Enterprise 2.0 in Action: Potentials for Improvement of Awareness Support in Enterprises

Hilda Tellioğlu
Vienna University of Technology
Institute of Design and Assessment of Technology
Multidisciplinary Design Group
Vienna, Austria
Email: hilda.tellioglu@tuwien.ac.at

Simon Diesenreiter
Vienna University of Technology
Institute of Design and Assessment of Technology
Multidisciplinary Design Group
Vienna, Austria
Email: simon@backlab.at

***Abstract*—In this paper we investigate conceptually and empirically in a software development company whether Enterprise 2.0 components contain awareness mechanisms. As a result, we introduce additional mechanisms to take the first step to improve awareness in complex cooperative work environments. After a short introduction to the concepts awareness, awareness mechanisms, and Enterprise 2.0 we describe a case study to find out patterns of awareness in collaborative work processes and the missing awareness support. We approach the problem by trying to understand which Enterprise 2.0 components are related to which awareness mechanisms, and to which degree Enterprise 2.0 fulfills awareness requirements of complex collaborative work. Our study results in the identification of two different categories: system- and user-related awareness mechanisms. Search, extensions, and signals – supporting system-related mechanisms – are the most common components that are already established by different tools. Authoring, links, and tags – assisting user-related mechanisms – on the other hand, have not been utilized yet. They are very powerful to create context and capture collective knowledge. To support this, we introduce additional awareness mechanisms like enter, annotate, rate, share, reference, select, mark, and label, to show how these three components can be implemented in enterprises. By doing so, we present the potential of Enterprise 2.0 to support awareness in cooperative work. The new map of awareness mechanisms to Enterprise 2.0 inform not only the developers of tools supporting (aware) collaboration but also practitioners working in teams to define their requirements to such tools.**

Keywords–Awareness; Enterprise 2.0; distributed software development; cooperative work; CSCW

I. INTRODUCTION

Enterprise 2.0 was introduced by McAfee to show how Web 2.0 technologies can be used in enterprises [21]. He focused on how to make work practices and efforts of knowledge workers visible. When introduced, the Web 2.0 technologies have fascinated almost everyone but unfortunately not the enterprises. To take these emerging technologies (or also any other) seriously, companies needed real case studies, in which the application of Web 2.0 has been already studied and analyzed from such perspectives which are relevant for enterprises, like business, cost and benefit factors as well as security, consistency, or scalability, to name some. They also needed best practice descriptions and guidelines showing how to introduce and apply Web 2.0 technologies in their organization.

How was Enterprise 2.0 discussed and studied in other areas? In Computer Supported Cooperative Work (CSCW) research Enterprise 2.0 was not really studied, understood, or accepted. Hence, there are no in depth qualitative analysis on Enterprise 2.0 in work context and its impact on work. Studies on social network systems (SNS), especially about Facebook and Twitter, on the other hand, exist in each direction, detail, quality, and format. In the meanwhile there are even qualitative studies available dealing with their application and real use. In the last years, some managerial journals published new results of applied Enterprise 2.0, some related books have been published with an emphasis on providing how to guidelines for management in enterprises. However, they did not really exhaustively and qualitatively describe how and in which areas such technologies are more efficient and of additional value to cooperative work. The research literature still lacks of in-depth studies and analysis of cooperation processes in complex development environments. We claim that to better understand the real use of Web 2.0 technologies in enterprises, we need to investigate these settings by basing the analysis on central concepts of CSCW, like awareness, trust, openness, sharing, etc. And, exactly that investigation is what we are presenting in this paper. Here we only focus on the aspects of awareness.

Enterprise 2.0 is a big chance for enterprises to learn from successful collaboration and cooperation patterns shown by Web 2.0 platforms [20]. We claim that Enterprise 2.0 basically builds a bridge between SNS and groupware. Using Enterprise 2.0 we can start applying SNS components in enterprises to meet groupware requirements. We can start to think about links and transitions between person-oriented and group-oriented communication, bottom-up and top-down implementation, voluntary and enforced participation, co-evolved conventions and pre-planned ways of group work, large number of users with no project limitation and small number of users over a limited period of time. By doing so, we can make use of Enterprise 2.0 components as a guiding template for analysis of SNS mechanisms and technologies in groupware infrastructures. This helps in identification of additional enterprise specific mechanisms, probably not considered in studies on SNS so far.

Most of the enterprises do have an Intranet and special solutions for their business and organization already established for a very long time now. Convertino et al. [5] studied tools and practices like the generation, sharing, and organizing con-

COLLABORATECOM 2013, October 20-23, Austin, United States

DOI 10.4108/icst.collaboratecom.2013.254066

tent with Wikis. The so called collective intelligence contain processes that allow users to adapt and control their social, informational, and physical environment. Studies tell that Web 2.0 tools are mainly used for personal purposes. However, the top-down structures of enterprises in compare to bottom-up organization of personal environments of single persons working in these enterprises facilitate completely different types of interactions when it comes to apply Web 2.0 technologies for exchange and collaboration. Skeptical managers refused for a very long time to include such SNS into their Intranet. They were not convinced of its relevance, efficiency, and focus. SNS would rather distract employees during work by occupying their costly time for nothing without offering enough benefit for the company as well as create a mechanism for leaking confidential information about the company or its products or services.

In the meantime Web 2.0 technologies have got the attention of enterprises. Companies try to find incentives to motivate their employees to participate in enterprise SNS. Farzan et al. [9] developed a particular participation incentive mechanism for employees at IBM to reward their commenting behavior. The impact of a points-based system on motivation was definitively high. Later on Thom et al. [30] studied the application of the same awarding scheme on contributions to lists, photos, and comments. Morrison [22] suggested that geographically distant members of an enterprise use SNS as a way to learn about their organizational culture and build one's reputation.

Given the fact, that Web 2.0 technologies and mechanisms are well-studied and -understood by researchers and software developers, now it is the time to map these mechanisms to concepts relevant for enterprises, like awareness, efficiency, organizational learning, knowledge management, trust, cooperation support, etc. This makes Enterprise 2.0 accessible for managers and so for enterprises. In this paper, we try to achieve this goal by focusing on awareness mechanisms related to Enterprise 2.0 on an example of distributed software development (DSD).

