Master Thesis
Designing a Game for Basic Process Model Elicitation

Bart Schotten

October 4, 2009

Supervisor: Stijn Hoppenbrouwers

Research Number: 108 IK

Radboud University Nijmegen

Contents

1 Introduction

Modelling is one of the most important parts of software engineering. During the software development process we create models to analyse an existing situation, to describe the required changes, and to design a new situation [16]. Requirements analysis and modelling is so important because it faces one of the greatest challenges of software engineering: bridging the gap between the domain expert and the software developer.

Modelling is often done by an expert whose task is to gather the knowledge possessed by domain experts and to translate and abstract this knowledge into a formal notation [8], often with the purpose of being understandable and useful to software developers. Of course, by summarizing it in one sentence like this we can't even begin to do justice to the complexity of this process. The modelling process is a popular subject of research for a reason. Just one example of a problem to be dealt with is the question which language should be used to describe a model.

Judging what a good modelling language is involves many different quality aspects. A language should be both expressive enough to describe all importants concepts in a domain and specific enough to accurately describe a design, and that is just the tip of the iceberg [12] [22]. Just within a theoretical debate over modeling languages there is a case to be made for many different choices. This often results in the YAMA (Yet Another Modelling Approach) syndrome [15], and this is without even considering the use of a particular language in practice.

This thesis does not concern a particular modelling language or meta-model, but instead focuses on the modelling process. We consider the method more important than the exact properties of the deliverables. What we do in this thesis is called method engineering [13].

What a good modelling process is, is even harder to say than what a good model is. Of course certain quality aspects can be identified [20], but we have to consider the collaborative nature of the modelling process [17], as well as the role modelling tools play. Method engineering also involves human factors and human-computer factors.

For this thesis we are especially interested in modelling tools. There are many tools that just focus on models. They support the notation of the deliverables of the modelling process, but that does not mean that they support the modelling process. Instead of just providing an empty canvas a modelling tool should consider the way the user thinks about modelling and support decision making [9].

Our goal is to create a tool that does support the modelling process, and shifts the focus away from the deliverables. A tool like that should be different from a normal modelling tool in a number of ways:

1. It focuses more on the initial stages of the process (elicitation), than on the final stages (notation).

2. It does not necessarily conform to a specific language, because differences between specific languages are not important in the initial stages of the process.

3. It may use schema representation, but does not see it as a deliverable.

According to the first points, we are going to try to automate elicitation. This sounds quite ambitious. Of course, getting information out of a user is not an uncommon task for a system. When it is simple, structured information you need, you can just present the user with a form, and when the information has a few more degrees of freedom, using a wizard or a conversational system might be a adequate solution [10]. However, when it comes to modelling, the information is not readily available. The information is highly unstructured and it can take many forms. This means we need a different approach.

We might learn something from a type of system that also requires a lot of input from a user, but at the same time allows this user a lot of freedom. A type of system that facilitates a process of working towards a solution, by providing the user with certain challenges and certain tools to meet these challenges, in a way that stimulates the user to give his best. A type of system that, like a good modelling tool, supports decision making. And most of all a type of system that makes the user's task more fun. It is clear that we are talking about games.

Eliciting information for a process model requires a technique that is very flexible, uses vague quality measures and provides subtle feedback. Game design provides excellent inspiration, because like process modelling, winning and losing in a game is not always black and white and there is not one correct way to play. Eliciting process descriptions requires a form of human-computer interaction that gives the user a sense of freedom. The system should not prescribe one correct way to act, but should encourage the user to improve.

The similarities between method engineering and game design invite us to look at the modelling process and the use of tools in a different way. Hopefully we can learn what makes games in general so much fun to play, and accessible to such a wide range of people, and apply this knowledge to the modelling process.

Our ultimate goal is to create a tool that supports the elicitation of process models in such a way that the role of the modelling expert is greatly reduced. This is called disintermediation [4]. The domain expert is the one using the tool, and the use of the tool results in all the information needed to create a model.

In this thesis we present a prototype of such a system. After stating the questions we want to answer, we will describe how we approached the development process of the prototype in section 3 followed by an overview of the actual development process in section 5. Next we will describe the theory behind the prototype in section 6, followed by a description of the actual gameplay of the prototype (section 7). In section 8 we will show that the prototype is compatible with XPDL. After that we provide the theoretical approach behind the testing in section 9, and the results of the actual testing in section 11.

Based on those results, we hope to answer our questions and suggest some improvements and future prospects.

2 Problem statement

In the introduction we outlined the need for a tool that truly supports the modelling process and we compared method engineering with game design. The goal of this project is to create a prototype of a modelling tool that is *game-like*, or at least inspired by game design, with the purpose of making modelling more accessible to non-experts, as well as:

1. Ensuring the quality of the modelling process [12] [22].
2. Improving the efficiency of the modelling process [6].
3. Making the modelling process more fun for the domain-expert.

These are ambitious goals, and we are only just exploring a new field of research with this thesis, so we do not expect to fully reach these goals right now. The goal of this project is just to make a prototype, so the main question we will try to answer in this thesis is:

- Does viewing method engineering as game design help to design a tool/game that allows non-experts to make a formal description of a process?

This question illustrates the role of this thesis as an exploratory project. The most important thing we hope to learn is the possibilities for the future. We will do this by creating and evaluating a prototype. We will try to answer the main question by answering two more concrete sub-questions that represent realistic short term goals for this project.

- Are players able to get through the game with little to no outside help from an expert?
- Does playing the game result in information that can be used to create a process model?

The goal of the game is not to create a process model conforming to a specific language. The game will only provide the information needed to create a process model, and it does not matter in what language exactly.

Admittedly these two points really only apply to the quality of the modelling process. When it comes to the efficiency and the fun-factor it is hard to set specific goals. We just hope that players will work efficiently, enjoy the experience, and feel like they are actually playing a game.

3 Way of working

The goal of this project is to develop a prototype of a game that supports basic process modelling. The development will be a creative process, but because this is a master's thesis, we will have to make sure it is a scientific process as well. The reality is that there is little existing literature on which we can directly base this prototype. What we have to do instead is systematically check if we are going in the right direction, by working iteratively and testing intermediate products. This is called *design science*.

Hevner et al. [5] argue that information systems research can combine both behavioral science and design science. If the development of a system is just a technical exercise, design science is sufficient, but since information systems research often involves human-machine interaction, aspects of behavioral science are important as well. It seems that the same can be said for game design. All of the game-related literature we studied digs deep into psychology, trying to understand moods, emotions and motivations of the player. We should be prepared to test these aspects as well.

Design science is a problem-solving paradigm. In information systems development and research, as well as in our case, these problems are often *wicked problems*; they are only understood fully *while* they are being solved. Therefore IT research is iterative by nature. Design science basically cycles between the design and the evaluation of IT artifacts. Depending on the nature of the artifact (ranging from a general description of a solution to fully implemented software), behavioral science is often used for its evaluation (or justification).

Hevner et al. provide seven guidelines for using design science:

1. "Design science requires the creation of an innovative, purposeful artifact"
2. "for a specified problem domain."
3. "Because the artifact is "purposeful," it must yield utility for the specified problem. Hence, thorough evaluation of the artifact is crucial"
4. "the artifact must be "innovative," solving a heretofore unsolved problem or solving a known problem in a more effective or efficient manner."
5. "The artifact itself must be rigorously defined, formally represented, coherent, and internally consistent."
6. "The process by which it is created, and often the artifact itself, incorporates or enables a search process whereby a problem space is constructed and a mechanism posed or enacted to find an effective solution."
7. "Finally, the results of the design-science research must be communicated effectively"
8. "both to a technical audience (researchers who will extend them and practitioners who will implement them) and to a managerial audience (researchers who will study them in context and practitioners who will decide if they should be implemented within their organizations)."

This approach seems well suited for this project. It is certainly our aim to create an innovative artifact that can actually be used, even though it is just a prototype.

- We have already described the problem domain in section 1.
- We will clearly define the artifact we have created and the process that has taken place to create it.
- We will also describe our evaluation approach in section 9.
- We can't promise that the results of our research will be directly applicable to a managerial audience, but we will try to explain what promises this research holds for the future (section 13).

We can therefore say the way we approach this project is soundly based on design science. Design science cycles between a design phase and an evaluation phase. The design will mostly be a creative process, while the evaluation phase will be based on theory about game design.

4 Game design

In this section we will give a short introduction to game design theory. We will expand on this when we describe the testing of our game(section 9).

What is it like do develop a game? How does it differ from the development of another system like a piece of software? Any system might be "playable" in a strict sense, as it is executable and usable by a human being, but whether or not it qualifies as a game is not certain. There are certain elements that make a game what it is. The things that make a game fun, addictive or immersive.

Järvinen describes nine game elements [11]. Any game can be explained in terms of these elements. However, certain elements are common to information systems in general and others are typical for games. We will give an overview of these elements and describe what each one means in terms of game design and how we can use this.

Components Components are all the objects that a player is able to manipulate and possess in the course of the game. In an object-oriented view of the world "components" describe pretty much anything, so it does not really guide us when designing a game.

Environment The environment embodies the constraints of system. It is often made up of components that can't be manipulated by the player. Much like components, the environment is not central to what makes a game a game.

Rules The rules are arguably the most important part of any game, but in our case we are somewhat limited in what we can do here. Rules encompass both goals and procedures, but our game already has an external goal. The external goal is of course to obtain the information needed to create a process model. We do not want the player to care about this while playing the game though. Therefore we present another, internal goal to the player. For example to score as many points as possible. The external and internal goals are different, but the internal goal must still be instumental to the external goal. This means we are not free to design this element as we please. We have more freedom when it comes to the procedures, but procedures are of course also based on the goals.

Game mechanics The ways in which a player interacts with a game (game mechanics) are very important. When done wrong, it can easily lead to frustration. For example in our case, the players have to enter a lot of information, but typing is not a particularly fun task. We should therefore perhaps try to change the game mechanics, and limit the amount of typing, by letting the system do most of the work (while still letting the player have a feeling of being in control). Typing could perhaps be replaced by dragging and dropping as much as possible.

Theme A game's theme is something that contextualizes the ruleset, and a good theme is the key to making the players believe they are doing something else than what they are actually doing. We would like to make the players forget they are basically filling in a form, but we are severely limited since the subject matter of the game

(the only thing that really changes like a theme can be changed) is actually the domain to be modelled. For example, it would be fun to model how the knight saves the princess from the dragon, but ultimately the theme is not something we can control.

Information The way the game's *state* is communicated to the players is very important, and luckily this is one area where we have a lot to work with. The main target is to give the players a sense of progress and an idea of the quality of their work (if possible compared to other players). This game element might be the most interesting, especially when it comes to communicating this "quality", because there is indeed a large body of work on "quality of modelling". It is a complicated and often quite vague collection of concepts, so here lies possibly the biggest challenge of this project: to translate quality to something the player can understand within a second.

Interface The interface is the physical way in which the player interacts with the game. If we choose to make a automated game we are pretty much limited to standard input devices (keyboard and mouse).

Player(s) The players are very important parts of the game system as a whole. In our case it is especially important how players respond to the game. For the game to succeed we have to be very conscious of the level of knowledge of the players, as well as their skills and abilities. These are things that are obviously also very much influenced by the *information* the game provides.

Contexts The context of a game can be something like the culture and tradition surrounding it, or the relative importance (what is on the line for winning or losing?). The context never really changes the game but it does change the behavior of the players. We have to consider that we will not be able to test our game in its proper context. In reality a tool like this will be part of a larger development process, a dynamic environment with many people involved, all with different things at stake. While this context is not a big factor for us now, it might be in the future.

