We Need Non-formal Methods Based on Formal Models in Interaction Design

Andreas Maier, Steffen Hess

We need Non-Formal Methods based on Formal Models in Interaction Design

Andreas Maier and Steffen Hess

Fraunhofer Institute for Experimental Software Engineering IESE
{andreas.maier, steffen.hess}@iese.fraunhofer.de

Abstract. According to our experience, early collaboration with non-expert stakeholders aimed at designing interaction in a user-centered way is mandatory if the goal is a great user experience. We have found that insistence on formal modeling when collaborating with non-experts leads to insufficient results. Therefore, we propose a user-centered approach in order to enable collaboration and communication among expert and non-expert stakeholders. This non-formal approach should be based on a formal model, which also builds the common ground for discussions between all involved project stakeholders.

1 Introduction

1.1 Motivation

Well-designed and usable human-computer interaction (HCI) is a key factor for successful software products. Offering a large number of features is no longer sufficient for achieving success on the market. We observe stronger interest in usability and user experience factors in the decisions people make regarding particular products in the same class of devices (e.g. iPhone vs. other smartphones, iPod vs. other MP3 players). Users expect good usability and want to enjoy a great experience when interacting with the system. But what makes an interaction a great experience? What are the elements of a great HCI? Which dependencies exist between these elements? How do we design such an interaction? These are the questions we want to answer in order to improve the results of interaction design and contribute value to the HCI community. To do so, we developed a formal interaction model, which shows the most important elements of HCI and their dependencies. This model is called MAInEEAC (Model for Accurate Interaction Engineering, Enhancement, Alteration, and Characterization) and is introduced briefly in section 2. Since we are aware of the fact that a model is not applicable to conversations between expert stakeholders and non-expert stakeholders , we also created and use the non-formal interaction design method “mConcAppt”, which is built on MAInEEAC and tailors HCI elements to the given context in which an HCI is designed. During this article, we refer to experts as experts in terms of software development.

adfa, p. 1, 2011.

After the description of MAInEEAC, we describe our interaction design method in section 3. Section 4 sums up the advantages of our approach. The article closes with a conclusion and an outlook on future work.

1.2 Our Model-Based Interaction Design Approach

In contrast to purely formal model-based approaches or the strict performance of formal methods, we apply a user-centered design approach [6], which combines the advantages of both formal models and non-formal methods.

In this approach, we involve non-expert stakeholders as early as possible, since only potential operators of the system under development know what constitutes a great experience in their context. When non-expert stakeholders are involved, experts use formal models to work effectively and efficiently: Every expert has a formal conceptual model of the target system in mind when designing an interaction [7], in this case a formal model for interaction design. With the help of this model, the experts are aware of every element of HCI and every dependency between elements. In conversations with non-expert stakeholders, the experts are able to ask for all relevant information regarding HCI, trace the development of the HCI, and find conflicting specifications immediately by matching the conceptual model to the non-expert stakeholders' mental models of the system [ibid.]. To prevent non-expert stakeholders from being confused, the expert does not show the formal model to them, but only uses it as preparation and for post-processing.

With the formal model in mind, the experts can describe a particular HCI on a very detailed level. They can easily show interaction elements and dependencies, can facilitate communication between stakeholders involved in the development process, and close the gap between interaction designers and end users.

The non-formal method is used for eliciting, analyzing, and specifying requirements, prototyping HCI according to the elements given by the formal model, and validating the interaction design.

2 MAInEEAC – A Formal Model for Interaction Engineering, Enhancement, Alteration, and Characterization

As the underlying model of our user-centered interaction design approach, we developed MAInEEAC - Model for Accurate Interaction Engineering, Enhancement, Alteration, and Characterization [4]. This model is view-based and currently consists of ten views, six of which we present in detail in this report. These six views are most relevant for this approach and build the formal basis of the non-formal method. The four views we do not describe in this report represent even more details of HCI and are mainly interesting for researchers in this area, currently. They deal with characteristics of users, with the process of information perception and its details, respectively, and with different types of interactions.

