Requirements Elicitation: Tools for End-Users

Denise Ammann

University of Zurich, Department of Informatics
Binzmühlestr. 14, CH-8050 Zurich, Switzerland
denise.ammann@uzh.ch

Abstract. Communication is difficult in requirements engineering. End-users and IT specialists have different backgrounds, thus many misunderstandings occur often without realizing it until later in the project. This causes financial and timely drawbacks, thus early involvement of end-users for requirements elicitation is essential. This paper discusses visualization and mobile tool approaches for requirements elicitation for end-user tools. OpenProposal is a visualization tool which expects the end-user to draw requirements onto their screen and send them to IT specialists. Immediate visualization integrates the end-user from the beginning into the description process. iRequire is a mobile requirement elicitation tool, which end-users use in situ. The ConTexter mobile tool is used in an IT ecosystem where wide-audiences report feedback for different systems which have to be identified. In the appendix an approach for teaching this topic to secondary school students is presented.

Keywords: Requirements elicitation, end-user tools, visualization, IT ecosystem, mobile tools

1 Introduction

Rashid et al. [14] say that "requirements analysts and end-users essentially speak different languages". Costabile et al. [5] state that "end users do not understand software developers jargon and developers often do not understand user jargon". Communication between end-users and IT specialists is a problem in requirements engineering.

Pohl et al. [13] explain that "requirements have to be communicated". For communication a mutual medium is necessary, mostly it is a natural language such as English. Even with a universal medium it has to be watched out for ambiguity, misconception and different previous knowledge. Exchange of information can be improved when there are common cultural, educational and social backgrounds and experiences between the communication partners. Often stakeholders don't have common backgrounds with IT specialists therefore this causes misunderstandings which lead to ambiguous requirements.

Studies show that 60% of project failures fall into the requirements engineering phase [4] and mostly aren't discovered until late during the project or when the system has already gone life [13].The later the error is detected the more expensive is the rectification [4]. Since missing or incomplete requirements cause projects to fail, it is important to find solutions for improving the quality of requirements.

Software engineers expect well-formulated requirements written in a detailed formal specification. For end-users to develop such specifications is very difficult and time consuming. That's why requirements analysts should write them with support of the stakeholders. There are many methods and techniques for eliciting user requirements which requirements analysts can use. Beyer et al. [2] claim that traditional interviews, surveys and workshops [8] are mostly used but not always suitable. To fill the communication gap between end-users and IT specialists these techniques are often used [16].

The diversity of domains of systems is a challenge for the IT specialists since their background is mainly IT. For each system they need to adjust to a new professional environment and a way of thinking. The end-users are the ones that have the knowledge of the non-IT topics and have to use the system in the end, thus they should really know what is expected. That's why it is important to find a way for end-users to communicate requirements in an accurate and understandable manner. For this reason it is important to have end-user tools where end-users can directly document their environment as well as their tasks and needs right when they occur during their daily work. Such tools are expected to have essential functionalities such as usability, efficiency and structure [14].

There are different approaches to find useful end-user tools for eliciting requirements. One approach is to find a common language between end-users, requirements analysts and software engineers through visualization. Another approach is recording a requirement or need when it occurs with a mobile tool.

2 Visualization

There are many solutions for visual requirements acquisition but mainly for requirements analysts and software engineers, e.g. UML Use-Case diagrams [13], mockups [11], rapid prototyping techniques [3]. End-users can mostly just give a feedback due to their level of IT background. That's why end-user tools are needed where they can describe their needs in a natural way through visual aid. Perez et al. [12] state that "visualization helps the end-user to identify their requirements" and it is more intuitive and easy to use than textual languages.

In the next two sections two approaches are discussed. The OpenProposal approach which is based on the Annotation Tool and Annotate!Pro, where end-users can annotate requirements right when they happen. And the immediate visualization for pervasive systems approach where the end-user is involved right from the beginning.

2.1 OpenProposal [15]

"OpenProposal is supposed to allow users to annotate their feature requests, error reports or enhancement requests directly on their applications workspace and send these requests to the requirements management. " [15] End-users actively participate in the software development process through submitting requirements for existing software as well as software under development.

Figure 1 shows the workflow in requirements engineering with end-user participation from Rashid et al. [14, 15]. The OpenProposal and Annotation Tool, discussed later, are based on this workflow.

There are three key players, the end-user, the requirements analyst and the software engineer, as well as five actions, specify, discuss, prioritize, decide and implement specified in the workflow. End-users participate mainly in the specification and discussion activities. Requirements analysts are responsible for the company's interests thus are involved in discussions, assigning priorities and making decisions. They may also propose new requirements. Software engineers' main focus is to understand the user's proposals, implement them correctly and contribute their professional technical knowledge to all other activities.

