An argument based approach for easier process modeling using dialog games
A procedure for BPMN process modeling

Author:
Rob Thijssen

Supervisor:
Dr. Stijn Hoppenbrouwers

June 24, 2013
Abstract

This thesis contains validating research regarding the combination of dialog games and argument based modeling. The goal is to combine and expand these conceptual modeling approaches in such a way that they form a complete procedure for creating conceptual models from expert knowledge.

A procedure for this is drafted. It contains three phases: input (elicitation), process (restructuring), output (writing to BPMN). The procedure is aimed at facilitators that already have some experience with gathering knowledge from experts. The procedure is implemented and tested with a prototype. Most of the weight of the research was put into the elicitation phase, where elements from dialog games and argument based modeling have to be combined.

After seven experimental sessions the procedure and prototype were good enough to validate the main question. When given some more attention dialog gaming with the argument based approach shows promise for practical applications.

A glossary containing specific language and abbreviations can be found at the end of this paper.
1 A word of thanks

This thesis is the crown on my career as a student. Why do I say career? I’ve been a student from my fourth to my twenty-sixth. Elementary school. High school. Eindhoven Technical University. And finally Radboud University Nijmegen. Not to mention all the various events, part time jobs and attempts to broaden my view and knowledge. Finishing this thesis and with that my education is a natural thing to do. But looking back I can see a long way I’ve taken to get that far.

This effort was not only made from my part though. Many people have helped me to get to this moment in time. I wish to thank them for that by these few words.

My parents and family have always been loving and supportive. No matter what I chose to do. What more can one wish?

My girlfriend for suffering through hours of rattling keyboards, experiments with varying levels of frustration, and a preoccupied boyfriend.

My friends and colleagues have helped me by keeping me company, be it in professional or personal matters. With discussing ideas. And finally some of them helped directly by taking part in my experimental sessions.

My teachers have learned me the necessary skills to become an educated and professional person, able to do research completely on his own.

Dr. Stijn Hoppenbrouwers has helped me tremendously. Not only did he give shape to my initial ideas for research during my bachelor thesis, he has (mostly) subtly guided me along the way of my master thesis as well. He is a professional, he knows how to motivate a person, and it has always been a pleasure working with him.

Everyone who has helped me to get where I am know: Thank you very much. It has not be for nothing. Three months before I am officially graduated I have already signed for a job at a promising little company as Business Intelligence specialist.
2 Background and significance

We now live the digital age. Everything revolves around connectivity, communication and streamlining. Behind this digital revolution lie uncountable pieces of digital machinery. Smart phones, tablets, laptops, servers, set top boxes, you name it. Almost everything we do is supported by some kind of computer and software. And in the time of a few months those fancy tools are all already outdated.

This enormously fast evolving organized chaos is happening everywhere. It puts a heavy and changing load on people who need to figure out how to use these digital possibilities to their full extent. If most normal people are having difficulty coping with staying up to date, imagine how the managers of organizations employing of over a thousand employees must feel. They need a clear picture of what is going on in their organization and how technology can support this. This picture will then enable them to better match their digital needs to their organizational needs.

Now let’s call the organizational needs the territory of business administration experts. Why? Well, business administration is a large field of expertise. It specifically focused on keeping an organization running towards its goal. This goal is often making money. Or at the very least “running a viable organization”. Let’s call the digital needs the responsibility of the IT department. Why? Because they always have been in charge there, and it makes sense to keep the digital experts in charge of the digital needs. Information and computer science is a very different field, with different fundamental approaches aimed at different problems.

A large organization without IT support will not be able to function in the current environment. That’s a business problem! But IT is nowhere without a goal. Without business there is little need for information technology. And that’s an IT problem. So both parties will have to work together in symbioses to facilitate a well running organization.

Not only that: the business administration and information technology have to work together more closely every passing day. In modern organizations it is very difficult to tell where one ends and the other begins. For organizations information technology offers both support in their traditional business cases and in new areas of business. Without the right information technology innovations an organization or business might not be able to compete.

No wonder then that the demand for information technology systems has grown enormously over the past few years (Freeman, Jarvenpaa, & Wheeler, 2000). The growth and usage of information technology outpaces the innovations in designing and building of these information systems. The result is many information technology projects failing. This is becoming an increasingly large problem. In some cases projects might be over budget or delayed. In others, the project is canceled before it completed. This is the case for as much as 50% of the projects. Clearly improvement in the project success rate would be a good thing, reducing cost and effort.

There are many different ways to increase the success rate of information technology projects. An obvious example: the software building process could be streamlined. That is the computer science side of the problem. Increasing software programming speed, efficiency and quality.

In this paper the background will be the information science and business administration side of the matter. This means that the emphasis will be on the information gathering analyses and design part of the process. While some might dispute this, I am convinced that the design phase of an IT project is at least as important as the creation phase. The design team of a project has to take countless variables into account. And it has to document the acceptable limits in which they can operate for the customer to benefit from the product. A few of these variables
might be the required functionalities, demands regarding stability and other non-functionals, cost, human measure, company culture, target consumer group, etc. This list would go on for pages if it were to be made exhaustive.

One way to tackle design problems is to facilitate the designing process by improving the communication between the parties involved. This might seem trivial. By providing clear, compact and unambiguous communication from “business client” to “IT-contractor” the design process is sped up. And the specification will be almost watertight. That would eliminate most project failure causes. Usually this communication consists of design documents in which different types of models and text describe what is going in the business side of things, and how information technology should support that. These models are made by, among others, information scientists. Although these documents are doing a better job than just improvising interviews and lists, it takes a lot of attention to detail and a lot of effort to correctly draft them. So there is still a lot of room for improvement.

It is also worth mentioning that IT and business are not necessarily at the same physical location. Outsourcing is one of the reasons for this. That means that the communication problems described above are aggravated. Most software design tools do not seem to be made for long distance collaboration. And they’re not used by the same “teams” as communication is inter-organization in stead of inter-department.

The problem described here is 25+ years old and widely known as the knowledge acquisition bottleneck (Hayes-Roth, Waterman, 1983).

When the generation of the information system design documents and models can be (partially) automated, both the business client and the information scientists have to put in less effort to get the same degree of design quality. And with equal effort, the quality will improve. Now we’re talking. The goal of this research is exactly that: to create an end-to-end process for quick, easy and reliable generation of process models using communication software.
3 Problem statement

The goal of an information technology project is usually to create an information system that supports the organizational processes. In doing so it is not uncommon to change the organizational workflow a little too so that it makes optimal use of the information technology. As organizations and businesses expand they have an ever growing demand for information technology support. An other reason for this growing demand is the fact organizational processes are becoming ever more complex. And because information technology enables them to grow more complex, newer generations of technology support are needed to keep that growth going. This means that the amount of information technology related projects is increasing steadily.

Designing information technology projects requires a lot of effort and resources from everyone involved. One of the most important resources is knowledge. Knowledge about the organization and its processes is needed to create supporting software. I would say that each software project is a custom fit. Like a custom-made suit, making such a custom project requires taking a lot of measurements. Gathering the needed knowledge for a IT-project is a difficult challenge.

Other important resources are familiar ones: time and money. Because the overhead costs of information technology are quite high, as is the failure rate, creating a new information system is generally considered expensive. Rather often unexpected problems cause the projects to be delayed. And that in turn again increases cost.

In order to solve those problems design experts, often called facilitators, set up meetings with key figures for a IT-project (Hayne, 1999). These meetings serve to accumulate knowledge about the organizational process. The analyst, often the same person as the facilitator, then uses all the knowledge that he was able to find to make a formalized description of the situation. This description is then used for (or adapted to) a design for the IT-project in question.

This classical example however yields new problems: this design process is a “problem without clear boundaries” (Hayne, 1999). There is no one right way to do that design. And every situation calls for a different solution to this design process. To worsen things, the parties involved are usually IT specialists and business specialists who both think in different terms. That means that after gathering knowledge it also has to be shared between different contexts.

This leads me to believe that it would be nice to have a guideline to address these new design problems. To have some support during the project design process. A piece of text or software that helps in structuring all the information that has to be exchanged in the software design process. As design for a large part consists of setting requirements, using information from a source, this should be the key part of the method. This means that the input from the client has to be processed, converted into usable information for the technical experts, and possibly be fed back to the client for verification.

This master’s degree research continues where my bachelor thesis left off. The way to semi-automatic reasoning was paved there, and now it can be used to solve the design problems. In this thesis I will develop, create and test an begin-to-end method for argument based modeling using dialog games.

The research question posed is:

Is it possible to make a viable modeling procedure by combining argument based modeling with dialog games?
In which we might be able to identify the following key points stated by (Schotten, 2009).

1. Ensuring the quality of the modeling process
2. Improving the efficiency of the modeling process
3. Making the modeling process more fun for the domain-expert
4 Possible solutions to the problem

Information, as a carrier of knowledge, has to come from somewhere. It has to be gathered, handled, structured, transported and shared to persons with different backgrounds. This is no trivial task. The problem surrounding getting and structuring the information quickly enough to be useful is described as the knowledge acquisition bottleneck. Transporting and sharing the information effectively is another important challenge.

Let’s focus on the first problem. How does one get knowledge? Well, in the context of an organization one could read all the paperwork that is available. Analyzing the organization from top to bottom is a method too. But both sources will only get you that far in getting the complete picture of the organizational process. In the best case they will give you a good context to work against, combined with a picture of how things should work.

No, the most important asset in organizations is the knowledge and skill of its members (or if you will, employees). These people, lets call them domain experts now, are a valuable source of information. They will be able to tell you how organizational processes are (in practice and not on paper!), and how they could or should be to make them work better. Working with different people introduces a new set of challenges though. Some are listed in an article that tries to summarize all types of communication and their problems (Craig, 2007). There are many, but the most important one is that their knowledge is raw and mixed with personal views and preferences (Berger & Calabrese, 1975). That’s where facilitators come in. Facilitators are people trained in knowledge gathering theories. They use their skills to set up meetings with one or more domain experts to gather and structure information, while facilitating the process of communication.

The second problem is transferring knowledge. When knowledge about an organizational process is collected it is still in a form that does not mean much for a computer programmer. There is a gap between the contexts and types of information between both worlds. Closing this gap is usually the task of the facilitator or a information scientist. They structure, rewrite and translate knowledge to other forms, with the goal of using it as input for a software project. Or, perhaps in improvement programs.

Requirement documents are one type of restructured information. They are full of (formal-ish) structured text and models, that make sense to software engineering teams. This translation from business to IT context is crucial. A good design based on that knowledge provides the basis for a good result. Many knowledge translation theories exist, but I find conceptual modeling most powerful in transferring knowledge between domains. Conceptual modeling languages follow rules and grammar that are better than those of natural language.

The theories and paradigms used to tackle the problems of acquiring and transferring the necessary design knowledge in this thesis are called dialog games and argument based modeling. A facilitator can use both theories to successfully gather and transfer knowledge. By using dialog game theory, a form of communication (or a meeting) between him and the domain expert is possible. This theory enables the facilitator to guide the conversation. He can do part of the information structuring and conversation control on the fly. The game component may give the domain expert a challenge, better involving him in the process of his design. Argument based thinking gives the facilitator a tool to give relevance to the information he or she is getting from the domain expert. Also, these arguments give an extra meaning to the information when restructuring and translating needs to take place.

As we’re looking for a method with corresponding tool to help facilitators in doing their job,
it is worth looking into the possibilities of combining both theories into a complete method for creating models. Together they may help to solve the knowledge acquisition and knowledge transfer gap. Both theories have existed for a few years. But to my knowledge they have not been combined yet. Most certainly there are no beginning-to-end tools available for facilitators and information scientists who want to model their information.

There are other theories around that might have the same goal, yet very few of them describe the process of conceptual modeling from beginning to end. Why then select argument based modeling and dialog games as a tool for this goal? Argument based modeling theories and examples are now available as a result of my bachelor’s thesis. It seems only natural to continue exploring possible uses for this. Dialog game design is an interesting new approach for meeting facilitation. They offer many possibilities, but much exploring on the subject still needs to be done. I would like to contribute to that.
5 Work plan

Again: in short, the goal is to create a beginning-to-end procedure for creating a type of conceptual model from domain expert information. Dialog Games (DG) and Argument based modeling (ABM) are selected as theories having a lot of potential in this method. Naturally these two theories alone are by far too little to go on to create a complete method. So more work has to happen. This chapter will list and describe the steps that will be taken. These goals are identified with the help of (Hevner, March, Park, & Ram, 2004).

1. Analyses of the exact goal(s) of the new procedure is needed
2. More relevant theories have to be found to close the gaps between the aforementioned main theories. These consist of:
   (a) Design theories
   (b) Communication theories
   (c) Modeling practices and theories
   (d) Facilitation practices and theories (of which the theories above can be subsets, see 7)
3. Selecting a modeling language that fits the goals
4. Defining a beginning to end procedure using ABM and DG.
5. Building a software prototype supporting that method
6. The prototype and method have to be tested
7. Test results will be analyzed and published
Figure 1: All theories in this thesis will be interlinked and sometimes overlapping
6 Goal analysis

According to (Schotten, 2009) we can identify three main goals of the new elicitation and modeling procedure.

1. Ensuring the quality of the modeling process
2. Improving the efficiency of the modeling process
3. Making the modeling process more fun for the domain-expert

Conceptual modeling needs domain knowledge as input. This knowledge comes from sources in different guises. The prime information source is (usually a group of) stakeholders in an information technology project. The stakeholders most often have an excellent business administration view of their wishes but are less aware of the IT implications. They have requirements for the IT-side coming from the responsibilities they have in the organizational context.

That stakeholder has knowledge about the goals, environment and context of the intended IT-project. This knowledge is crucial for a useful design and therefore has to be extracted and shared. Knowledge (information) can be shared or extracted by way of collaborative design meetings and interviews.

To avoid discussions about less important details let’s assume that for every project there are at least a few stakeholders that can provide the necessary knowledge. The process of gathering information itself is very difficult as, for example, stakeholders do not always know what they need and often have only one perspective on the matter. Furthermore, for various reasons they may be reluctant to disclose certain relevant information. A good facilitator has ways extract this information by guiding the conversation. There is quite some scientific literature about this which plays a valuable role in this thesis.

(Hayne, 1999) tells us about the general role of procedure and tools in improving a collaborative modeling session.
In short: there should be one or more meeting goals. In this case that's the mapping of the Universe of Discourse with help from a domain expert. The meeting goals dictate what modeling process needs to be used. That process is the new procedure that we're going to draft. What comes next is more interesting: the process should give a clear structure to follow. This structure in turn can be given by use of a modeling tool.

For the sake of assigning the different priorities to the participants in this knowledge extracting process it is important to know that at least two people are always present. The domain expert and the facilitator. One is representing the business world and one representing the world of information technology. The task of the facilitator is to bridge the gap. The steps of a rather general conventional knowledge gathering and modeling session are visually represented in figure 3. In this figure restructuring and transforming are separate steps.

Most parts of this approach is described by (Hoppenbrouwers, 2012).
So let’s see how these tasks can be modified to suit our three top level goals.

### 6.1 Ensuring the quality of the modeling process

Ensuring the quality of the modeling process and improving the modeling process are closely related. Both will have to be done by supporting the facilitator and domain expert in their efforts.

(Hommes & Reijswoud, 2000) define quality as a combination of three properties (note that this is a quote from page 3 of that work):

- **Expressiveness** - the degree to which a given modeling technique is capable of denoting the models of any number and kinds domains;

- **Arbitrariness** - the degree of freedom one has when modeling one and the same domain;

- **Suitability** - the degree to which a given modeling technique is specifically tailored for a specific kind of application domain.

Let’s focus on the facilitator for quality of the modeling process. Why? The domain expert is usually not familiar with the workings of a modeling process. Expressiveness, Arbitrariness and Suitability are goals that a domain expert can not be expected to reach on his own. That means the facilitator is mainly responsible for securing a high quality information transfer by applying the above. In other words: the facilitator will have to bridge the gap between the business and IT perspective by asking the right questions and giving the right examples. He has to integrate the three quality properties in his procedure.

Classically a facilitator can do this by adhering to the following six behavioral aspects of a successful facilitator (Niederman, Beise, & Beranek, 1996) (page 4):

1. Plans/designs meetings
2. Listens/clarifies/integrates
3. Demonstrates flexibility
4. Keeps outcome focused  
5. Creates open environment  
6. Selects/prepares technology

The goal of this research is to find a way to support the facilitator in such a way that a high quality information exchange is almost always guaranteed. This will be done in the form of a procedure. To make sure that procedure is followed, and to take make it not too taxing on the normal way a facilitator goes about his job, this procedure can be supported with the help of computer software.

Of course we cannot improve all aspects by just giving computer support. Facilitator tasks like making an agenda or giving an ad-hoc demonstration of an idea are things that computers aren’t very good at yet. This requires insight and improvisation. This is still beyond the reach of artificial intelligence. They will still be part of the procedure, but they will not be supported by software. Good old fashioned experience is needed in these instances.

Planning, designing, listening, clarifying, integration and focusing are all tasks that can be supported with computer software though. To narrow down the scope somewhat these tasks will be categorized in information gathering and restructuring of information. The information gathering category is directly visible to the domain expert. The restructuring is usually not, and for most purposes does not have to be.

The gathering stage of the conceptual modeling procedure will be visible to the domain expert. As a result this stage has to support both facilitator and domain expert. The quality of information exchange can be improved by giving a form to the communication. Also, the direction of the interview can be monitored and explicitly changed with help of software. By forcing both parties to think their statements over before uttering them the quality of information should improved. Dialog games have this effect and that’s the reason this thesis puts it to use.

The knowledge restructuring quality can be improved by making sure it always happens in a specific manner, so that the facilitator is less prone to make mistakes. On the other hand the restructuring support has to allow for the insight that the facilitator has and software lacks. Argument based modeling should give a handle on the information, that allows for (partially) automated restructuring.

Subgoals of improving the quality of the design process are:

- Give the right amount of freedom in communication while providing a baseline of useful questions and answers, thus increasing expressiveness.
- Increase consistency between different information sources/experts, in other words reduce arbitrariness
- Making sure the procedure is up for the job of modeling specific situations that suit the modeling language. This increases suitability.