2.1 General Overview

Figure 1 gives an overview of the interaction flow as described in MAInEEAC. This basic and coarse overview conforms to the unanimous view on HCI taken by almost every area dealing with HCI. Triggered by an *Intention*, the *Human* initiates an action on the *Input Interface* of the system. The *System* processes the given input and delivers a system reaction via its *Output Interface*. The *Human* again perceives this as a reaction to his *Intention* and evaluates if the perception is appropriate. In any case, this perception might affect the subsequent *Intentions*. This short description shows the basic concept of our model. As mentioned above, this view is common sense in the field of HCI. The concepts follow the Gulf Model published by Donald Norman in 1988 [8]. The distinct feature of MAInEEAC is that it is not restricted to the general overview shown in Figure 1, but shows all aspects in detail as well. It enables the different roles involved in system development to work with only one model. MAInEEAC represents the system in great detail, without being system-centered. In fact, it is interaction-centered, emphasizing the human at the same time. Cognition is not an explicit element in MAInEEAC, but implicitly covered by *Perception*, which covers the recognition, interpretation, and evaluation of information. *Media* and *Modalities* are well-defined distinct elements of HCI, both with an exact meaning: *Media* are representation forms of information, while *Modalities* are the concrete usages of human senses to perceive information. The user interface and possible system outputs are decomposed in order to describe HCI in even greater detail. Overall, MAInEEAC enables us to describe an interaction with a system without having to use another model.

We decided on using a view-based representation of the model to emphasize certain aspects of HCI. The following sections describe the most important aspects of MAInEEAC, including detailed views on its basic components (human action, system, system action, and interaction).

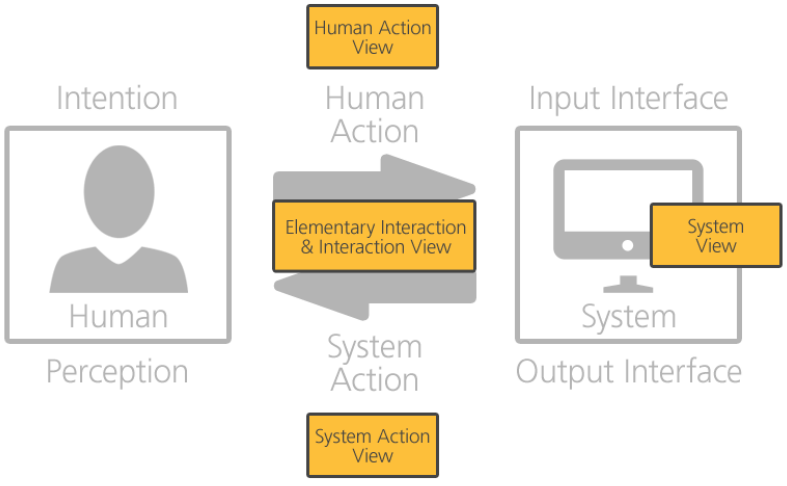

Fig. 1. MAInEEAC – General Overview

2.2 System View

The System View describes the system side during HCI (see Figure 2). MAInEEAC does not treat the System as a black box like many other HCI models do [8],[9]. The System components highly influence HCI and thus are shown in the same detail as Human elements and elements of the Interaction itself.

In general, we define *System* as a complex object based on software that fulfills a function by processing input and creating output. *Systems* in the application area of MAINEEAC might be PCs, industrial machines, handhelds, home appliances, consumer electronics, etc.

Each *System* consists of a number of devices. A *Device* is a technical aid which acts as interface for transmitting information from human to system and / or from system to human. Furthermore, devices are all parts of the system, no matter if these parts are actually used for the interaction or not (e.g. an electric shutter being controlled with the system). A device acting as *Input Interface* transmits information from human to system (e.g., keyboard, PC-mouse, touch-screen, microphone, and digitizer), whereas a device acting as *Output Interface* transmits information from system to human (e.g., screen, loudspeaker, and braille-display).

A System communicates with its environment through its *User Interface*, which presents the aggregation of all *Input Interfaces* and *Output Interfaces*.

The important fact we show with this distinction between device, input interface and output interface is that input and output might take place on different devices and even at different locations during HCI.

Every *Input Interface* is characterized by the *Usage Types* it offers. A *Usage Type* specifies how an *Input Interface* is used concretely to transmit information to a system (see also section 2.2 for a further clarification).

Furthermore, an *Input Interface* may give *Direct Input Feedback* which comes straight from the *Device* without the use of any *Medium* (for example, the sound that occurs when pressing a button on a mouse or keyboard). When a *Device* acts as an *Output Interface*, it gives *Application Feedback* and might give *Indirect Input Feedback* in addition. Both are transmitted via a *Medium*: *Indirect Input Feedback* is a reaction to the usage of the *Input Interface* to confirm the system's correct understanding of the human's action, for example, highlighting of a selected menu item transmitted via the *Medium* "graphic". It is always based on a *System Function*. The complete decomposition of system actions and detailed descriptions of each possible system action is represented in section 2.4.