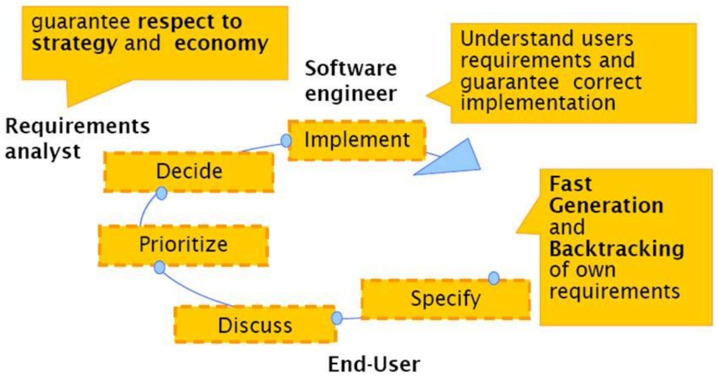

guarantee respect to strategy and economy
Software engineer
Understand users requirements and guarantee correct implementation
Requirements analyst
Implement
Decide
Prioritize
Fast Generation and Backtracking of own requirements
Specify
Discuss
End-User

Fig. 1. Workflow in requirements engineering with end-user participation [14, 15]

The OpenProposal approach is structured in three main components, the OpenProposal Annotation Tool, the OpenProposal Mediator and the OpenProposal Issue-Tracker. Figure 2 shows the architecture of OpenProposal.

The OpenProposal Annotation Tool is based on a previous Annotation Tool [14] and Annotate!Pro[1]. It gathers the annotated requirements (screenshots, annotations, metadata) in an XML specification which is sent to the mediator. The mediator creates a new issue in the Issue-Tracker with the specifications received. The issue can now be discussed and rated in the Issue-Tracker tool. A list of submitted requirements is available in the annotation tool.

OpenProposal is intended to support the software development process, e.g. global software development where team members are in different locations and can improve their communication with visuals of OpenProposal. Research done by Rashid et al. [15] has also shown that OpenProposal can improve collaboration between end-users and software engineers and it performs better than conventional tools. "OpenProposal was successfully realized in a real life scenario and is still in use" [15].

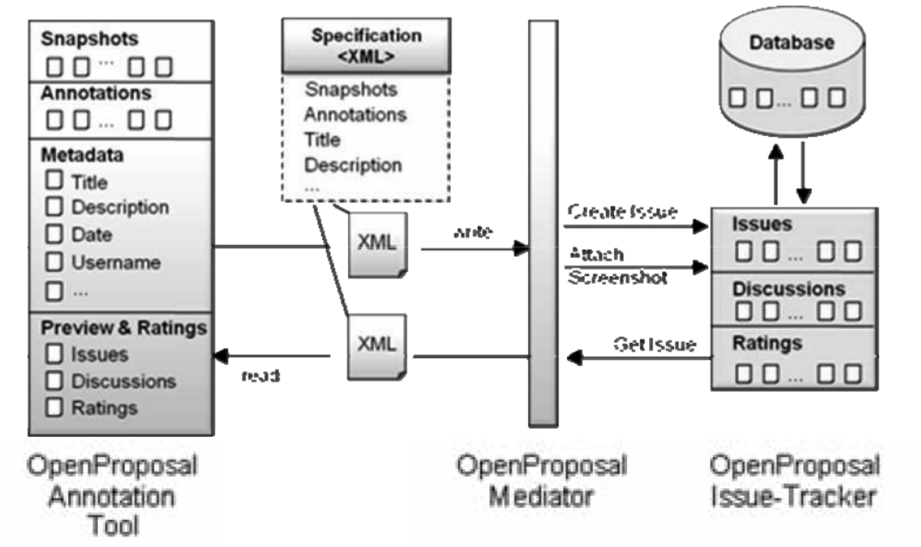


Fig. 2. OpenProposal Architecture [15]

Annotate!Pro [1]. This tool is intended to communicate between people during trainings, meetings, presentations, product development and help desk support. One can simply draw on top of any running application and save it as a screenshot. This approach was used for visually integrating requirements elicitation in the Annotation Tool and eventually the OpenProposal. End-users can easily draw and save requirements during daily work without much effort.

It is an easy to use tool, which doesn't require much time for getting used to it. The end-user simply has to start the program, a toolbar (see figure 3) appears with which then end-user can draw requirements directly on their screen. Once marked all the changes, a snapshot can be taken and then be sent to the requirements analyst and/or software engineer.

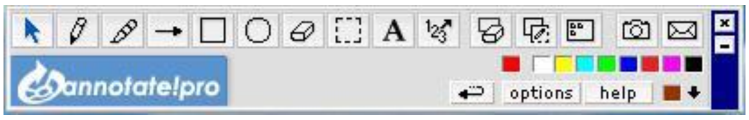
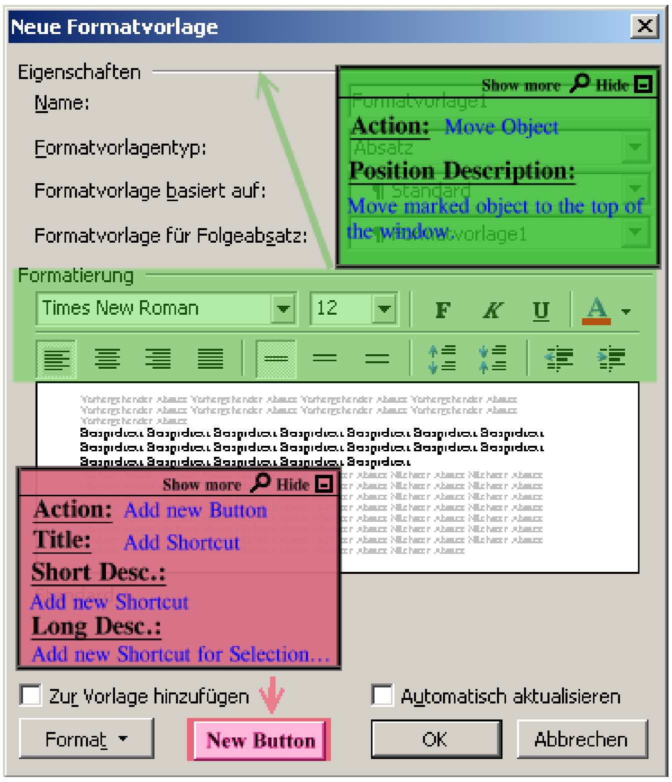


Fig. 3. Annotate!Pro [1]

Annotate!Pro doesn't provide a way of tracking the annotated requirements. Also there is no formal notation language thus there is no common language between the end-user and the IT specialist available. That's why Annotate!Pro's approach was a basis for the Annotation Tool where the limitations were solved.