While components, environments and rules usually make up the meat of a game, we are not really free to design them because of the specific purpose of our game. Our game represents a basic task that is next to impossible to make "fun".

Game mechanics, theme and information can still make or break a game though, and since we do not have full control over the theme either, we can conclude that there are just two game elements that we should really focus on when it comes to game design: game mechanics and information. In other words: input and feedback. These two things will be the key to making the game as enjoyable as it can be.

Obviously it is a matter of opinion if a game is fun, but that is exactly the reason why we need to test it. Literature on game design describes a lot of properties of succesful games, but that does not mean that there is a step-by-step guide to designing such a game. The reason for this is that, according to Salen and Zimmerman [18], "Game

design is a second order design problem". Meaningful game experiences are almost always emergent from the basic rules. You can't directly design human emotion, and that is why designing is less science than it is art.

The evaluation phase of our design cycle is therefore especially important when it comes to the "gameness" of our system. What we need to do is develop an evaluation method that incorporates what we can learn from the literature on game design.

5 Development

In this section we will provide a very short overview of the development process. For a detailed description of the different versions of the game we refer to appendix A.

We initially tried to stay away from automating the game, focusing instead on a pen and paper approach ("board game"). This gave us the chance to think through the basic setup and rules, without getting lost in the details of implementation. However, the downsides of board gaming also became apparent soon. Playing and therefore testing the game proved to be a tedious experience for both the facilitator and the player.

The first digital version of the game soon appeared, based on a standard spreadsheet implementation, which at least took the task of calculating the score out of the hands of the facilitator, and also made the input for the player a lot easier. This gave us the chance to do the first (successful) tests with "outsiders" (i.e. players other than ourselves). This proved that the game system was viable in principle and that players could at least get through the game.

It became increasingly clear at this point that the expectations for, and experience of, playing an actual computer game rely not just on the rules, but also on flowing interaction, animation and sound. We concluded the same thing in the previous section. Game mechanics and information, or input and feedback, are the two most important elements. We really cannot develop a proper game without focusing on these. The choice was therefore made to rebuild the game from the ground up in Actionscript 3.0 [19], which after a few iterations, led to the current prototype.

6 Theory behind the game

While working on the game we developed a particular view of how processes work. Where most process models focus on how steps are connected, we focus instead on the inner workings of the steps, and try to derive the connections from that information.

This may seem arbitrary, but by asking people that play the game to think about a process in a different way as usual we force them to think about it a little deeper than usual as well. The player should think about the process indirectly. The focus should be on internal consistency instead of the end result (which might otherwise just be a nice looking diagram).

The way we see it, a process P is:

1. A set of steps $S_p = S_1, ..., S_n$
2. Each step has a set of input items $S(I)$ and a set of output items $S(O)$, so that $S(I) \cap S(O) = \emptyset$
3. The production relation $S_x(I_a) \rightarrow S_x(O_b)$

By production relation we mean that the step produces something, and to do that, there should be a link between what goes in to the step and what comes out of the step. This is all the information that is required as input from the user. Strictly speaking the production relation is not even necessary for what follows, but in our actual implementation it is.

We can take a look at what more we can say about our view of a process. The following information can be derived from what we already know, and does not require any further input from the user.

- The equivalency relation $S_x(O_a) = S_y(I_b)$, where $x \neq y$
- S_y follows S_x if and only if $\exists S(O_a)$ and $\exists S(I_b)$ so that $S_x(O_a) = S_y(I_b)$
- S_y depends on S_x if S_y follows S_x. This dependency relation is transitive.

There can be problems deriving the equivalency relation from the informal input, since items that are actually equal are not always described in exactly the same way. You might have to use some sort of intelligent search method to find similar items[1]. However, for now we will just try to stimulate the player to provide correct input.

Looking at the *follows* relation we can see why the input and output sets can share no elements. An element that is present as both input and output of a step is what we would call a *tool*. Something that is used during the step but does not change at all. It does not make sense to include tools in the output set, because multiple steps using the same tool would then be seen as being somehow dependent on each other.

[1]This obviously is not a new problem. It is actually very similar to "model mining"; finding dependecies between steps based on logs of actual executions of a real world process [1] [21].

6.1 Deriving dependency from input and output sets

As we noted above, finding matching input and output items between different steps is not trivial. To make matters more complicated, in our implementation of the game you can have two kinds of input and output items. This is the result of the choice that has to be made between describing items as object/attribute or simply describing each occurence of an objects as a separate object.

For example: you can either describe mashed potatoes as the object "potatoes" with the attribute "mashed", or simply as the object "mashed potatoes". The latter is probably easier to implement, but depending on the situation, one of the two options might feel unnatural to the player. Therefore we have decided to leave it up to the player which one to use.

Aside from that, this would be the basic algorithm to derive the *follows* relation:

```
for each step x
  for each input item
    for each step y
      if step x is not step y
      for each output item
        if input item equals output item
          step x follows step y
```

In the next section we will see what this algorithm will look like in practice.

With this information you should theoretically also be able to generate a graphical representation of the model. This could be done by simply moving each step to the right of the step it follows, although this assumes that the process is acyclic. If there any loops in the process this implementation will break.

As we stated above, we think this way of looking at processes encourages people to think more thorougly about a process. In fact, it requires from the player an argumentation about why certain steps follow others. This works particularly well when it comes to whether two steps are performed in series or in parallel.

When describing a process in steps like you normally would in a conversation, you tend to describe steps in series by default, even if some steps might in reality also be performed in parallel. In our view however, parallel is the default, and argumentation in terms of input and output items needs to be provided to put steps in series.

This is likely to result in the emergence of AND-joins that may otherwise have been forgotten. The following diagram (figure 1) shows how describing the input and output of activities leads to AND-joins [7].

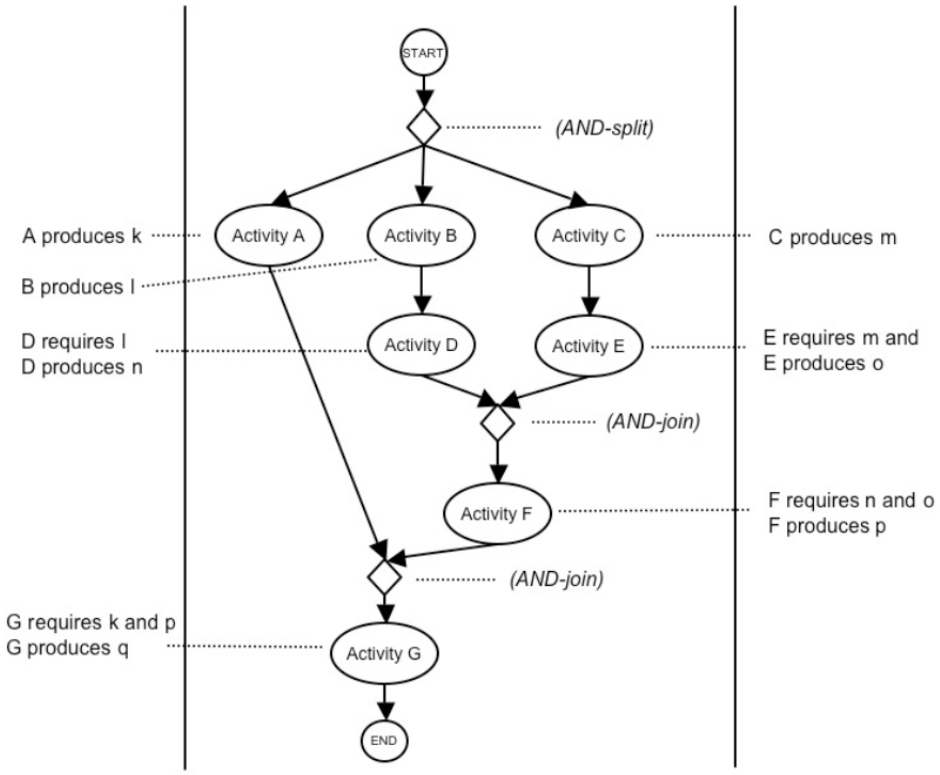

Figure 1: AND-joins

7 Description of the game

In this section we will try to give a detailed description of the experience of playing the game, illustrated with screenshots.

When the game starts the first thing the player sees is a screen with instructions. The appearance should be somewhat familiar to players because it is inspired by the way the instructions of most boardgames are set up. We have separate sections outlining the goal of the game, the way the game is played, and the end conditions.

The instructions are kept very brief on purpose. We do not expect players to be able to process more information at this point. The most important thing is that they grasp the general idea behind the game.

The full text is reproduced below. The actual screen can be seen in figure 2.

Figure 2: Entering the name of the task

The Describe-A-Task-Game

Goal of the game:

The goal of the game is to describe a task as clearly as possible. You can pick the task you want to describe. It is best to choose something you're very familiar with.

If you do well (your description has to follow the rules, so the computer can understand it) you score points. Obviously the goal is to get the highest score possible.

Playing the game:

Giving your task a name is a good start, but that's only the beginning. Each task consists of steps. That means every game starts by creating at least one step. After you've given the step a name you can describe what the ingredients of the step are, what the products of the step are and how the ingredients and products are linked.

If you do this well the game will find out by itself how the different steps are connected and in what order they belong. That means you're free to describe the steps in any order you like!

End of the game:
You're the one to decide when you're done with your description. The only condition is that all steps are connected. If any loose steps are left it might be best to delete them. You're final score will now be calculated based on the points you scored and the elapsed time.

Tip: pay attention to the text that appears on the bottom of your screen if you move your mouse over something.

After reading the instructions the player can choose to select two options: "Show extra tips for beginners during the game" and "Automatically move steps in the correct order". The former is definitely recommended for people playing for the first time, but the latter is still experimental and does not work well. As we described in the previous section, this tries to move steps based on which steps they follow. This tends to give strange results when a cyclic process is created (which sometimes tends to happen). The player can now click the button to start the game.

Just as we learned from the instructions, "giving your task a name is a good start". In figure 3 we can see an empty canvas and the place where the player can enter the name of the task. The speech bubble-looking thing is one of the extra tips for beginners. As an example we are going to describe the task "making coffee".

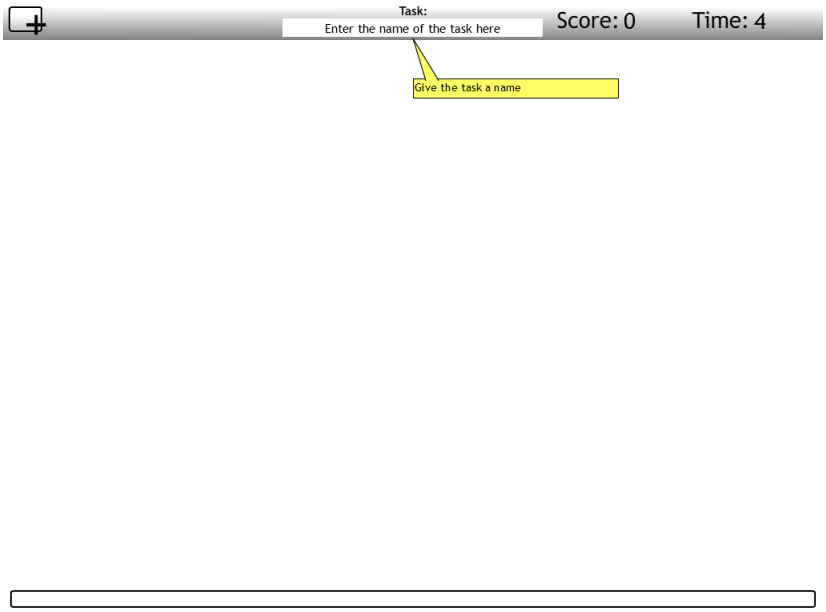

Figure 3: Entering the name of the task

After entering the name of the task the player is asked to create a first step by clicking the button in the top left (figure 4). A new empty step is created and the player is asked to give it a descriptive name (figure 5).

