A Survey of Social Software Engineering

Navid Ahmadi, Mehdi Jazayeri, Francesco Lelli, Sasa Nesic
Faculty of Informatics, University of Lugano
{firstname.lastname}@lu.unisi.ch

Abstract

Software engineering is a complex socio-technical activity, due to the need for discussing and sharing knowledge among team members. This has raised the need for effective ways of sharing ideas, knowledge, and artifacts among groups and their members. The social aspect of software engineering process also demands computer support to facilitate the development by means of collaborative tools, applications and environments.
In this paper, we present a survey of relevant works from psychology, mathematics and computer science studies. The combination of these fields provides the required infrastructure for engineering social and collaborative applications as well as the software engineering process. We also discuss possible solutions for the encountered shortcomings, and how they can improve software development.

1. **Introduction**

It is now commonly accepted that software programming is a social activity. Today software development is:

- Carried out in teams.
- These teams may include domain specific expert.
- Most of the time is spent in the division of the work among the members and in understanding the requirements that a particular component should meet.

All these activities involve a strong interaction between a set of people that collaborate in order to achieve a common goal: produce an effective, efficient software following the mantra "Maximum Results with Minimum Effort".

This raises on the one hand the need for an effective way of sharing ideas and requirements among the group members and on the other hand the need to create tools that can support such collaboration.

Sawyer in [55] mentions that the social perspective is more than just aggregation of individual software developer's attributes and actions, i.e., the team has to be seen as a single unit of analysis. Studies by DeMarco and Lister suggest that on large projects typical systems developers spend about 70% of their time working with others [41]. Jones reports that team activities account for about 85% of the costs of large software systems [47]. Therefore, it is clear that the social activity represents a substantial part of the everyday work of programmers. This raises the question of whether a systematic adoption of methodologies studied in social and psychological disciplines could improve the productivity of a team.

On the other hand many web based social tools have been developed in order to manage, maintain and improve the social

activities of people. These applications have grown in an independent way but they often share part of their use cases. Therefore there is the need to define techniques and guidelines for building such applications.

Considering this as a relatively new area, we have tried to collect the most interesting related work that have been done in different disciplines that could be used as basic building blocks for achieving the foreseen improvements.

The rest of the paper is organized as follows: Section 2 presents a set of relevant works from psychological and social areas that could be integrated in modern software development. Section 3 presents relevant works in building collaborative and social tools while Section 4 gives a set of references on mathematical techniques that have been or could be used in order to validate ideas around social software engineering.

2. Software Engineering as Social Activity

Many works in psychology and sociology disciplines have attempted to explain and classify the social behaviors of human beings. Unfortunately the software engineering community has spent little effort to integrate the outcome of these areas in the modern software development process. In this section we present a set of well-established works in disciplines like psychology and sociology that could be integrated in the software development process in order to improve the productivity of a team.

One of the first promoters of software development as a social activity is Gerald M. Weinberg in 1971 [34]. Most of his work is focused on reengineering the software development processes from a "people empowering point of view". In other words, he considers software development as a human-centric activity. Therefore, effective communication plays a key role in software development [21], [22]. Lui and Chan also have a human centric approach to software engineering: in [27] they present different ways to approach collaborative programming and they show how to combine and harmonize these practices. Therefore, they can be applied to common software management problems such as motivating programmers, discovering solution patterns, managing software teams, and rescuing IT projects in trouble. Finally, Yourdon in [28] identifies with the words "death march" a set of software projects that will have almost no possibility to succeed due to the way they are organized, managed or carried out by programmers. He also proposes a set of techniques for a rapid identification of these projects and a few ways to survive inside them.

Other useful results coming from the psychology communities [33] can explain: (i) Why in a certain kind of group we have a person/partner that tends to be excluded. (ii) Why in a group we have usually a person that keeps fighting against the project leader. (iii) What kind of people should never be grouped together in order to avoid group fragmentation. (iv) Why groups usually divide themselves into subgroups and (v) What is the difference between the "real" chain of command and the formal one. For example, during its lifetime, a group encounters 5 different phases that are characterized by a different behavior of the members and different productivity. This social behavior is common in every human activity. Knowledge of these phases and the adoption of techniques that could lead to a fast and safe "group cohesion" (last phase) should be adopted in order to improve the software engineering process. That was just an example; interested readers can refer to [23],[24],[25], [26] for additional information.