### 6.2 Improving the efficiency of the modeling process

Attaining a higher quality of modeling is an important goal. But improving the efficiency is not less important. Improving the efficiency has different benefits, mainly for the organization for
which the conceptual modeling is done.

Facilitators and everyone dealing with the design products will appreciate a higher quality. But efficiency helps the business administration side of the equation more directly in reaching their targets. Modeling processes take a lot of time and effort. Reducing those demands on the domain experts frees up resources otherwise used in "just getting IT-support".

The distinction between gathering information and restructuring of information is useful in the context of efficiency. Making the gathering of information more efficient is mostly beneficial for the domain expert. This is because the load of making a conceptual model is lightened. Increasing the efficiency of restructuring and transformation of information will mainly help the facilitator. The facilitator is the person integrating and transforming all knowledge. Supporting that process will increase efficiency in terms of modeling per time.

Reducing the time and effort needed will ultimately save the client or business money. Less paid working hours have to be invested by the participants of the design sessions. It will allow for quicker start-ups of information technology projects. This will in turn reduce the amount of project failures.

Now we know why it is important we can think about how to actually increase efficiency using a procedure. The design process should be streamlined by improving the speed of communication between the two parties.

Some sub goals for increasing efficiency are: (Some come from (Campschroer, 2007))

- Increase the information density of each statement in the communication
- Increase the processing and transfer speed of the communication carrier
- Reduce the number of steps needed for effective communication
- Reduce the number of errors in the communication (this is essentially quality as described earlier)
- Guide domain experts to the right level of abstraction quickly without overloading them

6.3 Making the modeling process more fun for the domain-expert

When the modeling process is made more fun for the domain expert he or she will feel more involved. A happy and more involved person will have less problems with sharing information (Berger & Calabrese, 1975). More involvement will also reduce the perceived stress of participating as an information source.

The goal is to make the domain expert want to talk and share in stead of having to convince that person to do so.

This goal will not play a large role this paper. Whenever the chance presents itself the domain expert will be challenged and stimulated in order to be more involved. We hope that by giving the domain expert more control than is usual during design meetings this goal is sufficiently met.
7 Theoretical framework

This chapter aims to identify and summarize the theories that are going to have a major role in the creation and execution of a process modeling method. A rather wide spectrum of theories is needed for the creation of a complete procedure that guides model design from beginning to end. Some are grounded in business administration research, others are grounded in information science research. But most are related to either design, communication or facilitation theories and practices. That is because there is a flow of communication from the business side to the IT-side.

The theories summarized here then form the foundation for the procedure to come. Design related theories give a solid base for the efforts in designing the procedure and the prototype that will implement it.

To reduce the scope of this research to a realistic size, some factors and approaches will be chosen beforehand. In this section these approaches and corresponding theories will briefly be explained. Their relevance should be self evident but will be explained concisely.

The key theoretical elements are:

- Business Process Modeling Notation (BPMN)
- Dialog games
- Questioning techniques
- Arguments
- FoCons
- Pre-formalization

![Figure 4: Literature of many sorts and disciplines](image)

7.1 Business process models

Relevance: The goal of this thesis is to create a complete method for creating the models that are needed in software design. A modeling language is a very basic but hugely important choice for this method. It will be the common language between the domain experts and computer experts.
A business process model is best explained as being a flowchart (White, 2006). This flowchart shows a hierarchical order of elements from a process. This process can be virtually anything: from the work flow of an assembly line to the flows of knowledge in a thinking process. The hierarchical order mostly models the temporal component, but could also express an iteration for example. The modeling language is not very demanding or enforcing. There are very little constraints. This makes this notation easily readable for business oriented persons. This relative lack of complexity and its wide range of use makes this modeling language a very good candidate for a pioneering design method.

The OMG group has assigned standard graphical elements to create an official process language. It is called the “Business Process Model Notation”. This Business Process Model Notation, from now on called BPMN, is designed as a high level business model language (White, 2006). The most basic of operators in the BPMN are shown below.

- Events
  - Flow begin
  - Flow end
  - Intermediate and customizable events (outside of thesis scope)
- Activities
- Gateways
- Connectors

The BPMN allows for the creation of junctions and splits. These are very important features for this research. This is because these junctions and splits are the core of the model, and are for a large part the carriers of information. Which junction(s) should be placed by which business arguments is one of the core questions. Note that splits and merges are determined by the in and outgoing connectors, not by model symbols.

The generation of a BPMN model essentially is translating information from one form to another. They essence is in doing this correctly and completely. My bachelors thesis (Thijssen, 2011) has made an attempt at listing pre-made model recipes for certain types of information, in the form of business arguments. This works in a number of cases, but the list of recipes is far from complete.

For a more complete description of the BPMN I would advise the reader to refer to the OMG tutorial to BPMN (White, 2006).

7.2 Dialog games

Relevance: The gathering of relevant and specific domain knowledge is a challenging task in itself. The dialog game theory is an approach that will help structure the process of information gathering.

Dialog games are not games in the common sense of the word. They are a way to create an environment for structured conversation. As argued by (Hoppenbrouwers, Schotten, & Lucas, 2010), dialog games consist of communication following rules between people for some (mostly common) goal. The goal can for example be to construct a model together from knowledge
distributed between the participants. The participants can be geographically separated, but their medium of conversation bridges the distance.

The game element in dialog games is adding certain rules to the communication. Electronic communication means such as chatting are easily modified to support this. These rules in turn allow for “easy” structuring of communication. This is of course a step in the right direction when it comes to preparing text for input in a model. An example of structuring communication can be dialog trees, or narrowing down open answers to a multiple choice box.

A practical example of a dialog game is an experiment in which people are only allowed to use a certain range of predefined openers and replies in a chat box-like environment. These confine the range of reactions people can use, but at the same time allow enough room to form structures in the communication. The type of openers and replies can be modified to suit a certain topic of discussion. Further structure can be forced by allowing only certain sequences of these openers and replies.

This structuring of communication is one step in the right direction of formalizing normal language. This makes it a useful tool in the quest for formalization of normal language for use in model constructing. The knowledge about using normal language is present in almost everyone so (partial) formalization for use in information systems can be of great use.

This topic will be expanded further in the creation of a procedure in chapter 9 and prototype in chapter 10.

7.3 Questioning techniques

Relevance: Dialog games provide a way to structure interaction. Questioning techniques will help in asking the right things.

As this thesis will attempt to (partially) support the information scientist (or facilitator) in gathering the necessary information from a domain expert, some attention must be paid to practices for doing this sort of thing. The question techniques will need to be modified for automation or support. Only then can computer software help in asking the right kind of things.

There is a lot of research available that focuses on asking the right questions in the right way. Some of my inspirational sources are:

- Stijn Hoppenbrouwers et al. (Hoppenbrouwers, 2012), (Hoppenbrouwers et al., 2010), “Asking questions about asking questions”. This title is self explanatory.
- Work by McBurney et al. (McBurney & Parsons, 2002), on how to transform questions into something a machine can interpret as well.
- Research by Hayne et al. (Hayne, 1999), on what types of question a facilitator should ask.
- Papers by Gordon et al. (Gordon, Prakken, & Walton, 2007), on how to logically verify statements by asking other questions.

And specifically there’s the question asking framework from (Hoppenbrouwers, 2012). This I discovered while already working on a prototype solution, but it still has greatly influenced the
way in which one can come up with the right questions. By selecting each variable with things that are already known at that stage one can more easily choose the remaining one(s).

![Diagram](image)

Figure 5: A visual representation of the question asking framework from Hoppenbrouwers

Citing from that same paper (Hoppenbrouwers, 2012) the variables stand for:

- Main conceptualization Goal(s) behind the questions to ask (G): pragmatic and possibly also semantic-syntactic goals underlying the creation of the model.
- The Questions to ask (Q): the actual, complete phrases used in asking focus questions within the conversation-for-modeling
- The Answers, which are the unknown variable: the result to be obtained (A)
- Possibly, Form/Meaning constraints on the answer (F): an intensional description of the properties the answer should have (for example, that it should be stated in a modeling language, or that it should be an activity or actor).
- Possibly one or more Examples (E) of the kind of answer desired: an extensional suggestion for the answer.

In the process of creating a suitable question tree for the dialog game sessions this framework is used (together with the other literature) to give more focus to what the question should be like. Variables G, E and F are usually known. A is the goal variable, of which the desired form is known. Q is missing, but can be “deduced” by the others.

7.4 Arguments

Relevance: The knowledge that is gathered has to be restructured and transformed. Argument provide a way of keeping the information intact and valid across languages and domains.

The word argument has more than one meaning. Here is are a few definitions from an on-line
Arguments in the context of this research project mostly fall under definition 2. Arguments are the very core of the information models. Arguments are the reasons behind decisions. Decisions are made when a model is constructed by people with knowledge about the process it is describing. The decisions captivate the actual information the flow model is showing graphically. This makes the arguments supporting these decisions very important. Without arguments a statement has no real worth. It’s the arguments supporting the statement that put in the weight of the statement.

The exact definition of arguments depends a little on the model they are used in. The dictionary definition fits natural language arguments. In the case of this research there is a little more to it:

Given a statement $S$ an argument for that statement can be denoted with $ax$, where $x$ is a number. This number is useful when multiple arguments are given for a single statement. The tuple $S\{a_1, ..., an\}$ is the notation for a statement and its arguments. It is useful to declare this basic piece of modeling information as a tuple to make the relation between statements and arguments clear. Relevant knowledge is bundled, and later on this can be used to more easily transform knowledge in text.

But why use arguments? What makes them so valuable?

- Arguments can be easily collected by - quite simply - asking for reasons using ‘how’, ‘what’ and ‘why’ questions. For example: *Why is this a valid event? How would you say that this*
is the case? These types of question are control questions, and they are already part of the repertoire of most facilitators. Arguments do not really require any extra knowledge on behalf of the domain expert, but do force him/her to think over the situation that is being described.

- Arguments also provide a way to either adopt or reject the initial statement. This means that arguments provide a lot more information than the statement alone.

- Arguments are in a certain way already abstract. The arguments implicitly carry the information value of the initial statement in them. But arguments can also fit multiple statements while they were given for only one. From a facilitator’s perspective that’s like hitting multiple birds with only one stone.

- This initial abstraction will probably help when further abstractions have to be made. And it means that the information that is contained in the arguments will be kept more intact than the information from only a statement as part of the reasoning of the domain expert is conveyed as well.

![ARGUMENTS](image)

Figure 6: Arguments according to internet memes. A simple but valid representation.

7.5 FoCons

Relevance: FoCons will help with determining abstraction levels and topics in the dialog game.

The term FoCon is an acronym for Focused Conceptualization. Paraphrasing from Hoppenbrouwers and Rouwette (Etiënne & Hoppenbrouwers, 2011), FoCons for example help giving focus during an interview. FoCons work by the divide and conquer principle.

Every Universe of Discourse has so many elements that is nearly impossible to describe them all in one go. The detail in description will vary between the elements as well. The result would be a large mass of information that is just as inaccessible on paper as it was in the mind of the domain expert.

FoCons are smaller parts of this Universe of Discourse. Ideally they are defined in cooperation with the domain expert. These smaller parts allow for more controlled discussion about the topic. The level of abstraction of the domain expert is also less important. The facilitator can more easily filter the relevant details, while discarding irrelevant ones.
Each FoCon then can be seen as a smaller piece of the universe of discourse that can be more easily interpreted. That view lets us treat each FoCon as a separate but small modeling process. Vogels has used this in his master’s thesis (Vogels, 2013) to successfully divide the modeling process in smaller more tangible bits.

7.6 Pre-formalization: Argument archetypes

Relevance: Arguments will be coupled to pre-formalized model snippets, and perhaps to other formalized or standardized elements.

The key theoretical elements so far have been relatively basic pieces. They are already rather well defined. The pre-formalization of arguments is however relatively new. As far as I’m aware my work in my bachelor’s thesis (Thijssen, 2011) was the very first research to venture in this region. In that work a number of underlying arguments for certain modeling decisions were condensed into a list of “standard argument types”. These standard arguments were then paired with a BPMN-model snippet. Whenever such an arguments came up during an experimental modeling session it could be used to model the intended situation with little or no modifications.

The list of argument types that was composed contained:

- Parallel processes
- Equal alternatives
- Independent acting parties
- Time or product dependency
- Actor Capacity
- Competence
- Cost effectiveness
- Resource sharing
- Employee satisfaction

These formalizations were done by looking at several leading factors in design decisions. The argument types listed here are not nearly plentiful enough to cover every possible design decision. When needed a few more arguments will be added to this list during the experimental phase. This will be done using the method from (Thijssen, 2011).

Not only argument types will need to be pre-formalized. The right questions to discover the existence of these argument types in the situation the domain expert is describing need to be made as well. Otherwise there is no way of discovering arguments without confronting the domain expert with modeling internals. This preparation of questions will be done using normal interview techniques as described in (Campschroer, 2007) and “asking questions about asking questions” (Hoppenbrouwers, 2012).
7.7 Design science

Relevance: Design science will give guidance in creating a new procedure and prototype
Design paradigms will help in finding the right way to set up and execute a new procedure (and prototype). They will gave a handle on what it is we’re trying to create, and how to take steps towards it.

Hevner et al. (Hevner et al., 2004) have created a list of design principles to stick to when designing a new artifact. These guidelines will directly help in creating a new modeling procedure. Especially the software supported part of it. They also help with the modeling itself as that is also a design process.

The following is an excerpt from Hevner et al. (Hevner et al., 2004) containing those guidelines:

1. Design as an artifact Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
2. Problem relevance The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
3. Design evaluation The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
4. Research contributions Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
5. Research rigor Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
6. Design as a search process The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
7. Communication of research Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Hevner’s guidelines will be followed when it comes to the creation, testing and revising of the prototype implementation. They will not be referred to directly, but it is important to know that these principles form the basis of all decisions in the prototype design. As a reader you will be able to link some chapters to each of the bullet points Hevner et al. have provided.
8 Methods and approach

The main research question will be answered in two stages. First a number of subgoals (milestones if you will) will be set and investigated.

Then the main question will be evaluated on the basis of the three important conceptual modeling criteria that were named earlier: Quality, Efficiency, Completeness. Quality and Efficiency come from (Schotten, 2009). Completeness is a quality indicator according to (Hommes & Reijswoud, 2000).

8.1 Subgoals

I will first have to get several parts working together to reach an overall conclusion on the main research question posed in the problem statement. One could call them subgoals or milestones. I have chosen these:

1. Is it possible to integrate fitting interview methods in a dialog game?
2. Is it possible to (partially) structure this information, using argument based modeling?
3. Can software (partially) create models from these arguments using my previous research?
4. Is the procedure complete and functional?

Completely automating this process is very difficult - if not impossible. The goal is to minimize the effort on the part of the information source and the facilitator. That means: the more work done automatically (by software), the better. An attempt is made to make a semi-structured argument system with the intervention of an expert during the modeling sessions where human interaction is really needed. Models are then generated with help of a human expert in stead of in a completely automatic way. This will still be an improvement on the current way of business process modeling because of increases consistency.

The goal of the first three questions is mainly to keep track of the progress in various stages of the new modeling process. They exist to give subgoals for the method that is being proposed. Their relevance in grading the total product however is limited, in that aspect they only serve as milestone check marks. How well those steps have been done is less important in this stage. As the matter is completely new, a functioning part with that goal is all I’m after for now.

The last subquestion, though fairly simple, is the pivot question. It is the last question that needs to be answered before any answer can be given to the main research question. For this reasons research steps related to that last question are more complex.

8.2 Operationalization

How will the subgoals be answered? Most subgoals will only need established yes/no criteria. These criteria are established in this section. The last subgoal is more complex as it largely answers the main research question. The operationalization for this subgoal is more elaborate. A score will be given on for model and modeling process quality in the areas of complexity, usefulness and completeness.
1. Is it possible to integrate fitting interview methods in a dialog game?  
   The result will either be “no” or a dialog game with rules that are suitable for use in an  
   information science model builders environment. The hypothesis is yes. It is however an  
   important step in the total process.

2. Is it possible to (partially) structure this information, to arguments, using a software tool?  
   The result will either be a software piece that automatically structures information from  
   the dialog games in known arguments, or a software tool that does the same with help  
   from a person. The hypothesis is that help from a real person (facilitator or information  
   scientist) is needed. But an effort will be made to automatize this process as far as possible.

3. Can software (partially) create models from these arguments, using my previous research?  
   The result will either be a software piece that automatically creates a BPMN model from  
   the arguments found in 2, or a software tool that does the same with help from a person.  
   The hypothesis is that software can do this task very well, minimizing the need for human  
   labor.

4. Is the procedure complete and functional?  
   This question is harder to answer. The rest of this chapter defines how this quality index  
   is established.

Improving the quality of building models - as well as model quality itself - is the focus of this  
research. To prove the effectiveness of the suggested method more operationalization is needed.  
The following summarizes the research elements, indicators and variables used for determining  
this quality.

The research elements, indicators and variables are defined as follows:

- **Elements:** The subject of this research paper will be a new BPMN model creation proce-  
  dure

- **Indicators:** The indicators for these process are efficiency, quality and completeness  
  (Schotten, 2009).

- **Variables:** The variables for the indicators mentioned above are:
  
  - **Efficiency:** The total number of facilitator-system-subject interactions needed
  - **Efficiency:** The amount of domain expert (or subject) actions needed
  - **Quality:** The amount of facilitator interventions needed
  - **Quality:** The amount of interventions needed outside of the system
  - **Quality/Efficiency:** The grade facilitators and subjects give to the procedure
  - **Completeness:** The score of how well generated BPMN models cover the gathered  
    information
  - **Completeness:** The amount of arguments or dialog rules found to be missing after  
    comparison to the case
  - **Completeness:** The grade a test subject gives to the amount of their information  
    given
8.3 Methods of acquiring data

Each subquestion requires different types of data to answer. Shown below is a numbered list with the data acquisition methods per subquestion. The numbers of the method description correspond to the number assigned to the questions.

1. This step is answered by using the design cycle (Deming Cycle) on a software prototype that implements this part of the procedure. Using the already existing components of a structured inquiry and dialog game rules, an attempt is made at combining the two. The design cycle will almost definitely refine the prototype to a working state. A working software prototype will therefore answer this subquestion with a definite “yes”.