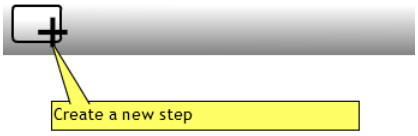

Figure 4: Creating a new step

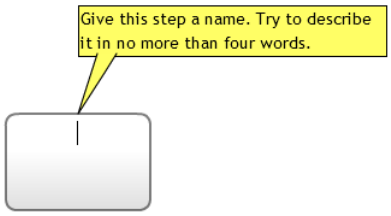

Figure 5: Giving the step a name

If you move the cursor over the step, a few buttons become visible. The buttons on the top allow you to move or delete the step, and by clicking the button on the left of

the step the player can enter ingredients for this step (figure 6). When an ingredient is added it also shows up on the top left of the screen in a list of all items currently in the game (figure 7).

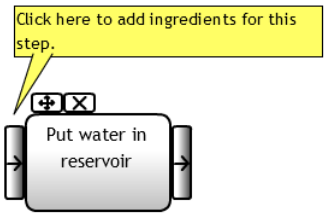

Figure 6: Expanding the list of ingredients

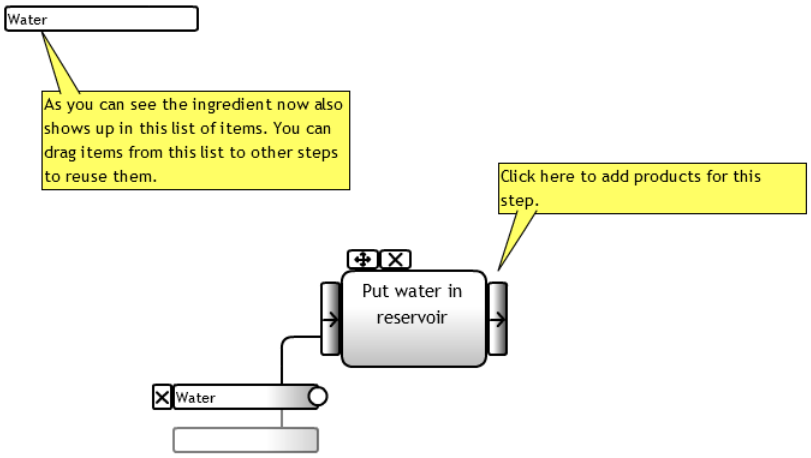

Figure 7: An item in the item list

Adding products for this step works the same way. When a product is added, the player is asked to connect an ingredient with a product by clicking on the bubble next an ingredient and then on the one next to a product. Only then does the product really become active (figure 8).

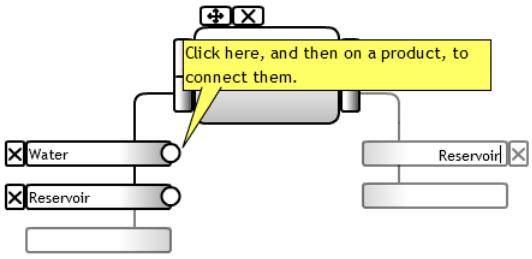

Figure 8: A product is added

When such a link is made, the player is asked to provide additional information about what is being changed about the ingredient (figure 9, 10). In figure 9 we can also see the animation that appears when ten points are scored for creating a link.

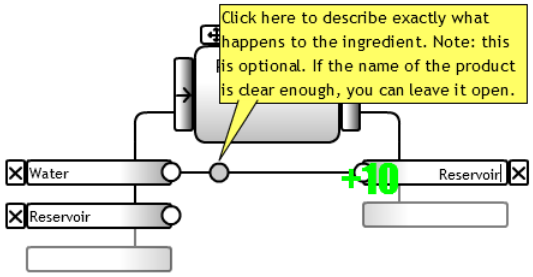

Figure 9: Creating a link

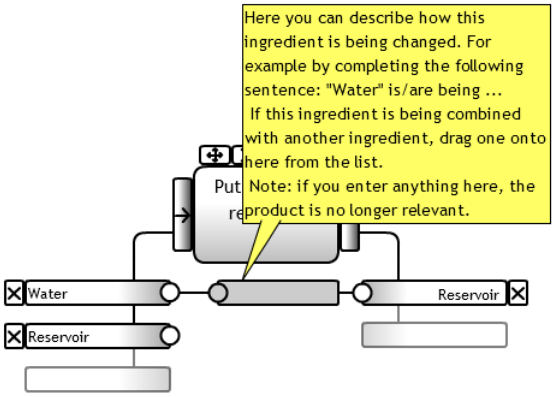

Figure 10: Entering a description of a link

One suggestion in figure 10 is to drag an item from the list to a link to combine the list item with the linked ingredient. If we link the ingredient *Reservoir* with the product of the same name and drag the item *Water* to that, the text “with Water”

is automatically added to the link and the two links in the step are also graphically combined (figure 11). The result is that we now have a *reservoir with water* and *water in reservoir*.

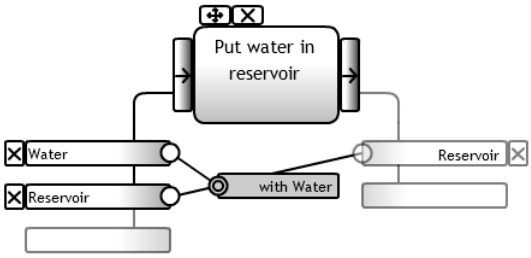

Figure 11: Two combined links

When we add the item "reservoir" as an ingredient to another step, the game recognizes that it has the attribute "with water" and that this is a possible precondition for this step, as can be seen in figure 12. If we check the box next to this precondition the two steps are linked and a triumphant sound (ta-daaaa) plays. Figure 13 shows that 100 points are scored when the connection is made.

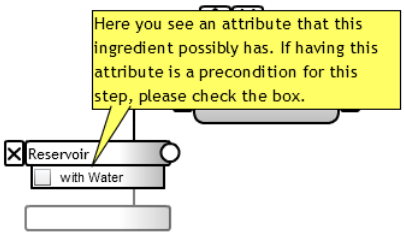

Figure 12: A possible precondition

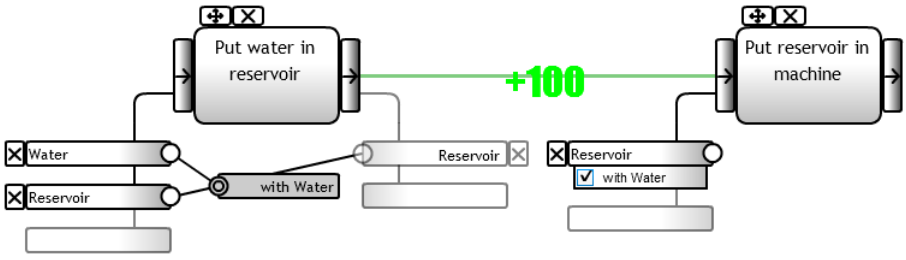

Figure 13: A connection is made

When all steps are connected to at least one other step, the task description is seen as valid and the player can finish the game by clicking a button at the top (figure 14).

A screen shows up that informs the player of the final score. The player is presented with the option to go back to the game to expand the description and score more points (figure 15), but usually this is where the game ends.

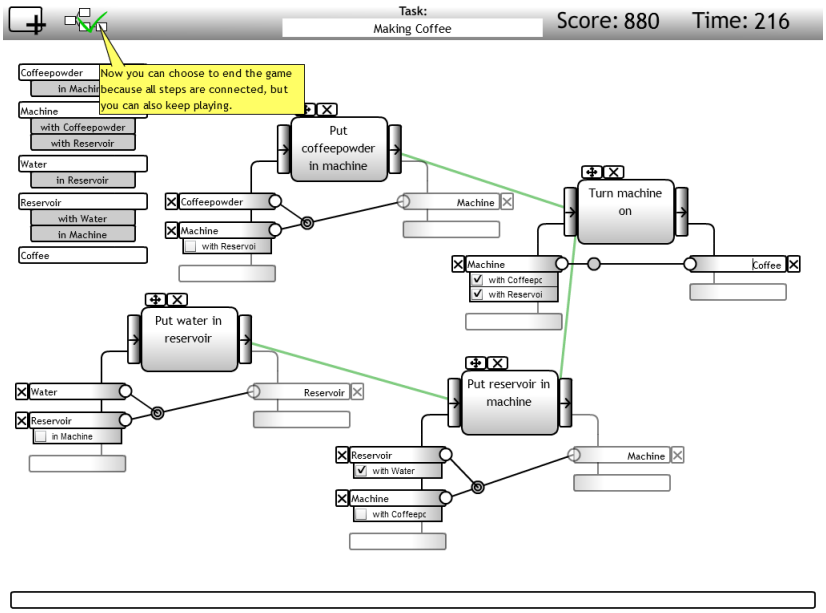

Figure 14: The finished product

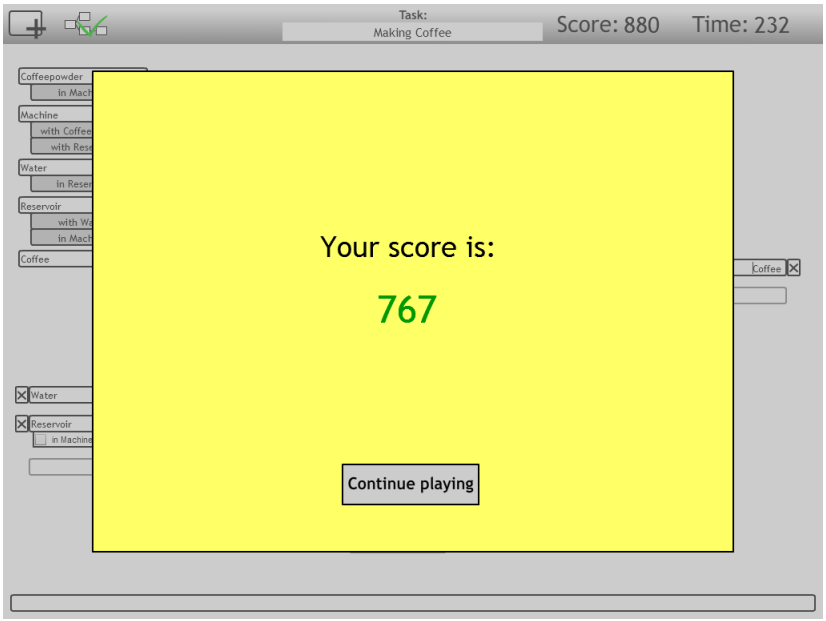

Figure 15: The final score

The score is calculated as follows:

- 100 points for each step.
- 100 points for each connection between two steps.
- 10 points for each ingredient, product or link.

The final score is the sum of those minus half a point for every second played. However, the amount of points deducted based on the elapsed time can never be more than half of the amount of points scored.

7.1 Algorithm to find step connections

In the previous section we promised to provide the actual implementation of the algorithm to find step connections based on sets of input and output items. Below we have tried to describe it in pseudo code. We start by finding the sets of input and output items for each step:

```
for each ingredient
  for each precondition
    if precondition is selected
      add "precondition_ingredient" to input set
  if no preconditions are selected
    add "ingredient" to input set

for each product
  if product is relevant
    add "product" to output set

for each link
  if link is not empty
    add "link_ingredient" to output set
```

Note:

- Whether a product is "relevant" (not overwritten by a link and not present as an ingredient) is known beforehand.
- A link always has a reference to its corresponding ingredient and product.