Other similar tools exist such as Jing [9] which focuses on annotated images, audio and videos. But all of other tools show the same limitations, there is no formal language and no way of tracking the requirements.

Annotation Tool [14]. The Annotation Tool eliminates the shortcomings of Annotate!Pro. A web-based collaboration environment, such as an issue tracker, provides traceability and a discussion platform for the submitted annotated requirements. The end-user specifies with help of a template a few actions for the visual annotations, such as moving, resizing, adding a button, changing the sorting of a list. This template is well-structured through assisting dialogs for the end-user. Moreover, window information like username, application title is automatically collected and saved with the annotation requirement. Figure 4 shows how an annotated screen could look.


Neue Formatvorlage

Eigenschaften
Name: Formatvorlage1
Formatvorlagentyp: Absatz
Formatvorlage basiert auf: ¶ Standard
Formatvorlage für Folgeabsatz: ¶ Formatvorlage1

Show more Hide
Action: Move Object
Position Description:
Move marked object to the top of the window

Formatierung
Times New Roman | 12 | F | K | U | A

Show more Hide
Action: Add new Button
Title: Add Shortcut
Short Desc.:
Add new Shortcut
Long Desc.:
Add new Shortcut for Selection…

☐ Zur Vorlage hinzufügen
☐ Automatisch aktualisieren
Format ▾ | New Button | OK | Abbrechen

Fig. 4. Annotated requirement including supporting template [14]

Rashid et al. [14] were not sure if end-users needed support with the specification templates and if the users' drafts were with reasonable effort understandable. Additionally, the topic of privacy is mentioned, since automatically information is collected, thus all users of this tool have to comply with the data privacy.

2.2 Immediate Visualization of Pervasive System Requirements [12]

Another approach for visualization is a tool for supporting immediate visualization of pervasive system requirements. Perez et al. [12] characterize a pervasive system by context-awareness, proactiveness, mobility and personalization. End-users have to be able to describe the physical environment, the users, the services and devices of a pervasive system [10]. This prototype allows end-users to describe their needs and visualize them immediately. There are 3 types of users:

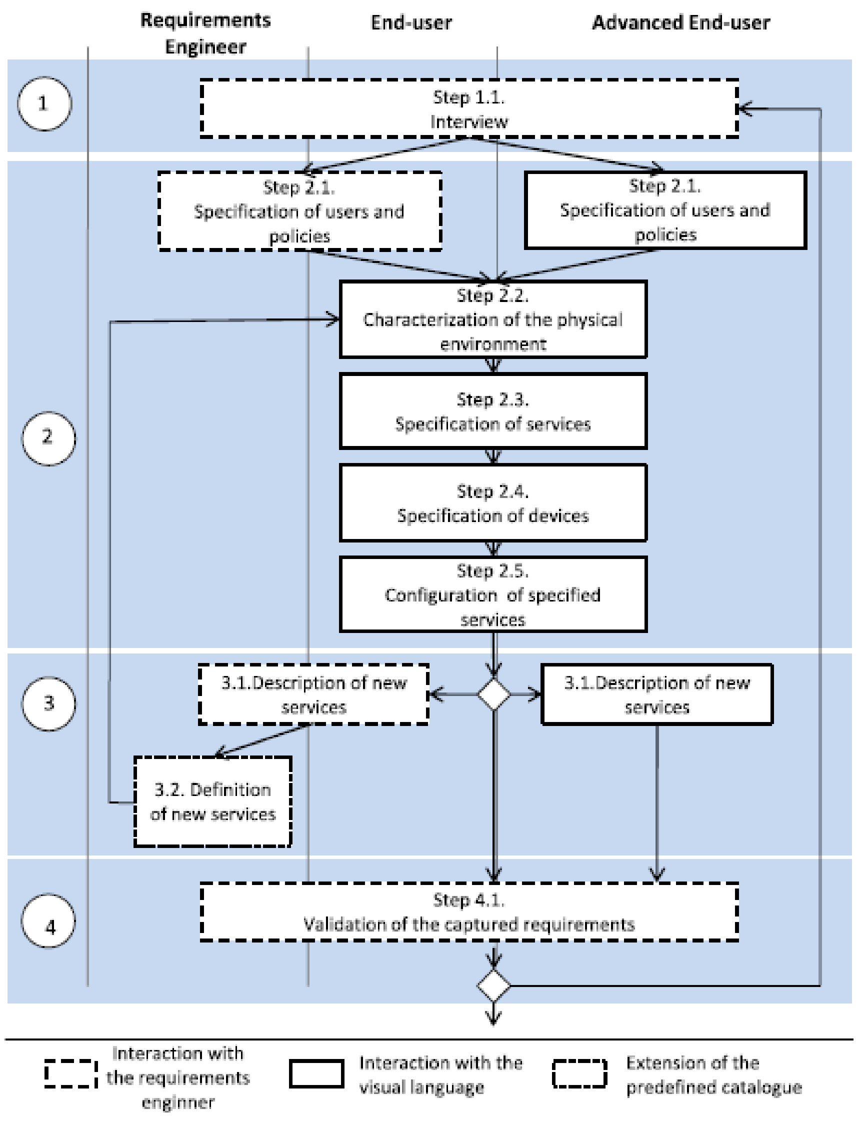
- End-user which are not familiar with computers.
- Advanced end-users that have some computer knowledge.
- Requirements engineers, who are professional computer experts and support the end-users.