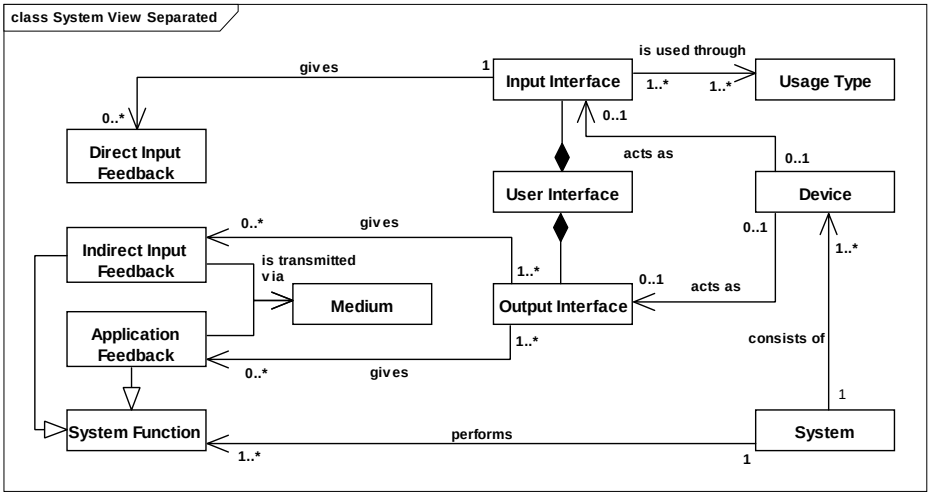

Fig. 2. System View

2.3 Human Action View

The Human Action view (see Figure 3) describes the way the *Human* accomplishes his *Intention* on the *Input Interface* of the *System*. In MAInEEAC, this accomplishment is called *Human Action* and involves the whole activity performed by a *Human* to transmit information to a *System*. Every activity might be influenced by the *Environmental Context* in which the activity takes place. For example, the *Human* might not want to transmit information to the *System* via speech when he is in a noisy environment or he might want to interact with the *System* from a distance when his environment is a huge warehouse. The *Environmental Context* is an attribute of *Human Action* and thus has to be specified when using MAInEEAC. Each *Human Action* consists of at least one *Action Method*, which specifies the action of the *Human*. It comprises generic human movements according to original human abilities. We distinguish four types of action methods:

1. Fine Motor Skills (e.g., typing, writing, moving an object, pressing an object)
2. Gross Motor Skills (e.g., gesturing, walking, jumping)
3. Facial Expressions (e.g., smiling, grinning, frowning)
4. Vocal Utterances (e.g., speaking, whispering, shouting)

Furthermore, a *Human Action* always has a *Method Type*, which specifies if the action is uni-methodical or multi-methodical. If there is only one kind of *Action Methods*, the type is uni-methodical. If there are at least two *Action Methods*, the *Method Type* is multi-methodical.

When performing the *Human Action*, every *Input Interface* features different *Usage Types* that specify how an *Input Interface* is exactly used to transmit information to the *System*. For example, ‘pressing object’ as an *Action Method* is underspecified. There is a set of concrete movements being performed within a *Human Action*. It

makes a big difference whether one presses the left or the right mouse button or whether someone performs a single or a double click. In addition, the *Usage Type* concretely specifies the *Action Method*. For example, if the *Usage Type* of the *Input Interface* is 'single left click', the *Action Method* is determined as 'pressing object'.

When characterizing an interaction with MAInEAAC, the elements *Action Method*, *Usage Type*, and *Input Interface* influence each other. When an *Action Method* is determined, the possible *Input Interfaces* and *Usage Types* are deduced from that *Action Method*. If, for instance, the *Action Method* 'Speaking' is determined, the number of *Input Interfaces* possible for sound input is low and the possible *Usage Types* are restricted to that *Action Method*. The *Action Method* in conjunction with the *Usage Type* restricts the number of possible *Input Interfaces* even more. With respect to the *Usage Type*, the same holds for the *Action Method* in conjunction with the *Input Interface*.

The *Usage Type* determines the *Action Method* and restricts the number of available *Input Interfaces* to those that offer the specified *Usage Type* and allow for performing the deduced *Action Method*. For example, when the *Usage Type* 'Natural Speech' is selected, 'Speaking' is automatically determined as the *Action Method*. The list of *Input Interfaces* is restricted to different microphones like desktop microphone, handheld microphone, wearable microphone, etc. When we decide on a particular *Input Interface*, the *Action Method* as well as the *Usage Type* are restricted at the same time. When *Input Interface* and *Usage Type* have been determined, the *Action Method* is deduced from those. This does not hold for the determination of the *Input Interface* and the *Action Method*: When the *Input Interface* and the *Action Method* have been determined, we still have a choice regarding the *Usage Type*. For example, when we decide on 'microphone' as *Input Interface* and 'natural speech' as *Usage Type*, 'Speaking' is automatically determined as the *Action Method*. When we leave the *Usage Type* open and decide on an *Action Method* from the class 'Vocal Utterance', we can still decide on which *Usage Type* to apply.

