Community Driven Elicitation of Requirements with Entertaining Social Software

Anna Hannemann, Christian Hocken, Ralf Klamma
RWTH Aachen University
Information systems
Ahornstr. 55, 52056 Aachen, Germany
{hannemann | hocken | klamma }@dbis.rwth-aachen.de

Abstract: With the new business models of the Web 2.0 the need for continuous requirements engineering becomes even more important. Future applications are in 'perpetual beta' and well-understood user needs are a competitive advantage in a billion dollar market. However, user communities have to be addressed with new Web 2.0 style elicitation tools, since support by communities is offered at will in the most cases. In this paper, we research community-driven elicitation processes and their tool support. Identification of user needs with and without proposed Web 2.0 style elicitation processes are modeled explicitly using the strategic modeling approach i*. In a case study we implemented a Bubble Annotation Tool (BAT) for enjoyable, intuitive and traceable interaction within communities performing requirements engineering processes. First experiences with the tool in a study conducted to elicit requirements for an iPhone application are reported and discussed.

1 Introduction

Recent developments in information and communication technology exhibit a rapid increase of web-based information systems. One of the key success components of Web 2.0 [O'R05] services is and will be the support for online communities, thereby introducing social concepts to enable people to meet and interact on the web. However, the application of social software in communities such as in organizations or enterprises is still limited. The essential research question is how to adapt social platforms to the needs of a certain professional community. The emerging business concept based on the Long Tail theory [And04] shifts the focus of many companies from few big customers to many small ones. Due to this business principle a customer turns into a long-tail community, with very diverse, spread all over the world members. This community is subject to an ongoing evolution process. Both the community experience changes over the time (because of learning and knowledge exchange processes within community), and also the community structure evolves (i.e. a community can either expand or shrink, get more loose or more tight, there can be many or no newcomers). Thereby, the requirements of community changes. Obviously, successful development of web services requires new requirements engineering (RE) concepts, which will consider not only technical but also social-psychological and structural aspects of communities.

Turning to research of RE processes, Loucopoulos and Karakostas [LK95] define RE as "a systematic process of developing requirements through an interactive co-operative process of analyzing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained". Other researchers point out interaction and cooperation as main aspects of the process. Jirotka and Goguen describe RE as the reconciliation of social and technical issues [JG94]. The authors stress that information needed for RE is embedded in social worlds of users and managers and can be extracted through interaction with these people. However, realization of an interactive co-operative process is not a trivial task. As it was stated before, requirements are distributed among many people with different backgrounds and interests. Therefore, Gause and Weinberg set the process of developing requirements equal to "a process of developing a team of people who: understand the requirements; (mostly) stay together to work on the project; know how to work effectively as a team" [GW89]. Many researches like Paiva in [Pai06] and Segal in [Seg07] propose to apply the concept of Community of Practice (CoP) created by Wenger [Wen98] in order to establish successful RE teams. Although the idea to create a CoP out of all people connected by a project seems to be very promising for RE, the diversity of potential team members can present a huge barrier for this strategy. An approach to overcome possible communication and co-operation difficulties has to be found.

Nowdays, Web 2.0 has a great impact on knowledge work processes [KCS07]. So why not apply this technique in order to facilitate community driven requirements elicitation? Bug tracking systems widely used within open source projects present a good example of web service application for project management. In our research we concentrate on involvement of peripheral community members in the RE process. We claim that enjoyable and intuitive RE concepts should be provided, so that everybody will be able to express his/her ideas easily. This was also previously recognized by other researchers in RE field [WA08], [BCRS08].