Then find the connections between the steps:

```
for each step x
  for each input item
    for each step y
      if step x is not step y
```

```
for each output item
  if input item equals output item
    step x depends on step y
```

This last part is actually exactly the same as the theoretical algorithm we suggested before.

7.2 Game elements revisited

We will now describe our prototype in terms of the game elements we first introduced in section 4.

Components The objects that the player can manipulate during the game are: steps, ingredients, products, list-items, links, preconditions, and various buttons on the GUI.

Environment The environment is the canvas on which steps are placed.

Rules The goals we present to the player are: describe steps in such a way that they all become connected and score as much points as possible. The procedures are obviously too complicated to reproduce here, because they are mostly embodied in the code, but they are described quite clearly in section 6.

Game mechanics We have really tried to make the game mechanics as user friendly as possible. We have replaced typing with dragging and dropping wherever possible, and by using visuals and sound as feedback we try to make the actions of player feel as natural as possible. How this works in practice is of course hard to say and only after testing the prototype will we know if it has the desired effect.

Theme Our game does not have a theme, unless we view the domain to be modelled as the theme.

Information We have focused a lot on how to communicate information to the player. The player must at all times know what the goals are, what tools are at his disposal to reach them and how well he is doing. In practice this means that the player needs to read the instructions well, that he knows what all buttons do, and that he is always conscious of the score.

Interface We were limited to a standard interface (keyboard and mouse).

Player(s) The player-element has turned out to be very important, especially in relation to the information element. It was a challenge to present the information in such a way that all players could understand the game, and during the tests it was interesting to see how different players with different mindsets responded to the game.

Contexts As we have said before, we cannot test the game in a realistic context, but the next section demonstrates that we are conscious of the context on a technical level. The game produces information that is useful outside of the game.

In our experience not all of the game elements are equally important in this development process. As we already predicted, the focus was mostly on the game mechanics and the information elements. That is what we spent most time on, and these are also the elements that have changed the most during the development process, because they are so dependent on the specific implementation.

8 XPDL compatibility

To prove the point that our game can elicit useful information, we can output a process defintion in XPDL. XPDL (or XML Process Definition Language) is a standard file format used to interchange process models between different tools. XPDL encourages vendors of different products to include the ability to import a process definition from XPDL and "export a process definition from the vendor's internal representation to XPDL" [2]. We can obviously only provide the latter.

An XPDL package corresponds to a Business Process Diagram (BPD) in BPMN (Business Process Modeling Notation [23]), and can also include graphical information. We will introduce the most important concepts of XPDL and show how we derive them from our game.

An XPDL file is an XML file that consists of a package that, aside from the appropriate meta-information, can contain multiple workflow processes. We will only use one workflow process though. This workflow process has a header that we will use to store the elapsed time and points scored in the game, as well as a timestamp. The actual description of the process consists of three essential parts:

Participants Participants (the actors who perform a task) are not defined in our game, so we will define a default participant simply called "Player". In BPMN a participant typically corresponds with a swimlane in a pool.

Activities The steps in our process are called activities in XPDL. We can easily derive these. All activities have a participant as a performer (in this case our default participant).

Transitions The transitions are the connections between our steps and can also be directly derived from the game. They contain a *from*-activity and a *to*-activity.

Below you will see an example of an XPDL file as outputted by the game.

```
<Package xsi:schemaLocation="http://www.wfmc.org/2008/XPDL2.1 bpmnxpdl_31.xsd"
Id="td1" Name="Task Description"
xmlns="http://www.wfmc.org/2008/XPDL2.1"
xmlns:xpdl="http://www.wfmc.org/2008/XPDL2.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<PackageHeader>
    <XPDLVersion>2.0</XPDLVersion>
    <Vendor>Describe-A-Task Game</Vendor>
    <Created>2009-06-13_01-19-07</Created>
  </PackageHeader>
  <WorkflowProcesses>
    <WorkflowProcess Id="wp1" Name="Making Coffee">
      <ProcessHeader>
        <Created>2009-06-13_01-19-07</Created>
        <Description>Score=880Time=138</Description>
      </ProcessHeader>
      <Participants>
        <Participant Id="p1" Name="Player">
          <ParticipantType Type="HUMAN"/>
        </Participant>
```

```
      </Participants>
      <Activities>
        <Activity Id="step1" Name="Put coffeepowder in machine">
          <Implementation>
            <No/>
          </Implementation>
          <Performer>p1</Performer>
        </Activity>
        <Activity Id="step2" Name="Put water in reservoir">
          <Implementation>
            <No/>
          </Implementation>
          <Performer>p1</Performer>
        </Activity>
        <Activity Id="step3" Name="Put reservoir in machine">
          <Implementation>
            <No/>
          </Implementation>
          <Performer>p1</Performer>
        </Activity>
        <Activity Id="step4" Name="Turn machine on">
          <Implementation>
            <No/>
          </Implementation>
          <Performer>p1</Performer>
        </Activity>
      </Activities>
      <Transitions>
        <Transition Id="t1" Name="Arrow1" From="step2" To="step3"/>
        <Transition Id="t2" Name="Arrow2" From="step1" To="step4"/>
        <Transition Id="t3" Name="Arrow3" From="step3" To="step4"/>
      </Transitions>
    </WorkflowProcess>
  </WorkflowProcesses>
</Package>
```

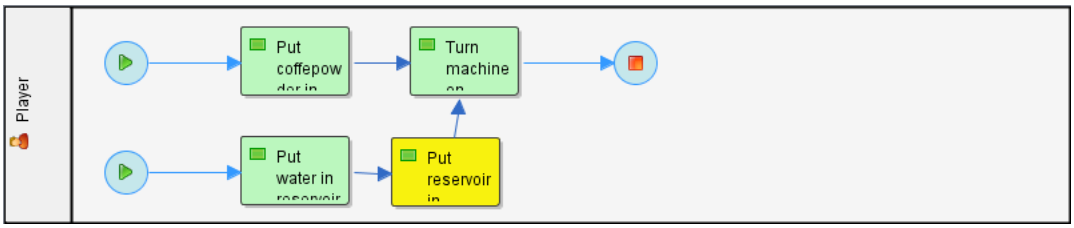

Figure 16: A diagram generated from the XPDL file

We can actually open this file in a different tool. Together Workflow Editor (Community Edition) [2] is a tool that works well with XPDL. When we open the above as an XML file in the editor the diagram below (figure 16) is automatically generated. Without providing graphical information, we only had to move the bottom right activity a bit.

[2] http://www.together.at/together/prod/twe/twecommun/index.html

9 Testing

Determining what a good game is, is far from easy. Common platitudes like "easy to learn, but hard to master" are only useful to a certain extent. We need a comprehensive list of properties of highly succesful games, especially those relating to the user's input and the system's feedback (game mechanics and information). We can then describe how we can see if the game we have designed possesses (parts of) these properties.

Based on game design literature, we have found a number of such concepts, and made some further observations.

Discernability/effectance, and Meaningful choice

Discernability simply means "Perceiving that ones actions have effect on the surroundings" [11]. Does the player feel like his actions have meaning (meaning is created by feedback)? Does he feel like he is in control? These may be vague questions to ask someone, but it is important to get an idea of the general experience of the player.

The player's action should also be "integrated", which means that the feedback is not only immediate, but it influences the rest of the game (with the purpose of attaining a goal) [18]. We might ask the player if he feels like his choices are arbitrary?

Purpose

Some authors [18] argue that games can never be serious or associated with material gain, or that they should take place outside of ordinary life. Games should be played voluntarily and should always be make-believe. This view matches with the view that games are a form of art, since art is also often defined as having no practical purpose outside itself.

This is not the kind of game we are trying to make. However, there is no reason games can't take the shape of both a work of art and a tool. Like a beautiful movie or a painting can have a purpose as propaganda, advertisement or instruction. Commercials can often be appreciated for their creativity and enjoyed as a work of "art", but still have a purpose.

This is the same thing we try to achieve. Our game obviously has a practical purpose, but what do the players experience when they play it in terms of the meaning of their actions? In our game a player's actions already have meaning in a modelling context, but the challenge is to obscure this and replace it with meaning in a game context. The modelling context is still present but it is another layer of meaning.

Our game is a very special case because it specifically leaves open the "magic circle", the boundaries of the game. The game-world is filled with real world content, and the game has a real world purpose. This will result in a conflict if the meaning within the game does not line up with the meaning outside the game.

Games usually reward people for forgetting the real world meaning of their actions. In this case the opposite should happen. If the "modelling" in the game does not make sense in the real world, it is not a modelling tool, but if it is just a modelling tool it is not a game. The big challenge is to get this right.

Quantifiable outcome

Salen and Zimmerman's [18] definition of a game states that: "A game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome".

We have already discussed how we try to create an artificial conflict (aside from the real conflict of modelling) and we certainly define rules. Our real challenge is to make the outcome quantifiable though.

One of the core elements of our efforts to turn modelling into a game is keeping a score. That is the quantifiable outcome we aim for, but how do we get there? We attempt to translate quality of modelling into a number. How can we know if we succeed?

We can ask the players if they felt like the score they received was a fair representation of how well they think they did. To properly test this, players should play the game multiple times.

Another important thing is that games should not have a dominant strategy: an ideal script that always gets the best result [11]. We can ask the players if they felt like there was a trick to win, or easily get a high score.

Rules

Although we focus on mechanics and information, we do not exclude the rules of the game. It is especially important how the rules are communicated to the players. In that sense, rules can also be viewed as information.

It is interesting that some games are designed to give the players imperfect information regarding the rules. There is a whole category of games where the goal is to discover the rules, such as "The Green Glass Door", where a few players know the rules and give examples of what objects do and do not belong behind the green glass door. The other players try to discover what the rules are that dictate which objects do what. This is similar to a game we used to play during the university introduction week, called "the land of no idea" (only words written without the letters i and d exist in this magical land).

There is yet another type of game with undisclosed rules, such as "Mornington Crescent" or "Stiften". In this case some players are led to believe that the game has a complicated and sophisticated ruleset, but in reality the rules are made up on the spot. Of course these are not actually games. They are intented to make fun of gullible new "players", or meant as a satire of other games.

What this sidenote shows is that obscuring the rules can have a function. In our case the purpose of could be different though. Maybe the player does not need to know exactly how scores are calculated, because this might lead to focusing too much on the score itself. A lot of digital games do the same thing.

What we want to know is whether the rules are communicated to the players in such a way that they know what to do and how to do it well. Are they too specific or too vague?

Motivation and enjoyment

There are two kinds of motivation, goal orientated and intrinsic motivation (aesthetic behaviour). It seems that a good game needs to possess both. That is why we distinguish telic and paratelic metamotivational modes [11]. The telic mode focuses on the goal of the activity. In our modelling game the motivating goal is not so much the resulting model, as much as the score. Although maybe if we could generate a full graphical diagram from the users input, this would be more interesting to see for the user. It could be a motivating factor, but it is not the most obvious.

Instead we should mainly focus on the paratelic mode: pleasure from the activity itself. Perhaps factors such as "satisfaction of skilled performance" could be attained by proper feedback. Other than that we cannot expect the player to truly enjoy the task, but it should be as inobtrusive as possible.