Awareness mechanisms in the CSCW literature mainly focus on system-related functions [28] [29]. They do not consider additional possibilities given if users become part of the system, like in the prosumer approach introduced by Web 2.0. This paper shows how user interactions and user involvement can improve awareness in enterprises. Based on our in depth study in a software development company and the literature research on concepts of Enterprise 2.0 and awareness in the area of CSCW, we identified awareness mechanisms in the utilization of tools. We could also identify new areas where such support was lacking. We analyzed our findings with available state of the art literature about awareness and introduced new awareness mechanisms that are embedded in some of the Enterprise 2.0 components. We present our study and analysis by putting our new ideas in a map (see Table I). In this map, we show how awareness mechanisms including the new ones we introduced are related to Enterprise 2.0 components. We claim that this enhancement will inform not only the developers of tools supporting (aware) collaboration but also practitioners working in teams to define their requirements to such tools.

In the next section, we summarize the familiar awareness mechanisms from the literature of CSCW to create a ground for our case investigations. In Section III we present our case – the setting, the processes, and the relevant tools. We describe where we found Enterprise 2.0 components in our case and what awareness mechanisms these contained. Finally in Section IV we present our results in detail. We propose three aspects – authoring, links, and tags – to enhance awareness mechanisms provided by Enterprise 2.0 so far. Additionally we map awareness mechanisms to Enterprise 2.0 components, before we conclude the paper.

II. Familiar Awareness Mechanisms

In this section we want to introduce briefly which awareness mechanisms we are dealing with in our analysis. We mainly refer to concepts from the CSCW literature because our focus is the collaboration support by computer systems. Awareness can be defined as providing a context for team members' activities in order to understand each others work [4]. It can be also considered as a mechanism to enhance coordination and efficiency in team work [15]. Stretching its definition, it has also been seen as an attribute of action, doing it "heedfully, competently, mindfully, accountably" [26].

In coordination research in distributed software development, awareness has been studied as a concept by considering several aspects [27], like information regarding the tasks of a project [10] [16], information on source code version management repository [17], social network analysis [25], visualization techniques [31], search for experts [8], tracking artifacts and relationships [18], quality metrics results [24], to mention few. There is still research needed on awareness based on recent information [23], the use of social networks in DSD, especially to maintain awareness on emerging and unplanned interactions. In this paper, we try to focus on the latter by using the concept of Enterprise 2.0 in DSD.

If we focus on group work dynamics during collaboration we can identify five awareness types [23, p.511f]: informal [12], group-structural [14], social [12], workspace [13], and context awareness [32]. Identity is the most studied element for workspace awareness, followed by (past) change awareness with history tracking or real time changes, objects used, and actions made [27, p.137]. Presence has been found as important for informal awareness, availability feature for social awareness, and roles and responsibilities for group structural awareness in DSD.

When considering awareness as a mechanism to facilitate its identification in Enterprise 2.0 components we have to look at the concept of awareness from a different perspective, namely as a mechanism with several elements enabling data exchange (but not completely in the sense of [15]). First of all we have to differentiate between the creation, share, and presentation of awareness information [11]. In this context the modes of active, passive, implicit, and explicit become relevant because they have impact on the people and the systems involved that deal with awareness information. If humans create information they enter data, e.g., a status, *actively* into a system. If data entry happens without involving a human interaction, like by a sensor, the data is gathered *passively*. The same differentiation can be made for the exchange of information. But in that case we have to think in terms of senders and receivers of data. Human can actively and

explicitly request or provide information from and to others or systems. A passive distribution of information occurs when a system forwards certain information to others or other systems without having a human being involved. In data processing or interpretation some data is *explicit*, e.g., status of something, some can be understood only in a certain context and is, that is why, only *implicit*, like data from the history of an artifact.

In the following we will briefly describe such awareness mechanisms that we want to study in the context of Enterprise 2.0: pull, push, check, poll, display, monitor, subscribe, notify, sensor, and filter.

- *Pull* occurs if data is actively accessed by a person or a system because of its relevance. Pull can be executed directly on an important object or an intermediary hosting that information. So, only through an explicit request an information exchange can take place.
- *Push*, on the contrary, means that information is pushed to the person or persons automatically (due to the configuration). Only if the receiver of the information thinks that it is relevant, an information exchange happens. The receiver can decide whether s/he wants to make use of the information or not. In a work environment, there is always a mixture of both mechanisms, whereas the pushing mechanism has been found more efficient [6].
- *Checking* means asking for clarification and verification. It can be also seen as a limited, compact version of pulling. It is like pulling information about a previous communication or about an information available from previous work. Check happens when the context of available information is not clear, or when the available information is not complete or is not enough for the receiver. For less complex activities checking is less needed than for more complex ones [6] [7].
- *Polling* is also a type of pulling in which the request is not defined by a person but by an information platform where persons actively look for status of users or other events [19].
- Transferring information about the work context is called displaying and monitoring [1]. *Monitoring* means observing the behavior of other colleagues or changes done to an artifact, *displaying* means "the implicit or explicit signals a given actor uses to show specific aspects of his or her current situation, which could be useful or relevant for the other actors in the context" [1, p.193].
- *Subscriptions* allow consumers to register on servers or data repositories for certain information or events, in order to be informed automatically if something is changed [2]. Changes can be notified to the subscribers either immediately per document or summarized over a certain period of time for one or more documents. This avoids the information overflow for the subscriber. Filters make possible to limit the areas in which a subscriber wants to be informed by the system automatically.
- *Notifications* are needed to be informed about the changes in the awareness system. In this case email or peripheral information displays are used. The latter is visible as such – like as a sidebar on the display, by taking a very small space on the screen, by disappearing when there is no event. The information can be transferred in real time and synchronously. The amount of data transferred or displayed is also small.
- *Sensors* are further mechanisms helping to register or gather awareness information. They track certain entities in an environment, gather changes occurred, and deliver all data to a component which takes care of the data visualization.
- To protect the privacy and self-determination in information sharing and provide users the possibility to configure the visualization of the selected data we need *filters*. Filters can be created automatically based on the context or manually by single actors.

Here we presented the following awareness mechanisms: pull, push, check, poll, display, monitor, subscribe, notify, sensor, and filter. We will refer to these later in our analysis. In the next section we will present our case.

III. The Case

Our case is about a software company – that we call *SoftCom* – with the head office in North Germany (with 140 employees) and two branches in South Germany and Austria (with 8 employees). Their product is a suite for insurance companies. In *SoftCom* we mainly studied the family of insurances like indemnity, liability, automobile, property and casualty, legal protection, cargo, and credit insurances except health insurance. Being part of the whole software suite, the modules we studied are related to other modules like insurance benefit or damage, document management system, etc. This case is studied in the scope of a master thesis [3].