Economists are facing the problem of improving the communication among people as well. Goleman in [35] puts a lot of effort into convincing the reader that working skills are fundamental but many other factors influence the success of projects. Successful people should have social skills as well in order to allow a good cooperation between people in the same group.

Other branches of psychology and sociology that could be useful in approaching the social software engineering problem include the so-called Persuasion field. On one hand, Cialdini in [32] describe how our brain is using seven different shortcuts in order to simplify the amount of information that we are receiving and how they are used in order to convince people to share a point of view. On the other hand, Fogg in [29] shows how the adoption of technology modifies the behaviors of a single person or of a group.

As final remark we need to point out that according to psychologists most of the social problems can be solved by a critical analysis of the dynamics between the people involved. Sometime this introspection needs to be performed with the help of an external entity that remains uninfluenced by the dynamics among people. This idea supports the approach of Weinberg [34], [21], [22] to software development. As a mater of fact by helping programmers to think critically we can avoid misunderstanding that could slow down the software development.

The next section will address the problem of engineering social applications by presenting the state of the art in relevant computer science areas like social networks and collaborative applications.

3. On Engineering of Social Applications

It is commonly accepted that many communities rely on computer support to attain their goals. A *social application* is software that takes the relationship among the members of the community into account and supports the social activities to the benefit of the community. With this definition all the applications that support communication and collaboration among users can be considered as social applications. Such support can be general and domain-independent, such as e-mail and instant messaging, or tailored for a specific domain and related community, which models intra-domain communications by virtual environments, such as tools that support collaboration in software engineering.

Social Network Services[1] (SNSs) [63] represent a subset of social applications that has gained attention recently. In addition to the usual collaborative features they also explicitly maintain the connection between people. These relationships among members are modeled as a graph that can be used to analyze and extract useful information to the benefit of target society.

SNSs are computer models of *social networks* that have emerged from human studies. For example, in [6] the authors propose social network analysis as a way to mitigate human health problems. According to [33] if a person is not inserted in a healthy group she/he can experience more diseases. Therefore, they propose social network analysis as a way to detect such problems. In [9] the authors identify these techniques as a way to understand possible communication weaknesses within a company and demonstrate that by solving this, productivity can increase.

In computer science, techniques on how to build social networking website are starting to be consolidated [30], [31]. Nerveless an appropriate visualization of a social network is still an open issue; for instance, in [8] the

[1] These links are examples in of Social network Services: http://www.friendster.com/ http://www.myspace.com/ http://www.linkedin.com/ http://www.facebook.com/

authors propose techniques and tools for browsing the graph that represents the collaboration between people.

Finally, as explained in the following sections many authors propose computer supported social networks as a way to increase the connection between people and their productivity [7] [10] [11].

Section 3.1 gives an overview of the social applications that are used in the software engineering domain. In Section 3.2 we discuss the SNSs in more detail and how semantic technologies can be leveraged in order to improve their usefulness.

3.1 Collaborative Applications

As pointed out in the introduction of this paper and in Section 2, software engineering is a social activity. DeMarco and Lister [41], and Jones [47] are examples of studies on the amount of time spent on communication and collaboration with others. Strubing [57] in his study found three other activities that developers consistently carry out beyond coding: (i) organizing their working space and process, (ii) representing and communicating design decisions and ideas, and (iii) communicating and negotiating with various stakeholders. Booch has conducted an experiment to obtain a snapshot of the daily life of a developer that conforms to Strubing's findings [39].

The social aspects of software process, as mentioned in Section 2, demand collaborative tools to support the collaboration among team members. Particularly, software development is a highly distributed task, involving programmers, designers and architects who work in different time/space, due to the ever-growth of offshoring, outsourcing and open-source communities. The majority of large software projects are undertaken by teams of software staff working across a number of organizations [50].

The computer-supported cooperative work (CSCW) community has investigated computer-assisted collaboration for more than two decades [45]. The CSCW application domains are numerous, but we consider them to be collaborative tools that are built in order to assist in software development. In the 1980s and 1990s, computer science researchers studied software processes as a way to improve productivity of software teams. Process-centered software engineering environments [61] is a collection of representative research of that era that placed the focus of collaboration on the process as opposed to on the people or the social aspects of the interaction.

Whitehead [58] distinguishes software engineering collaboration from the collaboration in other domains such as: 'artifact-based, or model-based collaboration, where the focus is on the production of new models, the creation of shared meaning around the models, and the elimination of errors and ambiguities within the models. Accordingly, he categorizes collaborative tools for software engineering, into four groups: model-based collaboration tools, process support tools, collaboration awareness tools, and collaboration infrastructure tools.