Järvinen states [11](p. 109): "Playing games has been characterised as an autotelic activity, which means that its purpose is itself only, and there are no other instrumental functions or purposes to it. Therefore the motivation to play is in most cases intrinsic rather than extrinsic. Extrinsic motivations manifest if the game system is an open system and it acknowledges information from system contexts (such as other systems) to be imported into it, meaning that there is a prize of some kind money, status, fame for the winners. The point is that it matters also outside the system, i.e. motivations expand to the contexts of a gaming encounter."

This is interesting because in our case the game is indeed an open system. We might want to obscure this as much as possible, but we need to recognize that extrinsic motivation does play a role. We need to find out what the players see as their goal.

Järvinen lists a number of possible goals. Those most applicable to our game are: configuration (especially logical), gaining competence, accomplish (merely getting trought the game in itself), and accumulating (points).

We can ask the players how much they felt they had the following goals in mind while playing the game: Solving a logical puzzle, trying to become better at the game, merely getting through the game, getting a high score (and beating other players). We can add to that the external goal of accurately describing a process, and maybe seeing the result of their work (in graphical form).

We could also ask what the players think they need to do better if they play the game again. Describing the process better or understanding the rules better.

Enjoyment is closely related to motivation. Järvinen states that there are some prerequisites for players to enjoy a game. Relevant here are the willingness to accept the ruleset (suspend disbelief), the desire to relate to the goals, and the desire to play well (virtuosity). We can ask the players if they felt this way.

Rhetoric

Games are often multimodal, featuring rhetoric in the shape of text, animation, sound, touch, etc. What needs to be communicated by rhetoric in whatever form is this [11]:

Gratification rhetoric Getting a player to start in the first place.

Motivation rhetoric Getting the player to get into the game, assuming the role he has as a player.

Goal rhetoric Communicating the goal of the game.

Means rhetoric Communicating what tools and option the player has at his disposal to reach this goal. This is related to the role player has and therefore with motivation rhetoric.

Feedback rhetoric Involves both *valence rhetoric*, which encourages, praises, or punishes the player regarding his performance, and *goal resolution rhetoric*, which is about communicating the progress towards a goal.

Outcome rhetoric Involves both *end rhetoric*, which lets the player know if the end has been reached (or is near), and *victory rhetoric*, which is what happens after the end. In our case, the way the final score is communicated.

This shows that the element of *information* has many purposes and is rightfully understood as very important for our game. All these types of rhetoric need to be considered.

9.1 Standard heuristics

Obviously we are not the first to think about the playability of games. In the existing literature we can find some standard heuristics for what makes a good game.

Desurvire et al. [3] provide a list of such heuristics. While not all of them are applicable (for example, there is a category called "game story"), some are useful.

1. Players fatigue is minimized by varying activities
2. Provide clear goals, present overriding goal early as well as short-term goals throughout play.
3. Game play should be balanced with multiple ways to win.
4. Player experiences fairness of outcomes.
5. Players should perceive a sense of control and impact onto the game world.
6. Provide immediate feedback for user actions.
7. Game should react in a consistent, challenging, and exciting way to the players actions.
8. The first player action is painfully obvious and should result in immediate positive feedback.
9. A player should always be able to identify their score/status and goal in the game.

10. The Player experiences the user interface as consistent (in control, color, typography, and dialog design) but the game play is varied.

11. Players should be given context sensitive help while playing so that they do not get stuck or have to rely on a manual.

12. The interface should be as non-intrusive to the Player as possible.

13. Player should not experience being penalized repetitively for the same failure.

9.2 Experiences during preliminary testing

Since we are interested in player's responses we would be well advised to learn from actual responses players have given during preliminary testing, to see which issues are most important to the user.

Below is a list of the most important remarks we made during an earlier round of testing with a previous version of the game. They serve just as example of the issues that we can expect.

1. It needs to be even more clear what gets you points and what does not. Player's attention needs to be directed towards the score. Once they get the hang of how the scoring works things seem to improve.

2. Most of the "tips" the game provides are noticed by the players and the information in them is seen as valuable.

3. It is also hard for the players that sometimes their natural way to explain a step does not quite fit the framework of the game. Frankly the game sometimes requires somewhat unnatural formalization.

4. On the other hand there is too much freedom. The game does not force the player to enter a *correct* process at this point. Players remarked that it is easy to get points by just entering nonsense.

5. Things like automatically keeping track of item/properties seemed to be appreciated by the players. On the other hand the step description was seen as rather unnecessary and time-consuming by the players.

6. It was interesting to see how players responded to the game with little introduction, but it soon became clear that more explanation is necessary.

Lacking a fully developed testing method, we used Nielsen's 10 usability heuristics [14] during preliminary testing to classify these remarks. While these heuristics are not specifically suited for games they are useful. Not only is there quite a lot of overlap between these heuristics and for example the heuristics of Desurvire et al. [3], we are also dealing with a system that is half game, and half modelling tool, so it is probably a good idea not to lose sight of what information scientists are familiar with.

9.3 Questions

To see where all this information leads to, we have made a table containing an overview of (possible) questions we can ask the players testing the game. The table shows how the questions are based on the multiple sources.

Salen & Zimmerman	Järvinen	Desurvire et al.	Nielsen	Suggested Questions
General experience				
Discernability	Effectance	Provide immediate feedback for user actions		Did you feel like had control?
Actions are integrated	-	Players should perceive a sense of control and impact onto the game world	-	Did you feel like the choices you made were arbitrary?
		desire to relate to the goals, desire to play well		Did you care about doing well?
	Valence rhetoric			Did you feel encouraged by earning points?
	Goal resolution rhetoric			Did you feel satisfied when you were finished?
	Gratification rhetoric			Did the presentation of the game interest you in playing?

Goals and outcome				
Telic motivation / Extrinsic motivation?	Gain competence, accomplish, accumulate, etc.	Provide clear goals	-	What did you think was the most important goal in the game?
		Willingness to accept the ruleset	Match between system and the real world	Did you feel like you played better the second time? What did you have to do well to reach the goal (modelling, or playing the game)? Did you feel like you were playing a game?
	Games should not have a dominant strategy	Game play should be balanced with multiple ways to win	Match between system and the real world	Did you feel like there was a trick to winning?
A game [...] has a quantifiable outcome		Player experiences fairness of outcomes	-	Did you feel like your score was a fair representation of how well you did?
Information				
		Provide clear goals	-	Did you understand the goal of the game?
		The first player action is painfully obvious		Did you think it was easy to get started?
	Means rhetoric			Did you understand what you could do to reach the goal?
	Outcome rhetoric			Did you know when you were finished?
				Did you understand how the scoring system worked?
	Feedback rhetoric	A player should always be able to identify their score/status and goal in the game	Visibility of system status	During the game, did you always know how well you were doing?
		Players should be given context sensitive help while playing	Help and documentation	Did the game give you enough help relevant to the context?

Frustration				
		Players fatigue is minimized by varying activities		Did you think the gameplay was varied enough?
		The Player experiences the user interface as consistent	Consistency and standards	Did you feel like the game worked consistently?
		Player should not experience being penalized repetitively for the same failure	Help users recognize, diagnose, and recover from errors	Did you feel like you could easily recover from errors?
Paratelic motivation		The interface should be as non-intrusive to the Player as possible	Aesthetic and minimalist design	Did you ever feel frustrated or hindered by the interface?
			Flexibility and efficiency of use	Did you feel like the game helped you to work fast?

9.4 What can we expect to learn?

When it comes to the actual testing, what is that we want to know? From a design science perspective, we are obviously interested in whether or not we have made a good game so far, and what can still be improved. From a more general perspective, we will focus on two questions:

- What are the differences between players with, and players without process modelling expertise?
- What are the differences between players playing the game for the first time and players playing the game for the second or third time?

Both will of course say something about the feasibility of taking the expert out of the modelling process (disintermediation), and therefore the feasibility of this whole project.

10 Test setup

We tested the game with ten different players. Five with prior experience in process modelling and five with little to no prior experience in process modelling. Each player was asked to play the game three times. Twice with a standard task and once with a task of their own choice. The standard tasks were "making coffee" first and "repairing a flat (bicycle-)tire" second. Two tasks almost everyone is familiar with, the second generally being a little more advanced than the first.

The players were given as little introduction as possible, forcing them to rely on the explanation provided by the game. Some subtle hints were given during the game, when it looked like they were really getting stuck. The players were asked to "think aloud" as much as possible and to voice their possible frustrations or confusion.

The players were closely observed while playing and the observations were written down, as were any interesting comments made by the players.

After the three games, the players were given a questionnaire. The first part consisted of 25 statements. The players had to say how much they agreed with them, choosing between: *strongly disagree - disagree - no opinion/neutral - agree - strongly agree*. The statements were based on those suggested by the literature (see section 9), and therefore all had to do with the players' general experience. The statements did not go into specifics of this particular game (aside from two questions about the time element).

These were the statements:

1. I thought it was a fun game.
2. I thought the game was varied.
3. I felt like I had control over what happened in the game.
4. I felt like the choices I made were important for the progress of the game.
5. I thought it was important to play well.
6. I felt encouraged by scoring points.
7. I felt encouraged by the time that was running.
8. I thought it was annoying that the time was running.
9. I was satisfied with the result when I was done.
10. The appearance of the game invited me to play.
11. I felt like I was playing a real game.
12. I felt like there was a trick to easily score well.
13. I felt like the score was a fair representation of how well I was doing.

14. I felt like the second (or third) time I played better.
15. I understood well what the goal of the game was.
16. I found it easy to get started.
17. I understood what I could do to reach the goal.
18. It was clear to me when I was done.
19. I understood well how I could score points.
20. During the game it was clear whether I was doing well.
21. The game provided enough context-sensitive help.
22. I thought the interface/controls worked consistently and clearly.
23. I found it easy to fix possible mistakes.
24. I felt frustrated or hindered by the interface.
25. The game helped me to work quickly.

The second part of the questionnaire consisted of 3 open questions and some space for general comments.

1. What was, according to you, the most important goal in the game?
2. What did you have to do well to reach this goal?
3. What went better when you played a second or third time?

11 Test results

11.1 Questionnaire results

In this section we will give an overview of the results of the questionnaire. While the number of responses is obviously too low to do proper statistical analysis, we will try to interpret what the results (along with our observations) suggest. For each question we provide the average of the responses of the "inexperienced" players as well as the "experienced" players. Both on a scale from 1 to 5, where 1 means "strongly disagree" and 5 means "strongly agree". We have also calculated the standard deviation. When the difference between the two groups is bigger than the standard deviation this might suggest that there is a significant difference (although we can't prove it).

1. I thought it was a fun game.
   Inexperienced players: 4.2 - Experienced players: 4.4 - St.dev: .67
   The players were generally very positive about the game. We presume that this does not necessarily mean that they would want to play it again in their own time, but it seems like they at least thought it was an interesting experience.

2. I thought the game was varied.
   Inexperienced players: 3.6 - Experienced players: 3.4 - St.dev: 1.08
   In hindsight, this question did not make a lot of sense, since the game is not really designed to be varied. The players did not really know what to make of this, and there is a lot of variation in the answers. Some players commented that the game is as varied as you make it yourself.

3. I felt like I had control over what happened in the game.
   Inexperienced players: 3.8 - Experienced players: 4.4 - St.dev: .57
   It looks like there is a significant difference between the two groups of players here. This corresponds with our observation that experienced players were a little quicker to get the hang of the game and that inexperienced players were a little lost at the beginning.

4. I felt like the choices I made were important for the progress of the game.
   Inexperienced players: 4.6 - Experienced players: 4.4 - St.dev: .53
   The high averages here might be indicative of the way people responded to the first connection made between two steps. That generally resulted in players suddenly understanding what the choices they made meant. The fact that the computer "understands" what you are doing might be stimulating for a player.