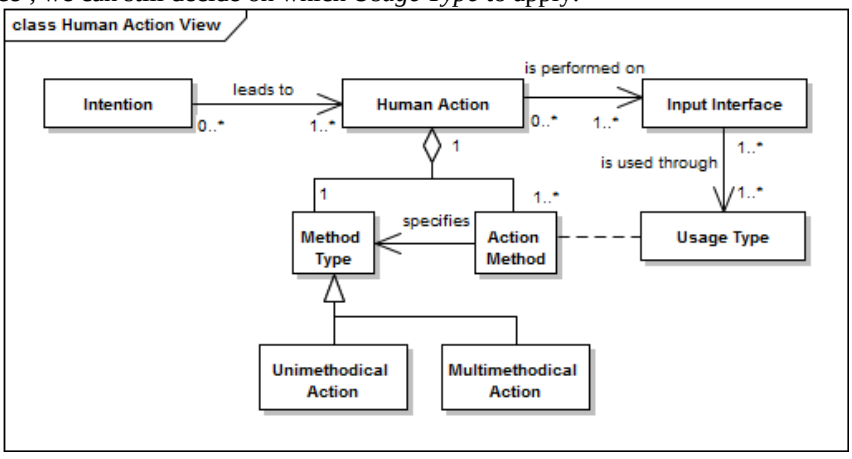

Fig. 3. Human Action View

2.4 System Action View

The System Action view (see Figure 4) describes the way the *System* reacts to the *Human Action* and the transmission of information from the *System* to the *Human*.

The *System Action* is a composition of *Direct Input Feedback*, *Indirect Input Feedback*, *Application Feedback*, and *System Function*. *Direct Input Feedback* is feedback the *Human* gets directly from the *Input Interface*, i.e., not via any *Medium*. Example: the physical resistance of a keyboard stroke or the sound when pressing a key on a keyboard or a PC mouse. In contrast to this, *Indirect Input Feedback* is input feedback the *Human* gets via a *Medium* as a reaction to his usage of an *Input Interface*. It is the *System Action* on a human's action to confirm the system's correct understanding of that action. *Indirect Input Feedback* is always based on a *System Function* and can be influenced by a designer. Examples: bordering of an icon that the mouse cursor points to or highlighting of a selected menu entry. A *System Function* is a *System Action* that is performed automatically by the *System* as a reaction to an external trigger. An external trigger may be, for example, a *Human Action*, a call from an external system, or an environmental context change. A *System Function* does not necessarily address a *Human* directly, but recognizes events, interprets and manipulates data, and plans and initiates *Application Feedback* and *Indirect Input Feedback*. A *System Function* cannot be influenced by an interaction designer. Examples: an internal change in the system state, working hard disk drives, control of external but system-related objects (e.g., lights).

From an interaction-centered point of view, a *System Action* is given when at least one *Application Feedback* exists. Furthermore, an arbitrary number of the other elements can be included. The rationale for this composition is that *Application Feedback* directly refers to the human's *Intention*. Without feedback to the *Intention*, we are not able to fulfill an *Elementary Interaction*. For example, if a user performs an action on a user interface and only receives the feedback that his input was successful, we do not refer to this as an *Elementary Interaction*. The user always has to perceive the part of the *System Action* that belongs to his *Intention* in order to complete an *Elementary Interaction*. This part is the *Application Feedback*.

Depending on the *Environmental Context* in which the *System Action* takes place, the *System Action* might be influenced by that context. For example, the *System Action* might be to transmit information to the *Human* by means other than sound when the environment is noisy, or to show relevant information to a particular person when it detects that that person is near.

The *System Action* triggers a human's *Perception*, which is composed according to the *System Action*. Because of this segmentation, we can trace which *Perception* is triggered by which kind of *System Action*.

For each kind of *Perception*, a *Modality Type* is specified that determines if the *Perception* is unimodal or multimodal. Unimodal perception is given when exactly one *Modality* is used to perceive the part of the *System Action*. In multimodal perception, at least two different *Modalities* are used.

In addition to that, for *Indirect Input Feedback Perception* and *Application Feedback Perception*, a *Medium Type* is specified that determines if the *Perception* is unimedial or multimedial. This is necessary for those two types because they are the only ones that are transmitted via a *Medium*. If one *Medium* is used for transmission, the medium type is unimedial; if at least two different *Media* are used, the medium type is multimedial.