SoftCom uses water fall model with three project milestones per year. Certain work packages are planned for each milestone. The work is done in cooperation of several departments. Each department has again groups responsible for certain parts of the product and responsible persons for certain components of the parts. The development process mainly consists of four steps run at different departments: product management, domain conceptualization, software development, and tests. These processes are in general sequential, but there are loopbacks possible to the previous steps. For instance, release management is carried out during deployment across all development processes.

Product management department together with the customers are in charge of defining the requirements to the next milestones. People from domain conceptualization take care of detailing and specifying by conceptualizing the requirements gathered by the product management. They prepare the user requirements for the software development department. The result is called "domain concepts" containing detailed descriptions of the functionalities of the product modules and management together with test lists and test case descriptions for the test department. Software is written in the software department and released for testing in cooperation with the

release management people, before handed over to the test department. They report the errors directly to the software developers and domain concept creators.

A. Methods

By means of an ethnographic study we investigated our users, their work environment, their individual and cooperative work processes with all artifacts and tools they used. This is a qualitative study and does not involve a comparison of different work environments or systems. As a consequence, there are no experimental numbers to present, but only narrative descriptions of studied work processes. This way, one user site gives us enough detailed insights on subjects of our attention.

Besides participatory observations (in the Austrian branch over six weeks, where we documented 15 single days with total 153 communication events) we carried out 17 in-depth semi-structured interviews with the most of the key actors. We also studied the Intranet and the tools used. For interviews we selected actors from different departments in Austria and in Germany. In total we needed 1,5 weeks for the interviews whereas each interview was between 25 and 75 minutes long. We audio and video recorded all observations and interviews that we could analyze later on in detail.

B. Tools

In *SoftCom*, work is carried out by using technical infrastructures with network connections, computers, servers, shared network spaces, local and online tools, and software like code repositories and tools for continuous integration like Cruise Control. The availability and proper functionality of these tools must be given any time. The response time when accessing the network is mentioned as an important factor by the developers. We could observe that developers were talking about the status of their technical environment several times.

SoftCom uses mainly two tools: Lotus Notes and JIRA. In this paper we will shortly present the use of TTS, an internal issue tracking system, and REA, the release management tool, both based on Lotus Notes and developed by *SoftCom* for internal use as well as off-the-shelf tool JIRA[1].

Lotus Notes contains an email client, instant messenger, and other intranet-based database applications (see Fig. 1). It is a distributed database-based groupware with a client server architecture. Besides ready to use applications, interfaces are provided by Lotus Notes, which can be used to create the own applications by integrating them with other modules easily through the shared database. As a base infrastructure, Lotus Notes hosts several applications used in *SoftCom*: internal issue tracking system (TTS), release management tool (REA), an application for database and system documentation, and internal time recording system.

Email client of Lotus Notes shows the current status of a sender of an email if logged in *Sametime* – the instant messenger of the system. The visualization is colored. Additionally, the tooltip displays the location of the sender. As usual, if a new email has been received a symbol notifies the receiver at the right bottom corner of the display, an example of peripheral

[1] www.atlassian.com

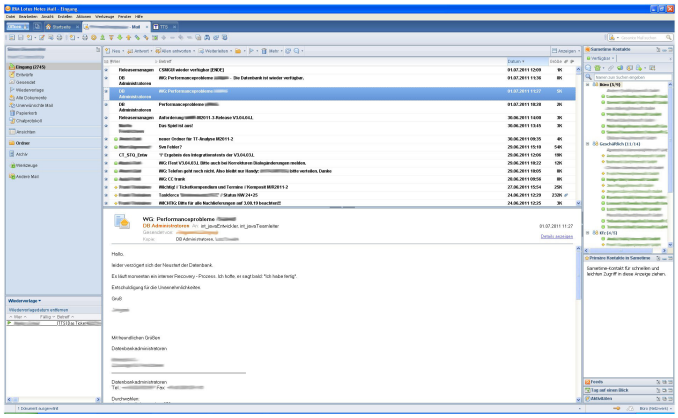

Figure 1. Lotus Notes with several views used in *SoftCom*.

awareness. *Sametime* displays the online status of people in color (green – present and active, yellow – present but not active). If one initiates contact with someone who is online, the telephone number, the local time, the location, a status notification (like “I am available”, “I am not available”, “in a meeting”, “please don’t disturb”) appear. The status notification can be adapted individually. One can also let to display an image of the contact. *Sametime* is always visible and notifies people who are online and using it. In the side bar of Lotus Notes there is other useful information displayed: a list of the favorite contacts, a display for RSS feeds, an overview of different calendar entries (e.g., day view), activities to organize tasks or todos with data of persons assigned, deadlines, documents attached, chat log files, or emails connected. In our interviews, we found out that these additional features are barely used by the team members.

TTS is the internal ticket system that is developed by *SoftCom*. Bug reports are documented and managed by using TTS. It shows an overview of all open tickets, which can be reduced by applying filters defined by users, like “assigned to me”, “opened by me”, “tickets for the customer XY”. The overview page also shows the status of the tickets, like “open”, “closed”, “reopened”, “waiting for customer’s feedback”, “ready to test”, the author of the ticket and the person dealing with, the responsible team, the program modules related, a priority (5 scales: 1, 3, 5, 7, 9, with 9 as the highest priority), an internal – and if available an external – ticket id, a short description of the failure, and other relevant information. The overview can be adapted by users mainly by sorting some of the data displayed. Users can see the single tickets also in detail and edit view, which contain more data. One can easily create an email related to a ticket opened in detail view, which is sent to the person assigned to the ticket. The ticket id is then used as the subject of the email, the description, the external id, and a link to the ticket as message body. One can also create new REA entries directly from TTS. So, TTS uses email notifications as awareness mechanisms. Seeing all tickets, also from other colleagues, in the overview makes all team members aware of the work still open or already finished in the whole team.

REA is another self-developed tool in *SoftCom*. Tickets, which are related to customers and need to be communicated with them, are managed in REA. This way, a new delivery

agreement is negotiated with the customer. It consists of one or more ticket ids, the related software components, the software version(s) that need to be modified, and a status (like “opened”, “accepted”, “rejected”, “withdrawn”, “ready to build”, “executed”). Also in REA there are overview and detail views with a similar filtering and sorting function, which allow to monitor a project. By means of automatic email notifications the people are informed about changes.