5. I thought it was important to play well.
   Inexperienced players: 3.8 - Experienced players: 5.0 - St.dev: .70
   Here we can see an interesting difference. The experienced players all found it very important to play well. They obviously felt more pressure and had something to live up to, while the inexperienced players were just happy to help out.

6. I felt encouraged by scoring points.
   Inexperienced players: 4.2 - Experienced players: 3.8 - St.dev: .82
   This differed quite a bit from player to player. Some people did not even notice the score during the game, while others were clearly seeking the limits of what is sensible in their quest for more points.

7. I felt encouraged by the time that was running.
   Inexperienced players: 2.4 - Experienced players: 2.4 - St.dev: .84
   Surpisingly low scores here across the board. Maybe it was not clear how the elapsed time factored into the final score, or maybe the influence on the final score was too insignificant.

8. I thought it was annoying that the time was running.
   Inexperienced players: 1.8 - Experienced players: 3.0 - St.dev: 1.17
   Most people (strongly) disagreed with the statement, aside from three experienced players who agreed with it, which explains the difference between the groups as well the high variation. It seems like some of the experienced players are just too busy "modelling", and rather not have a clock at all. The results here, together with the previous statement, suggest that the time aspects still needs some thought.

9. I was satisfied with the result when I was done.
   Inexperienced players: 4.0 - Experienced players: 3.6 - St.dev: .63
   Some players suggested they needed to compare their results with those of others to answer this question. Aside from that, the slightly lower score of the experienced players could suggest that they hold themselved to a higher standard.

10. The appearance of the game invited me to play.
    Inexperienced players: 3.0 - Experienced players: 3.0 - St.dev: .82
    The response here is mostly neutral. Most players had some nice things to say about the game, but as one player remarked, it is not quite a racing game.

11. I felt like I was playing a real game.
    Inexperienced players: 3.2 - Experienced players: 3.0 - St.dev: 1.10
    The opinions of the players here are all over the place. It seems that some people did not really know what to make of this statement, although it is one of the most crucial. Apparently people do not have a clear idea of what a "real game" even is.

12. I felt like there was a trick to easily score well.
    Inexperienced players: 4.0 - Experienced players: 4.2 - St.dev: .57
    All players eventually recognized that they did not have to enter anything sensible to score points, so the response here is expected.

13. I felt like the score was a fair representation of how well I was doing.
    Inexperienced players: 3.8 - Experienced players: 3.6 - St.dev: .95

There is some variation here, but in general the players tend to agree. This is a bit surprising considering the previous statement. Even though players recognize that there is a trick to easily score high, this trick is not really used.

14. I felt like the second (or third) time I played better.
    Inexperienced players: 5.0 - Experienced players: 4.6 - St.dev: .42
    Players overwhelmingly agree with this statement, and it clearly matches our observations.

15. I understood well what the goal of the game was.
    Inexperienced players: 4.0 - Experienced players: 3.8 - St.dev: .32
    The players seem pretty convinced that they know what the goal is. Although they indeed have a general idea of what to do, the response to the first open question shows that if you actually ask them what the goal is, the answers vary wildly.

16. I found it easy to get started.
    Inexperienced players: 3.2 - Experienced players: 3.4 - St.dev: .48
    There is little variation here. Compared to our observations the players are pretty positive about how easy it is to get started. The average of the inexperienced players is a little lower, which seems right, but it is still surprisingly high.

17. I understood what I could do to reach the goal.
    Inexperienced players: 3.8 - Experienced players: 3.8 - St.dev: .63
    The response is fairly positive here as well. It seems to fit our observations of the performance of the players during the third game, but certainly not their performance during the first game.

18. It was clear to me when I was done.
    Inexperienced players: 3.4 - Experienced players: 3.2 - St.dev: .95
    None of the players immediately noticed the button to finish the game, so this is quite surprising. Only two players (one from both groups) said they disagreed.

19. I understood well how I could score points.
    Inexperienced players: 4.2 - Experienced players: 4.6 - St.dev: .70
    This is also surprisingly high. We did not get the impression that everybody understood the scoring system in detail, but apparently that does not matter too much.

20. During the game it was clear whether I was doing well.
    Inexperienced players: 3.8 - Experienced players: 3.0 - St.dev: .84
    There is some difference between the two groups here, but that can be ascribed to a few outliers. In general it seems that there is some room for improvement here, since it is a very important aspect.

21. The game provided enough context-sensitive help.
    Inexperienced players: 3.8 - Experienced players: 3.2 - St.dev: .71

The difference between the two groups here is surprising, since the experienced players are expected to need less help. In general we noticed that the help that is provided is not read well by most players.

22. I thought the interface/controls worked consistently and clearly.
Inexperienced players: 3.8 - Experienced players: 3.8 - St.dev: .42
There were some obvious problems with the interface, but as the results here suggest, the players were generally positive. They eventually all learned to work with the interface efficiently.

23. I found it easy to fix possible mistakes.
Inexperienced players: 4.2 - Experienced players: 3.6 - St.dev: .74
A bit of difference between the groups here, but it is hard to find an explanation. Maybe the experienced players just expect a higher standard.

24. I felt frustrated or hindered by the interface.
Inexperienced players: 1.8 - Experienced players: 1.6 - St.dev: .48
Very low scores here. All of the players disagreed, which is definitely encouraging.

25. The game helped me to work quickly.
Inexperienced players: 3.6 - Experienced players: 3.6 - St.dev: .52
Most players did not take enough advantage of things like dragging and dropping items, but they were still fairly positive.

To summarize, here are the main areas that allow for improvement, as suggested by the results of the questionnaire:

- The whole aspect of time needs some thought. It does not seem to influence players that much, either positively or negatively.
- Two issues that are far beyond our reach right now are the fact that there is an obvious trick to easily score high, and the fact that some players do not feel like they are playing a *real* game.
- Feedback to the players about how well they are doing can still be improved somewhat.
- Some thought needs to go into how to communicate instructions to players. The instructions at the start, as well as the context sensitive tips, are not read well at all by the players. Maybe more visual presentation (for instance by using examples) might be needed.

The answers to the three open questions tell us something more about how players experienced the game. With the results of the first two questions, we can see if players really understood the game. The third will shed some light on the experience of learning to play.

What was, according to you, the most important goal in the game?

This question is particularly interesting because it really touches on the conflict between game-internal and game-external goals that we've been dealing with. Although the question specifies the goal *in* the game, it tends to get confused with the goal *of* the game.

The answers break down into a few categories:

Focused on points/results Answers include: "scoring points", "making as many steps in as little time as possible", "getting a high score with not many steps" and "reaching the end-product".

Focused on gameplay "describing steps and connecting them", "describing a process in clear steps and ingredients" and "generating associations between ingredients so the application can make connections".

Focused on the player's understanding "Seeing what links between steps in a task are" and "Thinking in steps instead of start- and endproducts".

Focused on ulterior motives "Judging how well people can model".

While the first two categories are the most "correct" in terms of the question, the range of answers reflects the way the game works on different levels.

What did you have to do well to reach this goal?

As we can see from the second and third categories of answers to the previous question, the distinction between this question and the previous one is not clear to everybody. It is interesting that the skills people identify in response to this question range from specific, such as "defining ingredients and linking them" to more general, such as "having a good overview", "abstracting well" and "thinking simplified".

It is remarkable that the players only identify general modelling skills, instead of game-specific skills. It looks like they think that you only have to be a good modeller to do well in the game. The next question provides conflicting results though.

What went better when you played a second or third time?

In this case players do identify game specific skills such as "figuring out how to link steps", "linking more things, resulting in more points" and "working with the interface and understanding the *language*".

It seems that the players do not see things like controlling the interface and understanding the game as core skills that determine how well you perform, but more as obstacles that need to be overcome at first.

11.2 Further observations

Our observations suggest that there is indeed a noticeable if not significant difference between players with and without modelling experience. Despite our intentions to make the game playable for players with little or no expertise in modelling, most of those players found it hard to get started. It took them a while to understand what was meant by a task, a step, an ingredient, and a product. It was somewhat of a surprise to us that people by nature do not seem to even make a sharp distinction between actions and objects, confusing the name of a step with its products and describing sub-steps instead of ingredients.

Generally the game has a hard time forcing people towards the correct way of thinking, if they are not inclined that way already. For example, while the game is supposed to be played by listing steps, some people naturally start describing a task by listing ingredients. They even search for workarounds to do this, like making one step a super-step encompassing the whole task, or describing a first step like "fetch all ingredients".

We had to point players in the right direction on multiple occasions, when it looked like they were really getting stuck, but we did not discourage them from making decisions that were technically correct but "strange" in terms of the game.

Most experienced players fare better, quickly grasping the concept of the game, although sometimes after some initial confusion. Some also started with the infamous "fetch an ingredient" step, but were quick to correct themselves when confronted with the lack of ingredients for that step. They also immediately recognize the importance of the difference between the ingredient "coffee" and the end product "coffee".

One player showed a tendency to list ingredients first, but instead of cramming it into the game somehow, he instead listed them in his head and then switched to creating steps. Not surprisingly, experienced players are also more conscious of the level of detail of their description, and sometimes consciously decide to abstract from something.

When it comes to learning to play the game, the one thing we can clearly see so far is the moment that players really get the hang of it. Experienced players generally start working quickly and efficiently near the end of the first game or at the beginning of the second, and can afford to start focusing more on how to get points (sometimes even searching for the limits of what is sensible). Less experienced players are still struggling during the second game, and only pick up pace during the third. On the up side, given that playing the game does not take all that much time and effort, this could still be considered reasonably rapid learning.

A further interesting observation is that advanced functions of the game are generally not used. Players simply look for the easiest way to succeed in the game. Only one player so far has properly used attributes in his description.

As we said above, some people focus more on steps and some focus more on objects. It was fascinating to see how people approach the task of describing something in different ways. It has to be noted here that three of the five inexperienced players had a background in engineering. This might explain their more technical, systematic point of view. One player went so far that, instead of describing how to make coffee, he started to describe how a coffee-machine works.

It was also funny that some players vastly overestimated the AI of the game. They seemed to think that the game could actually understand what they meant, and tried to formulate clear, full sentences because of it.

One last observation is that players do not seem to read the instructions very well. The instructions at the start of the game are closely read, but the players seem to forget them immediately when the game starts. For example, the fact that the game will find the connections between steps by itself seems lost on a lot of people. Many ask how they are supposed to connect the steps and start clicking around. However, this is all understandable because at the time they read the instructions, the players do not have a clear idea of what the game actually is.

More remarkable is the fact that the context-sensitive instructions that pop up during the game are sometimes completely ignored, even when they are obstructing the workspace. Maybe this has something to do with the many unnecessary pop-ups in certain operating systems that cause people to just tune them out.

11.3 Specific issues concerning interface and gameplay

Interface related issues:

- Players do not notice the "finish" button.
- Players cannot figure out how to expand or close the list of ingredients/products.
- Clicking the task name does not select the whole text.
- When entering long names in some text boxes, it is hard to change the text after it has been typed.
- When the screen is cluttered, a new step that is created can get lost.
- Step name cannot be entered when the mouse is not on that step.

Gameplay related issues:

- Providing a task name does not give you points. There is little motivation.
- It is not clear what linking really means, but since links score points, players tend to link everything.

12 Suggested improvements

In this final section, we summarize what went well, what went not so well, and what some suggested improvements are.