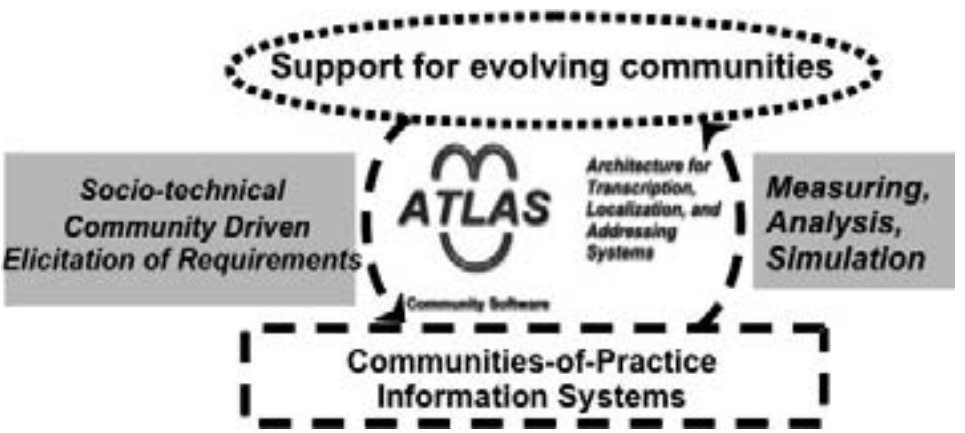

Figure 1: Self-reflexive Information Systems Architecture ATLAS (adapted from [KSC05])

Considering the aforementioned observation on community driven elicitation of require-

ments a reflective system architecture ATLAS [KSC05] previously developed by our group (Figure 1) presents an appropriate solution. The design principles of ATLAS build on ideas of joining usability and sociability by constantly assessing and supporting community needs. In its reflective conception the socio-technical information systems developed on the basis of ATLAS are tightly interwoven with a set of media-centric self-monitoring tools for the communities served. Hence, communities can constantly measure, analyze and simulate their ongoing activities. The usage of MPEG-7 as a basis of provided services is a core concept that helps to ensure interoperability and scalability of ATLAS community information systems. First, in order to analyze already existing possibilities for extraction of community requirements within ATLAS architecture both human and technological actors and their interconnectivity were modeled with i* [Yu95] , as i* has proven to be suitable to capture the various relationships and rationales of the actors involved. Then, the model was extended by tools for community driven elicitation of requirements. According to the sketches model we have designed and implemented Bubble Annotation Tool (BAT).

The rest of the paper is organized as follows: in Section 2 community driven elicitation of requirements modeled with i* [Yu95] is presented. To prove the proposed concept BAT was developed. Its functionalities, structure and also application scenarios are given in Section 3. The technology for data representation and architecture concepts used by BAT are introduced in Section 4. First testing results of BAT for RE are sketched in Section 5. Summary and outlook conclude this paper (Section 6).

2 Modeling Community Driven Elicitation of Requirements

The agent-based graphical modeling language i*, which was developed for early requirements engineering, has proven to be particularly suitable as a modeling instrument in RE because it explicitly deals with dependency relations, besides other notions like actors, goals, resources, and tasks. In order to conceptualize community driven elicitation of requirements techniques and their deployment in the RE world, we first modeled how community needs can be approached without extra facilities (Figure 2, green objects). The whole CoP consists of two main not necessarily independent communities: *developers* and *stakeholders*. The softgoal [Yu95] of the first group is to *satisfy users' needs* of the second group. As we assume that the stakeholders' community is extremely big and its members are spread all over the world, the standard approaches to collect user needs are not applicable. One method we identified is to monitor the system usage (*provide monitoring*). *Monitoring data* can be analyzed and knowledge about user preferences can be extracted (*analyze system usage*). However, this analysis answers only the questions "who" and "how", but not "why" and "what would be better". Next we presumed that several *communication media* (forums, wikis, blogs, etc.) exist, provided either by developers or by third parties, where CoP can exchange information via *communication entry*. These social platforms are normally used to ask questions, to exchange experience or to give advice. Analysis of these data (*analyze comm. data*) can also be used for requirements identification. A big challenge for the analysis presents data uncertainty, data amount and

the complexity of all aspects, which has to be considered. Due to the aforementioned observations, the requirements suggested by these two kinds of analysis are represented in the model by belief [Yu95] (*suggested requirements*). Developers still need to interpret the measured information by themselves, which is identified by the arrow *Help*.