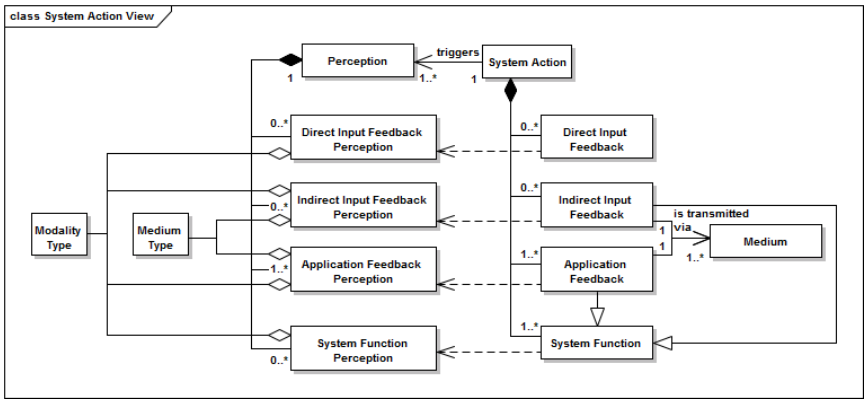

Fig. 4. System Action View

2.5 Interaction View

The interaction view (see Figure 5) gives a holistic overview of the views described so far and emphasizes the build-up of an *Interaction*. An *Interaction* consists of several *Elementary Interactions*. Furthermore, each *Interaction* consists of a *Human Action* that is initiated by an *Intention* and a *System Action* that triggers a *Perception*. This *Perception* either confirms or rejects the initial *Intention*. With the help of the interaction view, it is possible to describe an *Interaction* as a whole in a detailed manner, which can further be broken down by using more detailed views.

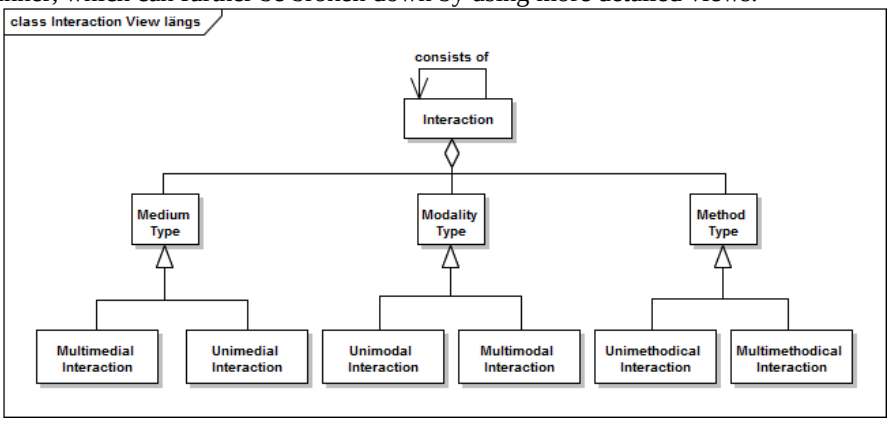

Fig. 5. Interaction View

2.6 Elementary Interaction View

The Elementary interaction view (see Figure 6) describes the composition of an *Elementary Interaction* and its specializations in detail. Furthermore, it is shown how an *Elementary Interaction* is constructed.

Elementary Interactions are restricted to exactly one *Human Action* and one to many *System Actions*. Example: Typing text into a word processing application (*Human Action*) and perceiving the written text on a screen in order to check the correct spelling (text output as *System Action*).

An *Elementary Interaction* always has three types, which specify how the *Elementary Interaction* takes place in detail.

First, the *Method Type* specifies if the *Elementary Interaction* is unimethodical or multimethodical. A unimethodical *Elementary Interaction* is an *Elementary Interaction* where the *Human Action* is unimethodical. A simple example is pressing a key on the keyboard in order to select a list item and visually perceiving the selected item. The attribute that is relevant for determining the *Method Type* derives directly from the *Human Action;* i.e., a unimethodical action remains unimethodical on the *Elementary Interaction* level, a multimethodical action remains multimethodical. Of course, in this simple example we have just one *Action Method*, namely pressing a key. A multimethodical *Elementary Interaction* is an *Elementary Interaction* where the *Human Action* is multimethodical. A simple example is pointing with a finger to an icon and saying 'open that' and visually perceiving the opening application. In this example, two different action methods are used in parallel during the action, namely 'Gesturing' and 'Speaking'. Therefore, the *Elementary Interaction* is multimethodical.

Second, the *Modality Type* specifies if the *Elementary Interaction* is unimodal or multimodal. A unimodal *Elementary Interaction* is an *Elementary Interaction* where *Perception* is unimodal. An example is requesting information from a speech dialog system on the phone and aurally perceiving the answers. A multimodal *Elementary Interaction* is an *Elementary Interaction*, where the *Perception* is multimodal. An example is requesting information from a speech dialog system on the PC and aurally as well as visually perceiving the answers.