2. This step is also answered by using the design cycle. Using the already existing list of arguments from my previous thesis, and the results from the dialog game with inquiry rules, an attempt will be made to structure loose information bits into their corresponding business arguments. The result either doesn’t work, works with help from a facilitator, or works on its own.

3. Again, this step is answered by the design cycle. Like the previous steps, a new component is made from already known theories. The goal is to semi-automatically create a BPMN model from the arguments that are found. The result either doesn’t work, works with help from a facilitator, or works on its own.

4. The last step is the most important of them all. About five modeling sessions with test subjects will be held using the software prototype and a facilitator. Each of the modeling session will take about one to two hours. The session will use an artificial case description with room from improvisation from the test subject. The latter is preferred as it will probably yield the truest results.

A short introduction will be given before each modeling-method session is started. In this briefing the test subject will be acquainted with the prototype software and the case. The hand-outs for this introduction are found in appendix A.

During the actual modeling all correspondence and results will be logged. An audio recording will register every interaction that takes place outside of the software prototype. The facilitator—that will be me—will also keep track of comments and activities that are not logged by the system, but are relevant to the process. These will also serve as input for the evaluation of the proposed method, by linking them to the quality, efficiency and completeness of the modeling process. More about the experimental sessions can be found in the next sub-chapter.

After the sessions the subjects will be handed a short inquiry. Its goal is to collect grades of their experience with the new modeling method. A few questions linked to efficiency, quality and completeness will measure how the participant felt with regards to those aspects of the modeling session. The questionnaire can be found in appendix A.

Questions related to efficiency are: a, b, i:

- Did you feel involved in the process?
- How did the tempo feel?
- Were the questions clear enough?
Questions related to quality are: e, f, j:

- Were the guiding text and example text good enough?
- Were you guided in the right direction?
- Was there a clear structure you could follow?

Questions related to completeness are: c and d:

- Could you give all information to the facilitator?
- Were there enough response options?

A few questions related to involvement were also asked. They are not part of the operationalization. Questions g and h:

- Was this session fun?
- Was this session useful?

Open questions at the end of the inquiry will be used to assess the modeling session in general. Problems documented there will be addressed in the next revision of the prototype.

The resulting model will also be compared to the case and the amount of information gathered to grade its quality using the operationalization. The models and inquiry are also present in the appendix.

### 8.4 Experiments

The practical sessions will take at least five different participants and confront them with the new method of modeling processes as it is proposed in this paper.

Important factors in an experimental context are:

- Experiment location and environment
- Tools and materials
- Participants and their treatment
- Procedure

The experiment will take place in a controlled environment. To maintain as much external validity as possible, the environment will remain the same throughout all of the experiments. An effort will be made to keep the amount of distractions to a minimum for the sake of internal validity.

This controlled environment is my work place. There is no background noise or music other than the humming of the computers that are used. Light conditions are moderate. Temperature is about 18-22 degrees Celsius. The room is decorated and might be described as being lived in: not overly neat and not messy. In other words: average room conditions.

The tools and materials consist of the following items:
- A laptop for the participant
- A desktop for the facilitator
- Pen and paper containing the introduction, case and questionnaires
- The prototype software running on the computers described above

The computers are not running any software other than the prototype, a voice recorder and optionally a notepad application.

The participants are selected by ease of access and not by many strict requirements. They need to be part of the working population and should be able to handle a windows computer. This is done because domain experts in non-experimental situations are not any special type of person, other than probably being part of the working population. The experience the subjects have with computers and modeling processes is shortly assessed by means of an intake form.

Participants are made to feel as comfortable as possible. They are offered a seat, and something to drink. In a short explanation the participants are told that they are allowed to talk and ask questions, but not about the case itself. When they are ready, the experiment begins.

The procedure for the experiment is as follows:

1. The participant is welcomed and seated
2. The participant indicated that he or she is ready to begin
3. The participant is offered the papers containing questionnaires, introduction and a case (see Appendix A)
4. The participant is asked to fill out the contact details and intake questionnaire
5. Then the participant is asked to read the introduction and case descriptions carefully
6. When they are ready, the person is then handed a laptop
7. An audio recording is started
8. After that the dialog game prototype is started on both computers, names are entered
9. The facilitator begins actual the experiment by starting communication using the prototype
10. During the experiment the facilitator takes notes on anything that does not pertain to what takes place in the software itself
11. During the experiment the facilitator can adapt questions
12. A few rounds of questions will be asked using the prototype
13. After the case is exhausted, or when the participant’s time slot is ending the facilitator ends the dialog session
14. The participant is asked to fill out the evaluating questionnaire
15. In some cases a verbal debriefing is held to supplement the facilitator’s notes
8.5 Methods of data analyses

A short recap of the subquestions:

1. Is it possible to integrate fitting questioning methods in a dialog game?
2. Is it possible to (partially) structure this information using argument based modeling?
3. Can software (partially) create models from these arguments using my previous research?
4. Is the process complete and functional?

The first three questions are design related and will be answered by checking the prototype progress against the criteria set earlier. As soon as a criteria is met the question is answered with a yes. Otherwise the outcome will be no.

Only the last subquestion really needs more thorough data analyses. Inquiries will consist of multiple choice questions with scored answers related to the research variables. An average grade will result from the combination of these answers. A better grade means that the person thinks the system performed well on that criterion, and vice versa. The quality of the BPMN model will be judged by the subject and facilitator. A 5-option score for each of the procedure goals will be given. A higher score means that the procedure is found to be of performing well. Each session will also yield a number of counted statistics, found in the research variables. For example: the amount of actions needed outside of the modeling system.

The grand total will be made of the average of each of those scores.

Please note that it is not the goal to statistically prove any of the performance indicators. The sample size is too small to do this. The goal is to see if the modeling procedure is viable, and that will be done by the grading described above.

8.6 Validity

To maintain internal validity, during all sessions of modeling different subjects with different backgrounds will be used. Participants of one inquiry will be excluded from other inquiries about the subject to ensure objective results. Ideally I would organize a large number of modeling sessions to improve statistical reliability. Because of limits in time and resources I will try to have at least 4 sessions.

External validity is maintained by keeping the research practice oriented. It will be focused on answering only the subquestions and research question. These are linked to real world use of models. Multiple moments of feedback on the arguments by people active in the field of BPMN (students, experts) will help to keep the research going in the right direction.

Statistical validity is only guaranteed for the answer of the main research question and the subquestions. It is not possible to make any conclusions or predictions on how the proposed method will fare within certain environments. To do this more trials than allowed by the time allotted for this thesis need to take place. In short, this work is explorative and not comparative.
9 The new methodology

This chapter will explain how the new modeling procedure will work. Please take into consideration that this chapter aims at a general explanation. Some important aspects are left quite open, as they depend on the choices of implementation. For an explanation of the actual implementation that was made see chapter 10 about the prototype.

9.1 Three parts

The modeling process has a beginning and an end. The beginning being the wish for gathering of information, and the end being a nice crisp model or other formal conceptual map of said information. Of course something has to happen in between. I have chosen to separate the workings of the proposed modeling procedure in to these three pieces.

- **The input** will be getting the required knowledge from the source by means of an interview.
- **The process** will be the structuring and transforming done to this knowledge to make it more suitable for BPMN output.
- **The output** has to be writing that knowledge to the BPMN language.

Each of the three parts has its own procedure description and prototype tool implementation (for prototype, see chapter 10). Each of the parts has its own goals and techniques to reach those goals. By using this division it is also possible to assign the available theories that were named earlier to the steps of the method. That enables me to give each step more focus.

![Diagram of Input, Process, and Output](image)

Figure 7: An overview of the usage of concepts in the proposed method

The definitions given by the various theories bring us somewhat closer to deciding how each piece of the puzzle will be put together. But the gaps in between the previously chosen concepts still need to be filled. This will be done “on the fly” during testing and adjusting, based on design paradigms and best practices found in the relevant literature.
9.2 The input step

**Goal:** *Methodically collect domain knowledge from a human source in the form of semi-structured text.*

The input step is created for gathering the information/knowledge that needs to be captured in the conceptual map. In other words: the Universe of Discourse (see glossary chapter 14) has to be explored and later mapped. Traditionally this exploring and mapping is done by setting up a meeting with persons that have knowledge about the relevant business processes. This person, called a “domain expert”, is then interviewed by the facilitator. And he writes down every piece of interesting information he hears, trying to capture knowledge as best as he can along the way. A facilitator is experienced in using various techniques to extract the required knowledge from the domain expert.

Dialog game theory is an interesting candidate for supporting that knowledge extraction task with information technology. Dialog game theory is a rather broad and general concept that does not fully accomplish “hosting an interview” by itself. It is not a complete theory, nor is it adopted in industrial use yet. Without implementation dialog games are not much more than “an interesting idea” without application. It means that one or more participants play “a serious game”. The goal of which is to have a dialog that is guided in the right direction by certain rules.

So the dialog game concept is meant to be implemented (for example in computer software) as a tool to structure conversations. Dialog games need rule sets. Most importantly to dictate the structure of the communication between participants. We will call that rule set the “communication rule set”. The rule sets for this procedure will be based on an argumentation approach.

Now that the concept dialog game is established we can look at how to actually “play” it. Before actually starting a game some general rules have to be chosen before the actual communication rule set can be drafted. Here those rules for this dialog game procedure:

- The dialog game is played by two people
- A participant can only speak when it is their turn to speak
- One participant has to role of domain expert and the other is the facilitator that guides him through the dialog
- The type of this communication is an interview, this means that the conversation will contain mostly questions and answers
- As the flow of information typically is from the domain expert towards the information gatherer, the questions will mostly come from the facilitator and the answers from the domain expert
- Important statements and elements that are recognized by either participant should be marked by using special symbols such as `,<,>`
- The domain expert has knowledge of the process that is being mapped
- A facilitator has knowledge of modeling principles and can act on that
The result of the dialog game will be a large amount of semi-structured natural text containing the knowledge we’re after. The real challenge lies in getting this text structured in a readable (for computers too) way without confining the communication options too much. Structuring the text allows easier extraction of that knowledge from the dialog game text.

This structuring aspect is provided by the rules of the dialog game. The structure that communication rule set provides helps with extracting and abstracting the knowledge from a domain expert by giving all of the response options beforehand. That means that the domain expert does not have to worry too much about how to give information, the rule set already gives him an example. It also helps the facilitator in selecting the right type of action to get more information than he would usually be able to.

How the game rules and dialog structure rules relate to the dialog game itself can be seen in image 8

![Figure 8: The dialog game needs rule sets and participants](image)

The communication rule set for both participants could grow rather large because of the many dialog options that need to be present. Only general demands for are documented here for this reason.

The procedure proposed in this paper will use argument based modeling approaches in the rule set. That means that argument based modeling, together with some other theories, will dictate the basis of the rules (question types and example text) in the dialog game. By asking for - or using the - arguments behind the statements the domain expert is giving the resulting interview dialog should contain information that would normally not be present.

This argumentation approach is based on the argument types that I have formalized in my bachelor’s thesis. (Thijssen, 2011) Each and every possible argument needs one, or preferably more questions linked to it. The goal of these questions is to “trick” the domain expert into naming activities that are related to the situation that is best described by the argument. “Trick” is the focus here, as directly confronting the domain expert with modeling practices is the opposite of our goal. So just asking to name “parallel activities” by itself will probably not work. The following is a fictive example for a smart question related to the “parallel activities”
Facilitator: Can you think of any activities that take place at the same time?
Expert: Well, yes
Expert: Driving and Texting.
Facilitator: A bit dangerous, but I think you’re right.

Now we have general conversation rules, and a rule set containing question types and examples generally based on argument collection. What else is needed to effectively gather the required knowledge from the domain expert? To answer that question we need to identify the exact type of knowledge that is needed to create a model. By looking at what is needed to complete a model segment in BPMN, I have identified the following types of text as being very important elements:

- Decision criteria. *i.e.* *time, money, people.* These are usually Nouns that describe some variable of importance.

- The names of activities. *i.e.* “Polishing shoes”, “Reading research papers”. These often are Verb-Noun constructions describing the tasks being done in the process flow.

- Relationships between the Nouns and Verb-Nouns. These are directly related to their respective argument types but also dictate the order of things. *i.e.* “…increases cost of …”, but also “… needs to be done before…”

That means that the communication rule set should accommodate these types of expressions from the domain expert. Whenever anything of importance is mentioned, the facilitator should be able to “dig deeper” and change focus to what has just been said. To do this the procedure will use FoCons (see chapter 7). The first two types of expressions make excellent FoCons but also carry information needed for various labels in the modeling language.

With these concepts in hand (BPMN structure, information gathering theories, argumentation, FoCons and rule sets), I have created a work flow for the gathering of the required information from a domain expert. The expert should not be confronted with the difficult logics and choices present in model generation. He or she will follow a pattern that is dictated by the structure of a much easier understood dialog tree. The dialog tree will contain example text that explains the function of each dialog node to the participant. A model of the work flow is shown in figure 9.
The communication rules come in the form of a dialog tree. This tree enables to define the structure of the communication relatively easily. Each node of the tree will contain at least one, but generally several response (question or answer) types. The response types fit in the context of the responses that are possible in the nodes that lead to and come out of that particular node. Each response type is linked to a piece of example text. Whenever a participant selects a response type during the dialog, the example text is shown. In some cases the participant does not even have to change this example text to continue the dialog.

Figure 9: The dialog flow in the input stage

Figure 10: The dialog tree determines which conversation path to follow.
Image 10 shows how the tree structure regulates the response options of each of the participants. The dialog tree contains a root node. In it are one or more response types, and each has its own example text. The different response options also lead to different tree nodes, that correspond with the required response options. It is also possible to have two consecutive tree nodes that belong to the same participants. That participant then needs to take two dialog turns after each other.

This example shows that the dialog game says that the Facilitator has to take the first turn. This is the root node turn. The facilitator has to options, a “what” question and a “where” question. The corresponding example text is also contained in the dialog node. After choosing the question type the example text is automatically displayed. The participant can then change that example text to suit the situation, or leave it as it is. After then ending the turn by sending that text to the other participant the dialog tree structure determines what node has to follow. This node is then read, and the participant who’s role corresponds to that of the node gets to take the next turn.

The interview is split roughly in three phases, each with its own goal and a higher level of focus. That means that every phase has different types of questions and example texts. The phases help in making a clear division in abstraction level and preciseness. The facilitator will decide when to go from one phase to another, although the expert may ask for a phase change. The phases allow the facilitator to “zoom in” on certain aspects while only glancing over other less relevant ones.

The following list summarizes the goals of each phase.

1. General discussion phase. In this phase the expert and facilitator are allowed to talk freely. No communication rules are imposed, but the facilitator is given a list of suggested questions. The goal is to get a set of interesting topics from the expert and to discuss new ones when they emerge. This is accomplished by asking for process goals, parts the expert thinks are important, questions about the order of the things already named. Typically Noun variables and Verb-Noun constructions (in other words: FoCons) are gathered in this phase. A list of interesting topics that have been collected so far is kept by the facilitator and repeated back to the expert every time it is deemed relevant. For example - right before going to the next phase by selecting a FoCon that will be reviewed in more detail. The facilitator tries to get the order of these thing clear as well, by starting with the FoCon that takes place in the beginning of the process of the expert.

2. Decision question phase. In the second phase the expert is asked for decisions that contain the FoCon or relate to it. The goal is to get a name for those decisions, their criteria and the relevant options. If the FoCon name already is or implicates a decision this naming of decisions is skipped and the FoCon name is chosen as the decision name in stead. After that the alternatives (or rather options) and their relations of each of the listed decisions are collected. The dialog option for open discussion is still available to solve any misunderstanding, to give some guidance and to give examples. Lastly the domain expert is asked for the decision criteria of each separate decision relating to the current FoCon. When this is done the next phase begins.

3. Argument Based Modeling phase. The deepest level abstraction level is the argument phase. Questions that allow the relations and criteria to be matched to a certain argument type are used for two reasons: To get more new relations and descriptions and to gather the arguments behind those relations. The facilitator has to look at the responses from the second level to suggest an argument type to the domain expert for confirmation. The
expert does not have to do much at all, except for validating the suggestions that are given to him. After this last phase the interview will go back to the first level of abstraction and the facilitator will (eventually) select a new FoCon or end the interview.

Eventually the facilitator will have collected all the relevant FoCons the domain expert can think of. After all FoCons are subsequently reviewed by taking them in to the next phases of the elicitation procedure, all necessary knowledge for a BPMN model of the case should be recorded in the interview log.

The following example is meant to explain how the phases help with getting to the right level of abstraction.

- **Naming**
  Before doing anything the expert is asked about what he is going to be interviewed. This for instance could be *A visit to the gas station*.

- **Phase 1: open discussion**
  Now an open discussion begins. The facilitator asks some general questions, like *What is your goal?* and *What do you think is important?* The expert can then answer in any form (s)he likes. Example responses are: *I want to fill the fuel tank of my car when it’s empty* or *The gas prices*. These contain valuable elements that the Facilitator recommends to be FoCons, for example *Fill fuel tank, Car and Gas prices*. These are renamed to suitable naming conventions and repeated back to the expert. One at the time is selected and reviewed further.

- **Phase 2: decisions and relations**
  When a FoCon is selected it can be mapped in more detail. We can now take *Gas prices* as our example focus. When the expert is asked for decision alternatives or options (s)he might say something along the lines of *Gas prices are high or low* or *I can pick various gas stations*. The expert then has to name his/her decision criteria. For example: *Gas quality for price offered* or *How badly I need to refuel*. The facilitator now adds any new elements the expert may have used in explanations to the list of possible FoCons and then initiates the last phase.

- **Phase 3: argument types**
  The facilitator suggests the best fitting argument type for each of the decision criteria. In this case *Cost effectiveness* and *Capacity enhancement* would probably fit best. The expert can confirm or modify the suggestion. Eventually (s)he confirms a best fitting argument type.

- **Phase 1: open discussion**
  Now the interview is back in the open discussion phase where the facilitator can explore any new elements or topics with the expert. After determining the right order of things a new FoCon is selected.
Image 11 depicts that each phase allows a higher density of (validated) information by use of focusing. Useful information (conceptual elements in some form or another) is allowed to be described further while irrelevant parts are filtered out by not passing into the next phase. This method of selection leans on the skills of the facilitator to recognize what is important and what is not.