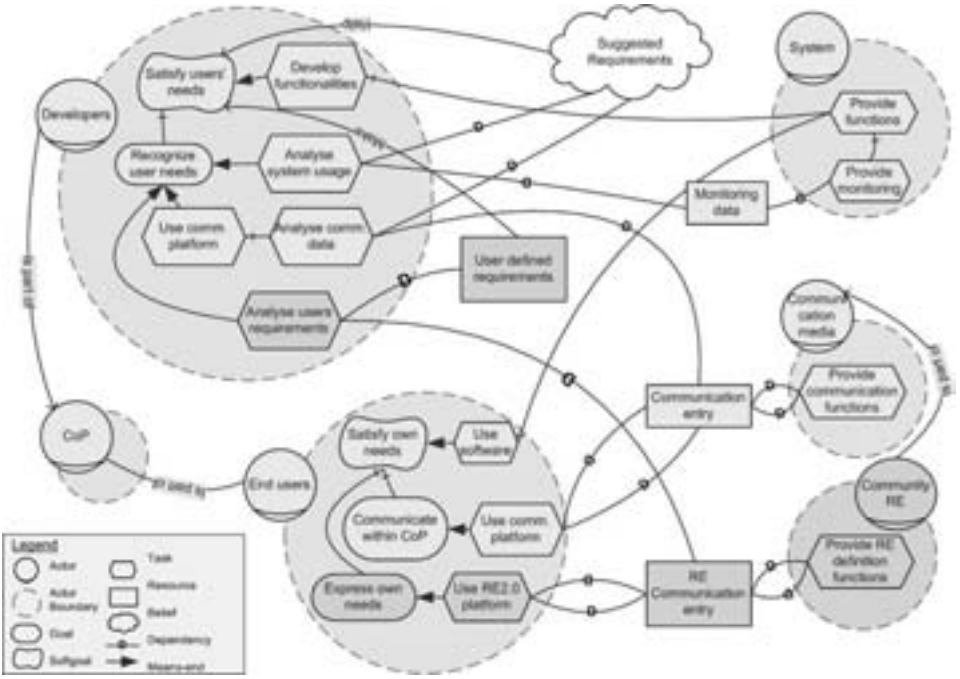

Figure 2: Model of Community Driven Requirements Elicitation

Based on the observation, that community members participate in social communication platforms, we propose to design social software with special target on requirements elicitation. With the lately emerging Web 2.0 technologies, social software delivers mash up platforms for information sharing and social networking. It has also reached out to organizations and enterprises, e.g. Enterprise 2.0 coined by McAfee in 2005 [McA06]. So why not create Web 2.0 services aimed at community support during RE processes? The extension of the model in Figure 2 by such services is shown by rose elements. Social technologies for RE process (*Community RE*) just like the other Web 2.0 should be available from anywhere and support requirements exchange within communities (*RE communication entry*). The user needs identified by the social software for requirements elicitation (*User defined requirements*) can be refined by the combined results from system usage and communication data analysis. A service for requirements elicitation should have a common character of Web 2.0, so that both stakeholders and developers can use it easily and intuitively. Such platform will present a shared repertoire for community and will bring both parts of the community together. However, such community driven requirements elicitation will be only successful, if RE processes will be organized in an enjoyable way attracting users to participate.

3 Proof of Concept: Bubble Annotation Tool

Bubble Annotation Tool (BAT) has been designed to support members of a certain community during requirements engineering processes for applications used by community. Our ambition is to benefit from structures within the community. Especially communities grouped around or linked-up by social software are in our focus, since they are both familiar with Web 2.0 principles and mostly well-connected. Members of such communities are used to exchange knowledge, ideas and desires regarding new releases via mailinglists, chats or instant messaging rapidly. Analyzing such information offers a deep insight into the community. However, attempting to extract desired features from mentioned communication resources is an awkward task since messages are neither classified nor standard formatted.

BAT can help to avoid these difficulties. BAT concept should enable community members to communicate about their requirements in structured, but also enjoyable way. Our intention is to create a typical Web 2.0 environment providing communities with collaborative online annotation functions. In addition, not only requirements stated via BAT but also their meta-data (i.e. author name, creation data, etc.) are stored persistently in a database. The collected data can serve as a source for the further analysis and traceability. BAT consists of two parts, part one is the frontend-client, called *BAT*. It holds the graphical user interface and communicates with an application server. Part two is a service, which extends the application server via a provided API. It interacts with a database, manages requests and connects any number of frontends. The addressed application server we are using is the Lightweight Application Server (LAS) which has been developed at our chair (Section 4).

Frontend

The BAT frontend offers a shared drawing board which allows to clamp several still images, whereas still images might be screenshots, pictures of mobile devices, respectively figures of every fragment end-users can get in touch with. After successful login a user has two possibilities: either to open an image, which has already been annotated by others or to upload a new image which is not known to the system yet. The selected image will be loaded and already existing annotations will be rendered on top. The user is enabled with functions to place comments, remarks or features he/she requests in the shape of speech bubbles anywhere onto the image and point them to arbitrary positions or to respond to already existing contributions. We have decided to use speech bubbles to represent annotations since they are well known from comics, easy to use and intuitive. Moreover, issues related to the interaction between software and user can be addressed directly if screenshots or pictures of input devices are annotated. Thus, remarks are bound to a determined location which helps in analyzing contributions afterwards. Local changes are synchronized instantly with the backend. That permits the frontend to send modified or new created bubbles to the server and to receive updates, which are committed by other users annotating the same image concurrently. Due to its capabilities, BAT is able to support collaborative real-time discussions on the one hand and non-real-time disscussion comparable to forums and mailinglists on the other hand.