Figure 5 shows the natural requirements elicitation process for this tool. The term natural states that something works the way people expect. This process is supported by Natural Programming, Visual programming and Programming by Example approaches. The process is split into 4 phases:

1. Context scope:
   The requirements engineer adapts the characteristics for the tool such as the profiles of the end-users and the domain of the system.
2. System specification:
   Describe the main characteristics of the pervasive system. Advanced end-users can define their own requirements. Regular end-users select predefined requirements from a catalogue and define requirements only with support of the requirements engineer.
3. Advanced system:
   The new requirements get refined through defining new services and their required configurations. The requirements engineer integrates the new predefined requirements into the catalogue.
4. Validation:
   The requirements engineer and the end-users validate the collected information or repeat the process to correct the information iteratively. The validated description is taken as a foundation for writing the formal requirements specification.

The architecture of this tool is composed of an interface, a controller, a feature model and a repository. End-users specify requirements using natural visualization techniques on the interface. The controller checks for completeness of the descriptions. In each step the controller is present. The predefined catalogue is represented by the feature model. The descriptions are stored in the repository.

This tool focuses on the integration of the end-user from the beginning in the requirements elicitation process. One reason is to reduce the difficulty that the client doesn’t know what he really wants. In the future the focus is on automatically transcribing the descriptions into a formal requirements specification.



Requirements Engineer
End-user
Advanced End-user
1
Step 1.1. Interview
2
Step 2.1. Specification of users and policies
Step 2.1. Specification of users and policies
Step 2.2. Characterization of the physical environment
Step 2.3. Specification of services
Step 2.4. Specification of devices
Step 2.5. Configuration of specified services
3
3.1.Description of new services
3.1.Description of new services
3.2. Definition of new services
4
Step 4.1. Validation of the captured requirements
Interaction with the requirements enginner
Interaction with the visual language
Extension of the predefined catalogue

Fig. 5. Natural requirements elicitation process [12]

3 Mobile Requirements Elicitation

Nowadays end-users are familiar with mobile devices. Using an application to document requirements during daily work is effortless and easy. Mobile devices also feature images, audio and video which can be integrated into the documentation.

Considering the huge variety of software which is available in one location, a so called IT ecosystem [16], there need to be ways for identifying the systems for which end-users want to report needs to. Mobile device applications can be very helpful for wide-audience requirements engineering (WARE) [18], since the end-users for such systems cannot easily be reached.

These two approaches were implemented. The first one is called iRequire, it is a tool to use in situ, where the end-user is known. The second approach is called ConTexter, it is a tool to use in an IT ecosystem where the end-user could be anyone.

3.1 iRequire [17]

End-users mostly take part in elicitation techniques such as interviews [7], workshops [6] to discuss their needs but many practices are forgotten if one is out of the context. This calls for a need to document requirements right when they happen.

The iRequire [17] is an approach for an end-user tool to use in situ. Nowadays end-users are familiar with mobile devices, therefore iRequire is implemented as a mobile device application. End-users can document their needs whenever and wherever they want. An advantage of mobile devices is that they support the documentation with different media types such as image, audio, video and text. Seyff et al. [17] introduced three elicitation steps for iRequire shown in Figure 6.