Summarizing the procedure for gathering information now has a form, the dialog game. Some basic assumptions were established. The general shape of the dialog options and the contents of those options were briefly explained. The needed level of detail is given by the different interview phases and the funnel model. What’s left is what needs to happen in between to guide the domain expert through the interview without too much confusion.

Between each interview phase the facilitator explicitly describes the goal of the next phase to the domain expert. When changing back or forth between phases the domain expert is informed again to avoid confusion. During the whole session the facilitator can suggest possible answers to the domain expert. In that way the procedure helps the facilitator without blocking his/her own expertise by disallowing additions. The questions types and example text in the dialog game are reminders of the things the facilitator can ask to help the situation while giving the expert a helping hand in answering. The communication rule set should also allow the facilitator to give situation specific examples to clarify things to the domain expert.

Now that the necessary contents of the rule set are chosen it is possible to choose the actual contents of the communication rule set. As mentioned before the rule set is a tree that contains dialog nodes. These nodes contain question types with pre-structured pieces of text. Appendix B contains the rule set containing all the questions for a relatively simple process modeling interview. Each interview phase has its own type of questions and answers these will be explained in the next few sub-chapters.

A rather general example list which I find works pretty well is listed in the following subsections. The question types and their example texts could (or should) be customized by the facilitator to suit the modeling goal.
It’s important to remember that any dialog suggestions made in the following few subsections are put in order of importance. The most important one will be on top of the answer selection list, and so on. This makes sure that both interview participants are most likely to select the right dialog options before they reply to each other.

9.2.1 General discussion phase

Goals: Establishing a list of interesting topics and then selecting one for further review.

As summarized before, during general discussion both parties can discuss whatever they think is relevant. The main method of communication is free text. No communication rules are enforced. Although a list of question suggestions is made available to the facilitator so he can methodically explore the Universe of Discourse.

The response options are those suggestions:

- Open question
- Goal(s) of the process
- (Un)important parts in the process
- Ask by example
- Select FoCon (relation phase)
- Next FoCon (relate to previous focon)
- Other FoCon (anything in between)
- Stop session

The domain expert can only answer in free text until the facilitator proposes a FoCon.

In this open phase the facilitator has to compose and maintain a list of interesting topics the domain expert has named. By making the distinction between decisions and criteria, variables and work descriptions a directed line of questions is made much easier.

That means keeping a separate administration with lists of multiple types of interesting topics:

- Decisions
- Variables
- Tasks / work descriptions

A list of decisions gives hints when more detailed questions will have to be asked in the interview next phase. The variables list is usually a list of Nouns. Work descriptions will have to be summarized as a Verb-Noun construction. This is almost always easy to do on the fly. And it will give the names of various (top or lower level) activities.

By repeating the contents of these lists back to the expert, he or she will be primed to think of more topics. Seeing the things they have discussed before in a neat and ordered list will trigger relational thinking. The result of which is more of those same knowledge elements.
these elements with marking signs such as <and> allows the terms to be picked up more easily in the processing part of the modeling procedure.

Also, in this interview phase the facilitator has to ask for the "natural order" of the things the expert is describing. The expert will then have to think of steps that he/she might have forgotten at first. Mapping the elements in order is of importance for the next steps.

Here's a short example of how this might work:

**Facilitator:** What parts of your process are important to reach your goal?
**Expert:** Driving to a gas station, and then filling up the car with gas.
**Facilitator:** Am I correct when I summarize this as <Driving to gas station> and <filling up the car>?
**Expert:** Yes.

The facilitator asks a question that is suggested by the dialog rule set. It works, as the expert replies with a few interesting topics. The facilitator can then ask for more things that are related to what was already explored.

By responding to these questions the domain expert will eventually have described his work flow from beginning to end in general terms that can be used as FoCons. The facilitator will help by steering the conversation and abstraction level during this phase, and by suggesting examples. Then the facilitator can choose a topic that - in his opinion - is interesting or important, and make it the Focus Concept.

9.2.2 Decision and relation phase

**Goals:** Exploring decisions, relationships, and criteria that are related to the chosen FoCon.

After a FoCon is selected in the open phase it’s time to ask for its relations to other topics, concepts or potential future FoCons. In this phase free text communication is still possible but the question types and example texts are more goal oriented.

This is the list of responses made available to the facilitator at this interview level:

- **Free type**
  
  *For general communication, suggestions, questions, etcetera.*

- **Ask for decision**
  
  *Ask for decisions made in the process that are related to the FoCon.*

- **Ask for decision options**
  
  *Ask for the options that are present for the decision that is being made.*

- **Ask for criteria**
  
  *Ask what elements or situation influence the decision outcome*

- **Suggest**
  
  *When quite sure, suggest relations or decision elements and ask for confirmation*

- **Adjust abstraction level**
  
  *Ask for more or less detail depending on the facilitator’s judgment*
• Next step (control questions phase)
  \textit{Go into the next phase}

• Step back (open phase )
  \textit{Go into the previous phase}

The facilitator can give suggestions or examples to help the domain expert. The main focus is on using decisions to get more knowledge and entity relation descriptions. Ask for these relationships between elements can lead to the discovery of new FoCons. The facilitator should repeat and restructure any new interesting concepts and add them to his FoCon list. Surrounding them with brackets or hooks in the dialog makes them easier to pick up in the following modeling procedure steps.

The domain expert can answer in a similar way:

• Closed answer
  \textit{For giving a specific answer to a specific question from the facilitator}

• Free type
  \textit{For general communication, suggestions, questions, etcetera.}

• Name a criterion
  \textit{Specifically give a decision criterion.}

• Ask for an example
  \textit{Ask the facilitator to describe an example situation to clarify things.}

• Don’t have an answer
  \textit{Ask the facilitator for another question.}

• Remind me of the current focus
  \textit{Ask the facilitator for the current FoCon again.}

• Step back (open phase )
  \textit{Go back to the open discussion phase}

By asking the right type of questions (opposed to just asking for all possibilities) the facilitator can save some time and frustration. The interview log will contain all important facets for the conceptual mapping later by the end of this phase. To make sure nothing is missed and no irrelevant relationships are described the facilitator can (and should) opt to ask control questions in the next phase.

\textbf{Facilitator:} (ask for decision) What decisions are related to <filling up the car>?
\textbf{Expert:} (free type) I often choose between different types of gasoline.
\textbf{Facilitator:} (ask for decision options) What options do you have then?
\textbf{Expert:} (closed answer) <Regular gasoline>, <premium gasoline> and <unbranded gasoline>.
\textbf{Facilitator:} (ask for decision criteria) What makes you choose between one or the other?
\textbf{Expert:} (closed answer) Mostly fuel prices. Sometimes the mileage or reputation I get.
\textbf{Facilitator:} (Suggest) But I think the fuel cost is most important?
\textbf{Expert:} (closed answer) Yes.
9.2.3 Argument phase

Goals: Collecting argumentation, suggesting a possible argument type for the FoCon decision reviewed earlier.

The argument phase is a type of reflection period for both interview participants. The facilitator can more explicitly ask questions to confirm suspicions regarding the Universe of Discourse. These questions are now mainly aimed at assigning a decision to an argument type, which might help in the following procedure steps.

Whenever the facilitator feels that certain aspects from the last phase might be not especially relevant he should ask some questions to confirm or deny that feeling. A few suggestions for the dialog response options are listed as part of the procedure:

- Suggest argument
  When quite sure, suggest an argument type for confirmation
- Ask about criterion
  Review why a domain expert has named that criterion, and if it is relevant
- Ask explanation
  Let the expert generally motivate his decision to name something
- Assert
  Ask the expert to assert a relationship example between elements he has named in the previous dialog phase
- Free type/open question
  Ask any question
- Done
  Exit this phase and go back to the open discussion

Again the response options for the Expert participant reflect these questions:

- Open answer
  Reply in a natural sentence
- Confirm suggestion
  Confirm a suggestion the facilitator makes
- Modify suggestion
  Change the suggestion the facilitator makes
- Don’t know an answer
  Signal the facilitator that the question was not clear enough, or has no answer
- Done
  Ask permission to exit this phase and go back to the open discussion

Again, an example:
Facilitator: (suggest argument) I’m guessing <Fuel Costs> is related to Costs in general.
Expert: (confirm suggestion) Yes it is.
Facilitator: (ask explanation) Could you tell me how exactly?
Expert: (open answer) By choosing a cheaper type of fuel I have lower fuel bills. This helps to keep the running costs down.
Facilitator: (assert) Okay, fuel choices influence running costs.

9.2.4 How the dialog game is played

After all this procedural explanation I am sure you are becoming a little confused. Hopefully giving an example will help understanding how the dialog game is played.

First of all, the rules need to be set. The few base rules named earlier don’t change. But the communication rule set might change depending on the interview subject. Getting to the point, this means: the facilitator should prepare a suitable rule set/dialog tree for the required interview type beforehand!

Because this rule set is in a structured tree form, it also contains the role distribution. It determines who gets to take the first turn, and the turn after that, etcetera.

Let’s say that the facilitator has prepared the dialog tree that can be found in appendix B. The game always start at node 1 in the dialog tree, the root node. That very first tree node contains this entry:

Step=1
Role=F
QuestionOption =Introductie
QuestionExample =Hallo. Welkom bij de dialog process modeling sessie. Met een interview ga ik je situatie die je wilt beschrijven schetsen

Meaning that the facilitator gets to take the first turn. His response options are limited. The only thing that can be chosen is to do an “introductie”. The example text belonging to this response option does not really need any editing. Let’s say the facilitator chooses to send that example text as it is. The actual message sent to the other participant is:

Facilitator - Hallo. Welkom bij de dialog process modeling sessie. Met een interview ga ik je situatie die je wilt beschrijven schetsen.

With sending that message the facilitator ends his turn in the dialog. Then the “2” at the bottom of the tree node tells the game that tree node 2 is the one that should follow after sending that particular (and only) response option.

Node 2 looks like this:

Step=2
Role=F
QuestionOption =Vraag een naam voor het proces
QuestionExample =We beginnen met het vinden van een naam voor wat je wilt beschrijven. Wat gaan we bespreken? Bijvoorbeeld: Klanten screenen en registreren
As we can see, the facilitator gets another turn, not the domain expert. This time there are more response options for him to choose from. In this example, the facilitator chooses the first response option. The response type is “asking for a process name”.

The example text corresponding to this is “Let’s find a name for what you’re going to describe. What will we be talking about? For example: Screening and registering of customers. Again, the facilitator can send this message without any editing of the example text that was already provided by the dialog rule tree. Sending this message in this example leads the communication path to tree node 3.

Node 3 looks like this:

Step=3
Role=E
QuestionOption =Ik weet een naam
QuestionExample =Ik denk dat <...> een goede naam is
4
QuestionOption =Ik weet geen naam
QuestionExample =Ik weet op het moment geen geschikte naam
4

Now the domain expert receives a turn. He has two response options: one when he knows a good process name, and one when he doesn’t. Which one he chooses does not matter much for the dialog path, as in this example it will continue with node 4. The text examples do differ. When the expert thinks he knows a good name for the process that he is going to describe, he is tempted by the example text to put that name between < and >. That’s exactly what we want to happen. In this way the example text helps to get the right knowledge out of the participants without over-complicating matters.

Eventually the facilitator can decide to take the interview to the next phase. He would only do this when he thinks he has collected to knowledge he needs for continuing on a more goal oriented level. When he selects the response option that is named “go to decision phase” the next interview phase will begin.

In this way the interview goes through to three phases as shown in figure 12. The dialog rule set should be designed in such a way that each of the phases contains questions that are more in depth than the others. When all required information about one FoCon is collected, the cycle starts again.
This exchange of messages continues until the facilitator proposes to stop. When the domain expert agrees, the dialog game is ended.

Summarizing, the dialog tree is in charge of which way the dialog goes by:

- A tree structure (with node numbers) that links consecutive question to each other, guiding communication flow
- Providing example texts that only need minor adjustments, such as blanks being filled in
- Distributing the turns between each of the participants

And the collected results contain, per message sent:

- The tree node label to which that message belongs
- The response option chosen by the participant in that turn
- The actual text responses

9.3 The process step

**Goal:** Extract domain knowledge from the semi-structured text and transform it in a form that can be generally applied.

The process phase embodies the interpreting, ordering and transformation of the dialog log that is gathered in the input step. What’s going into the process step is a text log of the dialog
game with their respective tree nodes and choices. The resulting product from this process is an ordered list of decisions tuples that carry relevant activity names and criteria in tuples.

Why were decisions chosen for this? Decision statements are the core of the required domain knowledge for most types of conceptual modeling (Campschroer, 2007). This was deducted from the practical trials.

Thanks to the text log, dialog tree and required BPMN structure from the input section of the modeling method, we can point out most of the elements that play a significant role in transforming and restructuring.

- **Complete text sentences** that contain the actual domain expert knowledge, in natural text form
- **Tree nodes** that contain useful contextual information about the text sentences
- **Conversation structure choices** that contain useful information about the semantics of the text
- **Activities** that are core statements with regards to the process activities related to an argument type
- **Argument types** that allow for pre-formalized mapping of those activities to specific junction types in the modeling language

To begin with, the process step first needs to pick up the important parts from the dialog while ignoring unimportant pieces of text. This helps the information analyst in his task to sort through the information. This means going through all the sentences, looking for FoCons, statements and relevant pieces of texts. Let’s call this process the pruning of the conversation.

After the relevant knowledge is accumulated it needs to be restructured. This is the transformation task the information analyst has. The (often chronological) order of events needs to be established. Most of the ordering of elements happens during the dialog game, as the domain expert is questioned in chronological order. When this is not enough, extra sorting and ordering can be done by sorting the FoCons after pruning.

Chances are that each and every FoCon plays a role in the Universe of Discourse the domain expert has described. That means that they have to be linked together in a way that fits the description well. By matching the FoCons to decisions these relationships can be established and later modeled. Each of these decisions is then matched to an argument type that best describes the situation the domain expert has elaborated. The result is a tuple of information that contains the knowledge to create one BPMN model piece that describes what the domain expert has said.

The following sections will explain each of these three techniques in more detail.

### 9.3.1 Pruning the conversation

The dialog log going into this process step has a structure that is almost completely created by the dialog tree rule set from the input phase. Also the response options (carrying semantics) and actual text responses between the facilitator and expert are logged during the conversation. We can use this to our advantage when we need to parse and prune the dialog text from the input stage of the modeling procedure.
That predefined structure enables the information analyst to skip irrelevant pieces of text and focus on the core elements that the domain expert has contributed. For example: one can choose not to look at responses that are of “free text” type, but only at the ones that are of “list decision criterion” type which is probably more important. The information analyst should be familiar with the dialog tree in order to fine tune the pruning method however. If he is not he can only use part of the dialog structure to his advantage.

The structure in the text has two helping factors:

1. **Marking** Brackets [...] and hooks <...> in the dialog game mark places that contain important elements. The domain expert uses these whenever he recognizes something he describes as important, but the facilitator should always repeat those elements back surrounded by this marking. Without knowing the grammar of the dialog tree one can just filter out all of the text that is marked. The result is a bunch of important elements, almost exclusively FoCons, ready to be sorted.

   For example, a sentence from the domain expert could be:
   
   **Expert** I guess I like <birds>, <bees>, <flowers>and <trees>
   
   While this example contains relatively many marked FoCons, other parts of the conversation will not. Pruning by looking at previously marked text eliminates the need for scanning all of the text for important elements. The result is a small list containing only the elements that are important for the facilitator:
   
   - Birds
   - Bees
   - Flowers
   - Trees

2. **Patterning** By looking at the grammar in the dialog tree that lies behind to chosen dialog text, the facilitator can learn more about the marked text. The response type gives a meaning to the element that is being mentioned. And when looking even further back, one can deduct information by looking at the type of responses from both expert and facilitator leading to the actual response element being reviewed.

   For example:
   
   **Expert** I guess I like <birds>, <bees>, <flowers>and <trees>
   
   Says very little about what type of elements are being named. When adding the response types to the example things become slightly more clear:
   
   **Expert** (name options) I guess I like <birds>, <bees>, <flowers>and <trees>
   
   We now know that the expert was naming options, because that was the response type he selected before typing his actual answer.

   Looking at the actual pattern of response types leading to that one response clarifies matters even more:
Facilitator (Name decision) What would you call this decision?
Expert (Free type) Naming my favorite word in a song.
Facilitator (Ask decision) What is the most important criterion this decision?
Expert (Give criterion) They must rhyme a little bit.
Facilitator (Ask options) Could you name the options you have in this decision?
Expert (name options) I guess <birds>, <bees>, <flowers> and <trees>

Because of the pattern we know these options belong to a certain decision the expert makes. He has also given a criterion related to that decision.

Lets do another example: an actual text response from a test subject would be:

Yes. The car would be most expensive (if it is my own). Maybe public transport would be as expensive or more, depending on the place of the store. The bike and on foot are almost free, not counting maintenance of shoes and bike.

The facilitator would then recognize and repeat the elements [car], [public transport], [bike] and [foot]. These are all Nouns that appear to be important in the story the expert is telling. They’re probable worthy FoCons. In this step of the procedure they would automatically be identified when pruning by marking because during the dialog game the facilitator has marked them.

When including patterning we see that in this example the domain expert had selected “free type” as response option. Meaning that he wanted to express something that was not among the other possibilities (those would be giving a direct response or asking for an example). We can then deduce, looking at the sentence, that he is giving an explanation for elements he named earlier.

Now also including a previous conversation segment it turns out that the facilitator asked:

Can you name anything you would relate to or changes <transport> because of [cost]?

So all elements named are related to FoCon transport because of cost. We also know that the facilitator made a suggestion by using an argument type from the list. The elements should then be grouped accordingly.

This in itself will be a modest win in modeling efficiency and possibly quality. But the gains are not as large as the ones we are looking for. Therefore we need something more: argument analyses.

9.4 Sorting and rewriting

After the dialog log has been pruned the elements that remain deserve some extra attention. As mentioned earlier, most elements (such as the FoCons) should be in the Noun or Verb-Noun form. Whenever this is not the case, the facilitator should rewrite the elements so that they are. Otherwise the BPMN model snippets might not make any sense later on in the modeling process.
The dialog rule set guides both participants in naming elements in the order the domain expert thinks they should be in. But sorting could still be necessary. The order in which the elements are placed is also the order in which they will be put when writing to the modeling language. For this reason the facilitator should group all elements belonging together. Then he has to sort them in the order in which they belong.