Sarma [54] provides a classification framework for collaborative tools based on the user's effort required to collaborate effectively. The framework is represented as a pyramid, consisting of five layers and three strands. Tools at higher layers provide more automated support and reduce the required effort for collaboration. Strands are critical needs that crosscut all aspects of collaboration. According to Sarma, the main three strands are *communication*, *artifact management*, and *task management*. At the highest layer of the framework, *collaborative development environments* (CDEs) provide a seamless environment where the

collaboration tools have become integrated into the development environment, thus lowering the context switch barrier to the developers. Booch [39] defines a CDE as: "a virtual space where the stakeholders of a project – even if separated by time or space – can meet, share, brainstorm, discuss, reason about, negotiate, record, and generally labor together to carry out some task, most often to create some useful artifact and its supporting objects".

A number of CDEs have been proposed in the context of software engineering. Boldyreff et al. [38] presented two development environments to support collaborative software engineering: GENESIS (GEneralised eNvironment for procEsS management in cooperative Software Engineering) and OPHELIA (Open Platform and metHodologies for devELopment tools IntegrAtion in a distributed environment). Dossick et al. [42] propose CHIME, a semi-automatically generated 3D virtual world representing the software system. Users interact with project artifacts by "walking around" the virtual world, where they potentially encounter and collaborate with other users' avatars. CHIME aims to support large software development projects, in which team members are often geographically and temporally dispersed, through novel use of virtual environment technology.

Considering that the Web can potentially be the most collaborative environment and that modern Web 2.0 presents a new paradigm for collaboration among people we continue the rest of the subsection in two parts: (i) the impact of the Web in software development, and (ii) Web 2.0 as the more recent trend for collaboration on the Web.

3.1.1 Web-based Collaboration on Software

The Web has had a great impact on software engineering. Oreizy et al. [52] consider the Web as an enabling technology with the potential to change software development by providing a cheap medium for large-scale software distribution, update, and reuse. The Web provides the required infrastructure for transiting from a syntactic level of collaboration on the Internet to a more semantic level by means of hypermedia in a globally accessible information space. Open hypermedia systems [49][60] had been well investigated even before the invention of the World Wide Web, and had been used in the context of software engineering [37].

The Web has been considered as a key environment for collaboration, due to the fact that it relies on the Internet, an infrastructure that enables the creation of highly scalable distributed systems. Such inherent properties of the Web accurately meet the need for collaboration beyond the time and space limits. Besides, the simple architecture of software development on the Web, for example REST [43], has played an important reason for its acceptance as the main infrastructure for building scalable collaborative applications.

Booch and Brown [39] provide a survey of collaborative sites. According to them, such collaborative sites, i.e., Web-based CDEs, can be classified into six groups based on their application domain: non-software domains, asset management, information services, infrastructure, community, and software development. They suggest a list of features for CDEs to be added to the feature list of virtual meeting spaces provided by Fournier [44]. They organize the CDE features into three categories, namely: *coordination*, *collaboration*, and *community building*. Furthermore, they propose a conceptual model for a software development-specific CDE, consisting of three layers: project workspaces, team tools and development resources.

3.1.2 Web 2.0 Fosters Software Engineering

Web 2.0 is a trend that leverages end–users' participation to build collective intelligence [51]. Wiki systems are one of the most concrete examples of gathering such collective intelligence [46]. Wikis are a form of content management system, in the form of Web pages, which enable a repository of information that may be updated easily by its users, using a simplified markup language.

Wiki systems have been exploited in the context of software engineering [48]. Rech et al. [53] use a wiki-centered framework that lets development teams quickly share experiences and reuse project-related knowledge in small- and medium-sized enterprises. Aguiar and David [36] suggest a wiki that helps in quickly weaving different kinds of contents into a single heterogeneous document, with the goal of reducing the development–documentation gap by making documentation more convenient and attractive to developers. Schuster et al. [56] exploit a wiki system to capture and share software architecture design rationale in enterprise systems. These are many other examples of the use of wikis in the context of software engineering.

Using wiki as a collaborative development environment is one step beyond using it only as a documentation repository. Xiao et al. [59] present Galaxy Wiki, a wiki based environment in which programmers can collaboratively write source code, build projects, execute programs and debug defects. Cheyer and Levy [40] introduce WubHub, a wiki-based collaborative programming environment with the notion of executable pages that act as functions, i.e., are executable and able to call each other and pass defined data types. Users can collaboratively write and compose services, and build upon each other's work, assembling new services that provide new or more customized features.