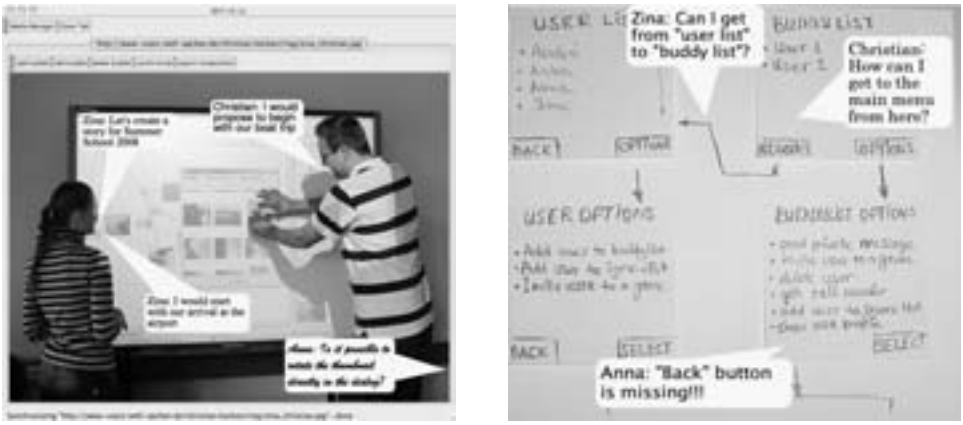

Figure 3: Image on the left hand side shows an application scenario, image on the right hand side shows the annotated paper prototype

In BAT each speech bubble is represented by an independent object. A speech bubble object stores all required data like its *creator*, *content*, *position*, *creation date*, *modification date* and *size* and it can paint itself to a passed graphics object. Thus, every speech bubble is easy to manage. The underlain drawing board, that holds image and bubbles, just has to implement a surface, which offers a canvas, a storage structure to keep track of every bubble and a function, which determines, whether a bubble was highlighted. Another functionality that deals with a certain annotation can be handled within the appropriate speech bubble object. As mentioned above, BAT interacts with a backend and speech bubbles are passed to and received from an application server. Serialization (and deserialization at the counter part) is completly handled by the underlain LAS system.

Application Scenarios

BAT presents an interactive communication environment for all participants in a certain RE process. Any number of users from anywhere can collaboratively annotate any image. The left hand side in figure 3 and the right hand side represent two different application scenarios.

The figure on the right hand side shows a picture of a paper prototype of a new application for a mobile phone. After the creation of the prototype, the designers can upload their results via BAT thereby making them available to the rest of the community. Now, the community members have the possibility to ask questions, to report problems, etc. by means of BAT. Also the designers are not excluded of the communication process: they can create bubbles in order to answer questions. The figure on the left hand side presents a system scenario. Here, designers and developers have annotated the picture to make the system usage more intuitive and clear to others. In a case a user has another scenario in mind, he/she can easily present it to the CoP by modifying the bubble with BAT.

The usage of BAT is not restricted to these two cases. In language of bubbles a project plan, a software architecture or a concept map can be discussed. In order to tighten the

community we suggest to use BAT for annotating not only project materials but also community pictures: e.g. from together visited conferences, exhibitions, etc. In that way, even the community experience will be shared, as those who have not participated in the pictured situations can gain information about missed events.

BAT-Service

The backend called *BAT-service* and presents a service, which is executed by the application server (LAS) [SKJR06], developed in our research group. BAT-service implements an interface with methods that can be invoked by the frontend. These methods offer a bunch of functionalities like *creation*, *retrieval* and *storage* of speech bubbles but also *access control* and *management functions* for pictures. The task of the service is to map speech bubble objects received from clients to valid MPEG-7 and, additionally, build speech bubble objects from existing MPEG-7 data which can be transmitted back to the frontend. Thus, complex operations like storing annotations in an appropriate format are swapped to the backend and the underlain application server. Due to the fact that LAS offers an API with management functions dealing with the manipulation of XML data, there is no need to parse single MPEG-7 documents by the backend itself. Bubbles and image metadata are sent to or received from an XML database (see Section 4 for further details).