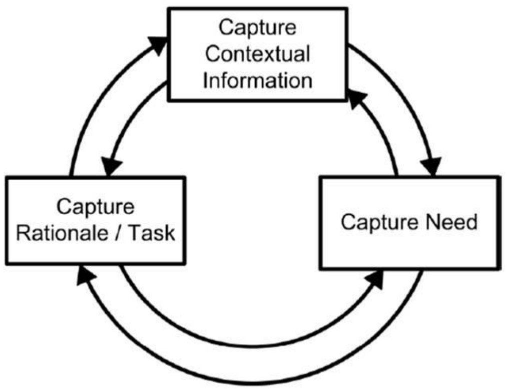


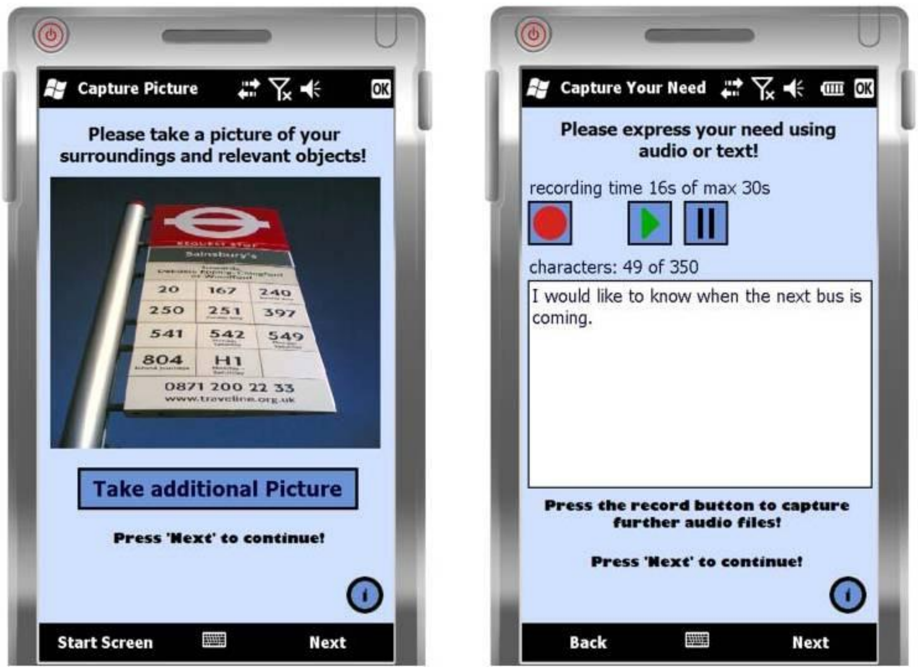
Fig. 6. The iRequire Approach [17] – this figure shows the C*apture Ration / Task* and the *Capture Need* and the *Capture Contextual Information*.

1. Capture Contextual Information:
   Give insights of the end-users environment with text, image, audio and video.

2. Capture Needs:
   Document upcoming needs with textual descriptions or audio recordings.
3. Capture Rationale / Task:
   Explain textually or with an audio recording the importance of a requirement or which task does get supported by the requirement.

To keep flexibility the order of the steps can be carried out as one likes, although the sequence described above is suggested. Important to note is that the iRequire approach is not a brainstorming tool, it focuses on requirements discovered during daily work. After gathering the requirements, requirements analysts can analyze and transcribe them into a formal requirements specification.

The implementation of the iRequire prototype is based on a Windows Mobile smartphone. The screens are structured into a four-step wizard (see figure 7 and 8), which allows a step-by-step guidance for the end-user.


Capture Picture

Please take a picture of your surroundings and relevant objects!

Take additional Picture

Press 'Next' to continue!

Start Screen | Next

Capture Your Need

Please express your need using audio or text!

recording time 16s of max 30s

characters: 49 of 350

I would like to know when the next bus is coming.

Press the record button to capture further audio files!

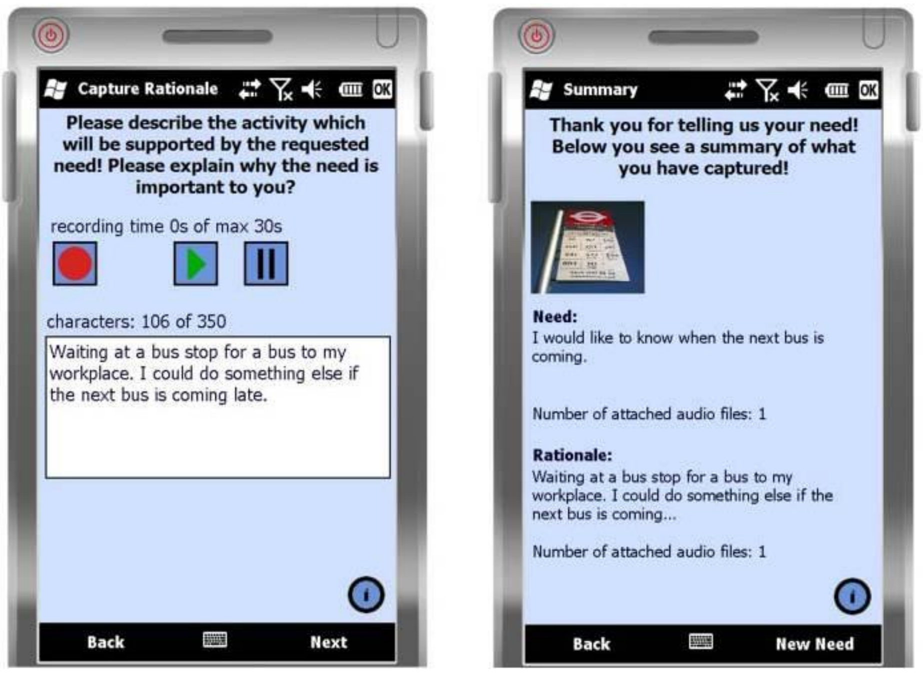
Press 'Next' to continue!

Back | Next

Fig. 7. Taking a picture of the environment (*left*) and documenting a need (*right*) using iRequire [17]

On screen one (see figure 7, left), the surroundings and relevant objects are captured with pictures. Screen two (see figure 7, right) is where the actual need is described with either recording an audio or writing a text. On screen three (see figure 8, left) the rationale or task is recorded or described. Screen four (see figure 8, right) shows a summary of the captured requirement, which has to be confirmed before it is stored in the database.

All screens show back, next and information buttons as well as a virtual keyboard. Besides the input of the end-user, relevant information like location or time of documentation is automatically colleced using GPS and time stamps.



Please describe the activity which will be supported by the requested need! Please explain why the need is important to you?

recording time 0s of max 30s

characters: 106 of 350

Waiting at a bus stop for a bus to my workplace. I could do something else if the next bus is coming late.

Thank you for telling us your need! Below you see a summary of what you have captured!

Need:
I would like to know when the next bus is coming.