First of all, it is encouraging that players generally do not take very long to learn how to play the game and how to produce a reasonable task description. It seems that the main strength of the game is that it provides players with tangible feedback based on what they are doing, giving them at least a general sense of direction. This is an improvement over having to explain quality of modelling in abstract terms to people with little proces modelling experience.

Aside from a few specific problems, the players seem to be satisfied with the current interface. By keeping things simple it manages to provide a good overview of the whole process within one screen, and the feedback it provides by using graphics and sound seems to work well.

On the other hand, we can't say that the current score system properly represents the various quality aspects of modelling. Players recognized that they did not really have to provide sensible input to get a good score. There is certainly a lot of room for improvement here, although quantifying quality will never be easy.

It was also somewhat disappointing that even though the scope of the game was already limited, most players did not use the advanced functionality such as attributes. Players may be more encouraged to do this by providing examples of more complicated descriptions, but we avoided the use of examples so far. Instead we used a wizard-like system to get players started, which is easy to understand, but seems limited when things get more complicated.

A more specific problem we encountered is the use of links between ingredients and products within a step. When attributes are not used, the purpose of these links is unclear to players and they consistently tend to forget about them. In fact, there aren't any clear rules about when links are required, so the confusion is entirely understandable. This is something that still requires some thought.

13 Conclusion and future prospects

Whether modelling can ever be as fun as an actual game remains to be seen, but we can certainly learn a lot from how games are designed. A good game is more than just a goal to work towards. It lets you know what tools are at your disposal, it lets you explore what effect your actions have on the world and it gives you a sense of how well you are doing.

Similarly a good modelling process is more than just a model to work towards. With our prototype we have attempted to give meaning to the modelling process aside from the end result. Our experiences are generally positive, but considering the complexity of process modelling in practice, the scope of this thesis was of course limited.

We focused mainly on one aspect of process modelling. That is, the argumentation behind ordering steps in series or in parallel, resulting in the elicitation of AND-joins.

In section 2 we proposed two questions to evaluate the prototype we were to create. They were:

- Are players able to get through the game with little to no outside help from an expert?
- Does playing the game result in information that can be used to create a process model?

The test results have shown that the we can give a positive answer to the first question, even though the amount of outside help required by the average user could still be reduced. By adding XPDL compatibility we have also demonstrated that we can give a positive answer to the second question, although we are only working with a limited set of concepts at this point. These answers should help us answer our main question:

- Does viewing method engineering as game design help to design a tool/game that allows non-experts to make a formal description of a process?

We have created a tool/game inspired by game design and the sub-questions show (with some reservations) that non-experts can indeed make a formal description. It is safe to say that viewing method engineering as game design helps. This means we can also give a positive answer to our main question.

In section 2 we also stated that a tool like this might in the future improve the efficiency of the modelling process. This was not our goal for this prototype, and we have not tested if this was the case. To test it, one would need to compare the game to a regular tool, and it is hard to say what the results would be. However, if we include learnability in efficiency, a game like this might be promising.

Another long term goal was to make the modelling process more fun for the domain-expert. While we do not claim to provide first class entertainment, reactions from players were generally pretty positive. Again, this is almost impossible to objectively measure, so we will just have to wait and see what is possible in the future.

Finally, the somewhat overarching goal was to make modelling more accesible to non-experts. We cannot say that we achieved true disintermediation, but a game like this does give those non-experts a chance to become acquainted with process modelling in a playful way. It also spurred a lot of discussion about the way people think about processes, which everyone seemed to be open to.

We have said a lot of positives things about the current prototype, and it is certainly interesting to further develop this, or a similar, game. As we said our focus was quite narrow, so there is still a lot of ground to cover within process modelling alone.

Of course it was not all positive. There is still a lot of room for improvement. This project opened up a whole range of new challenges regarding the combination of game design and process modelling. Just as an example, we need to understand more thoroughly the different aspects of quality of modelling: both how they can be quantified and how they relate to (or conflict with) the way people conceive processes.

Whether this was just an interesting bit of research into the relation between process modelling and game design, or one of the first steps into a new field of study that uses games as elicitation tools, we do not know yet. It is hard to imagine exactly what the future holds. What we can say is that it has been an interesting and somewhat succesful experience so far.

A Notes on the design process

A.1 First version

The first version of the game was played by one player and one facilitator using only pen and paper. It introduced the basic concept of *step cards* and *item cards*. The rules of this version were:

1. Make a list of items.
2. Name and describe the steps. Is it a removal step?
3. Identify the items that belong to each step (tools, ingredients and products). Are the products end-products?
4. Perform a step try.
5. The facilitator helps to identify items and properties.
6. Make an item card for each item. Describe what changes and what stays the same.
7. Describe the order of appearance of properties.
8. Perform an item try.

A removal step is a step which has a product that is not used again in the rest of the process. A "try" means that the player submits what he has written down to the facilitator, who can then calculate a score. The step try and the item try indicate the ends of two phases in the game. A step try can be done multiple times, but you cannot return to the first phase of the game after you enter the second.

Problems with this version

- The distinction between the two phases (or tries) was not clear.
- The concept of removal does not work. When something is split up or decomposed, is it removed? It can reappear later.
- Providing an order of properties is hard, because certain properties can appear multiple times.
- The distinction between tools and ingredients is not clear either. Something can be a tool in one step and an ingredient in the other.

Solution

A solution may be to make no distinction between items and properties. There is no such thing as a removal step anymore, because items never "change" anymore anyway. They are always "removed".

We do not identify tools on a process scale anymore. But we can identify a tool within a step easily by comparing input and output. When something appears in both input and output it is a tool.

A.2 Second version

The main difference in this version that it focuses on input and output of steps instead of global items and how they change. We have not done away with the concept of properties completely, as we suggested above. We think that it can still work after we redesign the rules.

1. Make a list of steps in the process.
2. Describe the resulting items of each step.
3. Are these end-products (i.e. they are not used later on)?
4. Describe which items are needed to get this result.
5. Describe what happens which each of those input items (stay the same / change / disappear).
   (This is what used to be a step try.)
6. If an item changes, describe what changes (these are the properties). One property per step per item.
7. Match properties to items on different step cards. What other properties are relevant to which item?

The idea is that the order of the steps can be deducted by comparing the input and output of different steps.

An item can take on three roles in a step:

- Staying the same: it is a tool.
- Disappearing: it is involved in composition or decomposition.
- Changing: it has properties.

In the first phase you can see steps as entirely separate.

If you look at the output of a step you will see that all items that were not tools in the step and are not used as input in another step are end-products or waste-products (there is no fundamental difference).

Only the final step can produce only end-products. If there are more steps that do this some steps must be irrelevant.

Problems with this version

- Describing input and output on one line does not work. Multiple input items can lead to one output item and vice versa.
- The distinction between staying the same, disappearing and changing does not really work. It especially does not feel natural when discussing information. When it comes to information an item can become something but still remain.

Solution

- Instead of writing corresponding input and output items on one line, we should draw marked arrows between input and output.
- Instead of staying the same, disappearing or changing, we should separate how an item transitions and whether or not it persists.
- Transitions are now: changing into, becoming part of, and falling apart into.

It will be interesting to see certain patterns of transitions within a step emerging. Are certain patterns typical for certain domains?

A.3 Third version

While developing the third version it turned out that, despite what we suggested above, arrows between input and output do not actually have to be marked at all. Also, it seemed sensible to reintroduce item cards.

1. Make a list of steps in the process.
2. Describe the resulting items of each step.
3. Are these end-products (i.e. they are not used later on)?
4. Describe which items are needed to get this result.
5. Draw arrows from from input items to output items.
6. For each input item write down if it persists after this step.
7. If an item changes, describe what changes (these are the properties). Also make an itemcard for this item.
8. Match properties to items by looking at the item cards. What other properties are relevant to which item?

An item *changes* when it does not persist and there is a one on one relation with an output item.

Problems with this version

It seemed like the game was at least logically consistent at this point, so it was time to check if it was actually playable. We provided a player with all the necessary forms and a description of the rules (similar to normal boardgames). We first asked the player if he understood the rules, and he said he did.

However, actually playing the game turned out to be much harder than expected. It was hard for the player to follow the strict guidelines. The case was "baking an apple pie", and the player thought it did not make sense to start by describing the steps, but instead wanted to list the ingredients first. The game did not allow for this, but he creatively solved it by introducing a step called "collecting ingredients".

It was hard for the facilitator as well, because we could not look on while the player was filling in forms, moving towards what we already saw was a dead end. Making mistakes is frustrating and time-consuming after all.

Playing the game quickly dissolved into a meta-discussion about the game. Interesting, but not what we intented.

Solution

Having to follow strict rules, making mistakes and staying focused on the game were all frustrating (not to mention keeping the score). All these problems pointed to one thing: the game had to be automated.

The theory was that people are more accepting of strict rules when working with a computer. They are willing to dumb down their description to the necessary level for a computer to understand. Exactly what we need.

It is also obviously easier to edit mistakes on a computer.

In theory the game does not even have to "work". Just the interface alone would solve a lot of problems. It is possible to take a Wizard-of-Oz like approach, where a facilitator tries to keep the score.

A proposed platform for a first implementation was Excel (or equivalent program), because the game is very much form-based, and does not require fancy graphics. It also means we do not need a database.

A.4 Fourth version

We decided to build the first automated version in OpenOffice.org Calc.

Each step card is its own separate tab in the spreadsheet, and each has its own score counter. Apart from ten step card tabs there is one tab that includes the step list and the item list, and keeps the total score.

We decided to put the step list and item list on one tab, so players are free to choose which one to fill in first.

Because it is not possible to draw arrows on a spreadsheet. The player will have to list the numbers of the matching products for each ingredient.

This version of the game included the following features:

- When you enter a step name in the step list, it is copied to the respective step card. When a step name is filled in the player gets 100 points.
- The game checks the length of the step name. If it is more than four words, the player loses 10 points.
- When the player lists the ingredients and products, the game checks if they are also present in the step description. The player gets 10 points for each item present in the description.
- The game recognizes items that "change" when ingredients are not marked as persisting beyond the current step and have a 1 on 1 relation with a product. Changing items are required to be listed again at the bottom of the step card. Each item that changes and is named in the "changing-items-list" gets the player 10 points. Items in the "changing-items-list" that do not actually change cost the player 10 points. This means that the check goes both ways.
- Any item that changes should also be present on the item list. If this is not the case, the player loses 10 points. Note that the player is free to add any extra items to the item list.
- For every item on the item list, the game searches the step cards for related properties and lists them all. The search is somewhat fuzzy to catch similar item names.
- There is a rule that there can only be one step with only end products. The game checks for each step how many non-end-products there are (this number should be larger than zero for most steps). In the main tab these numbers are then collected and when there is more than 1 "end step", the player loses 100 points.

Testing

To test this version of the game we need to formalize the tests somewhat. We are not worried about how it works as a game at this point, but we are interested in how it works as a system. We will therefore focus on usability. Typical factors of usability are:

- Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?
- Efficiency: Once users have learned the design, how quickly can they perform tasks?
- Memorability: When users return to the design after a period of not using it, how easily can they re-establish proficiency?
- Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?

- Satisfaction: How pleasant is it to use the design?

Efficiency and memorability are hard to judge when players only use the system once, so we will focus on the other three aspects for now. They all seem very important.

Learnability is a key aspect because we are trying to let non-experts perform a complicated task without any prior experience. Satisfaction is something that seems especially important since we are talking about a game, even if we are not focusing on "fun" at this point. The amount of errors players make influences the former two points directly.