Annotations in MPEG-7

Listing 1: MPEG-7 annotation example

```
<Mpeg7>
 <Description type="ContentEntityType">
  <MultimediaContent type="ImageType">
   <Image id="img_160440431">
    <SpatialDecomposition gap="true" overlap="true">
     <StillRegion fontSize="25" fontType="Arial" textType="scene"
                  id="srt_163011957"  type="ImageTextType">
      <SpatialMask>
       <SubRegion>
        <Polygon>
         <Coords dim="2 5">
          501 482 701 482 701 582 501 582 795 540
         </Coords>
        </Polygon>
       </SubRegion>
      </SpatialMask>
      <Text lang="en">
       Anna: Is it possible to rotate the thumbnail directly in the dialog?
      </Text>
     </StillRegion>
    </SpatialDecomposition>
   </Image>
  </MultimediaContent>
 </Description>
</Mpeg7>
```

In order to represent annotations the *ImageTextType Description Scheme (DS)* of MPEG-7 (Section 4) is applied. Speech bubbles are treated as polygons defined by a quintuple holding the coordinates of the pointer plus all anchor points of the bubble's body. MPEG-7

entry of one of the bubble annotations illustrated in Figure 3 is presented in Listing 1.

In this format all interactions with BAT are saved to an XML database. Thus, during the RE an easy to analyze data pool is created. The collected data presents a rich source for both requirements evolution and cross-media network analysis. In [Poh96] Pohl emphasizes that the RE process should be traceable. The author names three applications for the trace information: integration of changes, during system development and for improving acceptance of the system. Hereby, BAT collects all data needed for full traceability: user id, image id and content (Listing 1). The date and time of record creation is saved by the database automatically. Thus, requirements emerged during communication through BAT can be traced.

The three dimensions - user, content, image - present a very encouraging resource for cross-media analysis. Various networks based on different aspects can be extracted from this data. Usually, only two dimensions: user and content are available. The media dimension opens a new possibility for analysis. By means of the analysis of the CoP network roles of community members can be identified and, therefore, the knowledge creation and sharing can be approved [RCB02].

4 Technologies

Lightweight Application Server

The Lightweight Application Server (LAS) [SKJR06] is a system developed and continuously improved in our research group. LAS is completly written in Java and therefore platform independent. Current installations are running on *Solaris*, *Linux*, *Mac OS* and *Windows*. LAS is designed as a modular system and provides an API, whereby developers can simply write extensions. Such extensions - in LAS terminology "services" - enhance functionality and implement new features, which can be executed by connected clients (e.g.the BAT backend introduced in previous section is a service). One advantage of LAS is out of the box support for tasks dealing with MPEG-7. Applications, which draw on LAS, are able to handle a huge amount of metadata easily. The API provides functions for interaction with an XML-database storing MPEG-7 documents. Examples of offered methods are *add*, *remove*, *update* or *execute xPath*. Further advantages affect the processing of MPEG-7 data within services. There is no need to parse XML files. Data received from or sent to the database are expected to be binding class instances, which present objects created from *Apache XMLBeans*. *Apache XMLBeans* is a set of Java classes that can be built automatically for a given XML schema representing XML tags as Java classes [Fou07]. During processing XML data LAS checks all changes against the description scheme in order to verify, whether only valid changes are committed to the database. Otherwise an exception is thrown.

Interaction with users is handled through connectors. Two server-side connectors implementing HTTP and SOAP are available at present. We also provide client-connectors written in Java either using HTTP or SOAP, which can be integrated in new projects. Con-

necting LAS from PHP or Python is possible, too. Thus, new applications are not limited to Java which is indispensable for web 2.0 software. In this way LAS gives us a chance to assemble new ideas related to MPEG-7 rapidly.

Users can connect to LAS and invoke designated service methods, in case they have sufficient access rights. The rights management is very fine-grained within LAS. Nearby the possibility to assign access rights to a single user, they can also be granted to a predefined group of users. Access can be granted to whole services, but also to a single method. Hence, the principles of LAS rights management is comparable with those of the rights management in *UNIX* filesystem.