Number of attached audio files: 1

Rationale:
Waiting at a bus stop for a bus to my workplace. I could do something else if the next bus is coming...

Number of attached audio files: 1

Fig. 8. Capturing rationale / supported tasks (*left*) and summarizing captured information (*right*) using iRequire [17]

An evaluation study was conducted which concluded that end-users were able to follow through their regular tasks without getting disrupted by recording their needs. For the future it is planned to do more evaluation studies for iRequire to explore more precisely its benefits or limitation. The results from the first evaluation phase will be integrated into the iRequire tool as well as additional technologies such as Bluetooth and RFID. At this stage the collected requirements are stored locally, thus it is of interest to distribute upcoming needs right away and to have an automated transcription into a formal specification.

3.2 ConTexter [16]

An ecosystem in nature is an area where all living organisms interact autonomously with each other and their non-living environment. Similarly an IT ecosystem emphasizes on the emergent behavior of always changing systems, they evolve and are not designed in one piece. Traditionally requirements are elicited before they are implemented, in IT ecosystems they are identified while the system is up and running. Regular workshop and interview techniques are not efficient with wide audience end-users (WAEU) [18].

In public place people are confronted with many different systems for which weaknesses and faults are noticed. People often like to give feedback if it can be done in an effortless and easy manner. The ConTexter tool [16] provides such an approach.

It is implemented as a mobile device application. Since people are familiar with mobile devices, the ConTexter application is effortlessly accessible and feedback is fast and easily provided. But before providing the feedback, the receiving system has to be identified. Mobile devices provide context information, such as geographical and logical positions in the IT ecosystem, which can be identified through sensors e.g. GPS, WLAN, Bluetooth, currently connected URLs.

Figure 9 shows a simplified IT ecosystem with two objects for which feedback can be given. The stakeholder provides context information with the mobile device, such that the ConTexter application can identify the location of the stakeholder as well as objects in the range. The best addressees are selected and presented to the stakeholder so he or she can decide who to give the feedback to.

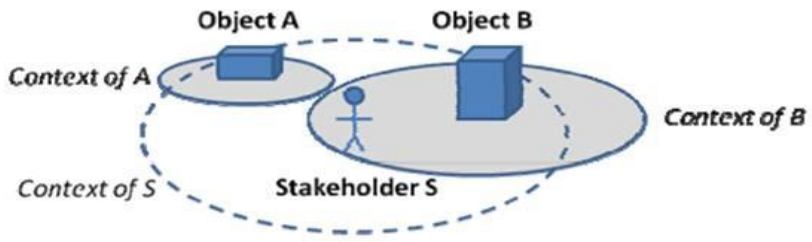


Fig. 9. Context perspective [16]

These objects, which can also be software-free objects, have to register in order for them to get feedback. They are called SmartObjects and are managed by the central administration unit, logically seen as one unit, which can be distributed for robustness or efficiency reasons. Once stakeholders want to give feedback they connect to the central unit. Their location is identified and a list of SmartObjects is presented to the stakeholder. The stakeholder can make a final decision on the addressee. There may be feedback interactions necessary for determining the system. In figure 10 the architecture of the ConTexter framework is shown. The architecture is described in four parts:

- The GUI is the administration (create, define, edit) of the SmartObjects. It is a Java Swing Application that works as a client and connects via Java Remote Method Invocation (RMI) to the SmartObjectAdministration (part of the Central Unit) and the Central ConTexter Unit.
- The mobile device is a G1 Smartphone with Android operating system.
- The Central unit identifies SmartObjects and informs on how to connect to them with the context data provided by the Smartphone.

- The SmartObjectAdministration provides an interface between the Smartphone and selected SmartObjects.

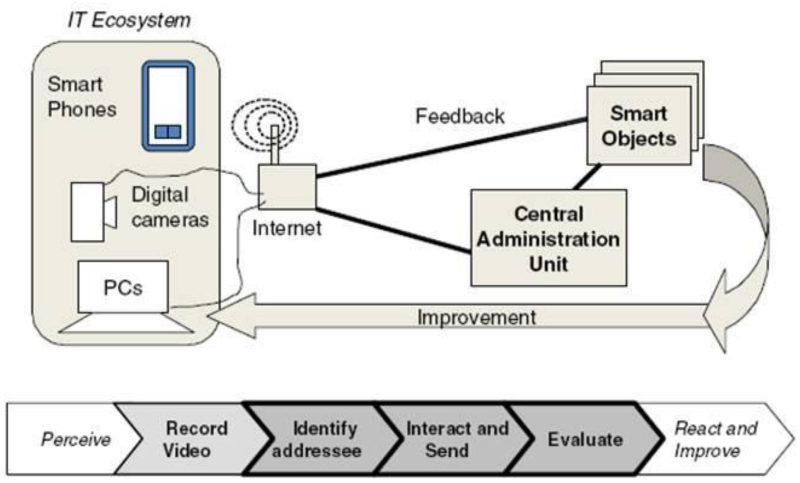


Fig. 10. Architecture of the ConTexter framework [16]

Figure 11 shows screenshots of the ConTexter mobile application used by the UniImprove case study [16]. For this case study the scenario was to improve the university services, software and other university objects using the ConTexter application.

The left screen shows how to choose a feedback category, the middle view presents the decision of a final addressee and on the right the feedback form for submitting is displayed.