To determine the modality types, all *Perceptions* are subsumed and it does not matter which kind of *Perception* the modality derives from.

Third, the *Medium Type* specifies if the *Elementary Interaction* is unimedial or multimedial. A unimedial *Elementary Interaction* is an *Elementary Interaction* where the *Perception* is unimedial. An example is using a command-based input shell (as in UNIX), perceiving only alphanumerical text. A multimedial *Elementary Interaction* is an *Elementary Interaction* where the *Perception* is multimedial. An example is using a GUI-based operating system (like Microsoft Windows) and perceiving alphanumerical text, graphics, animations, and icons during one single elementary interaction. To determine the *Medium Type*, each medium involved in *Indirect Input Feedback Perception* and *Application Feedback Perception* is considered.

An *Elementary Interaction* always has to be characterized through the triple *<method type, modality type, medium type>*. For example, the *Elementary Interaction* of clicking the recycle bin icon on a MS Windows desktop to open the context menu

is multimethodical (moving an object and pressing an object), unimodal (visual perception of the moving mouse cursor and the opening context menu), and multimedial (representation of information via the icon, text, and graphics within the context menu). An *Elementary Interaction* specified by only one or two of these types does not exist from our point of view.

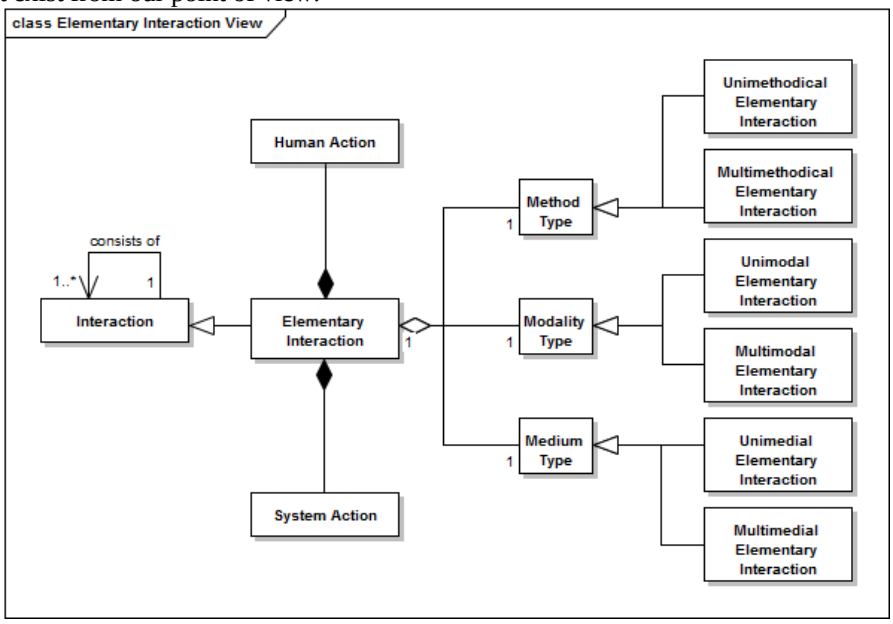

Fig. 6. Elementary Interaction View

3 A Non-Formal Method Based on MAInEEAC

When collaborating with non-expert stakeholders in order to define an interaction concept for a particular software system, MAInEEAC as a formal model is not applicable in its original condition.

Therefore, we have developed non-formal methods based on this formal model to facilitate this collaboration. After having accomplished a huge number of software engineering projects with interaction design activities, we have found that these methods, where stakeholders can speak freely, lead to more relevant information than strict formal and restricted approaches. One of these methods is the mConcAppt method [2],[3], which is a method for the user-centered conception of mobile business apps. In this chapter, we focus on the part of mConcAppt that is relevant for the construction of the HCI.

3.1 Upfront Requirements Elicitation and Analysis

In this practical approach for mobile interaction design, we face typical challenges like short time to market, high user experience, and less focused user attention than

with desktop systems. Furthermore, the user interface design is related to device, platform, and technology. It comprises upfront requirements elicitation and analysis in combination with an iterative interaction design (see Figure 7). Both phases are conducted with close user involvement.