3.2 Social Network Services

Social Network Services (SNSs) provide a multitude of ways for users to interact. They have been promoted as central to the second generation of the Web (i.e., Web 2.0), which is characterized by mass participation and collaboratively generated Web content. The usual functionalities, which are provided by a majority of existing SNSs (e.g., Friendster[2], MySpace[3], LinkedIn[4] and Facebook[5]) include: a personal profile page for the participant, a network of friends listings, private messaging, discussion forums, events management, blogging, commenting and media uploading. The SNSs have brought a new way of interconnecting online content and networked people for various social and professional purposes. People are now better connected and can easily access, reuse or comment on content that is authored by other people from the network. However, in spite of being well connected within particular social networks, the diversity of SNSs hampers the interoperability between people from social networks, which adhere to different SNSs. Current social networks act as isolated islands of connected people and their contents. One possibility to overcome this lack of network interoperability is to leverage semantics into social networks – interconnecting both content and people in meaningful ways [1]. In this section, we first outline problems with today's SNSs that prevent them from accessing the full range of available content and networked people online. Then, we briefly explain the Semantic Web technologies and how they can be leveraged into SNSs. Moreover, we discuss potential benefits of enhancing SNSs with the

[2] http://www.friendster.com/

[3] http://www.myspace.com/

[4] http://www.linkedin.com/

[5] http://www.facebook.com/

Semantic Web technologies as well as initial steps towards building a social networking infrastructure as a fundamental part of the Internet infrastructure.

3.3.1 Fundamental problems with today's SNSs

The majority of today's SNSs are based on the notion of object-centered sociality [2] instead of explicit connections (i.e., person-to-person) among people. In other words, people do not just connect to each other; they connect through shared objects. Karin Knorr-Cetina defines shared objects as the reason why people get in touch with each other. A date and job, for instance, connect us to very different sets of people. The SNSs enable the creation of object-centered sociality, via people's actions, around the content they create together, comment on, link to or annotate with controlled vocabularies such as DublinCore[6] and LOM[7] or with uncontrolled vocabularies (e.g., social tagging). For example, Flickr[8] does this to photos, YouTube[9] to videos, Del.icio.us[10] to bookmarks, and so on. In general, the shared object could be any digital or non-digital resource as an instance of the appropriate concept from the world modeling domains.

The existing SNSs form bounded communities of users and their shared objects of interests. Although most of them provide the usual set of functionalities listed above, each SNS represents users, objects of interests and links between them in an application specific way. This hampers the interoperability among SNSs and, ultimately, content creation facilities on the Web. Users of different SNSs cannot interact with each other and can not access, reuse or comment on shared objects that belong to other communities. This is caused by the fact that today's SNSs do not have agreed-upon semantics to describe shared objects, that is, there is no shared understanding of the objects' meaning in different SNSs. Accordingly, it is difficult to transfer the users' properties such as blog posts, comments, photos and videos, from one SNS to another. The next problem is that a user who wants to join more than one social network must create a new profile for each network from scratch. Also, contacts that a user has in one network cannot be automatically transferred to other networks. This causes the same person to have many profiles and identities (e.g. account data), one for each network. The maintenance of a number of different profiles and account data is an error-prone and tedious task.

In addition, as their name suggests, the SNSs mainly focuse on the human dimension, that is, all information is available only in a human readable way. By embedding both people (i.e., their profiles) and shared objects in more machine processable (understandable) context the interconnectivity between people can be further enhanced. Intelligent software agents could infer new relationships between shared objects and hence between people who are interested in them.

3.3.2. Leveraging the Semantic Web technologies for SNSs

The Semantic Web aims at providing an environment in which both humans and software agents can unambiguously determine the meaning of resources and make better use of them [3]. Moreover, data and knowledge stored within a resource should be easily accessible across application, enterprise, and community boundaries. In accordance to this, the Semantic Web

[6] http://dublincore.org/documents/docs/

[7] http://ltsc.ieee.org/wg12

[8] http://www.flickr.com/

[9] http://www.youtube.com/

[10] http://del.icio.us/

community works on providing: a) common formats for integration and combination of data drawn from diverse sources, and b) languages for the conceptualization of real world objects. The Semantic Web principles are implemented in layers of Web technologies and standards. URI (Unique Resource Identifier), RDF (Resource Description Framework), ontologies, and structured query languages such as SPARQL[11] are the key Semantic Web technologies. We now briefly outline each of them and discuss how they can contribute in solving the outlined problems of today's SNSs.