9.4.1 Argument matching and tuple creation

Argument matching will be the part of the process that allows for easy translation to (any suitable) conceptual modeling language. It is an abstraction from the form of knowledge that it was in and allows it to be put into another knowledge carrier. So an argument embodies the meaning of a certain statement. It is more abstract than the statement itself.

This abstraction helps keep the information it carries intact when it is transformed into another domain. For example: from text to a conceptual model. Coupling the names of the activities that are part of that argument type to the argument type itself would mean that all information for a piece of pre-formalized model is present.

This argument matching can be done by looking at the key lines distilled from the dialog game output text. The questions in the dialog game already point towards the formalized arguments and should therefore help in matching their answer to the actual arguments.

During the dialog (input) stage of the procedure the domain expert is asked for decisions. According to the test subjects these decisions are the most important statements to record. These can be explicit decisions, but they can also be left somewhat implicit. The facilitator asks for the related decision criteria. He also tries to get all the options of a decision out of the domain expert. Later on in the interview this decision is also coupled to a preliminary argument type. This means that all the required information from the UoD is in the text in the form of elements.

It is up to the facilitator to get all of the required information. He has to read the dialog log and put all the required elements in a row to create a decision tuple. He then has to assign an argument type to this that describes the relation between the decision options best. Luckily this task is relatively easy with the help of the elements and the right dialog structure. The decision criterion and the contextual information (elements) are rich enough to decide which argument type fits the situation best. The facilitator assigns this argument type to the decision.

The result is a tuple of information:
Decision { name, {criterion 1, ..., criterion n }, {option1, ..., option n }, argument type}

The knowledge necessary for the creation of the tuple is found in the dialog text log. The rule set used in the experiments makes sure the responses that contain that specific knowledge are close to each other. The domain expert is asked for a decision name and type, criteria, options and so on in the second interview phase. In the third interview phase a first hint towards a fitting argument type is collected as well. By looking at that segment of text the facilitator can copy paste what was said into the tuple form without too much effort.

This tuple, together with a list of elements, is passed on to the last procedure step: the output step.
### 9.4.2 Example situation

Here is an example situation that should clarify what marking, patterning and argument matching do.

Let’s assume that the dialog structure worked and has delivered this text. Actually, it has. That’s why the example is in Dutch, my apologies to any international reader. Note that this example is special. The participant of the experiment did not answer the questions as one would expect because of misinterpretation of a question, but the information he gives still fits the procedure model.

The expert is asked to focus on a FoCon called *Boodschappen beslissen*. This FoCon was named earlier by the domain expert in the general dialog phase. As this FoCon already implies a decision the facilitator skipped asking for decisions related to the name of the FoCon. “Can you name any decision that involves *Boodschappen beslissen*” is a redundant question so it is not asked again. The domain expert instead responds by giving criteria, not using decision options. Luckily the structure and facilitator are flexible enough. This confusion is not a large problem as the context gives away what the expert actually means and the facilitator recognizes this. Implicitly the domain expert means that he either does or does not add groceries to his list depending on the criteria he has just given.

Now this dialog log text is processed. First by applying pruning, by listing the elements marked during the dialog game. That yields a list of elements of interest to the facilitator:

- boodschappen beslissen
As one can see not all elements from the text are truly relevant. Patterning means looking at the response choice and previous responses of both parties in the interview. Applying that we can eliminate some “useless” elements from this list.

The decision criteria for example were named twice because the facilitator chose to repeat the names back in a structured Verb-Noun manner. The longer names can be eliminated from the element list. The result:

- boodschappen beslissen
- boodschappen
- behoefte
- hoeveelheid in voorraad
- hoelang moet je met boodschappen doen
- wensen vriendin
- correct
- voorraadbeheer

Also placing correct between hooks was a joke from the participant. By looking at his response choice from the dialog tree (patterning) we can see that his response was not of a valuable information giving type. Eliminating that element too leaves a shorter but more useful list of elements:

- boodschappen beslissen
- boodschappen
• behoefte
• hoeveelheid in voorraad
• hoelang moet je met boodschappen doen
• wensen vriendin
• voorraadbeheer

Sorting and rewriting turn out to be not necessary in this example case. Facilitator naming conventions for activities and FoCons have done their job during the dialog game.

Now for the last technique: argument type matching and tuple creation. The goal is to create a tuple that looks like this:

Decision \{ name, \{criterion 1, ..., criterion n \}, \{option1, ..., option n \}, argument type \}

We can identify the name as the name of the current FoCon in the dialog text, *boodschappen beslissen*. There are several criteria for the decision the expert makes in his process. They are in the text: *behoefte, hoeveelheid in voorraad, hoelang moet je met boodschappen doen, wensen vriendin*. These are put in the tuple as well as separate criteria. The facilitator can use his/her own judgment to remove criteria he or she thinks are unnecessary.

Now the facilitator has to identify the decision options to put in the decision tuple. In this case these are only referenced implicitly. The facilitator can look at the dialog text and conclude that the domain expert means that he either does or does not add groceries to his list. *Boodschappen toevoegen, Boodschappen niet toevoegen* are added to the tuple as decision options.

Now the facilitator has to decide which argument to assign to this tuple. The dialog text and resulting elements point him in the right direction: stock management or capacity. By looking at the list of formalized argument he can select the one fitting the description best.

Now this decision tuple is done and the facilitator can go on to the next.

9.5 The output step

**Goal:** Convert knowledge from the general form into a BPMN model.

The output step is more or less the most straightforward one. After the process step has analyzed and sorted all of the argument related information it gives a list of decision tuples. These tuples contain the necessary argument types, criteria and related activity names:

Decision \{ name, \{criterion 1, ..., criterion n \}, \{option1, ..., option n \}, argument type \} The goal is to put this in the right BPMN representation of that tuple.

Each of the formalized argument types has a predefined model snippet that belongs to it. An example can be seen in the image below.
Figure 13: A blank model snippet with no gateway type, labels and only two activities

This snippet then forms the basis for the split/join types and activities placement. In other words, a blank snippet is filled with the correct gateway type according to the argument type of the decision it is going to represent. Then it is given labels and the activities are named. When necessary the amount of activities in this snippet can be varied to match the amount that are present in the decision tuple from the process procedure step.

For example: Let’s review the parallel activity argument. The definition of the argument type tells us that all of the activities that belong together under this argument take place at the same time. The BPMN model should reflect that. This can be done by choosing the right BPMN split and join gateways, so that all activities must be completed before continuing the process flow. The resulting model snippet would look something like this:

Figure 14: An example BPMN model snippet for parallel arguments

When more activities are related in that way, more activity boxes and connectors can be added. The gateway type should remain the same.

Let’s take a closer look at the subset of BPMN that’s being used in this procedure to make a work-flow model. The elements that are being used hold certain knowledge that can be found in the elements that are distilled from the dialog game. Each of the elements has a certain knowledge value:

- **Events** The BPMN token set has a large number of events. But the ones most important for a simple flow model are the *start* and *stop* events. These do not hold explicit knowledge. But in order to place them one has to know the order in which things happen in the Universe of Discourse. The beginning and end are deduced from the order in which the domain expert has named and explained the various steps of his process.

- **Connectors** BPMN connectors carry very little informational value. In some cases they’re labeled to make the conceptual model easier to understand. When labels are used they add contextual information.
• **Activities** Activities are the describing elements in the BPMN set. They represent knowledge about actions in the Universe of Discourse. They usually contain a description in the form of *Verb-Noun*. The text has to describe that action at the right level of abstraction for the goal of the conceptual model.

• **Gateways** Gateways are the trickiest element to generate. They carry implicit information about relations between activities. They are often used as a trigger, and they signify a decision with a criteria that needs to be met. Gateway types mainly function as a type of constraint. The type of gateway determines in what direction the process flow continues. Needless to say that is a very important part of creating a useful flow model. All this knowledge again comes from the UoD, and it is of great importance that the right criteria and names are connected to the right activities.

![Figure 15: An overview of BPMN elements used in a simple flow model](image)

All these BPMN elements are present in the model snippet. Visually they are arranged in the right order, but they contain no explanation, names or specific types.

The goal of the output step is to give the BPMN elements the right “value” using the knowledge contained in the decision tuples from the previous modeling step. The information we have gathered from the dialog text is packaged into these tuples like this:

```
Decision { name, {criterion 1, ..., criterion n }, {option1, ..., option n }, argument type }
```

How do we get from that to a model snippet that represents the knowledge captured in this tuple? Each of the elements in the tuple will take a place in the model snippet. They will act as labels. Because earlier on the elements in the tuples were written in the Verb or Verb-Noun form this should fit well. Image 16 shows where the elements are placed.

![Figure 16: Label positions in the BPMN snippet](image)

1. The argument type that was assigned to the decision tuple dictates the type of BPMN gateway that should be used.
2. The options are placed in the activities that take place between the gateways. The number of activities corresponds to the number of options.

3. The decision criteria are placed under the gateway, as to show what leads to that type of decision.

4. Optionally it is possible to place the decision name over the model snippet.

Here’s an example.

Assume we have the following decision tuple:
Decision \{ Buy fruit, \{The fruit least in stock\}, \{Apples, Oranges, Bananas, Pears\}, Capacity management \}.

The capacity management argument means that the BPMN model snippet will look like this:

![Figure 17: The pre formalized model snippet for the capacity management argument type](image)

The argument dictates what split/merge gateway type is to be used.

We can then name the activities using the options from the decisions. We have four options, so we also have to expand to four activities in the BPMN example. It works well when we use the format Decision name + Option name. The result is shown in image 18

![Figure 18: The snippet expanded with the option names](image)

Now only the decision criterion is missing. This will make the example (a lot) more clear to unskilled BPMN readers. The decision criterion/criteria is placed in the label of the gateway like in image 19.
And now we’re mostly done! The next snippet can be chained to this one in place of the “stop” event.

Per default these snippets will be linked to each other in the order they are presumed to be happening. This is based on the order of which the various elements are mentioned during the dialog game. The order of these snippets might not be quite correct. During this output step the facilitator has to shift the snippets around when necessary.

Lastly the facilitator can name the model links when he feels that it’s needed. This is optional. The complete result will be named after what the domain expert has explained during the very first question of the dialog game.

9.6 Facilitator code of behavior

Before giving an actual implementation of this procedure in the next chapter, we’ve summarized how facilitator behavior can improve the results of the modeling method. To start with, a facilitator that will use the new procedure should behave and act like an ordinary (computer assisted) collaborative modeling facilitator. But there is more to effectively using the procedure than just that.

Recapping, according to (Niederman et al., 1996) (page 4) the tasks of such a facilitator are:

1. Plans/designs meetings
2. Listens/clarifies/integrates
3. Demonstrates flexibility
4. Keeps outcome focused
5. Creates open environment
6. Selects/prepares technology
While the approach and design of the dialog game takes points 1, 4 and 6 for itself and supplements some others, the facilitator still has tasks left. That means that a facilitator using this procedure still needs experience with setting up modeling sessions.

The meeting still has to be planned. The design however is in the dialog tree structure that for the most part is already complete. The selected technology is also a given factor in this procedure. The outcome is somewhat focused by the design of the dialog tree, but the facilitator is still responsible for some intricacies. More on this follows in a moment.

Based on the findings in the experiments and the literature that was studied we would like to bring forward a few points of attention per facilitator task.

**Listens/clarifies/integrates.** The facilitator needs to make optimal use of the questions that are available to him. Whenever the opportunity presents itself (s)he should repeat what the domain expert has said into a logical form. The goal is to keep the domain expert thinking in the right direction while showing that he was paid attention. By giving suggestions and examples the facilitator should restructure the information the domain expert has given on the fly. In this process it is very important to keep the questions relevant, embedded into the current context and as precise as possible to keep the knowledge flow going.

This specifically means that the example text in the procedure is *just an example!* Although the basic meaning of the sentence must be preserved, it is important to fill the blank spots with exactly the right things. Sometimes modifying the whole example is the only right way to go.

Furthermore, a facilitator should take his time and ask all questions available to him in the dialog tree. Only when (s)he is extremely sure, or when the desired answer has already been given can questions be skipped while keeping good results.

**Demonstrates flexibility.** A good facilitator should be able to deal with a wide range of situations. And (s)he should solve (or work around) all encountered problems as quickly as possible.

The procedure somewhat limits the response options. Whenever a difficult situation is encountered the facilitator should go to some extra lengths to get everything back on the rails. Using the open chat function can greatly improve results by going off the record when a domain expert is not thinking in the same direction as the facilitator is.

**Keeps outcome focused** The dialog tree uses FoCons to keep the outcome focused. So part of this task is already been done. But the effectiveness of these FoCons depends greatly on how the facilitator indexes, renames and structures them. It is important to take what the domain expert says, and convert that into a clear and describing Verb-Noun structure. Repeating back all the FoCons and selecting them one by one means that the expert has a good idea of where things are going and what (s)he has to do.

**Creates open environment** By creating an open environment we mean to make sure the domain expert can and will say anything that is relevant to the goal of the interview. That means leaving as many options open as possible while giving enough direction to get the right answers.

The dialog tree guides parts of this, but it is up to the facilitator to decide when to narrow down or widen the perspective. Careful use of the different interview phases is needed for this. There is no clear path to follow here, this comes down to experience and interpretation of the signals the domain expert is giving.
10 The prototype

This thesis will utilize a software prototype in trying to prove that a new modeling method is effective. The product of this work is an actual method that guides information scientists in developing a model from stakeholder information. But this method is implemented in software to alleviate the work load on the information analyst. The result is that the method corresponds very closely to the software prototype, as lessons learned from either are shared to the other.

It is the software that contains all the necessary steps for computer aided process modeling. Understanding the prototype is understanding the method and vice versa. For this reason the prototype is the leading factor in finding a working method to combine argument based modeling and dialog games.

This computer supported method of creating a model from information implies that software will do all of the work. Although every effort is made to make this statement true, the facilitator still plays a the most important role during modeling sessions. His or her experience is still needed for many steps in the method.

This chapter will serve as a log book and show case so that all important decisions will be traceable.

10.1 Design

Using the very simple input, process, output scheme we can identify three parts that need to be constructed. This abstraction helps in defining the necessary components for a prototype that embodies an end to end method.

- **The input** will be information from the domain expert.
- **The process** will be the theoretical tricks done to this information to make it more suitable for the necessary output. As the output has to contain only relevant information, the process needs a set of assumptions.
- **The output** has to be a BPMN model that fully envelops the input information.

Furthermore, the software tool needs to built around the two main concepts:

1. Dialog game theory
2. Argument based modeling

A secondary design concept to adhere to is “KISS”, or Keep It Simple, Stupid! The prototype does not have to do much more than prove the hypothesis that argument based modeling and dialog games can be combined by implementing them in software. KISS will ensure that the focus of the prototype goes to exactly that, and not the overly complex added functionality of the tool.

Lastly a few of Hevner’s (Hevner et al., 2004) design guidelines will be used to give focus in the design.
The dialog game theory determines that the input part of the software needs to be a dialog environment. As computer based messaging is not a very new concept - in fact it is rather old - dialog environments are plentiful.

The choice of what that should look like is rather obvious as chat box environments are not very complex and rather common. One example of a dialog game embedded in a chat box like environment is InterLoc (as mentioned in (Etienne & Hoppenbrouwers, 2011)). The look of the prototype will be a little like InterLoc. See image 20 for a screenshot of InterLoc. But there are differences. InterLoc only uses openers, and the prototype will use a mix between completely open text and completely predefined answer types. InterLoc is designed for modeling on the fly; while (at least the front end of) the prototype is only meant for gathering and pre-structuring information. Lastly the dialog part of the prototype will only accommodate two participants: the facilitator and the domain expert.

Figure 20: A screenshot of the InterLoc dialog software

The simple “chat view” and “message typing box” that every computer user has seen will be the core of the visible part of the prototype program. I have chosen not to adopt InterLoc or another existing alternative, with specific customization towards my own dialog trees in mind. The need for structured communication and dialog rules is not met by any commercially available chat programs. The logic for satisfying this need will be invisible for the users but is an essential part of the required prototype. This means that a new but simple looking chat box environment with support for structured communication is necessary.

Argument based modeling related features are also not necessarily visible for the user of the program. They however will seep trough in the form of rules for the communication. The ABM logic will be independent from the users as much as is possible. This essentially means that the
ABM logic will use the dialog results as input, but will not be actively triggered by either the domain expert or the facilitator.

Looking into several research papers from (Hayne, 1999), (Hoppenbrouwers et al., 2010) and (Niederman et al., 1996) some other basic concepts come to mind:

- Rule set for facilitator and domain expert customizable
- Automatic control questions
- Argument distillation, or at least elicitation
- Facilitator involvement (even if the program could be fully automatic, there is no substitute for people skills)
- Example sentences with blanks, again customizable
- Turn by turn communication
- Effective and fair solving of disagreements
- Clear role distribution
- Simple but effective interface
- Balance between guiding the user and freedom of conversation

While it would be a great feature to have support for messaging across large distances this is not a high priority feature for the prototype. It might be useful for the participants to see in which part of the dialog process they are as well. This is not a priority for the prototype either.

![Figure 21: An overview of the different components of the prototype software](image)

**10.1.1 Input**

The input stage of the prototype is where information is distilled from the knowledge of the domain expert. Several design choices are made here.
The communication has to follow certain rules to be structured enough to interpret by a relatively simple computer program. This is the reason why dialog games exist: to structure communication between agents, be it people, computers, or both. The computer program will follow a pre-programmed dialog structure based on the formal dialog game model created by (McBurney & Parsons, 2002). A dialog structure of this type gives form to the concept of applying rules to conversation. An added advantage is that the software can take care of most of discussion flow, while the Facilitator remains in charge when it comes to content. McBurney et al. make distinction between several forms of dialogs. The program will focus on so called information seeking dialogs. While other goals could be persuasion or choosing a best alternative, gathering knowledge is the most important facet here.