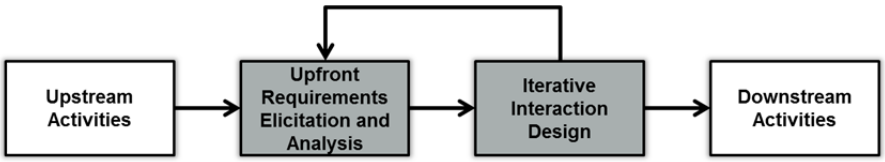

Fig. 7. Brief overview of mConcAppt

During the execution of mConcAppt, different kinds of artifacts are produced. Upfront requirements elicitation and analysis are mainly performed by means of a requirements elicitation workshop that involves end users as well as other project-relevant stakeholders (see Figure 8). In addition, several requirements elicitation phone calls or interviews might be performed in order to clarify information elicited in the workshop as well as to elicit further requirements. As a result from upfront requirements elicitation and analysis, the following artifacts are produced in close collaboration with non-expert stakeholders:

1. List of involved stakeholders
2. Stakeholder goals
3. Stakeholder description
4. Description of a user persona [1]
5. Description of the as-is situation
6. Problems in the as-is situation
7. Description of the to-be situation
8. Technical constraints
9. Exchanged domain data

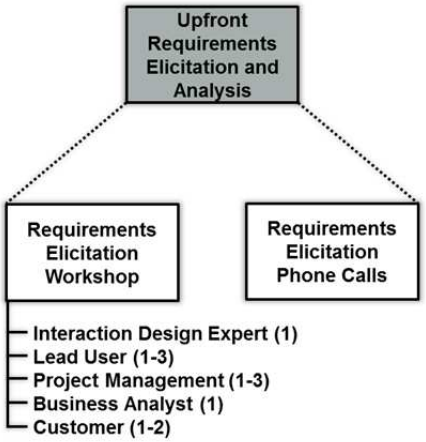

Fig. 8. Upfront Requirements Elicitation and Analysis Phase

Requirements and interaction design experts already have the formal MAInEEAC model in mind while eliciting and documenting those artifacts. From the list of involved stakeholders, the definite end users (cf. *Human* in MAInEEAC) are derived. Especially stakeholder goals elicited in combination with the description of the to-be situation will ultimately lead to the interaction created using the formal model. Already at this point in time, communication between the requirements engineer and the interaction designer is improved, since they both have a conjoint formal model in mind when discussing different aspects.

All mentioned artifacts are used as input for the development of the initial interaction design, which will be described in the following chapter.

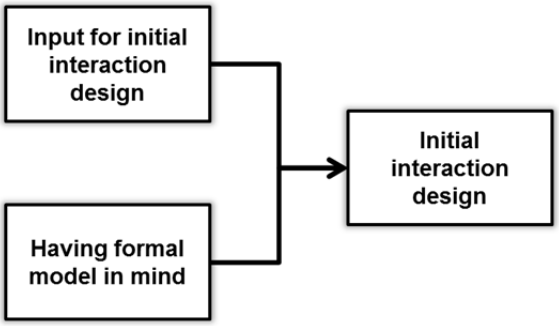

Fig. 9. Initial Interaction Design Phase

3.2 Iterative Interaction Design

The second phase during mConcAppt is the actual interaction design phase. Figure 9 shows that the artifacts elicited in the first phase and the formal model that the interaction designer has in mind are combined to create the interaction design. Therefore, interaction cases (see the example in Table 1**Table 1**) are created that structure the elicited to-be situation into a textual description using elements of the MAInEEAC model. During the textual specification of Interaction Cases, *Action Methods*, *Usage Types*, *Input Feedback*, and *Application Feedback* are defined. Using the interaction cases, the actual wireframes are assembled by combining exactly one *Human Action* and one *System Action* into one wireframe. In early iterations, wireframes are usually produced using paper prototypes and evaluated with real end users in a Wizard-of Oz setting. During those early evaluations and especially during later ones, when the prototype is already in a more mature state, the interaction designer benefits from having the formal model in mind when talking to the end user in order to check whether all recommended elements (e.g., giving *Input Feedback* at each *Human Action*) have been considered. In this phase, it might also beneficial to apply a co-design practice: interaction designers and end-users might create the interaction cases and wireframes conjointly; this will probably minimize the effort necessary for reworks of the interaction design after it was evaluated by other end-users.

Item	Description
ID	IC2: Track time
Usage Context	Business travel for multiple days. Meeting an industrial partner in Leipzig. The user is on his way to the hotel and uses the device while walking.
System Action 1	The system recognizes that the user arrives at his destination and notifies him that the time of arrival is tracked (via the notification center).
Human Action 1	The user taps the notification to directly open the app.
System Action 2	The system opens the app and immediately shows the current trip itinerary and the proposed time.
Human Action 2	The user confirms the proposed time.
System Action 3	The system provides feedback on the confirmation to the user.
Human Action 3	The user closes the app.
Post-condition	The arrival time is stored persistently.