JIRA is in fact the Enterprise 2.0 system established in *SoftCom* (see Fig. 2 and Fig. 3). It is the web-based issue tracking system used mainly by the developers. JIRA helps to organize and manage projects, tasks (issues), and subtasks (subissues). Additionally, there are possibilities to order the data by details (specified further as “subjects” or “domain concepts”), persons (identity of the author or person who is currently working on the task) and time and date (of creation and modification of tasks, or finishing a task), attachments (file size, date of upload), tasks (mandatory or optional comments, time records of people related, change history with date, person, comments), and subtasks (name, type, status, person assigned). Versions can be created or assigned easily, either to single entities or to the development stage as general. Each project gets a version number. Projects or tasks can be searched easily by entering keywords. Tasks and solutions have status, like “open”, “processing”, “finished”, or “not finished”. Tasks can be rated and monitored by all. All system users know who has rated and who is monitoring a certain task. JIRA allows integration with other software development tools like code repositories or build systems. If it is configured it shows the repository commits or results of build processes on the status bar.

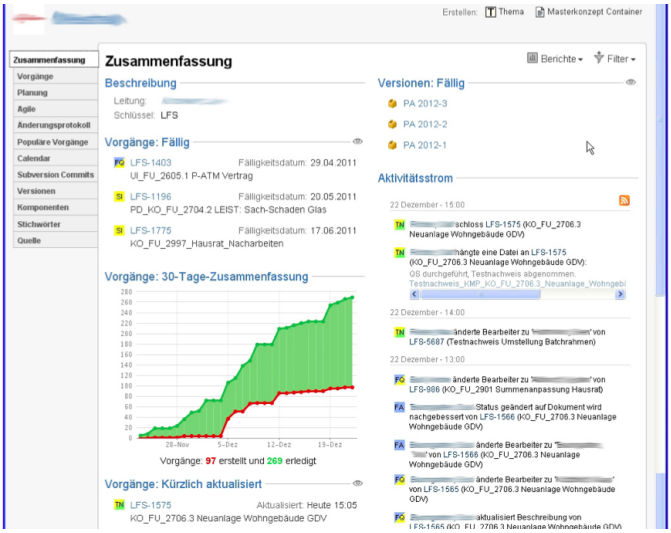

Figure 2. JIRA used in *SoftCom*: A project overview.

The main function of JIRA is to assign tasks to persons or groups as well as to add status to tasks. Status can be configured by users. Assignment is done partly automatically (the creator is always referenced to the task s/he creates) and manually. Each task is connected to a person. Persons or groups must be logged in the system before accessing the JIRA data. Each user must be registered and have a profile. The link between tasks and persons makes task-related

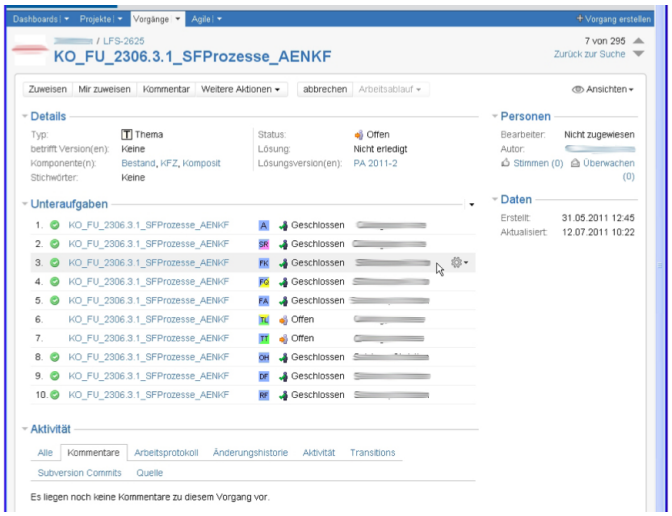

Figure 3. JIRA used in *SoftCom*: A task overview.

browsing through persons possible. Dashboards offer an additional overview of data generated by registered users by using widgets. Widgets are small tables, diagrams, calendar views, or similar things, which can be located easily on the dashboard. They visualize the data that is filtered by person, status, or date. In JIRA, workflows can be modeled and implemented, to manage actions to and status change of tasks depending on their types. Several business rules can be integrated into the system by using workflows.

IV. Results

There are six important components of Enterprise 2.0, the so called SLATES [21]: search, links, authoring, tags, extensions, and signals. Considering the awareness mechanisms briefly presented in Section II (pull, push, check, poll, display, monitor, subscribe, notify, sensor, and filter) and investigating our case (especially the cooperative work practices and the use of tools in Section III) from the awareness point of view, we identified Enterprise 2.0 components used in our case. Furthermore we found out that there are additional awareness mechanisms embedded in some of the Enterprise 2.0 components, like authoring, links, and tags, which were also partly present in *SoftCom*. In this section we present the main results of our analysis that we try to justify with empirical evidence.

A. Enterprise 2.0 components comprise awareness mechanisms.

Finding information is essential in projects, especially if time or other resources are limited. *Search* algorithms and the definition and handling with search criteria depend among others on data format. Structured data is mostly indexed, key worded, and stored centrally or replicated. Unstructured data is tagged, ranked, annotated, and can also be stored distributed. The availability and reliability of enterprise data are completely dependent of the success of search mechanisms established. Persons can initiate a search process and pull the result lists if they want to ask something someone verbally and explicitly (*pulling*) or they want to search an artifact in the

system to access particular data (*polling*). These might happen event driven, which uses notification mechanisms to inform the persons sending the request. Persons might also *check* with others if there are questions and needs for clarification. *Filter* mechanisms help users in searching, by providing them configurable environments. They also help configuring views based on certain filter parameters of their choice. Changes in JIRA views or overviews in TTS created by user-selected filters trigger events showing changes in data. This invokes attention of the observers and they initiate consequently an active pull of information to catch up. This can only be done if observing happens regularly. Otherwise they cannot recognize a change in the state of the data, or own or others' work progress. REA provides different views (overview or detail view) to common data by means of filters and sort algorithms that facilitate monitoring of projects. This type of monitoring is very well established in *SoftCom*. In the following, one of the developers describes how he is using the filter view on a daily base:

> "Exactly, yes, this is the main view that I am working with. What is still open, what is in progress, and what is already in quality assurance. And how many. I can use the number to compare it with my performance, I know then approximately 'ok, where are you, is there something new'. If I check it in the evenings, how many issues I have still open, and see 'how many, I did not have 25 tasks today, there must be something new added to my list.' Something like that. And that is all actually. It is only an overview for me."

