp. Visual scenemaker—a tool for authoring interactive virtual characters. Journal on Multimodal User Interfaces, 6(1):3–11, 2011.
6. Johan Jeuring, Frans Grosfeld, Bastiaan Heeren, Michiel Hulsbergen, Richta IJntema, Vincent Jonker, Nicole Mastenbroek, Maarten van der Smagt, Frank Wijmans, Majanne Wolters, and Henk van Zeijts. Communicate! — a serious game for communication skills —. In P*roceedings EC-TEL 2015 Design for Teaching and Learning in a Networked World: 10th European Conference on Technology Enhanced Learning, volume 9307 of LNCS*, pages 513–517. Springer International Publishing, 2015.
7. Raja Lala, Johan Jeuring, Jordy van Dortmont, and Marcell van Geest. Scenarios in virtual learning environments for one-to-one communication skills training.
8. Anton Leuski and David Traum. NPCEditor: Creating Virtual Human Dialogue Using Information Retrieval Techniques. *AI Magazine from the Association for the Advancement of Artificial Intelligence*, 32(2):42–56, 2011.
9. Jonathan Posner, James A. Russell, and Bradley S. Peterson. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and Psychopathology*, 17(3), pp. 715–734, 1980.
10. Jeroen Wauters, Frederik Broeckhoven, Maarten Overveldt, Koen Eneman, Frederik Vaassen, and Walter Daelemans. delearyous: An interactive application for interpersonal communication training. In Serious Games: The Challenge: Joint Conference of the Interdisciplinary Research Group on Technology, Education, and Communication, and the Scientific Network on Critical and Flexible Thinking Ghent, volume 280 of Communications in Computer and Information Science, pages 87–90. Springer, 2012.

Appendix 1: Interview

General Information

1. What is your profession, what is your working domain?
2. How many scenarios have you created with Communicate?
3. Have you used Communicate in a classroom situation?
4. Do you make use of the validate option?
5. Do you make use of the see parents option?
6. Do you make use of the calculate points option?
7. Do you usually work node for node (completing, the dialogue, prerequisites, emotions, parameters) or do you first write all the dialogue and then at parameters and prerequisites?

Their Ideas

8. What kind of problems do you experience in creating Communicate scenario's?
9. Are there problems that are specific to scoring?
10. Do you have any ideas how we could solve those problems?
11. What kind of steps do you take to avoid problems in scoring?

Assumptions

I now would like to ask you some questions about scenario's. The editor gives you a lot of freedom in how scenarios should work. This can be a problem for creating validation tools. We can try to make some assumptions about scenario's however that can help us define what is a 'good' scenario. Can you describe how true you think the following assumptions are by giving a number between 1 and 5, where 1 is completely false and 5 completely true.

12. The longest path yields the highest score.
13. If the player chooses the best option (the option with the highest score increase) at every node, he / she should have the highest possible score at the end of the scenario.
14. The sequence in which subjects have been dealt with, should not matter for the end score.
15. It should be possible to get the maximal score.
16. It should be possible to get the maximal score on every parameter, but not necessarily the maximal end score.
17. Every node (even those with prerequisites) should be reachable in some way.
18. Parameters should have no correlations with each other.
19. The longest path should yield the lowest score
20. It should be possible to get the minimal score

21. It should be possible to get the minimal score on every parameter, but not necessarily the minimal end score.
22. The player should sometimes sacrifice a scoring opportunity to receive a better opportunity down the line.
23. Switching from subject is never a good idea.
24. All the (scoring) parameters should use the same scale (1 – 10, 0 – 100 etcetera).
25. It should be possible to score above 50% on every parameter.
26. Parameters are always positive.

27. Are there any assumptions we have missed?
28. Can you tell me the three assumptions that you currently find the hardest to validate in your own scenario's?

Optimal Path

In discussions about (dialog) trees we often talk about optimal paths. This is a term that usually described the sequence of nodes that rewards us with the most wins (in games like chess) or the highest score. We could therefore conclude that the path with the highest overall score is the optimal path in the case of Communicate, but this is not necessarily satisfactory. I will now state a couple of ways in which we could determine the optimal path. Can you tell me for each definition if you agree or disagree that this would be a workable definition?

29. The highest average of all the parameters (no weights)
30. The path that includes the highest score for an individual parameter
31. The weighted highest average of all the current parameters (current total score)
32. The quickest path
33. The path that has the highest score on the lowest parameter.
34. The slowest path

35. Would these definitions improve if we added the extra requirement that all the scores need to be above a certain threshold
36. Which of these definitions do you thinks works the best? Are there any other possible definitions that we didn't think of?

Solutions / UI

I know want to talk about possible improvements of Communicate. These are hypothetical improvements, so we don't need to think about feasibility. For each of this improvement, can you tell me what you think about the following aspects:

- Will using it improve your scenario's?
- Do you think you would make use of this improvement?
- How much margin for error is there, ea. the tools needs to give me an exact answer, the tool needs to give me a reasonable answer etcetera.

- Would it be okay if we ignore some part (requirements, subjects etcetera) of the scenario in the tool?

You don’t need to give me an exact answer, so feel free to give me all your thoughts.

37. Visually displaying all the parameter changes for every node.
38. Color coding all the nodes based on the parameter changes. Green for a positive effect on the overall score, orange for no effect and red for a negative effect.
39. Visually displaying the change in the overall score for each node
40. A button which shows the optimal path through a subject
41. A button which shows the optimal path through the entire tree
42. A button which shows the optimal path from a specified node
43. An option to repeatedly show optimal paths from different start nodes, whereby the start points are randomly chosen by the computer.
44. An option to mark nodes as desirable and to get a warning if a desirable node isn’t included on the optimal path
45. An option to mark nodes as undesirable and to get a warning if a desirable node is included on the optimal path
46. Statistics about the scores like correlation, minimal value, maximal value etcetera
47. A profile of every subject that shows you how often a parameter is used in that subject, what the maximum changes (both positive and negative) are etcetera.

Finishing Thoughts

That was my last prepared question. Now that we have finished the entire interview, do you have any last thoughts. Did you think of any new possible improvements that could be made to Communicate in respect to scoring? Did is miss something important in the discussion about this subject?