For good measure the following paragraph will summarize the workings of McBurney’s model. The model discerns different type of rules. For example, beginning, stopping, combinations, commitment and locutions. (see page 3&4 of (McBurney & Parsons, 2002) for a more complete explanation). Next to different categories of rules, the model also defines several layers of communication. The topic, control and dialog enable the separate different type of messages. Among others this useful for sifting modeling relevant information from control structure information. Lastly the paper defines communication as tuples of rules and content. Only slight modifications are needed to integrate argument based thinking into this formal message structure.

The above fits well into a tree-like model for structured conversation. Because a tree is relatively ease to make, use and understand it is selected as the main structure and information carrier for the dialog game.

The prototype software has to provide a framework in which the dialog game rule tree is used to regulate communication. That means that the dialog game software has to distribute the turns between participants, depending on what is recorded in the dialog tree. Following the tree nodes in the correct way is also the responsibility of that program. Image 22 shows the structure of the dialog tree nodes from the method description.

![Figure 22: The dialog tree grammar](image-url)
From that we can see that the dialog software needs to show both response types and example text. The difference between them is that the response type gives a general direction to the conversation, and determines what tree node should follow. Where the example text is the actual message the participant is sending. The example text should be editable, and to be able to use the processing and pruning should also contain <and>markings where the participant can enter his key concepts in the message.

What this is going to look like will be shown later on.

10.1.2 Process

Matching the argument types to knowledge from the domain expert is of course an important part of the process step. The workings are very intertwined with the act of getting information from the expert in the input stage. Examples of this are marking and Noun and Verb Noun restructuring during the dialog game.

The focus of the process design will therefore be on filtering and transforming the dialog text into something the output step can interpret. The information in the text will need to be matched to the available argument types from my earlier argument based modeling research. At the moment of writing there really are no documented ventures in this area so I will still rely on KISS and educated choice of methods.

It follows from the procedure description that elements need to be recognized and filtered, sorted and then listed for the information analyst. He or she should then be able to use these elements, together with the parts of the dialog log they come from, for creating decision tuples. Any support of software to make this job easier will help the information analyst work more efficient.

The patterning and marking techniques discussed in the procedure description should be applied automatically. During the dialog game phase the facilitator surrounds important elements with <and>. By selecting all words between these markings and putting them in a list, pruning by marking is implemented.

Implementing by patterning is more difficult. Interesting response option patterns depend on the dialog tree that is used. The facilitator or information analyst will have to identify all interesting response patterns by hand before the software can detect them automatically. For this reason automatic pruning by patterning is dropped from the prototype software. Luckily, the same can still be done by hand. For this reason response options should be visible together with the actual responses during the processing of the dialog log.

The prototype should provide a way to edit and sort the elements that were found. And it should support tuple creation. Preferably by selecting the elements for them or copy/pasting elements in there.

Each of the available variables in this tuples can (or has) to be filled with a core statement. These core statements are called elements and as discussed in the procedure section9 they usually look like Verb-Noun or Noun mini sentences. During the input phase the dialog game used brackets [,] and hooks <, >to delimit important statements. The software can apply to principle of pruning to effortlessly compile a list of things that were placed between those delimiters. The information analyst can then place these elements into the tuple structure.

Not much else is needed to implement the absolute basics of the method description.
10.1.3 Output

The last piece of the puzzle is converting the list of decision tuples with their arguments into the BPMN language.

To make this work means that every argument type should have a pre-formalized model snippet. Otherwise the argument can not be expressed in BPMN.

The software must also be able to match the model snippets to the amount of related options in the decision tuple. And, when the information analyst spots a mistake, it must be possible to correct it. Some graphic manipulation is needed for this.

The chosen solution is a preview field, where the user can look at the model snippets that are generated for each of the decision tuples. When all snippets are reviewed the software must write the complete BPMN model to a file.

10.2 Creation

While writing the prototype software several choices were made. These were made in line with the design goals as much as possible.

As an inexperienced domain expert has to work with the prototype, the user interface was the first component on the list. The goal was to keep that part as simple as possible, while still incorporating the required features. These features are shown in figure 23. A user can keep track of the conversation, as one would in any on-line chat program. What’s special is that the user first has to select a type of answer, before being able to actually free type. Also, free type is only relative, as the program will automatically give pre-structured sentence after the user selects a type of answer.
The second decision was made shortly after writing the invisible parts of the prototype program. At first it did not seem necessary to create the software in such a way that communication over networks or the Internet were needed. However, during very early user interface testing it became clear that two participants communicating with the program on one single computer will eventually start discussing outside of the software. The simplest countermeasure for this phenomenon is splitting the participants up physically. And that means that the software has to run on two different locations. This led to very basic networking support being introduced, together with a client and server model. That means that as long as an IP-address is open directly for the right connection, any computer can connect with the server.

The client/server model has some additional benefits. It enables the actual conversation logic and everything model related to run on a different computer. This reduces the risk of tampering or circumventing parts of the software. The actual logic can run on a computer that is easy to access for the facilitators, while not risking any access from other parties. Hopefully this means that the gathered information can’t fall in the wrong hands.

The third important decision is to use a tree representation of the conversation structure as used in (Gordon et al., 2007). A tree is a relatively easy but reliable way to make a computer program follow a path, while allowing for different options along the way. Most techniques for elicitation are easily converted to tree structures, which means that conversations are a suitable means for the goal of the prototype. The decision to use trees also means that the program is not limited to only one type of questioning. By designing and using the right kind of tree, the prototype program can be adapted for the required situation. Because this tree can be created relatively easy by people without experience in computer programming languages, the actual facilitating process is still in the hands of the facilitator, and not in the hands of the programmer. More information on the syntax for these trees can be found in chapter 11.3.1.
The process and output steps of the procedure were less challenging to get right. And less effort was put into them, has the key goal is to prove that argument based modeling and dialog games have a future in a modeling procedure. One by one the techniques the procedure describes were implemented. When a feature proved to be too time consuming, challenging or not directly needed, it was omitted.

As said before, the goal was to keep the tools as simple as they can be. An effort was made to get as much relevant information in the user interface as was possible. Lists of various sorts are used to keep track of what is relevant and what is not.

A few preliminary screen shots of the tool that is made for facilitators to create and assign decisions to arguments are show here. More about the workings of the prototype tool-set can be found at the end of this chapter.

Figure 24: A screenshot of the UI for the facilitator in the processing phase of the procedure
10.3 Testing

The prototype was tested during experimental sessions. The prototype software had only minor problems. Most of those were not fixed because they did not obstruct the correct working of the modeling procedure.

The dialog tree however was fine tuned after each round of tests. A better dialog rule set will lead to better results throughout the whole modeling procedure. This is probably because more knowledge is extracted by use of a good rule set, and that knowledge then needs less work to be of any use.

It turned out this tree component deserves most of the attention during the prototype development. That is why most of the finding below are related to the dialog tree.

1. **Round 1**
   Before testing an initial conversation tree was created. It is closely modeled after some of the resulting models from my previous thesis. (Thijssen, 2011) This tree served to try out the dialog software. It was found satisfactory apart from some slight non blocking glitches.

2. **Round 2**  
   It was found that during the very first ‘real user’ testing the prototype program itself would perform well enough. The conversational aspect of dialog games and argument based questions seemed to work. The same could not be said for the dialog tree that was used in this phase. The amount of response options was found lacking. It was difficult for the test subjects to determine where in the tree the conversation was. A second problem was that the tree did allow for describing process steps that were composed of sub activities, but it did not allow for singular process activities. The dialog tree for initial testing was not intended for real experiments, but some valuable lessons were learned.

3. **Round 3**  
   After modifications to the dialog structure used in round 2 a new attempt was made. While there was some improvement there were still many major flaws in the design of the
dialog tree. The test subject in round 3 was especially good at verbalizing the problems he encountered. Together with the notes that were taken during the interview this gave valuable insight in points of improvement. The subject found the level of abstraction required difficult to establish. And the facilitator has little ways of giving examples to clarify things. Some argument based question were perceived too vague as they were meant for too many different contexts. All in all, the concept showed promise but there were too many obstacles in the dialog set up at that moment.

4. Round 4 \( n=2 \)
The dialog structure was rewritten to follow a new but simpler design. In short: FoCons together with separate interview phases were introduced to help determine the level of abstraction. The result of this new design was a lot better. But while the interview went more smoothly the level of pre-structuring declined. Also there was some ambiguity when it came to the naming of different steps from the example case. At one point facilitator and expert were talking about different things and didn’t know about that until after the session. One of the subjects, more business oriented, noted that some steps felt repetitive as they asked for things he had already talked about in free type mode. During the two separate sessions decisions were implicitly named but deserved some more attention. Some things went rather well. The level of abstraction was almost exactly where it needed to be, mostly thanks to the FoCons. The option for open discussion solved many smaller problems. The results were more promising and some fine tuning might get the troubles sorted.

5. Round 5 \( n=2 \)
After some further tuning of the dialog structure another session was held. The main change was in the way decisions were approached and later assigned to a specific argument. The test subject did not perceive many problems, although some questions were not understood immediately. These problems were then discussed outside of the software, while that could have been done inside the dialog. In some instances responses differed in detail from what the facilitator suggested, but I was able to restructure in such a way the meaning was captured. The largest downside now is the repetitiveness of some of the questions. Upsides are that the order of elements that were described are now chronological by default. The abstraction level was mostly on point. The decision mapping approach in many cases gave direct hints toward the type of argument that would fit it’s description better. The facilitator has to be careful to do the element naming as the experts use different names for the same thing. But this was found to be easily manageable.

10.4 Adjusting

Each round of tests results in major or minor tweaks to the prototype program. This subsection will briefly discuss each adjustment.

1. Round 1
After round 1 some minor changes in the layout of the UI were made. The text fields were made a little wider to accommodate longer sentences. The dialog tree was checked for spelling errors, not much more was added.

2. Round 2
After the second round of experiments it seemed that the dialog tree approach showed
great promise. More response options for both facilitator and expert were added to increase expression room. It is important to note that this tree was fully closed with no option for open communication. The option to ask for singular process activities was added as well. Some example texts were rewritten a little so that the expert would have a better idea of what was asked, and what questions could be expected next.

3. **Round 3**
   When testing round 3 was well and done it was clear that the fully closed tree was too rigid. It could not be made flexible enough to accommodate all levels of abstraction. Successive questions were difficult to adapt for coherence. So the very long single stage tree was ditched and new design was made. Now three main levels of detail were separated by a clear description of what was to follow. FoCons allow to take only a part of the Universe of Discourse to the next level of detail. Each level or phase was iterative with its own goal, meaning that the dialog goes back and forth between expert and facilitator until the phase goals are met. Each phase has room for open conversation but still gave the option for closed predefined responses. After naming the model the open discussion phase was started. In this interesting topics were accumulated. Then each of these topics was made a FoCon and in the second phase relations describing this FoCon were mapped. During the mapping of relationships arguments formed the basis for the type of relations suggested and mapped. In the last phase some control questions made sure the right pieces of knowledge were collected. All of the phases allowed for suggestions containing to the point examples that would dictate the level of abstraction the expert was thinking at.

4. **Round 4**
   Following round 4 the design from round 3 was polished. The relation mapping phase was dropped in favor of decision mapping. Relations became clear during questions related to decisions, and sometimes even in open discussion. The control question phase was changed to the argument phase where each decision could be categorized using one or more arguments. Control questions turned out to work better mixed in with the interview in stead of after and are now mixed in with the questions of each phase. Lastly, the dialog text was translated into Dutch. As all participants are native Dutch speakers this helps in conveying small nuances better.

5. **Round 5**
   Round 5 gave few new insights. The subjects found the dialog pattern repetitive but understood why it worked in that way. This gives a reason to rewrite some expressions to “lighten up the mood”. I think a rewarding system that motivates the expert to sit through until the end of the session would be a good idea, but is not necessary for the experimental set up.

10.5 **The resulting prototype**

This section is meant to show the resulting prototype and instruct on how it can be used. The prototype was written in Java. The software is split up into three parts, just like the modeling procedure. Namely: input, process, output.

There are a few advantages to this choice: the domain expert is only needed in the first part, and won’t be confronted with any of the rest. It allows the facilitator and/or information analyst to divide the work into pieces as well. That means that a facilitator who isn’t responsible for
the rest of the modeling process can pass on the results to someone who is after the first stage. Lastly it makes possible that each of the three parts of the prototype can be improved without having to change the other ones.

10.5.1 Input: the dialog game software

The first part of the prototype software has two separate programs that work together. There’s a server program that should be started first. The server program reads a dialog tree file and will distribute the turns between the clients. There is not much more to it, the server instance does not have a GUI, and is only meant to act as a fair referee enforcing the previously established dialog game rules.

The dialog tree strictly speaking is not part of the prototype software. But it does have a very large influence on its workings. It contains all the dialog rules, contained in nodes that store response options and example texts. The dialog tree is a complex thing, and its development and evolution have been discussed before. The dialog tree rule set in the form of that text file determines everything the dialog participants are going to see and do. Creating a decent rule set with just the right response options and examples is work for an experienced facilitator, not for a computer programmer.

The dialog rule set is a file that is made up from textual representation of the tree nodes. Here is an example of one rule node:

```
Step=2
Role=F
QuestionOption =Vraag een naam voor het proces
QuestionExample =We beginnen met het vinden van een naam voor wat je wilt beschrijven. Wat gaan we bespreken? Bijvoorbeeld: Klanten screenen en registreren
3
QuestionOption =Proces naamgeving overslaan
QuestionExample =Laten we beginnen met het interview
5
```

It contains five key components:

1. A step number. This is a label that the dialog software uses to find this tree node.

2. A role identifier. F for facilitator, E for expert. The software uses this to give the turn to the right participant for that tree node.

3. An unrestricted number of question options (named response options throughout this paper). These appear in the dialog game interface, but cannot be changed.

4. An example text, related to this question option. This appears in the dialog game interface, but may be edited to convey the actual message the participant wants to send.

5. A corresponding node reference. This number is the label of the tree node that should follow after that response options was selected by a participant.

Note that the example text contains - as the name implies - as many general examples as possible. The facilitator and domain expert can change these examples to suit the situation
that is being discussed at that point in the dialog game. It also contains marking signs such as \textless \text{and} \textgreater. The participants can (and should) enter their important elements between them so that they are marked for later use. Whenever the domain expert fails to do this properly, the facilitator should do this for him and repeat back the marked elements for validation.

The full tree can be found in appendix B.

The client software is the visual representation of the dialog game. Two instances are started, one for each of the participants of the dialog game. When it is started the user can enter his screen name and a role: expert or facilitator. Note that there must be one of each for the dialog game to function correctly. When both are present the dialog game is started automatically.

The picture below shows what the user interface of the dialog client looks like.

![Figure 26: A screenshot of the UI for participants](image)

The participant can:

1. Review the conversation history
2. Select a response type (only when the participant has a turn)
3. Then review or edit the example text associated to that response type
4. Send his or her message to the other participant (again, only when the participant has received the turn)

In this way the participants of the dialog game can exchange messages back and forth. The general three phase structure keeps the conversation on topic and focused. The response types steer the general direction of the conversation. Only the actual response text is shown in the
history view. All is logged however, so that they can be used in the second part of the prototype program.

10.5.2 Process: filtering, sorting, restructuring and creating decision tuples

The second part of the prototype is designed with straightforwardness in mind. It implements the three features mentioned in the procedure section of this paper:

1. Pruning (mostly automatic)
2. Sorting and editing
3. Tuple creation and argument matching

How? Shown beneath is the user interface of the processing software.

![Figure 27: A screenshot of the UI for the facilitator in the processing phase of the procedure](image)

Let’s take a look at the various options this gives the facilitator.

Pruning (by marking) is done automatically. When the program starts it reads a dialog log and goes through it in a matter of milliseconds. All the marked pieces of text are then put in a list in the upper right corner of the interface.
Patterning is not automated here. It can still be done by hand because the corresponding dialog choices are shown between the replies. By investing some more time it is possible to automatically highlight important parts of the dialog that contain information about the decision tuple that is created at that moment.

For the information analyst this is the result:

![Figure 28: A screenshot of the list of elements that was automatically created from the dialog text](image)

To the left of image 27 the dialog log is shown. The buttons above allow the user to step back and forth between statements in the dialog log. Using the scroll bar for this is also possible. The chosen response types are put in between the text responses as well, so that the information analyst has a better understanding of what leads up to a certain statement. This, for example, can be used to identify new important elements that were missed during the dialog game.

Of course not all of the elements found during pruning are in the correct naming convention. The information analyst can delete irrelevant elements, rename incorrect named ones, and add new entries with the help of the dialog text. He or she can also change the order of these elements. That’s the sorting and editing taken care of.

Lastly, the procedure dictates that we convert the knowledge in the text into tuples looking like this:

Decision \{ name, \{criterion 1, ..., criterion n \}, \{option1, ..., option n \}, argument type\}

The elements identified earlier then form the basis of the actual tuple creation step. When the facilitator feels all the elements he needs are present he can start creating “decisions”. By following the dialog text (visible to the left in image 27) he can fill out a form that essentially translates the typed elements to the decision tuple.

An example of the decision creation menu is shown below in image 29.
Which element goes were can be quite easily established by following the dialog structure, as each of the information elements is asked for separately. There are response options that are labeled “give decisions criterion” and such, and looking at these will immediately give the required information. By using patterning (or simply looking at the response type) it’s quite easy to establish which element fits where. Assigning an argument can be done by selecting the best fitting one from the drop down box at the bottom of the form. The facilitator or information analyst have to use their own skill for this.

When all decision tuples are created they can be exported by pressing the “export button” at the top of the window. The exported file can the be used to create a BPMN model by using the third part of the software.

10.5.3 Output: writing to BPMN

The prototype tool set also implements the output phase of the procedure. The support is quite basic and simple as little of the research weight is put into this part. Not to mention that with the abstractions, linkages and assumptions I have already made the printing process is quite straightforward.

Image 30 show us the user interface of this tool. Again KISS is applied by only implementing the minimum of needed functions.
The facilitator can change the order in which the decisions are taken by clicking the respective “up” and “down” buttons above the decision list shown left. This of course affects the order in which the decisions will be linked in the BPMN representation. Usually this is not necessary. But when the natural ordering that follows from the question structure during the dialog game is disturbed the information analyst can correct the problem.