Extensions are systems that capture and follow users' interests by analyzing their behavior. Based on that information, they then recommend users certain actions or access onto specific information in particular areas. Extensions are in general related to artificial intelligence and semantic analysis. *Displaying* can be implemented as an extension to show particular predefined information about all types of content captured in the system. Displays can be triggered explicitly, or implemented *peripheral* for *monitoring* of users' activities and any kind of relevant changes in the (common) system. Capturing relevant data by using *sensors* to delegate systems certain business logic, complex analysis and filtering are normally needed to *poll* data for users, which they might *subscribe*. Specific visualization forms must be provided. All these mechanisms are facilitated by extensions.

In *SoftCom*, we could find several extensions in place. Information about tasks are essential for developers, domain concept creators, and testers for several reasons. First of all, their own work is defined practically by changes and new requirements addressed to tasks. They must actively carry out these tasks modified or arrived newly. Background information is needed to estimate the work load or possible problems which can occur during the implementation. Such considerations (doubts, questions, etc.) are also communicated with the corresponding team coordinator. Information about team members are important because they need to know who to contact in case of questions and uncertainties about the tasks, or coordination of activities in the team. Knowing who has initiated a change or added a remark to a task is mostly used to assess the importance of the task and has direct impact on the priorities to set. The status of a task shows whether a task is tested and the results are positive to publish for a new release. Programmers need to know what testers found out. If needed, they need to rework on the task they have already finished.

There were generally four aspects actors were interested with respect to tasks:

1) The question of "what": Are there modifications to a task? Are there new requirements added to a task? Are there new domain concepts added which need to be implemented next? Are there tasks discarded completely?
2) The question of "why": Why has been a task modified? Why is there a new version of a domain concept?
3) The question of "who": Who has created a task or a ticket? Who was the tester of a ticket? Who has been working on a task?
4) The question of "status of the task": What is the progress of the implementation of a task? Is there any task already completed? Which subtasks are still open and need to be finished next?

Whereas team members are mainly interested in information about their individual tasks, team coordinators need to have an overview of the status of all tasks carried out by their teams. They are in charge of managing the new requirements and changes notified and to meet the expected deadlines for delivery. Reassigning tasks, redefining priorities, estimating the effort needed and risks which may occur, planning the time of development, etc. are their responsibilities. In case of contingencies or unclarity, they initiate meetings with actors related to the issue to solve, no matter from which department. Direct communication is usually the main approach of team coordinators in such situations. The following quote by a team coordinator describes this very well:

> "I prefer in principle the direct exchange with the colleagues. Of course we get the emails by JIRA if something is added, changed, assigned, or if certain things were changed, or new versions of documents are created. We get all information automatically by the tools. That is very useful. Nevertheless, we prefer rather the direct contact, if something defers, or if there is a new requirement to the tool documented. This is then for me a sign to initiate contact to the colleague and arrange a meeting. It is not enough for me to read in the tool "ok, the version 3 has been created". I need to know what the reasons of the change are, whether we can cover it in our ongoing process, whether it was really necessary or not, whether it is possible to prioritize it differently in order not to disturb the ongoing development process in the team, etc. etc. etc. The main problem that we usually have is that the deadlines are always planned beforehand and we cannot modify them. That is why everything which is supposed to be different than the plan is a huge problem that we have to solve in the process. And that is the reason why the direct interaction with our colleagues is the best."

The question about the background reasoning for events and states in the project is not always of same size and importance. If status changes of tasks are planned, developers do not ask for

a reason. But if something is changed unexpectedly, everyone involved starts asking questions about why. The timing of the events is also very crucial: Conceptual changes at the beginning of production phases are perceived as normal. If they occur close to the delivery date, then they represent a risk or probably a problem.

Peripheral displaying of information is coupled to notification and subscription mechanisms. In *SoftCom*, several mechanisms have been established for this purpose: small visualization of relevant data in case of communication, i.e., by using tooltip to display the location of the sender of a message in *Sametime*; color- and short-text-based notification mechanism of *Sametime*, i.e., the online status of people; tickets created by TTS. It does not mean that notification is always useful and is welcome by the team members. The content notified must be relevant for others, e.g., when they trigger certain actions, or provide crucial work- or activity-related information. Several notification functions of Lotus Notes are not even used in *SoftCom* except the online status information of persons in the team. We could observe that the most of the emails, sent as notification of some events or changes, landed in some mailboxes without being read by the addressed people. Still, the number of the unread emails in the mailboxes made them peripheral aware of the increasing number of the actions taken.

Subscriptions as extensions are densely used in *SoftCom*. In JIRA, the subscription to a content (like a task or a ticket) is done automatically for its owner or modifier. With an email notification the information is sent to the relevant persons by the system. No one makes use of the monitoring feature of changes in JIRA. But everyone uses the explicit subscription possibility connected to a person by stating that one wants to be informed if an artifact is ready to hand over by this person. The following quote shows how the product manager expects that the team leader of the domain concept department notifies him about any possible delay in the project.

> "I have the team leader in my *Sametime* contact list. Of course. But I prefer to ask him explicitly what is the status of the development, are you still on schedule. If he seems that they will not be able to deliver at the right time, then I expect that he contacts me and informs me about it. Because we are also responsible to the customers. We have to inform them too. Otherwise we have problems."

Signals are notifications sent to users to inform them about news in the system. Traditionally, emails are used for this purpose [21]. Unfortunately the increasing number of emails makes email exchange more and more unpleasant for users. The overhead spent for the management and maintenance of emails and mailboxes is becoming a serious problem for a lot of users. RSS feeds and feed aggregators are taking over the role of tracking changes and triggering user notification. *Pushing* information means, on the one hand, the direct verbal mediation of data by a person to another person or group. On the other hand, entering information into the system (e.g., into an artifact) is a way of transferring data by pushing. *Notifying* the recipients of such information automatically or active *polling* the relevant information by receivers are only possible if signals are implemented in the IT infrastructure. Polling needs to be supported by identification and monitoring of changes transferred to recipients.