URIs provide means for uniquely identifying the resources in the Web. Using global URIs for the identification of people and shared objects (e.g., photos, videos, blog posts, etc.) within SNSs, enables them to be identified independently of particular SNSs. Once an URI is assigned to a shared object within a SNS, it can be linked to other resources on the Web. In a similar way, users of the SNSs can use their unique identity for all social networks they belong to. OpenId[12] is an example of an open and decentralized identity system, which allows Internet users to log on to many different SNSs or Web sites using a single digital identity.

The RDF is a standard model for data interchange on the Web. It allows structured and semi-structured data to be mixed, exposed and shared across different applications. By publishing shared objects' description as RDF, the SNSs enable them to be shared with other SNSs. However, in order to understand the meaning of the shared objects, different communities should also use the agreed-upon semantics that lie behind these descriptions. Ontologies, as the next Semantic Web technology, can offer possible solutions to this problem.

An ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. SNSs can use ontologies to describe people, groups, and actions that people perform within the groups, as well as shared objects of interests. There are several initiatives for creating universally recognized ontologies to describe social networks such as FOAF[13] (Fried-of-a-Friend), SIOC[14] (Semantically Interlinked Online Communities) and PIMO[15] [64] (Personal Information Model), which have gained significant acceptance. We believe that over time and with further enhancement they will become standardized. However, we do not exclude the emergence of new ontologies within specific communities at any time. The interoperability between SNSs that use different ontologies should be solved by ontology-mappings [4]. The FOAF ontology describes persons and their relationships to other people and objects. FOAF documents are RDF instances of the FOAF ontology. People can manually create their FOAF documents and link to them from their homepages or use an SNSs with capability to export the FOAF file (e.g., hi5[16], LiveJournel[17], Tribe[18], etc.) based on user profiles. FOAF allows groups of people to describe social networks across a number of SNSs without the need for a centralized database. The SIOC ontology provides a mechanism to integrate online community information (e.g. discussion methods such as blogs, forums and mailing lists). It is commonly used in conjunction with FOAF

[11] http://www.w3.org/TR/rdf-sparql-query/

[12] http://openid.net/

[13] www.foaf-project.org

[14] www.sioc-project.org

[15] http://www.semanticdesktop.org/ontologies/2007/11/01/pimo/

[16] http://hi5.com/

[17] http://www.livejournal.com/

[18] http://www.tribe.net/

for expressing personal profile, user-generated content and the activities of online communities on the Web.

PIMO is based on the idea that users have a mental model to categorize their environment. It represents the user itself and the fact that he has a Personal Information Model. Each concept in the environment of the user is represented as *Thing* in the model, and mapped to documents and other entities that mention the concept. *Things* can be described via their relations to other *Things* or by literal RDF properties.

The SPARQL is an RDF query language and data access protocol. It enables unified queries to RDF data regardless of where the data comes from. The SPARQL endpoint is a SPARQL protocol service that enables users (software agents) to query remote RDF data stores. By publishing RDF data such as instances of the FOAF and SIOC ontologies into RDF stores with a SPARQL endpoint, the SNSs make this data accessible to other SNSs. In this way, SNSs can share user profiles as well as RDF descriptions of shared objects.

3.3.3 Towards unified social networking infrastructure

Combining SNSs with semantic technologies gives benefits to both social networking and the Semantic Web. By using agreed-upon semantic formats to describe people, shared objects and the connections that bind them all together, SNSs can interoperate via common semantics. Thus, the Semantic Web has the potential to connect disconnected social networks into a unified network potentially encompassing all Internet users. On the other hand, SNSs provide the possibility of creating valuable semantic metadata – and the Semantic Web still lacks sufficient (valuable) metadata.

Rather then building separate social networks, the Internet infrastructure should be augmented to include a social networking infrastructure, making social networking a shared component across various desktop and Web applications. NEPOMUK – the social semantic desktop (SSD) [5] provides an operating system layer that extends the personal desktop into a collaboration environment that supports both personal information management and information and content sharing and exchange across social and organizational relations. NEPOMUK is based on a set of carefully designed and integrated ontologies for the Semantic Desktop like PIMO. It also uses a P2P infrastructure as a communication medium, thus avoiding centralized SNSs whose maintenance requires a major investment. Profile and user information remains the property of individual users and multiple social software applications can crawl it. The SSD opens the way for a range of new social software applications that can employ all person-to-person connections and access shared objects of interests through the social network infrastructure.