A preview of each of the decisions will be generated based on the arguments, options and labels of each decision. This is shown in the middle. By using the buttons above the preview panel, it is also possible to add or remove more activities between the two BPMN gateways. The facilitator can then make small adjustments when he feels they are necessary by typing them in the preview screen. Those are then committed by clicking “confirm preview”.

By clicking the export button the tool then generates the complete model in the ASCII tokenset (for simplicity). And that concludes the procedure! A BPMN model generated from the user input is available for use.
11 Results

After drafting a procedure, then implementing and testing it we have now arrived at evaluating the results. The result come from the experiments that were held with test subjects from all sorts of backgrounds.

The experiments have yielded several types of results. Let’s take a quick look at them.

1. The most important result is a new type of conceptual modeling procedure.
2. The second is a working prototype of this procedure, including a working dialog tree.
3. More arguments needed for complete modeling.
4. There are results from using the modeling procedure and the prototype in an experimental setting.
5. The participants of those experiments have expressed their opinion in a small questionnaire after working with the procedure.

One could say the modeling procedure as it is described in the chapters above is the foremost result. Yes, a method and prototype were created before the first of the experiments. But the feedback and observations gathered during these experiments have made the method and prototype into a working example.

During the experiments the original list of formalized arguments was not complete enough to describe everything in the example case. This led to the analyses of the arguments on the original list. Some were changed, and others were added based on what needed to be described.

The last two types of results are more directly linked to the experimental sessions. One is a “qualitative”, the result from applying the modeling procedure on a case and then trying to build a BPMN model with it. And one is “quantitative”, numbers about the quality of the experiments.

11.1 The method and prototype

The procedure and the prototype are very much two separate products. This is because the prototype is one and not necessarily the way to implement the conceptual modeling procedure that is being proposed. Yet because of the strong relation between the two I will treat the results of the prototype testing as an indicator for the procedure’s potential.

After many refinements the result of all of the effort consists of a beginning-to-end tool-set that guides and supports the beginning-to-end procedure of conceptual modeling. A facilitator or information analyst can use this tool-set to interview a domain expert with a consistent structure and quality. An example of this is show in image 31
Figure 31: A screenshot of the dialog game UI for participants

Furthermore the prototype offers guidance and support in the process of converting dialog text to another more abstract form: decisions and arguments. The facilitator can more easily recognize important elements from the dialog text, and use them in making sense of the desired situation. For an example of this, see image 32 and 33.
Figure 32: A screenshot of the UI for the facilitator in the processing phase of the procedure. Left: the dialog text. Top right: element list. Bottom right: decisions mapped so far.
Lastly, the prototype converts the structured information to a conceptual model with minimal effort.

The facilitator can shuffle around the order of the decisions. He can also edit the previews if he
noticed mistakes, or just wants to add a note.
In short, the working tool-set is a result by itself.

11.2 Arguments

During the experiments a number of explanations resulted in modifying or adding argument types. The list is now composed of the following argument types:

- Parallel alternatives
- Equal option alternatives
- Independent acting parties
- Capacity management
- Risk management
- Competence
- Preference
- Cost effectiveness
- Time effectiveness
- Resource dependency
- Satisfaction
- Availability

The arguments all imply a certain split/join structure with a specific gateway type in the BPMN language.

For easy use the formalization of the existing arguments and new arguments was simplified somewhat. Each corresponding model snippet was changed in such a way that only one process flow exit is present. This is because the prototype (in contrast to the procedure) can not handle multiple flows and nested activities.

The use of the argument type is somewhat restricted by this, but the core feature of providing the right split/join gateway type remains intact.

An example of a “blank” model snippet can be seen in image 35.

![Figure 35: A blank model snippet with no gateway type, labels and only two activities](image35.png)
For more information on the process of formalizing more argument types one can fall back on my bachelor’s thesis (Thijssen, 2011).

11.3 Dialog tree results

The experiments resulted in different conversation trees. The grammar that was created for this tree was not changed in any major way.

An actual result - like the modeling method description and prototype is the contents of the dialog tree. A simple tree was tested. Then a very complex and tedious tree was made to support the argument based modeling approach, but this failed to work well.

Eventually a dialog rule set was found through evolution. It was tested, improved and when the results matched the expectations finalized. This version is the tree as described in the prototype 10 section of this thesis. It is also added for reference in the appendix in B.

11.3.1 Dialog rule set grammar

The tree syntax is simple but quite effective.

1. Step=\(n\), unique number of this tree node
2. Role=\(F\) for facilitator’s turn or \(E\) for domain expert’s turn
3. QuestionOption = Any text describing the overall question direction/meaning, possibly with a word of advice
4. QuestionExample = Actual example text, complete sentences with <and> as blank spaces for input
5. \(x\), the number of the node to go to after this question or answer has been given.

Note that each step can have as many QuestionOption, QuestionExample and \(x\) combinations as is necessary. The tree may also contain cycles. Allowing cycles keeps the tree much more compact but can introduce problems when the question options are not prepared for this. It’s also possible to have one participant take several turns as long as the next nodes are also destined as a node for that participant.

Here’s an example for one question with two separate options for a facilitator.

Step=5
Role=\(F\)
QuestionOption = Agree with name
QuestionExample = I like that name!
5
QuestionOption = Disagree with name (give suggestion/instruction)
QuestionExample = Could you rephrase that name? For example: [...]?
3
11.3.2 Dialog tree contents structure

The creation (and maybe even the appearance) of this tree may seem quite simple. But don’t let looks deceive you, fine tuning this procedure implementation to a usable level took well over 80 hours. This is excluding the time it took to go through the experimental sessions and reviewing those results. Perfecting it will take even longer but will yield better results in return. More on suggestions for this can be found in chapter 13.

During testing it was found that the level of abstraction is best regulated by using different layers of interview. These layers are called “interview phases”. Each phase phase after the open and general phase zooms in further on the subject matter.

Testing the several versions of the tree contents and tree structures resulted in seven dialog logs. Two of those are in the format that is dictated by the procedure as it is described in this work. The others were made with an older less successful version. The resulting logs are quite long and therefore not included in this paper, but they can be found on the software disc.

11.4 Resulting models

The two “modern” logs were used for the rest of the procedure. One was put trough the process and output stages of the tool set.

The other session was a little too short for this to provide a good looking result. This results in the following model:

-x-(wat is er nodig?)--toevoegen aan boodschappenlijst-x--
    -niet toevoegen aan boodschappenlijst

-x-(prijs producten)--speciaalzaak-x--
    -supermarkt
    -markt

-o-(beschikbare tijd)--auto-o--
    -fiets
    -bus
    -lopen

-x-(beschikbaarheid kar)--kar gebruiken-x--
    -kar niet gebruiken

11.5 Research variable count

During the last two “final version” experiments the following counts were made for the research variables:

• Efficiency: The total number of facilitator-system-subject interactions needed. 98 and 100.
With the note that very little typing had to be done on account of the example text.

- **Efficiency:** The amount of domain expert (or subject) actions needed.
  
  41 and 41.
  
  With the note that both participant did about the same amount of case exploration, about two thirds of the complete process.

- **Quality:** The amount of facilitator interventions needed
  
  4 and 2.
  
  Most interventions were questions the participants asked about how to interpret a question, but only because they were in the same room. The interventions were non critical.

- **Quality:** The amount of interventions needed outside of the system
  
  None and none.

- **Quality/Efficiency:** The grade facilitators and subjects give to the procedure.
  
  See the questionnaire results 11.7. In the open questions the method was received generally well with little adoption problems.

- **Completeness:** The score of how well generated BPMN models cover the gather information.
  
  None of the two experiments describe the whole case because of time limits given by the test subjects. However, the procedure appears to be working well in finding all important decisions. Minor are not always picked up, but one can also claim these were dropped by abstraction. That means the method automatically filters irrelevant decisions, which is a good thing.

- **Completeness:** The amount of arguments or dialog rules found to be missing after comparison to the case.
  
  Some were missing, but about five new ones were discovered. In general most situations from the example case can now be described by the current list of argument types. Continued use of the procedure might yield more.

- **Completeness:** The grade a test subject gives to the amount of their information given.
  
  See the questionnaire results 11.7. Almost every participant felt they had enough freedom to share information they had.

These will be used to answer part of the research questions per the operationalization.

### 11.6 Observations

During and after each of the experimental sessions notes were kept. From this we can derive a number of unquantifiable results. These are addressed in this section.

All of the participants had no trouble adapting to the two-stage answer form in the dialog game interface (first selecting answer direction, then editing the example). This is probably thanks to the incorporation of chat messaging into the society, and that’s beneficial for this approach.

Because experiments were not held in separate rooms participants sometimes tried to explain or ask things outside of the dialog environment. Sometimes they needed a little help, which was fine. But modeling related questions outside the environment were ignored best as possible. The improvements in the dialog tree were noticeable as problems, facilitator interventions and questions declined gradually.
The participants were always keen to start and generally liked the controls and questions. Most sessions were about an hour long. At that point the interest declined somewhat because of the perceived repetitiveness of the procedure.

11.6.1 What’s good

Some key points that seemed to go rather well during the experiments and subsequent use of the tool-set are:

- Participants picked up the dialog game workings quickly
- The order of elements and steps seemed to come naturally
- The example text stimulated participants to put elements they thought important between the markers <,> themselves
- Processing the dialog log and creating a model took little effort
- The analyst did not seem to lack major knowledge from the expert during the processing step

11.6.2 What’s not so good

Some key points that could use some improvement during the experiments and subsequent use of the tool-set are:

- Participants were left with some sense of repetitiveness
- Going through the process of modeling even one decision takes about 20-25 minutes
- The BPMN writer software is rather limited by the ASCII representation of models

11.7 Questionnaire results

Six different participants worked with me in the experimental session to create seven dialog logs. Before and after each session they filled out a questionnaire.

The first questionnaire is meant to give insight in the competence of the participant and their experience with conceptual modeling and models. The latter one was used to measure the participant’s view on the modeling procedure after participating.

It is important to review the latter inquiry in light of the first one because of skill level and related problems or successes. The table below shows the self assigned skill level in the second column followed by the grades given to the main goals of the procedure.
<table>
<thead>
<tr>
<th>Participant</th>
<th>Skill (0-5)</th>
<th>Quality (0-5)</th>
<th>Efficiency (0-5)</th>
<th>Completeness (0-5)</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stefan Joosten</td>
<td>4.2</td>
<td>4.0</td>
<td>4.3</td>
<td>4.5</td>
<td>4.3</td>
</tr>
<tr>
<td>Femke van Egteren</td>
<td>1.4</td>
<td>2.5</td>
<td>4.0</td>
<td>3.0</td>
<td>3.2</td>
</tr>
<tr>
<td>Pim Sewuster</td>
<td>4.4</td>
<td>3.8</td>
<td>3.7</td>
<td>4.5</td>
<td>4.0</td>
</tr>
<tr>
<td>Erik Thijssen</td>
<td>3.0</td>
<td>3.3</td>
<td>2.7</td>
<td>3.5</td>
<td>3.2</td>
</tr>
<tr>
<td>Bas Gerritsen</td>
<td>3.2</td>
<td>3.7</td>
<td>3.0</td>
<td>4.0</td>
<td>3.6</td>
</tr>
<tr>
<td>Femke van Egteren</td>
<td>1.4</td>
<td>4.7</td>
<td>4.7</td>
<td>3.5</td>
<td>4.3</td>
</tr>
<tr>
<td>Stefan Joosten</td>
<td>3.8</td>
<td>4.3</td>
<td>3.7</td>
<td>4.5</td>
<td>4.2</td>
</tr>
</tbody>
</table>

The scoring above is calculated by taking the average of that category of questions. Grades are rounded to one tenth of a measure.

The participants were also allowed to give suggestions during and after the experiment. These results are not strictly related to the grading of the procedure. Because I feel they do give an unofficial impression of the experimental results I’ve summarized them nonetheless.

During the first three sessions comments were mostly aimed at the questions that the facilitator asked. They were too rigid. They were not specific enough for every situation the domain experts were describing. The goal of the session was not clear enough to some participants.

The following two session participants mainly commented that the pattern was repetitive, and that they had to repeat some answers they gave to make things fit into the procedure.

The participants of the last two sessions were using the “final” version of the prototype. There were less comments here. Only that the process felt somewhat repetitive still.
12 Conclusion

First an answer to all of the subquestions that were posed earlier on.

1. *Is it possible to integrate fitting interview methods in a dialog game?*
   Yes, this is possible. The input stage prototype toolkit is evidence of this. The result is in the form of a tree like dialog structure containing the right type of questions and answers for both facilitator and domain expert.

2. *Is it possible to (partially) structure this information, using argument based modeling?*
   Yes, this is also possible. The process stage of the prototype toolkit has implemented this successfully. The result is a small program that allows for easy translation between domains by the use of argument types and the right kind of dialog structure.

3. *Can software (partially) create models from these arguments using my previous research?*
   The output stage of the prototype procedure toolkit demonstrates that this is also possible. The result is a rather simple model structure containing the right kind of junction and merge gateways.

4. *Is the procedure complete and functional?*
   All other milestone questions were answered with “yes”. This gives a positive feeling for the answer to the last and most complex subquestion. With some weak points (or rather, points that deserve more attention in following research) the procedure is deemed to be complete and functional. It’s possible to create a BPMN model with the use of the procedure, prototype toolkit and a domain expert.

And now for the top level research question.

<table>
<thead>
<tr>
<th>Is it possible to make a viable modeling procedure by combining argument based modeling with dialog games?</th>
</tr>
</thead>
</table>

Looking at the results of the subgoals I would say the answer to the main research question is a *yes*.

Splitting up the research question in two simpler parts enables us to clarify this.

1. *Is it possible to make a modeling procedure by combining argument based modeling with dialog games?*
   Yes it is. For each of the three parts of the procedure (input, process, output) it was possible to create an approach and a tool to support this. This is supported by subgoals 1, 2 and 3.

2. *Is this procedure viable?*
   Judging the viability of the procedure is almost as difficult as creating it. The more complex subgoal 4 is a testament to this. The results of the experiments suggest that every new revision of the prototype performs better than the last one. The number of unexpected and unintended events outside of the modeling environment was also declining. At the end of the experimental sessions both the participants and the facilitator thought the procedure performed relatively well for the example case. This leads me to conclude the procedure is in fact viable and should be tested and improved further.
Both questions combined indicate that dialog games with argument based modeling integration offer a interesting new opportunity. Before using the procedure in production circumstances I would first improve some features. How that could be done is written in the next chapter.
13 Future research

Looking back now we would say there are two possible approaches to continuing this research. One can focus on expanding and improving my theoretical procedure, and one can choose to improve the implementation.

13.1 The procedure

The goal of this thesis was to prove that the dialog game/argument based modeling combination could form a complete and viable procedure for conceptual modeling. This was proved by drafting the procedure, implementing it and testing it with some successful results.

But to really say more about the success of this procedure it would be a good idea to fine tune it more using a very much larger group of test participants. It would then be possible to statistically say things about the success of the procedure regarding skill level or occupation of the participant. One could also choose to select a certain target group for the procedure and see if it works better when tuned to this. I think it would.

Another possible line of research is comparative research. For example: compare this procedure to ordinary or classic conceptual modeling using the same quality, efficiency and completeness indicators. Improvements and faults would become much more evident when the two are compared.

The last suggestion I would want to give is to implement the third goal element: participant engagement. There is no real reward element in the experiment set up so far, and adding this might improve results. For example: award score points for each useful reply or topic the domain expert can give. At the end let the participant choose something from the soda or snack machine from these points to promote for gathering them.

13.2 The implementation

The software prototype is only a very simple implementation. Its development was restricted by time, my limited skill in computer programming and resources. When the focus is not on improving the procedure but the implementation, a simple but well working tool could be developed.

My first focus would then be expanding the tree and its options. Most of the research in this thesis was focused on this dialog tree and most of the gains are to be had there as well. During my experiments the largest remaining problem was the repetition the participants felt. By adding more or different questions this problem could be reduced. But too many options will over complicate matters.

The much needed specificity of the example questions had to come from the facilitator adapting the example text to the current context. Perhaps creating different dialog trees for different situations is a solution for that sort of problem, effective when interviewing many employees with the same background and function. In short: the core of the implementation is the dialog tree. Improving this in any way will be a direct gain in effectiveness and quality of the method.

Adding support for embedded activities (i.e. a decision in an activity) will give some more modeling freedom. This was omitted in the prototype to reduce complexity.

The restructuring tool could be improved by making elements drag and drop compatible. A
facilitator could then drop elements in the desired places of the decision form and do his work much quicker. Patterning is not really intelligently supported by this tool yet. Mapping and implementing all the interesting question combinations can improve automatic parsing of the dialog text by a large amount. It is however quite time consuming.

Lastly the BPMN writer tool could be hugely improved by adding graphics in stead of ASCII representations. The current BPMN writer was meant as a placeholder to just demonstrate that writing decision tuples is in fact possible. Most of the functionality of this tool was not implemented yet. Doing that will be a good start. Implementing some more model builder functions so that the facilitator can more easily edit and verify the generated examples can save time.
14 Glossary

- **Argument** A statement meant to support a position taken by the domain expert

- **Argument type** A more general category description in which arguments can fall, which allows for further abstraction

- **BPMN** Business Process Modeling Notation, a standardized symbol set from the OMG group for drawing flow models of business processes

- **Dialog rule set** See dialog tree, with the inclusion of some extra assumptions such as the amount of dialog participants and turn taking

- **Dialog tree** A text file that contains question types, question examples and role distribution in tree nodes

- **Example text** A stub containing most of the sentence structure of the actual message that is going to be sent to the other participant during the dialog

- **Domain expert** A person with knowledge of the specific situation that is relevant for making a conceptual map

- **Facilitator** A person skilled in supporting the process of information gathering

- **Information analyst** A person skilled in the task of information restructuring transformation. Often the same person as Facilitator, but not necessarily

- **Information technology** Technical tools such as digital hardware and software that facilitate the storage, exchange and use of information within an organization

- **KISS** Keep It Simple Stupid!, a rather basic design philosophy dictating that anything complex or irrelevant to the core function of a computer program should be omitted or hidden

- **Model** A conceptual and formal representation of a real world aspect or artifact at a certain level of abstraction

- **Model snippet** A small piece of model, containing the correct gateways and activities for exactly one argument

- **Organization** A social unit of people that is structured and managed to meet a need or to pursue collective goals. All organizations have a management structure that determines relationships between the different activities and the members, and subdivides and assigns roles, responsibilities, and authority to carry out different tasks. Organizations are open systems—they affect and are affected by their environment

- **Response option** A piece of text summarizing the type of response in the dialog that is being chosen

- **Universe of Discourse (UoD)** All concepts, relations and events that are relevant to the world in which the domain expert plays his role
References


A Introduction argument based modeling using dialog games

Note that this introduction is in Dutch as all participants in this research were native Dutch speakers.
1 Inleiding

Beste deelnemer,

Hartelijk welkom bij het de testsessie van mijn prototype. Allereerst wil ik je graag bedanken voor het deelnemen. De resultaten zullen mij en mijn vakgebied helpen om betere procedures te ontwikkelen voor het informatiekundige werk.