In *SoftCom*, pushing can be triggered by events or done regularly by persons, e.g., the daily reporting of time records by all. Tasks and work progress of the developers are monitored and signals are invoked in case of overdue of any task, new entry of a new task, or the occurrence of a problem. In very problematic cases, oral exchange is initiated to assure the reception of the information by the recipient. Everyone knows that being notified by an automatically created email is not enough to get really the attention of a person, e.g., to fix a problem. A direct active information push is better than the exchange based on pulling in cases of clarification and justification of the information exchanged, e.g., if details are needed to understand an issue, reasons for its creation, or clarify its urgency or accuracy.

In JIRA, notification happens via emails and RSS feeds. No one we have interviewed mentioned explicitly that they use RSS. Emails need to be checked by single persons, but this is obviously no problem at all. What they on the other hand confessed is that they trash automatically the most of the emails they receive or they filter them directly and move to a specific mailbox without reading. The reason is the large amount of emails they receive. Some said that they can judge an email by just reading its subject. In case of relevance they change from the email client to JIRA and check the detailed information about the issue. The following quote describes what a developer answers to the question how often and when he accesses JIRA:

> "Hmm, actually only when I get emails. So, always when any task is assigned or changed you get an email. ··· Usually, you receive a set of tasks, like 'test prove', 'concept', etc., a lot actually, instead to read all, I think it makes no sense, I delete all and think then, ok, something has happened in JIRA, I will have a look. ··· Normally I check evenings JIRA to plan the next day. What is to do next, what is still open for me, did I already finish a task but not closed. Yes, I do it once every evening."

B. Enter, annotate, rate, and share are identified as emerging awareness mechanisms related to authoring.

Authoring means in Web 2.0 the possibility of collaborative creation and editing of text in the Internet. Examples are Wikis and Blogs [21]. Authoring has positive impact on information update and linkage of related content in enterprises. In this sense, authored text must be published immediately. Handling with tags and links belong to authoring if these are dealt with by humans. There are two means of capturing a new entry: active generation of awareness information by users and passive generation by the authoring platform. Besides *entering* whole entities, *annotations* can be added to existing content, in collaborations usually by several people. In some areas it can be more useful to capture the opinion of people by using a *rating* mechanism. This can vary in form and range. By *sharing* the captured data in enterprises, common information spaces can be built and maintained.

Team members in *SoftCom* are authors of several entities in JIRA, like tasks. They need tasks not necessarily for the

coordination of the team, rather to manage their own work, as one developer states in the following quote:

> "· · · There was a new requirement, he said 'we would like to have a search function here'. And then he entered this into JIRA, to avoid to write this down on a piece of yellow paper. And one can then add sometimes a comment to it, and it is very handy, because everyone can see it. This doesn't happen often, but still now and then. In this case it was a comment that referred to the time when he is going to deal with the ticket, or that the ticket will not be handled at all, until the server is restructured. And such things · · · The best is that I know that RMM_DB_2 will receive an email when I close it. Or if I comment something. Then he knows exactly what I have done."

C. Reference is an awareness mechanism for links.

Links are crucial for effective search. Through automatic analysis of links and their modifications over time, systems can be used to identify emerging topics or interesting subjects in a work environment [21]. Links make contents and relations visible and, through this, provide context awareness in organizations. In order to build context from content *references* are created, which enable access from one point (source) to another point of content (target). If links are a collection of data in documents they are called cross-references. By externalizing context information into links, information amount and informational benefit of contents can be influenced both positively and negatively.

D. Select, mark, and label are awareness mechanisms for tags.

Tags are keywords that are used to categorize available information to improve the search process in common information spaces. Tagging supports user involvement and participation, which enriches the knowledge capturing process in enterprises. Tags can be predefined and stored in the system, or defined ad hoc during tagging process by users. The latter makes patterns and processes in knowledge work visible. Active generation of tags means active generation of context information which of course produces at the same time context awareness in the work environment. Tagging requires, first, *marking* an entity. The object *selected* can be a piece of text, image, or a relation between two entities. Besides adding a term or a keyword to *label* the marked entity, supplementary data can be attached, which helps identify and categorize certain content. Lightweight tags are pointers to a specific comment, whereas annotated tags can be composed. Tags can be shared if they are pushed to a server after their creation.

E. From the awareness point of view, Enterprise 2.0 components are interrelated to each other.

Some awareness mechanisms are included in more than one Enterprise 2.0 component, like polling in extensions and signals. Considering context and real use of Enterprise 2.0 in enterprises, some mechanisms need the others to make sense and to be useful at all. Extensions contain composed elements, that are built in cooperation of several Enterprise 2.0 components. Search enables configuration and personalization before accessing the common data and delivers the results in a preferred format. Extensions use generally both search (i.e., filter and pull) to retrieve data from the common information space and signals (i.e., notifications) for capturing events caused by changes. The combination of the results of search and signals must be of course presented in a useful way, mainly by applying display and monitoring mechanisms. Tags and links are kinds of authoring. They are created by humans as authors or by systems semi-automatically if defined properly, usually by means of rules.

F. Intranet solutions established in enterprises are mainly system-related and system-driven.

Intranet infrastructures established by medium and large enterprises are utilized by enterprise software, covering several areas like content management, customer relationship management, supply chain management, enterprise resource planning, business intelligence, project management, financial management, human resources management, manufacturing, etc., as well as, accounting software, office suites as a collection of productivity programs used by knowledge workers, multimedia editing and storage software, and much more. The most common collaborative client-server software platforms used by enterprise systems are, e.g., SAP's NetWeaver, Oracle's Fusion, IBM's Lotus Notes, and Baan's ERP. All these system are customized and prepared for use in different departments by different work groups. Employees are registered and access those parts of the system, which are permitted to them. In some environments users can personalize their interface. Functionality, usability, efficiency, evolution, communication, and business intelligence are in fact predefined and configured by the system, and cannot be modified by single persons or (work) groups. Changes required need to be communicated to system administrators and further on to system vendors. Humans are only consumers of the data and functions tailored. This has been seen as stabilizing, consistent, trustworthy managing body of an enterprise, as long as humans feed the system with syntactic and semantic correct data and use the system for their daily work continuously so that no data is missing for reporting. Such environments are not flexible, not (easily) adaptable, not scalable without huge additional effort. Single person cannot change anything in the system's structure – related to communication, workflow, or data. These are the reasons, why we call such Intranet solutions in enterprises as system-related and system-driven.