MPEG-7

MPEG-7 [MSS02] also known as Multimedia Content Description Interface is a powerful ISO standard (ISO/IEC 15938) defined by the Moving Picture Experts Group.

MPEG-7 is designed to store metadata for any kind of media. Thus, it is not a standard to encode content like MPEG-1 and MPEG-2, but to describe it. Typical application areas of the standard are digital libraries, broadcasting stations, etc. or, summarized, all institutions, which have to handle a huge amount of media, whereby handling implies effective search and retrieval.

MPEG-7 is XML-based and roughly defines three parts:

Descriptors Descriptors are used to define specific features of multimedia content like spatial regions or titles within a scene.

Description Schemes Description Schemes are predefined structures which manage the relationship between different components. Components may be Descriptors or Description Schemes.

Description Definition Language The Description Definition Language is xml-based and defines the structural relations between descriptors. Furthermore, new description schemes can be created or existing ones can be modified.

Due to the fact that the MPEG-7 language is an extension of the XML language one can revert to technologies like xQuery and xPath to query for media information. This becomes even more powerful, if all data is stored in an appropriate XML database. One benefit of using MPEG-7 in requirements engineering tools is its interoperability. Due to the fact that it is developed by a huge community a lot of tools exist, which can handle MPEG-7 data. Thus, annotations are easy to handle even, if a tool which generated metadata for the media file is not available.

5 First Results on Requirement Elicitation with BAT

One of the core community supporting services of LAS is New Media Viewer (NMV) [KCG+08]. NMV provides a wide range of image editing and annotation functions. Users can tag, upload and search any type of media e.g. images or videos. This media management service runs on both desktop and mobile devices like Nokia N95. However, hardware does not stay constant. Emerging new technologies result in an evolution of community needs. The iPhone presents such an innovative technology, which finds continuously increasing popularity among people. Therefore, our community decided to develop an iPhone version of NMV.

The RE process for the new NMV version for the iPhone (iNMV) was executed in two separate groups of community members. The first group was equipped with printed out NMV screenshots. The second group was equipped with BAT. Also the possibility to upload images (e.g. screenshots) to the media repository, which can then be annotated via BAT, was provided. Both groups worked very efficiently and generated many requirements for iNMV. After "RE session" surveys with questions about the RE process were filled out by every participant. One of the important findings was, that all members of the BAT session rated the question *"it made fun to express my requirements"* with *"strongly agree"*. Whereby, in case of session without BAT almost 30% selected an option *"undecided"*. These results supports our hypothesis, that BAT serves enjoyable RE. Further, more people using BAT felt, that it was easy *to specify the requirement they wanted to discuss* (80%, *"strongly agree"*) and *to understand what their colleagues were talking about* (40%, *"strongly agree"*), than participants using paper screenshots (42% and 28% respectively). An explanation for these results was found in the process reflection after the session. Participants of the session without BAT pointed out that *"a facility to make an issue under discussion available to everybody was missing"*. As everybody was working on his/her personal sheets, it was difficult to explain own ideas and understand those of others. In case of using BAT this problems did not occur, because the actions of the group were visible for everybody after each synchronization. Although, there were only 9% more satisfied users in session with BAT (80%) than without BAT (71%), several participants of the latter one pointed out, that *RE worked so well, only because of a small group size* (7).

Based on the RE workshop results, it can be concluded, that BAT was accepted by the community. The application of BAT can increase the funniness of the RE process. Also the processing of workshop outcomes generated with BAT was easier than results sketched on printouts. Bubble annotated pictures are saved in MPEG-7 format in a DB2 database and, therefore, can easily be analyzed with xPath. Whereby, the paper prototypes can only be analyzed manually. In spite of generally positive feedback on BAT from the users, many suggestions for the improvement of BAT were made by the community. Also participants presumed, that *"BAT will have greater advantage in cases, when RE participants have to work distantly"*. In the next future a distant RE session with and without BAT is planed.

6 Summary and Outlook

In this paper we proposed to apply social software for co-operative and interactive RE. We analyzed existing concepts for RE process. The i*-model of possible requirements elicitation for world wide spread communities was presented in this paper. Shared repertoire presents one of the three dimensions by which practice becomes a source of coherence of a community. Concentrating on the aspects of CoP building, we defined the need for intuitive, enjoyable and interactive tools. We claimed that social software presents a very promising approach for community driven requirements elicitation. As a proof of concept Bubble Annotation Tool has been created. BAT allows community discussion in form of speech bubbles. Any number of users from anywhere in the world can communicate by means of bubbles, synchronously. A great benefit of BAT is that all data created by this service is saved to a database in MPEG-7 format. Thus, the RE process is continuously traced and a data analysis based on different aspects (e.g. media, author, content) can be easily performed.