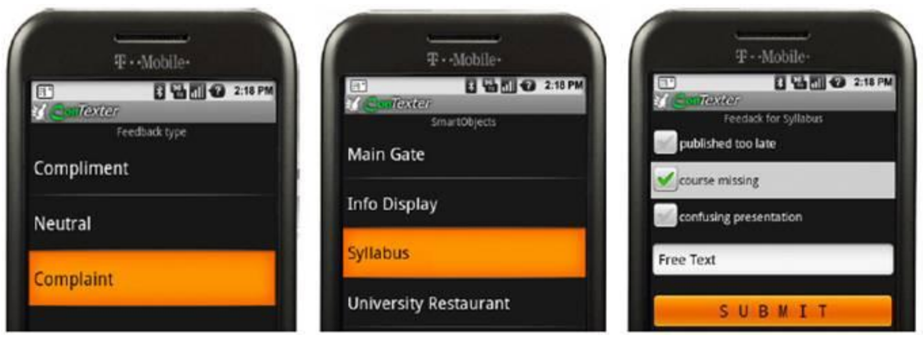


Fig. 11. Choosing category of feedback (*left*), selecting final addressee (*middle*) and submitting feedback (*right*) [16]

Many open questions are still unanswered in this field therefore extended research will have to be done. One possible extension could involve ad-hoc video clips attachment. Currently Schneider et al. [16] are working on an iPhone interface to compare with the Android application.

4 Conclusion

To fill the gap of communication between end-users and IT specialists, different approaches for end-user tools were detected. With visualization tools end-users can annotate requirements in a natural way. Mobile applications are based on a device the end-user is already familiar with, thus documenting needs can easily be integrated into daily activities. In an IT ecosystem everyone can be an end-user. In such a system there are many systems which have to be identified in order for the requirements to be submitted to the correct system.

Table 1. Benefits, limitations and outlook of the discussed tools.

Tool	Benefits	Limitations	Outlook
Annotate!Pro	End-users draw their own needs	No tracking Not formal language	Idea was reused in the Annotation Tool
Annotation Tool / OpenProposal	End-users draw their own needs with help of a template Tracking and discussion platform Automatic information collection Improving collaboration between end-users and IT specialists	Usage of templates may need support End-users have to comply with data privacy No automated transcription	Annotation Tool led to OpenProposal OpenProposal: Extended studies Additional technologies Automated formal specification
Immediate Visualization	Early integration of end-users Clarifying needs of end-users	No automated transcription	Automated formal specification
iRequire	End-users collect their requirements during daily activities without disruption of their daily task	No automated transcription Requirements are stored locally	Extended studies Additional technologies Automated formal specification
ConTexter	Feedback to multiple systems	No automated transcription Find participating end-users	Extended studies Additional technologies Automated formal specification

All these different approaches were implemented as a tool or prototype. Studies were conducted and mostly concluded improvements and for the future more extended studies will be done. In the future these tools will be more refined and provide deeper insights for end-user involvement. Table 1 shows a summary of the discussed tools. I would like to emphasize on the effort to involve end-users from the beginning and the easy to use approaches. The focus is on tools where end-users can work on their own. One of the main limitations and something to work on in the future is to transcribe the collected requirements automatically into formal specifications.

Combining aspects of the approaches could probably help gaining a successful tool. The mobile device is already an important component for the end-user in different approaches, such as the iRequire and the ConTexter application.

One aspect which was barely talked about are privacy issues. Many of the tools include automatic collection of information for providing insights into the environment of an end-user. This could cause end-users to be reluctant in communicating their needs and desires. Letting end-users previewing and agreeing on the automatically collected information could cease this problem.

References

1. Annotate!Pro, http://www.annotatepro.com/ (viewed on 25.03.2011)
2. Beyer, H., Holtzblatt, K.: Apprenticing with the customer. Communications of the ACM, vol.38, no. 5, pp.45-52. ACM Press, New York (1995)
3. Beynon-Davies, P., Holmes, S.: Integrating rapid application development and participatory design. In: IEE Proceedings – Software, vol. 4, pp. 105-112. (1998)
4. Boehm, B.: Software Engineering Economics. Prentice Hall, Englewood Cliffs (1981)
5. Costabile, M., Mussio, P., Provenza, L., Piccinno, A.: End Users as Unwitting Software Developers. In: Forth Workshop on End-User Software Engineering (WEUSE IV). Leipzig (2008)
6. Duggan, E.: Generating System Requirements with Facilitated Group Techniques. In: Human-Computer Interaction, vol. 18, pp. 373-394. Lawrence Erlbaum Associates Inc. Hillsdale, NJ, USA (2003)
7. Goguen, J., Linde, C.: Techniques for Requirements Elicitation. In: Proceedings of IEEE International Symposium on Requirements Engineering, pp.152-164. San Diego, CA, USA (1993)
8. Hickey, A., Davis, A.: Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes. In: 36th Annual Hawaii International Conference on System Sciences (HICSS'03), vol. 3. Hawaii (2003)
9. Jing, http://www.techsmith.com/jing/ (viewed on 25.03.2011)
10. Kolos-Mazuryk, L., Poulisse, G., van Eck, P.: Requirements Engineering for Pervasive Services. In: Second Workshop on Building Software for Pervasive Computing. San Diego, CA, USA (2005)
11. Nielsen, J.: Usability Engineering. Academic Press, Boston (1993)
12. Pérez, F., Valderas, P.: Allowing End-users to Actively Participate within the Elicitation of Pervasive System Requirements through Immediate Visualization. In: Fourth International Workshop on Requirements Engineering Visualization (REV'09). Atlanta, GA, USA (2009)
13. Pohl, K., Rupp, C.: Basiswissen Requirements Engineering. Dpunkt Verlag, Heidelberg (2010)
14. Rashid, A., Meder, D., Wiesenberger, J., Behm, A.: Visual Requirement Specification In End-User Participation. In: First International Workshop on Multimedia Requirements Engineering (MERE'06 - RE'06 Workshop). IEEE Press, New York (2006)