G. Awareness mechanisms can be categorized into system-related and user-related mechanisms.

As presented above, several Enterprise 2.0 components embrace certain awareness mechanisms which are either system- or user-related. So, we differentiate between two categories and assign Enterprise 2.0 components with their related awareness mechanisms to one or the other (Table I). *System-related* mechanisms are triggered by systems configured to capture, save, retrieve, and display awareness data. Search, extensions, and signals are system-related. On the other hand, *user-related* mechanisms are under control of users. Users can capture, select, edit, share, refer, label data they find interesting. Authoring, links, and tags are user-related.

TABLE I. AWARENESS MECHANISMS SUPPORTED BY ENTERPRISE 2.0.

AWARENESS MECHANISMS			
System-Related		User-Related	
Enterprise 2.0	Mechanism	Enterprise 2.0	Mechanism
Search	Pull Check Filter	Authoring	Enter Annotate Rate Share
Extensions	Display Monitor Peripheral Display Poll Subscribe Sensors	Links	Reference
Signals	Push Notify Poll	Tags	Select Mark Label

It is clear that there is a trade-off between how much pre-defined and controlled mechanisms, implemented in the systems used, are needed in an enterprise and how much flexibility and configurability have to be provided for the users, by enabling them to be prosumers of data used in the enterprise mainly for awareness purposes. In general, awareness information can be captured and prepared by systems established. This makes sense in several areas, like if the log data of user actions represent certain crucial issues in work processes, or lack of actions in a workflow can be monitored by just having the system configured properly for monitoring and notifying the right persons to take action. In some work environments and for certain critical work processes it is essential to use the system for gathering and communicating awareness information with other systems or humans. In more human centric areas it is a disadvantage if systems used are still in charge of dealing with awareness data. In such settings users are the active ones to capture, select, edit, share, refer, label data they find relevant and interesting. Their motivation to create and share data is the main reason why others can be aware of their actions, knowledge, or purposes. The communication between human actors can be most supported if users see the benefit and affordance of creating text or other type of data related to the work on different levels, relating data connected to work processes or products with other data available but loosely coupled in the system, adding semantic to the references they introduce to create a context which makes the captured data accessible and usable for others, tagging existing data to modify their relevance to address others explicitly sharing the same environment, and so on.

JIRA is a powerful Enterprise 2.0 system in *SoftCom*. Searching and linking are mainly used to enable and improve navigation within the system and offer an interface to other systems. Authoring is provided by different commenting functions. Registered users can create and manage their own projects and issues. They can provide access to others by using filters. Tags are both pre- or user-defined. Extensions are facilitated by widgets and visualizations. Signals are implemented, e.g., with email notifications if tasks are modified.

JIRA implements all system- and user-related awareness mechanisms. Push is initiated by events, which occur in case of changes in tasks and projects. Usually it is a notification via email or RSS feed. Notifications contain data about the task and mainly about the modifications: who changed what at what time. User can register – subscribe – for notifications and the system executes automatically the notification process as configured. Through subscription monitoring has been made possible. Users also can access awareness information stored in JIRA by pulling, like status data of tasks in detail view, change history of tasks showing the reason, the old and new values of changes. Filters and widgets enable overview pages and customization for pulling, by configuring the dashboard.

V. CONCLUSIONS

In this paper we summarized the familiar awareness mechanisms from the literature to create a ground for our case investigations (Section II). Then we presented our case (Section III) and used several observations to explain and justify our findings and results (Section IV). We proposed three Enterprise 2.0 components – authoring, links, and tags – to enhance awareness mechanisms. We ended up with additional awareness mechanisms like enter, annotate, rate, share, reference, select, mark, and label (Section IV). We tried to explain what we mean with all components and awareness mechanisms assigned to them. We came up with two different categories of awareness mechanisms in Enterprise 2.0: system- and user-related mechanisms. We concluded our results with the following: Authoring, linking, and tagging are very powerful mechanisms to create context and capture collective knowledge in enterprises. Intranet solutions for enterprises have to consider authoring, linking, and tagging as essential mechanisms for knowledge management and coordinated collaboration.

Our study is qualitative. We captured the cooperative endeavor in a software development company and studied their tools and work practices by focusing on the mechanisms of awareness. The main reason was to understand what really is going on in such environments. We do not claim that this is always the same in all other enterprises. We know that some companies are switching to cloud-based collaboration. This changes a lot in the ways how people collaborate with each other. In these cases the user-related awareness happens automatically because of the features provided by the systems they use, like Google Docs. The prosumer approach is in most cases implemented and people have to use them to make collaboration easier among the collaborators. In this paper we address the enterprises who are still using their Intranet solutions for collaboration. It is a fact that cooperation – no matter for which purposes and under which circumstances – requires certain parameters and establishment. We have just started to think about awareness mechanisms that became relevant because of the development of Web 2.0. The main contribution we make is to introduce the user-related awareness mechanisms for Enterprise 2.0 and stress out that Intranet solutions need to include these possibilities into their portfolio.

To sum what has been already done is to understand the technology and mechanisms applied in enterprises. What we have done in this paper is to understand these mechanisms by using concepts relevant for enterprises like awareness. We tried to introduce the differentiation between system- and user-related awareness mechanisms and aligning mechanisms to Enterprise 2.0 components (see Table I). In the future, these mechanisms need to be integrated into existing enterprise

IT systems, to enhance their functionality and application in new use contexts. The goal is among others to improve the data consistency, integrity, availability, sharing, and user participation. Furthermore, we also want to mention that the newly introduced systems in enterprises need to be evaluated qualitatively, which again would inform all stakeholders about further development and improvements in use and technology.

ACKNOWLEDGMENT

We thank all members of *SoftCom* for their openness, patience, and cooperation during our study. We are also grateful to all constructive comments of the reviewers of this paper.

REFERENCES

[1] J. E. Bardram, and T. R. Hansen, "The AWARE architecture: Supporting context-mediated social awareness in mobile cooperation", Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW '04). New York, NY, USA: ACM pp. 192–201, 2004.

[2] A. J. B. Brush, D. Bargeron, J. Grudin, and A. Gupta, "Notification for shared annotation of digital documents", Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02). New York, NY, USA: ACM, pp. 89–96, 2002.

[3] S. Diesenreiter, Awareness im Enterprise 2.0. Master thesis. Institute of Design and Assessment of Technology, Vienna University of Technology, 2013.

[4] P. Dourish, and V. Bellott, "Awareness and coordination in shared workspaces", Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work (CSCW '92), pp. 107–114, 1992.