First experiences with the tool in a study conducted to elicit requirements for an iPhone application showed, that BAT makes the RE process more enjoyable for the users. The users provided several suggestions on BAT improvement. Their realization is current work in progress. At the same time we started creating a navigation dashboard composed of various cross-media analysis methods applied to the data collected by BAT in order to realize community-awareness.

References

[And04] Chris Anderson. The Long Tail. *Wired*, October 2004.

[BCRS08] Bernd Brügge, Oliver Creighton, Max Reiss, and Harald Stang. Applying a Video-based Requirements Engineering Technique to an Airport Scenario. In *Third International Workshop on Multimedia and Enjoyable Requirements Engineering (MERE'08)*, September 2008.

[Fou07] The Apache Software Foundation. Apache XMLBeans, Juni 2007.

[GW89] Donald C. Gause and Gerald M. Weinberg. *Exploring Requirements: Quality Before Design*. Dorset House Publishing Co., Inc., New York, NY, USA, 1989.

[JG94] Marina Jirotka and Joseph A. Goguen, editors. *Requirements engineering: social and technical issues*. Academic Press Professional, Inc., San Diego, CA, USA, 1994.

[KCG+08] Ralf Klamma, Yiwei Cao, Anna Glukhova, Andreas Hahne, and Dominik Renzel. Nonlinear Story-telling in a Mobile World. In K. Tochtermann and et. al, editors, *Proceedings of I-Know'08 and I-Media'08, International Conferences on Knowledge Management and New Media Technology*, pages 218–225, Graz, Austria, September 3-5 2008.

[KCS07] Ralf Klamma, Yiwei Cao, and Marc Spaniol. Watching the Blogosphere: Knowledge Sharing in the Web 2.0. In N. Nicolov and et al., editors, *International Conference on Weblogs and Social Media*, pages 105–112, Boulder, Colorado, USA, March 26-28, 2007.

[KSC05] Ralf Klamma, Marc Spaniol, and Yiwei Cao. Community Hosting with MPEG-7 compliant Multimedia Support. *Journal of Universal Knowledge Management*, 1(1):36–44, 2005.

[LK95] Pericles Loucopoulos and Vassilios Karakostas. *System Requirements Engineering*. McGraw-Hill, 1995.

[McA06] A. P. McAfee. Enterprise 2.0: The Dawn of Emergent Collaboration. In *MIT Sloan Management Review*, volume 47, pages 21–28, 2006.

[MSS02] BS Manjunath, P. Salembier, and T. Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*. Wiley, 2002.

[O'R05] Tim O'Reilly. What Is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software. *www.oreillynet.com*, 2005.

[Pai06] Elizangela Andrade Paiva. The Test Community of Practice Experience in Brazil. *Global Software Engineering, 2006. ICGSE '06. International Conference on*, pages 247–248, Oct. 2006.

[Poh96] Klaus Pohl. *Process-Centered Requirements Engineering*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[RCB02] Andrew Parker Rob Cross and Stephen P. Borgatti. A bird's-eye view: Using social network analysis to improve knowledge creation and sharing, 2002.

[Seg07] Judith Segal. Some Problems of Professional End User Developers. *Visual Languages and Human-Centric Computing, 2007. VL/HCC 2007. IEEE Symposium on*, pages 111–118, Sept. 2007.

[SKJR06] Marc Spaniol, Ralf Klamma, Holger Janßen, and Dominik Renzel. LAS: A Lightweight Application Server for MPEG-7 Services in Community Engines. In K. Tochtermann and H. Maurer, editors, *Proceedings of I-KNOW '06, 6th International Conference on Knowledge Management, Graz, Austria, September 6 - 8*, J.UCS (Journal of Universal Computer Science) Proceedings, pages 592–599. Springer-Verlag, 2006.

[WA08] Amanda M. Williams and Thomas A. Alspaugh. Articularing software requirements comic book style. In *Third International Workshop on Multimedia and Enjoyable Requirements Engineering (MERE'08)*, September 2008.

[Wen98] Etienne Wenger. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press, Cambridge, UK, 1998.

[Yu95] Eric Yu. *Model ling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, 1995.