Table 1. Interaction Case Example

4 Advantages of Our Approach

The application of the approach we propose in this chapter bears a number of advantages: Non-expert stakeholders are not forced to worry about formal models, since expert stakeholders do not discuss formal models with non-expert stakeholders, but rather prepare their conversations with the help of the model. Due to the unrestricted and non-formal conversations, all stakeholders can speak freely and more relevant information is gathered than with strict formal and restricted approaches like workshops and interviews. During the conversation, the expert stakeholders know which information is missing and can describe and discuss the impact of decisions made collaboratively. But it is still the interaction designer who creates the final interaction design and the developer who implements the interaction. Benefits of our approach can also be found within the groups of expert stakeholders and non-expert stakeholders, where communications are facilitated, since a single model with a single terminology can be used. When single group members are familiar with or used to another terminology, both terminologies can be mapped very easily. Besides requirements engineers, interaction designers, and developers, visual designers, architects, business analysts, customers, and end-users will also benefit from this facilitation of communication. The requirements elicited, analyzed, and specified by these communications while applying the non-formal method can be traced to concrete interactions and their

highly detailed elements. While the formal model represents the elements, the non-formal method assures traceability.

5 Conclusion

From our point of view, early collaboration with non-expert stakeholders is mandatory. This is done best by applying non-formal methods based on formal models. Insisting on formal modeling when collaborating with non-expert stakeholders leads to insufficient results, since non-expert stakeholders have to familiarize themselves with unusual formal models and unusual formal thinking instead of focusing on considerations about needs, wishes, and demands in terms of HCI.

In this article, we presented our user-centered design approach for an interaction design based on our non-formal method mConcAppt, which follows our formal model MAInEEAC. Due to the early involvement of non-expert stakeholders, the software development process can be shortened and thus could be applied to a wide range of software development projects, especially to projects using an agile development approach. Such agile development approaches are often applied to mobile business applications, for example, which need a lightweight user-centered approach because of their special challenges. Formal approaches do not satisfy this requirement sufficiently. The approach can be applied to a large number of domains due to its flexibility and the formal structure in the background.

The approach presented (MAInEEAC, mConcAppt and their interrelation) is based on best practices resulting from many projects in which we have designed interaction in collaboration with non-expert stakeholders, and is even supposed to enable semi-experts to design a well-conceived HCI. An example of the application of the approach in an actual project can be found in [5]. However, the approach is still evolving and work is in progress. We plan to integrate interaction-related aspects such as user experience and architecture in order to be able to cover a holistic view on HCI and to discuss all relevant aspects with non-expert stakeholders. We also believe that this approach is able to create new value through its capability to apply the co-creation practice. However, investigations of such effects by applying the approach were not conducted yet and remain an open issue to carry out as future work. Eventually, we hope to achieve a huge increase in interaction design quality with the help of our approach.

References

1. Cooper, A.: The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity. Indianapolis, USA, 1999.
2. Hess, S., Kiefer, F.: mConcAppt Methode - UX und Interaktionsdesign für mobile Business Apps in Usability Professionals Association, German Chapter: Usability Professionals 2012 Tagungsband. Konstanz, 2012.
3. Hess, S., Kiefer, F.: Quality by Construction through mConcAppt - Toward Using UI-Construction as a Driver for High Quality Mobile App Engineering. QUATIC 2012.

4. Hess, S., Maier, A., Trapp, M.: Differentiating between Successful and Less Successful Products by Using MAInEEAC – A Model for Interaction Characterization in Human-Computer Interaction. Design and Development Approaches, Springer Berlin / Heidelberg, 2011.
5. Hess, Steffen ; Kiefer, Felix ; Carbon, Ralf ; Maier, Andreas: mConcAppt - A Method for the Conception of Mobile Business Applications. In: Uhler, David (Ed.) ; Mehta, Khanjan (Ed.) ; Wong, Jennifer L. (Ed.): Mobile Computing, Applications, and Services. 4th International Conference. MobiCASE 2012 - Revised Selected Papers. Berlin: Springer-Verlag, 2013, 1-20. (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 110).
6. International Organization for Standardization. ISO 9241-210:2010 - Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems, 2010.
7. Norman, D. A.: Some observations on Mental Models. In D. GENTNER, and A.L. STEVENS, eds. Mental Models. Lawrence Erlbaum Associates Inc. pp7-14, 1983.
8. Norman, D. A.: The Design of Everyday Things. New York, Doubleday, 1988.
9. Schomaker, L., Munch, S., Hartung, K.: A taxonomy of multimodal interaction in the human information processing system. Technical report, ESPRIT BRA, No. 8579, 1995.