[5] G. Convertino, A. Grasso, G. De Michelis, D. R. Millen, and E. H. Chi, "Clorg: Collective intelligence in organizations", Proceedings of the 16th ACM International Conference on Supporting Group Work (GROUP '10). New York, NY, USA: ACM, pp. 355–358, 2010.

[6] G. Convertino, H. M. Mentis, M. B. Rosson, J. M. Carroll, A. Slavkovic, and C. H. Ganoe, "Articulating common ground in cooperative work: Content and process", Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08). New York, NY, USA: ACM, pp. 1637–1646, 2008.

[7] G. Convertino, H. M. Mentis, M. B. Rosson, A. Slavkovic, and J. M. Carroll, "Supporting content and process common ground in computer-supported teamwork", Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09). New York, NY, USA: ACM, pp. 2339–2348, 2009.

[8] K. Ehrlich, C.-Y. Lin, and V. Griffiths-Fisher, "Searching for experts in the enterprise: Combining text and social network analysis", Proceedings of the 2007 international ACM conference on Supporting group work (GROUP '07). New York, NY, USA: ACM, pp. 117–126, 2007.

[9] R. Farzan, J. M. DiMicco, D. R. Millen, C. Dugan, W. Geyer, and E. A. Brownholtz, "Results from deploying a participation incentive mechanism within the enterprise", Proceedings of CHI2008. New York, NY, USA: ACM, pp. 563–572, 2008.

[10] W. Geyer, H. Richter, L. Fuchs, T. Frauenhofer, S. Daijavad, and S. Poltrock, "A team collaboration space supporting capture and access of virtual meetings", Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work (GROUP '01), C. Ellis, and I. Zigurs, Eds. New York, NY, USA: ACM, pp. 188–196, 2001.

[11] T. Gross, and N. Koch, Computer-supported cooperative work. Germany: Oldenbourg Wissenschaftsverlag, 2007.

[12] T. Gross, C. Stary, and A. Totte, "User-centered awareness in computer-supported cooperative work-systems: Structured embedding of findings from social sciences", International Journal of Human-Computer Interaction, vol. 18, nr. 3, pp. 323–360, 2005.

[13] C. Gutwin, S. Greenberg, and M. Roseman, "Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation", Proceedings of HCI on People and Computers XI (HCI '96), Germany: Springer-Verlag, pp. 281–298, 1996.

[14] C. Gutwin, Workspace awareness in real-time distributed groupware. PhD Thesis, Department of Computer Science, University of Calgary, 1997.

[15] C. Gutwin, and S. Greenberg, "Effects of awareness support on groupware usability", Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '98), pp. 511–518, 1998.

[16] C. Gutwin, K. Schneider, D. Paquette, and R. Penner, "Supporting Group Awareness in Distributed Software Development", in R. Bastide, P. Palanque, and J. Roth, Eds., Lecture Notes in Computer Science, vol. 3425, pp. 383–397, 2005.

[17] L. Hattori, M. Lanza, and R. Robbes, "Refining code ownership with synchronous changes", Empirical Software Engineering, vol. 17, nr. 4–5, pp. 467–499, 2012.

[18] J. Helming, M. Koegel, H. Naughton, J. David, and A. Shterev, "Traceability-Based Change Awareness", in A. Schürr and B. Selic, Eds., Lecture Notes in Computer Science, vol. 5795, pp. 372–376, 2009.

[19] A. King, and K. Lyons, "Automatic status updates in distributed software development", Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering (Web2SE '11). New York, NY, USA: ACM, pp. 19–24, 2011.

[20] M. Koch, "CSCW and Enterprise 2.0 – Towards an integrated perspective", 21st Bled eConference eCollaboration: Overcoming Boundaries Through Multi-Channel Interaction. Bled, Slovenia, June 15-18, 2008.

[21] A. P. McAffee, "Enterprise 2.0: The dawn of emergent collaboration", MITSloan Management Review, vol. 47, nr. 3, pp. 20–28, 2006.

[22] E. W. Morrison, "Information seeking within organizations", Human Communication Research, vol. 28, nr. 2, pp. 229–242, 2002.

[23] I. Omoronyia, J. Ferguson, M. Roper, and M. Wood "Using Developer Activity Data to Enhance Awareness during Collaborative Software Development", Computer Supported Cooperative Work, vol. 18, nr. 5–6, pp. 509–558, December 2009.

[24] T. Proenca, N. Moura, and A. Hoek, "On the Use of Emerging Design as a Basis for Knowledge Collaboration", in K. Nakakoji, Y. Murakami, and E. McCready, Eds., Lecture Notes in Computer Science, vol. 6284, pp. 124–134, 2010.

[25] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, "Tesseract: Interactive visual exploration of socio-technical relationships in software development", Proceedings of the 31st International Conference on Software Engineering (ICSE '09). Washington, DC, USA: IEEE Computer Society, pp. 23–33, 2009.

[26] K. Schmidt, "The problem with 'awareness': Introductory remarks on 'awareness in CSCW"', Computer Supported Cooperative Work, vol. 11, nr. 3–4, pp. 285–298, 2002.

[27] I. Steinmacher, A. P. Chaves, and M. A. Gerosa, "Awareness support in distributed software development: A systematic review and mapping of the literature", Computer Supported Cooperative Work, vol. 22, pp. 113–158, 2013.

[28] H. Tellioğlu, and I. Wagner, "Software Cultures. Exploring Cultural Practices in Managing Heterogeneity within Systems Design", Communications of the ACM, vol. 42, nr. 12, pp. 71–77, December 1999.

[29] H. Tellioğlu, "About Representational Artifacts and Their Role in Engineering", Chapter in the IGI book on Phenomenology, Organizational Politics and IT Design: The Social Study of Information Systems. G. Viscusi, G. M. Campagnolo, and Y. Curzi, Eds., IGI Global, pp. 111–130, 2012.

[30] J. Thom, D. Millen, and J. DiMicco, "Removing gamification from an enterprise SNS", Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12). New York, NY, USA: ACM, pp. 1067–1070, 2012.

[31] E. Ye, L. A. Neiman, H. Q. Dinh, and C. Liu, "SecondWATCH: A workspace awareness tool based on a 3-D virtual world", Proceedings of the 31st International Conference on Software Engineering (ICSE 2009), pp.291–294, 2009.

[32] Webster, Unabridged dictionary. Springfield: Merriam-Webster, 2006.