We will not use quantitative testing using metrics yet. Instead we will just take a close look at what kind of errors users make. We will try to categorize them according to Nielsen's 10 heuristics [14], and see exactly in which ways the game falls short.

The setup

We have put the players in front of the game, running on a computer. The players have never seen the game before. We made notes while the players played the game.

We gave the players the following introduction:

> The goal of the game is to describe a task or process as clearly and as completely as possible. Try to make it clear enough that a computer can understand it. Depending on how well you perform you will be given a score.
>
> Start by naming the task you are going to describe. Next list all the different steps. Use a maximum of ten steps to avoid going into too much detail. If you want you can also name some of the items involved in the task.
>
> Next describe all the steps in detail. Each step has its own tab. Name the products and the ingredients needed to produce this products.
>
> If an item changes during a step, decribe what changes. Also make sure this item is present on the item list on the task tab.
>
> Try to figure out how it works as much as possible, but if you don't understand something, feel frustrated or confused, or if you find yourself making a mistake, please try to "think aloud". Explain what you think goes wrong.

Thinking aloud is a well known concept in usability, and it worked fairly well in this case.

We will now list the current flaws according to Nielsen's 10 heuristics:

Visibility of system status It needs to be even more clear what gets you points and what does not. Player's attention needs to be directed towards the score. Once they get the hang of how the scoring works things seem to improve. After finishing their first draft, we did compare the player's score with the previous player, and this really worked as an incentive to improve the score. Most of the "tips" the game provides are noticed by the players and the information in them is seen a valuable.

Match between system and the real world This is the hardest part at this point. Things like persistence of ingredients and the connection between ingredients and products are pretty confusing for the players. There is not really an elegant way to explain it in way that makes it click immediately. Maybe use examples? It is also hard for the players that sometimes their natural way to explain a step does not quite fit the framework of the game. Frankly the game sometimes requires somewhat unnatural formalization.

User control and freedom The game gives a certain degree of freedom. Errors can be quickly corrected, and when steps are forgotten the players appreciate that steps do not have to be in the correct order, so a new step can just be added at the bottom of the list. On the other hand there is too much freedom. The game does not very much force the player to enter a *correct* process at this point. Players remarked that it is easy to get points by just entering nonsense.

Consistency and standards Not really applicable.

Error prevention Since this is a game we do not really want the players to play flawlessly, so this is not really applicable.

Recognition rather than recall The game tries to organize information for the player as much as possible, so this was not really a problem. It would make sense to provide more extensive written instructions though, so they are easily retrievable by the player.

Flexibility and efficiency of use Efficiency of use is certainly the main goal of automating the game in the first place. Things like automatically keeping track of item/properties seemed to be appreciated by the players. On the other hand the step description was seen as rather unnecessary and time-consuming by the players.

Aesthetic and minimalist design It is more likely that there is too little information at this point.

Help users recognize, diagnose, and recover from errors There are not really any error messages in the game, but it is certainly interesting too see how the feedback from the score can influence players to fix errors. This is not really the case at this point and it is one of the main goals for the next version.

Help and documentation There is a lot of room for improvement here as well. It was interesting to see how players responded to the game with little introduction, but it soon became clear that more explanation is necessary. As we said, things like persistence are hard to explain in one line of text in a tooltip.

Other lessons:

- Some players find it hard to think of a product of a step. We suggested that they look at the step name for inspiration. This often works really well.

- Criteria for which items change still are not correct. Changes need to be made.
- Correctness needs to be forced more!
- Usability is all about subtleties. How visible one tip is can change the whole thing. A lot of finetuning will have to be done.

Solution

- Change the column "stays" into "reusable". It functions the same way but the description is much more sensible.
- Defintion of which items change made less strict. As long as they are not reusable any item can change.
- Not every item on the itemlist gets you points anymore. Only items with properties.

A.5 Fifth version

While the last version was theoretically sound, the limits of the implementation in OpenOffice.org Calc were becoming more and more clear. The experience of playing a game relies heavily on fluid interaction, animation and sound. That is why we decided to make a radical change, and rebuild the game from the ground up in Adobe AIR / Actionscipt 3.

One of the most striking differences in this version is that the steps are no longer represented by different screens, but are all elements floating on the same canvas, so it is possible to see how they are connected. This also allows the amount of typing to be minimized, because the player can easily drag and drop items.

Although the experience in this version of the game is much richer, some functionality from the previous versions was actually dropped. Players now do not have to mark items a "reusable" or as end products anymore. This is still interesting information, but simplicity is important as well. When you are not working with a spreadsheet and want to have all steps on a single screen, things can get cluttered easily.

A discription of the game experience in this version

At the beginning of the game the interface provides the player with only one button, allowing the player to create a new step. Other than that the player can give the task a name.

When a new step is created it is just a rectangle that the player can drag and drop around, so the next thing to do is to give the step a name. A guideline for this is to describe the step in no more than four words. There is no hard limit though.

The player then has to add ingredients and products to the step. Each ingredient or product that is added to a step also shows up in a list in the top left of the screen. Items from this list can then be dragged to a different step to be reused.

Ingredients and products of a step can be connected by clicking on a little bubble next to an ingredient and clicking again on a bubble next to a product. This creates a link. A link can be used to provide extra information about what happens to an ingredient during that step. When an "ingredient" is really just a tool that does not change itself, no other information is required.

However, when an ingredient that does change gets linked, the link should describe what happens to the ingredient like this: "this ingredient is being ...". The grammatical trick is that the player provides a verb that can also be used as an adjective, and therefore as an attribute of the ingredient.

This is useful, because when the ingredient is again used as an ingredient in another step, the game recognizes that this ingredient has attributes, and the player only has to select which of these attributes the ingredient has to have as a precondition for this step.

When the player has described multiple steps this way, the game can derive the connections between the steps. Each step has a set of input items (ingredients with possible preconditions) and a set of output items (both products and ingredients with added attributes). When an input item of one step matches with an output item of another step, the two are connected.

Note that there can be some redundancy in the output sets. For example, we can describe the step "mashing potatoes", with "potatoes" as an ingredient, "mashed potatoes" as a product and "mashed" as an attribute of "potatoes", resulting in an output set of "mashed potatoes" and "mashed+potatoes".

The game can automatically put connected steps in the right order by moving them when they are not being used by the player.

The score is calculated as follows:

- 100 points for each step.
- 100 points for each connection between two steps.
- 10 points for each ingredient, product or link.

The final score is the sum of those minus one point for every second played.

The player can only finish the game when all the steps he created are connected to at least one other step.

At the opening screen of the game, the player can choose to get extra help. This help consists of pop-ups that guide the player through the early stages of the game. Other than that, hints are also shown on the bottom of the screen when the player moves his mouse over any object.

The player can also choose whether or not connected steps are automatically placed in the correct order.

Problems

- Typing names that are longer than textboxes is not supported in some cases.

- Steps are being connected too quickly sometimes, even while the user is typing. This is confusing.
- The time element in the final score is way too harsh. You should not be able to get negative scores.
- Because of the redundancy in the output sets, the item list gets messy very quickly.

A.6 Sixth version

To keep the output sets and the item list as clean as possible, products are now only relevant (visible) when they are linked to an ingredient with an *empty* link, and they are not the same as the ingredient. When the link is not empty, the resulting ingredient/attribute combination overrides the product.

Some extra functionality is also added in this version. The user is now able to express that two ingredients are combined into one product, by typing into the link of ingredient A: "with *ingredient B*". You can also drag a listed item to a link, or you can connect two links by dragging one over the other. These are all ways to "join" two links together.

The way the final score is calculated in this version is changed as well. The final score is now half of the base score, or the base score minus a half point for each second. Whichever is highest.

References

[1] Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. In *Advances in Database Technology EDBT'98*. Springer Berlin / Heidelberg, 1998.

[2] Workflow Management Coalition. Process definition interface - xml process definition language. http://www.wfmc.org/xpdl.html, 2008.

[3] Heather Desurvire, Martin Caplan, and Jozsef A. Toth. Using heuristics to evaluate the playability of games. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1509–1512, New York, NY, USA, 2004. ACM.

[4] J. Faget, M. Marin, P. Mégard, V. Owens, and L. Tarin. Business processes and business rules: Business agility becomes real. In *Workflow Handbook 2003*, pages 77–92. Future Strategies Inc., 2003.

[5] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.

[6] Bart-Jan Hommes and Victor Van Reijswoud. Assessing the quality of business process modeling techniques. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 1*, page 1007, Washington, DC, USA, 2000. IEEE Computer Society.

[7] S.J.B.A. Hoppenbrouwers. Community-based ict development as a multi-player game. In *What is an Organization? Materiality, Agency and Discourse*. University of Montreal, 2008.

[8] S.J.B.A. Hoppenbrouwers, H.A. Proper, and Th.P. van der Weide. A fundamental view on the process of conceptual modeling. In *Conceptual Modeling - ER 2005 - 24th International Conference on Conceptual Modeling*. Klagenfurt. Springer: Lecture Notes in Computer Science, 2005.

[9] S.J.B.A. Hoppenbrouwers, P. van Bommel, and A. Järvinen. Method engineering as game design: an emerging hci perspective on methods and case tools. In *Exploring Modelling Methods for Systems Analysis and Design (EMMSAD'08), held in conjunction with CAiSE'08*, 2008.

[10] Dietmar Jannach and Gerold Kreutler. Personalized user preference elicitation for e-services. In *EEE '05: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service*, pages 604–611, Washington, DC, USA, 2005. IEEE Computer Society.

[11] Aki Järvinen. *Games without Frontiers: Theories and Methods for Game Studies and Design*. PhD thesis, University of Tampere, Finland, 2007.

[12] John Krogstie, Guttorm Sindre, and Havard Jorgensen. Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, (15):91–102, 2006.

[13] Isabelle Mirbel and Jolita Ralyté;. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requir. Eng.*, 11(1):58–78, 2005.

[14] Jakob Nielsen. How to conduct a heuristic evaluation. online, 2001. http://staff.unak.is/not/nicolaw/courses/hci/HCILab7papers.pdf.

[15] J.H.L. Oei, L.J.G.T. van Hemmen, E.D. Falkenberg, and S. Brinkkemper. The meta-model hierarchy, a framework for information systems concepts and techniques. Technical Report 92-17, Dept. of Information Systems, University of Nijmegen, 1992.

[16] Roger S. Pressman. *Software Engineering: A Practicioner's Approach*. McGraw Hill, 6th edition, 2005.

[17] Peter Rittgen. Negotiating models. In John Krogstie, Andreas L. Opdahl, and Guttorm Sindre, editors, *Advanced Information Systems Engineering, 19th International Conference, CAiSE 2007, Trondheim, Norway, June 11-15, 2007, Proceedings*, pages 561–573, 2007.

[18] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press, 2003.

[19] Adobe Systems. Actionscript 3 language specification. online, 2006. http://livedocs.adobe.com/specs/actionscript/3/wwhelp/wwhimpl/js/html/wwhelp.htm.

[20] P. van Bommel, S.J.B.A. Hoppenbrouwers, H.A. Proper, and J. Roelofs. Concepts and strategies for quality of modelling. In *Innovations in Information Systems Modelling: Methods and Best Practices*. IGI, New York, 2008.

[21] Wil van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[22] I. Vanderfeesten, J. Cardoso, J. Mendling, H. A. Reijers, and W. van der Aalst. Quality metrics for business process models. In L. Fischer, editor, *BPM and Workflow Handbook. Future Strategies and Workflow Management Coalition*. 2007.

[23] Stephen A. White. Introduction to bpmn. online, 2009. http://www.bpmn.org/Documents/Introduction to BPMN.pdf.