Voor we gaan beginnen zijn er eerst een aantal persoonlijke gegevens nodig, die later helpen met het verwerken van de informatie. Na deze vragen volgt een korte uitleg over de sessie en de gebruikte software. Afsluitend worden er nog een aantal vragen gesteld over het verloop van de sessie.

2 Persoonlijke gegevens:

1. Voornaam en achternaam: 

2. Leeftijd: ____

3. Wat is je opleidingsniveau?
   □ MBO
   □ HBO
   □ WO
   □ Anders, namelijk: 

4. Wat is je belangrijkste betaalde functie?: 

5. Bij welk bedrijf of welke organisatie vervul je de bovenstaande functie?:

6. E-mailadres: 

3 Evaluatie van je voorkennis

7a. Je kennis van computers   geen □□□□□□□□□□ erg veel

7b. Je kennis van formele logica   geen □□□□□□□□□□ erg veel

7c. Je kennis van procesmodellen   geen □□□□□□□□□□ erg veel

7d. Je kennis van modelleerprocessen   geen □□□□□□□□□□ erg veel

7e. Je kennis van chatprogramma’s   geen □□□□□□□□□□ erg veel
4 Een korte omschrijving

Door mee te doen krijg je de kans om een expert te zijn op het gebied van een proces dat plaatsvindt in je dagelijkse leven. Een expert heeft doorgaans veel kennis van een bezigheid. Maar hij of zij heeft misschien moeite om deze kennis op een volledige manier over te dragen, zodat er een goed procesmodel van gemaakt kan worden. Een zogenaamde facilitator kan dat wel. Maar deze heeft de situatiekennis van de expert - jouw kennis dus - nodig om het benodigde procesmodel te maken. Daarvoor is een nieuwe methode ontworpen.

Het experiment waaraan je mee gaat werken dient dan ook om aan te tonen of deze nieuwe aanpak voor het inzamelen van gegevens en daaruit opstellen van procesmodellen levensvatbaar is. Dat wil zeggen: het onderzoek zal aantonen of de methode een goede kans maakt voor doorontwikkeling. Je gaat helpen door met behulp van een gereedschap die de nieuwe methode ondersteund informatie te geven over een bekend process.

DIT PAPIER GEEFT JE EEN GEBRUIKSAANWIJZING VOOR HET MEEDOEN AAN MIJN EXPERIMENT. LEES HET A.U.B. ZORGVULDIG DOOR.

5 Wat is een procesmodel?

Een procesmodel, ook wel bekend als flowmodel, is een aaneenschakeling van activiteiten. De activiteiten zijn de stappen van het proces. De aaneenkoppelingen en volgorde beschrijven hoe het process wordt uitgevoerd.

Hier is een plaatje van een procesmodel om je een beeld te geven van waarvoor je de informatie gaat leveren:

![Diagram van een BPMN procesmodel](image)

Figure 1: Een klein voorbeeld van een BPMN procesmodel

De namen van de stappen van het procesmodel zijn vaak van de vorm *zelfstandig naamwoord - werkwoord*.

Bijvoorbeeld: koffie maken. Samen vormen deze stappen een volledige werkbeschrijving.
6 Wat moet je doen?

Op de volgende pagina’s staat een uitleg over het gereedschap dat in dit experiment wordt getest.

Daarna staat er een casusbeschrijving. Het doel is om deze casus te beschrijven in je interview. Je wordt door de vragen aan de hand geleid. Wees niet bang een vraag te stellen als je even niet begrijpt waar het over gaat, maar alles zou voor zichzelf moeten spreken. Je moet er van uit gaan dat je chatpartner deze casus niet kent, want hij probeert namelijk deze situatie te modelleren en wil de kennis uit jou verkrijgen!

Het gereedschap voor dit interview is een soort chatprogramma. Dit interview is zo opgezet dat je delen van de casus kan beschrijven aan de persoon die de vragen stelt. Je moet daarbij aannemen dat de interviewer helemaal niets weet van de casus die je beschrijft.

Er zijn drie fases waar het interview tussen op en neer gaat. Iedere keer wordt een klein onderwerp verder uitgediept in de volgende fase. Deze fases zijn:

1. Het geven van een naam aan het proces. Dit is slechts eenmaal in het begin.

2. Een vrije discussie over hetgene wat je wil/moet gaan vertellen. Samen kiezen we hier deelonderwerpen. Hierop wordt steeds terug gekomen na het bespreken van een deelonderwerp.

3. Een vragenfase, waar over een deelonderwerp specifieke vragen worden gesteld.

4. Een koppelfase, waarin we de specifieke vragen evalueren en proberen te koppelen. Daarna keren we terug naar de vrije fase.

Wanneer je denkt een belangrijk antwoord te geven kan je antwoorden van de vorm *zelfstandig naamwoord - werkwoord* tussen vishaken zetten. Wanneer je vishaken of vierkante haken ziet, dan moet je daartussen je kernwoorden plaatsen en de haakjes laten staan.
7 Het gereedschap

Deze methode maakt gebruik van een soort chatbox. Te vergelijken met facebook chat, msn of zelfs irc. Als dit je niets zegt, geen probleem. Hier is een voorbeeld van hoe de chatbox eruit ziet:

![Figure 2: Het programmavenster](image)

Er zijn slechts vier belangrijke onderdelen. Hier staan ze op een rijtje:

1. Het eerste onderdeel is het chatvenster. Hierin staat de geschiedenis van je conversatie. Je kan de scrollbalk gebruiken om terug te kijken.

2. Het tweede onderdeel is de vreemdste eend in de bijt. Hier moet je eerst aangeven wat je antwoord het beste beschrijft, alvorens zelf te gaan typen.

3. Het derde onderdeel is het typvak. Je krijgt hierin een voorbeeldtekst nadat je het soort antwoord bij 2. hebt gekozen.

4. Als je klaar bent kan je met de knop send je bericht versturen.

Nu zijn er nog een paar zaken die van belang zijn.

- De voorbeeldtekst is slechts een voorbeeld. Als je deze wilt aanpassen of aanvullen moet je dat absoluut doen. Wijk s.v.p. alleen niet teveel af van de bedoeling.
• Voordat je je antwoord typt, kijk dan even welk antwoordtype uit de lijst het beste bij je antwoord past. Je kan dit gemakkelijk en zonder gevolgen veranderen.

• Als er puntjes tussen brackets [...] of vishaken <...> staan, probeer dan daar je daar je de kern van het antwoord in te typen. Laat de brackets of haakjes staan.

• Als je je antwoord niet passend genoeg kwijt kan kies dan vrije text. Als het dan nog niet lukt, laat het me dan even weten.

• Verder moet je weten dat je niets kan selecteren als het niet jouw beurt is om te typen. Dit is normaal, wanneer jouw beurt komt wordt de invoer vrij gegeven.

Houd in gedachten dat het programma nog in de kinderschoenen staat. Als je er even niet uitkomt mag je mijn hulp inschakelen.

Tijdens het experiment worden gesprekken opgenomen. Dit geld voor zowel de tekstgesprekken in de software, als de gesprekken in de kamer.
8 Scenarioschets

8.1 Casus Boodschappen doen


Op de onderstaande punten mag je terugvallen om je antwoord te geven.

- Je stelt je boodschappenlijst op door te kijken wat er niet meer in je voorraad aanwezig is
- Terwijl je dat doet overleg je met een vriend(in) of hij/zij ook nog iets nodig heeft
- Je gaat met de fiets, auto, bus of lopend naar de winkel
- Er zijn verschillende winkels waar je heen kan gaan, en je kiest degene met gemiddeld de laagste prijzen op die dag
- Soms doe je voor een week boodschappen, andere keer slechts voor een dag
- Je besluit alleen een winkelwagen te pakken als er niemand in de weg staat, anders pak je een mandje
- Je bezoekt de servicebalie op te zoeken voor vuilniszakken, maar dit kan voor of na het winkelen
- Je zoekt de boodschappen op je lijstje. Soms neem je een dure variant, soms een goedkope.
- Je komt altijd wel iemand tegen die in de weg staat, deze vraag je om even aan de kant te gaan
- Tijdens het kiezen van de boodschappen doen bel je soms met een kennis
- Er zijn twee winkelmedewerkers, en je kiest de ogenschijnlijk best opgeleide om iets voor je op te zoeken
- Er zijn drie kassa’s waar je kan afrekenen. Je kiest de kassa die niet te druk is
- Je gaat naar huis
- Je leest je bonnetje door op fouten
# Evaluatie

1a. Voelde je je betrokken bij het process?
   helemaal niet ☐ ☐ ☐ ☐ ☐ — volledig ☐ ☐ ☐ ☐ ☐

1b. Hoe was het tempo?
   te traag ☐ ☐ ☐ ☐ ☐ — te snel ☐ ☐ ☐ ☐ ☐

1c. Kon je al je informatie kwijt?
   helemaal niet ☐ ☐ ☐ ☐ ☐ — volledig ☐ ☐ ☐ ☐ ☐

1d. Vond je de keuzemogelijkheden breed genoeg?
   niet breed ☐ ☐ ☐ ☐ ☐ — zeer breed ☐ ☐ ☐ ☐ ☐

1e. Vond je de begeleidende en voorbeeldtekst duidelijk genoeg?
   niet duidelijk ☐ ☐ ☐ ☐ ☐ — zeer duidelijk ☐ ☐ ☐ ☐ ☐

1f. Werd je genoeg de goede richting in gestuurd?
   teveel sturing ☐ ☐ ☐ ☐ ☐ — te weinig sturing ☐ ☐ ☐ ☐ ☐

1g. Zou je een sessie als deze kunnen omschrijven als leuk?
   helemaal niet ☐ ☐ ☐ ☐ ☐ — volledig ☐ ☐ ☐ ☐ ☐

1h. Zou je een sessie als deze kunnen omschrijven als nuttig?
   helemaal niet ☐ ☐ ☐ ☐ ☐ — volledig ☐ ☐ ☐ ☐ ☐

1i. Waren de gestelde vragen duidelijk genoeg?
   helemaal niet ☐ ☐ ☐ ☐ ☐ — zeer duidelijk ☐ ☐ ☐ ☐ ☐

1j. Was er een duidelijke structuur aanwezig?
   helemaal niet ☐ ☐ ☐ ☐ ☐ — zeer duidelijk ☐ ☐ ☐ ☐ ☐

2. Wat mistte je, indien je de keuzemogelijkheden beperkt vond?

3. Wat beviel je aan deze sessie?

4. Wat kan er beter?

5. Heb je nog andere opmerkingen of suggesties?
B Dialog tree

The final revision of the dialog tree in Dutch. As participants were native Dutch speakers, the tree was translated into Dutch early on and later revised in that same language.

Step=1
Role=F
QuestionOption =Introductie
QuestionExample =Hallo. Welkom bij de dialog process modeling sessie.
Met een interview ga ik je situatie die je wilt beschrijven schetsen

Step=2
Role=F
QuestionOption =Vraag een naam voor het proces
QuestionExample =We beginnen met het vinden van een naam voor wat je wilt beschrijven.
Wat gaan we bespreken? Bijvoorbeeld: <Klanten screenen en registreren>

Step=3
Role=E
QuestionOption =Ik weet een naam
QuestionExample =Ik denk dat <...> een goede naam is

Step=4
Role=F
QuestionOption =Ik weet geen naam
QuestionExample =Ik weet op het moment geen geschikte naam

Step=5
Role=F
QuestionOption =Eens met de naam
QuestionExample =Dat is een geschikte naam!

Step=6
Role=F
QuestionOption =Introductie vrije gespreksfase
QuestionExample =Goed! Nu hebben we een naam. We gaan nu je situatie in het algemeen doornemen. Pas later zal ik je over de details ondervragen.
QuestionOption =Herhaal terug
QuestionExample =Ik heb <...> op mijn lijst van interessante onderwerpen. Heb je nog toevoegingen?

QuestionOption =Doel(en)
QuestionExample =Welk doel - of welke doelen - wil je bereiken met het proces?

QuestionOption =(On)belangrijke delen
QuestionExample =Wat zijn volgens jou de belangrijkste onderdelen van het proces die nodig zijn om je doel te bereiken?

QuestionOption =Vraag met voorbeeld
QuestionExample =Zijn zaken zoals [...] ook aan de orde?

QuestionOption =Selecteer FoCon (relatiefase)
QuestionExample =Zullen we nu concentreren op <...>?

QuestionOption =Volgende FoCon (relateer aan vorige focon)
QuestionExample =We hebben zojuist <...> besproken. Wat volgt volgens jou daarna?

QuestionOption =Ander FoCon (zit er iets tussen twee focons)
QuestionExample =Vind er misschien iets plaats tussen <...> en <...> dat het beschrijven waard is?

QuestionOption =Stop sessie
QuestionExample =Bedankt voor al je hulp. Zullen we het voor vandaag hier bij laten?

Step=7
Role=E
QuestionOption =Vrij antwoorden
QuestionExample =

Step=8
Role=E
QuestionOption =Eens
QuestionExample =Ja, akkoord

QuestionOption =Oneens
QuestionExample =Nee, want [...]
QuestionExample =
11
QuestionOption = Keuze
QuestionExample = Welke keuze(s) in je proces bevatten <...>?
11
QuestionOption = Keuze alternatieven
QuestionExample = Bij de keuze [...], wat zijn dan de opties voor <...>?
11
QuestionOption = Keuze criterium
QuestionExample = Wat is dan je beslissingscriterium voor die keuze?
11
QuestionOption = Suggereer
QuestionExample = Klopt het als ik zeg dat [...]?
11
QuestionOption = Geef voorbeeld
QuestionExample = Bijvoorbeeld:
11
QuestionOption = Verander niveau van abstractie
QuestionExample = Je vertelt me iets teveel details. Kun je meer op het niveau denken van [...]?
10
QuestionOption = Volgende fase (ABM fase)
QuestionExample = Bedankt! Denk je dat we alle belangrijke beslissingen omtrent <...> hebben gehad?
12
QuestionOption = Vorige fase (open fase)
QuestionExample = Bedankt. We gaan nu terug naar ons open gesprek om een nieuwe interessant onderwerp te kiezen.
6
Step=11
Role=E
QuestionOption = Gesloten antwoord geven
(zet meerdere beschrijvingen evt los van elkaar tussen <>)
QuestionExample = Dat is <...>
10
QuestionOption = Open antwoord geven (vrij typen)
QuestionExample =
10
QuestionOption = Noem een criterium
QuestionExample = Ik beslis op basis van <...>
10
QuestionOption = Vraag om voorbeeld
QuestionExample = Kan je daar een voorbeeld van geven?
10
QuestionOption = Ik heb geen passend antwoord
QuestionExample = Ik weet even niets dat aan je beschrijving voldoet
10
QuestionOption = Vraag naar de focus
QuestionExample = Kun je me er even aan herinneren waarop we ons nu richten?
10 QuestionOption =Ik wil terug naar de open gespreksfase (iets algemeens schoot je te binnen)
QuestionExample =Ik wil graag terug naar het open gesprek, omdat <...> me te binnen schoot
10 Step=12
Role=E
QuestionOption =Ja
QuestionExample =Ja, dat denk ik wel
13 QuestionOption =Nee (blijf in de huidige fase om beslissingen te beschrijven)
QuestionExample =Nee, ik wil nog meer vertellen over de huidige focus
10 QuestionOption =Nee (terug naar open gesprek fase)
QuestionExample =Nee, ik wil terug naar het open gesprek
6 Step=13
Role=F
QuestionOption =Introductie verificatiefase
QuestionExample =Ik ga nu vragen stellen achterliggende argumenten voor je besliscriteria vast te stellen.
14 Step=14
Role=F
QuestionOption =Suggereer reden
QuestionExample =Klopt het als ik zeg dat het criterium <...> te maken heeft met [...]?
15 QuestionOption =Vraag naar reden criterium
QuestionExample =Bij de keuze tussen <...> en <...> was je criterium <...>. Wat is daarvan de achterliggende reden?
15 QuestionOption =Laat expert reden uitleggen
QuestionExample =Welke reden past het best bij [criterium]
15 QuestionOption =Stel vast: vaak het geval?
QuestionExample =Zijn <...> en <...> vaak door [...] aan elkaar verbonden?
15 QuestionOption =Open vraag/Eigen controlevraag
QuestionExample =
15 QuestionOption =Klaar (ga naar open gespreksfase)
QuestionExample =Bedankt. We gaan nu terug naar het open gesprek om een nieuw onderwerp te kiezen.
6 Step=15
Role=E
QuestionOption =Open antwoord geven (vrij typen)
QuestionExample =
14 QuestionOption = Bevestig suggestie
QuestionExample = Ja, dat klopt
14 QuestionOption = Verbeter suggestie
QuestionExample = Nee, ik zou zeggen [...] 
14 QuestionOption = Ik weet daarop geen antwoord (vrij typen)
QuestionExample =
14 QuestionOption = Ga terug naar open gespreksfase
QuestionExample = Ik wil graag terug naar het open gesprek
14 Step=998
Role=E
QuestionOption = Beindig interview
QuestionExample = Klik op Send om het interview te beëindigen
999 QuestionOption = Ik heb nog dingen niet besproken (ga verder met interview)
QuestionExample = We hebben het nog niet over <...> gehad
6 Step=999
Role=F
QuestionOption = Exit987654321
QuestionExample = Exit987654321
999