15. Rashid, A., Wiesenberger, J., Meder, D., Baumann, J.: Bringing Developers and Users closer together: The OpenProposal story. Multikonferenz Wirtschaftsinformatik (2008)
16. Schneider, K., Meyer, S., Peters, M., Schliephacke, F., Mörschback, J., Aguirre, L.: Feedback in Context: Supporting the Evolution of IT-Ecosystems. In: Ali, M., Vierimaa, M., Oivo, M. (eds.) Product-Focused Software Process Improvement. LNCS, vol. 6156. Springer 2010
17. Seyff, N., Graf, F., Maiden, N.: Using Mobile RE Tools to Give End-Users their Own Voice. In: 18th IEEE International Requirements Engineering Conference, pp. 37-46, IEEE Press, New York (2010)
18. Tuunanen, T., Peffers, K., Gengler, C.: Wide Audience Requirements Engineering (WARE): A Practical Method and Case Study. Helsinki School of Economics. Sprouts: Working Papers on Information Systems, vol. 4 art. 27. Finland (2004)

Appendix: Pedagogical Approach – How to teach this topic in secondary school?

This appendix shows the preparation done for teaching this topic in secondary school. The learning objectives shown in table 2 describe the time frame of the lessons, who is the intended audience, what is the topic, what methods will be used during the lesson and what tools are used and have to be available. The most important part is identifying the different learning objectives, the mission statement, the disposition objectives, the operational objectives and the fundamental idea.

Table 2. Learning objectives.

Time	3 lessons
Class	Students in the last years of secondary school (age: 17-18)
Topic	Requirements Elicitation – Tools for end-users
Methods:	Discussions with plenum and in groups of two Teacher presentation and demonstration Individual exercise
Tools:	Blackboard / overhead projector Computer: • PowerPoint Presentation • Annotate!Pro • Individual computers for students
Learning objectives:	1. **Mission statement:** Studies show that 60% of project failures fall into the requirements engineering phase [4] and mostly aren't discovered until late during the project or when the system has already gone life [13].The later the error is detected the more expensive is the rectification [4]. Since missing or incomplete requirements cause projects to fail, it is important to find solutions for improving the quality of requirements. 2. **Disposition objectives:** Students know that the communication of requirements is difficult between end-users and IT specialists therefore it is important to find a common language through end-user tools.

	Students have learned about the different approaches for end-user tools and are aware of the importance of good requirements. 3. **Operational objectives:** Students know how important the requirements engineering phase is. They can argument that eliciting good requirements are difficult and end-user tools are necessary to overcome the communication gap Students can explain visualization tools and mobile tools and know at least one example of each. **Additional: Fundamental idea** Introducing the requirements engineering phase, why end-user involvement is important and getting to know visualization and mobile tools for end-users.
Comments:	• For these lessons no previous knowledge is necessary. The focus is on the end-user which are people without or little IT background therefore the tools should be common sense and easy to use. • Depending on the overall plan of the course, I could imagine that a software development project could be part of it. This project could involve all software development phase such that requirements engineering is part of it. Students can work after this introduction on their project and implement what they have learned. I could see some role playing games where they split up in three groups, the first group is the end-user (client), the second is the requirements analyst and the third represent the software developer.

Table 3 shows a precise schedule for three lessons. This topic covers short teacher presentations and many interactions with the students through discussions. Because of the recency of the end-user tools and their implementations, it is difficult to give the students more hands-on experience. That's why the focus is on the discussions for introducing at least the mindset of this topic.

Table 3. Schedule for three lessons.

Time	What?	Medium	Who?
15'	Homework discussion • Prepare at least 3 things you would change or are needed around you, e.g. digital timetable at the bus stop is missing, unsatisfying features of the iPhone.	Blackboard / overhead projector	Students and teacher
15'	Introduction into requirements engineering • What are requirements? • How are requirements elicited? • Importance of end-user involvement. • Communication problems between IT specialists and end-users. • End-user tools: visualization tools.	Presentation	Teacher
5'	Introducing Annotate!Pro exercise • Where is the tool? • How can they start the tool?	Presentation / Demonstration	Teacher

	• Short example how it works.		
45’	Exercise: Annotate!Pro • Get familiar with Annotate!Pro. • Chose a website or application to annotate. • Think about why you would change, add, remove or do something different, annotate this and take a snapshot. • Team up with someone and send your annotated screenshot. • Let the other person tell you what requirements you communicated in your screenshot. Discuss what was meant and what was understood and why. Take the break individually.	Individual computers	Students
15’	Discuss and gather findings of the exercise.	Discussion	Students and teacher
15’	Introduce mobile tools: • iRequire: tool to use in situ. • ConTexter: used in an IT ecosystem.	Presentation	Teacher
15’	Discuss the different tools - advantages and disadvantages, traceability, privacy issues, etc.	Discussion	Students and teacher
10’	Summarize, outlook and finish up the lesson.	Presentation	Teacher