4. Mathematical Methods for Social Software Engineering

In this section we present and describe some of the mathematical techniques that have been used to validate the theories presented in section 2 and 3 or that could be considered useful. In this section we will not present possible mathematical open issue. We just mention a set of models that may be useful for the evaluation of new ideas related to Social Software Engineering.

The most used techniques and probably the most relevant is represented by the Social Network Methods [12], [18], [19]. In a nutshell, it is commonly accepted that

analyzing a social network consists of analyzing a graph, represented by a matrix. Different properties of the matrix are mapped to properties of the social network. In Social Network Analysis the attributes of individual (such as friendly, unfriendly, beginner, smart etc.) are less important than the relationship between people. The scientific community has accepted a few properties while many authors in different papers propose new metrics in order to provide a better understanding of the social network structure. The work of Borgatti [16] is an example in this direction. While examples of accepted properties are the *Structural Coesion* that is The minimum number of members who, if removed from a group, would disconnect the group or the *Centrality* that count the number of ties to other node in the network.

It has also been demonstrated that social networks usually follow a scale free structure [13], [17] instead of a random network structure. Therefore they share all the strength and weakness of scale free model. In random network theory, despite the random placement of links, the resulting system is deeply democratic: due to the fact that the connections between nodes are added randomly, most nodes have approximately the same number of links. In a random network, it is extremely rare to find nodes that have significantly more or fewer links than the average. In scale free networks the connection between nodes follow a power law distribution. Therefore some nodes have a high number of connections to other nodes, while most nodes have just a few.

Considering that Social Networks follow a scale free model they share also the strengths and weaknesses. For example they are robust against accidental failures but vulnerable to coordinated attacks. In addition, in random networks, the propagation of the information needs to surpass a critical threshold (a number of contacted nodes) before it propagates to the entire system. Below the threshold, the information may not reach all the nodes. Above the threshold, the information spreads exponentially. In Scale Free network the threshold for the information propagation is close to zero.

The adoption of different human organizational models could strengthen and weaken the communication. Gleiser in [15] claims that all networks constructed by humans follow a scale free organization by demonstrating that Marvel Superheroes' communications follow the power law distribution. In any case, this idea does not represent an endpoint in this direction: different power law distributions can lead to different structures especially for small groups of people [17].

Another mathematical model that could be useful is the Nash Equilibrium [14], [20]. In a nutshell, it represents a set of strategies, one for each player, such that no player has incentive to unilaterally change her action. Players are in equilibrium if a change in strategies by one of them would lead that player to earn less than if she retained her current strategy. In the Prisoner's dilemma [62] it is shown that we can maximize the profit in the situation where players cooperate with each other. From an intuitive point of view we would expect that on the one hand the set up of a social relation between two people could be mapped in a situation where both players are not in Nash Equilibrium. On the other hand we could identify new opportunities in situations where the equilibrium leads to non-cooperation between the players.

A final remark is that the consistence between the social model and the reality is hard to identify and maintains. In [11], the authors propose a set of questions for evaluating the social network parameters that are used in order to build the graph.

5. Conclusions

In this paper we addressed the idea of Social Software Engineering by providing a survey of contributions in related areas. We identified and discussed two main subcategories: on the one hand the need to integrate results from social and psychological sciences in the software lifecycle; on the other hand, we outlined the need for engineering social networking services and collaborative tools. As a matter of fact, these applications have been developed independently but they share many use cases. Therefore, by approaching the problem in a systematic way we can try to formalize and improve these classes of applications.

Finally we presented a set of mathematical techniques that have been used for experimental validation of scientific contributions: these techniques may be used as basic blocks for understanding the ideas proposed for social software engineering.

6. References

[1] J. Breslin, and S. Decker, "The Future of Social Networks on the Internet", *IEEE Internet Computing*, vol. 11, no. 6, pp. 86-90.

[2] K. Knorr-Cetina, "Sociality with Objects: Social Relations in Postsocial Knowledge Societies", *Theory, Culture & Society*, vol. 14, no. 4, 1997, pp. 1-30.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", *Scientific Am*., vol. 284,no.5,2001,pp.34-43.

[4] Euzenat, J., and Shvaiko, P., *Ontology Matching*, Springer-Verlag, Berlin, 2007.

[5] S. Handschuh, et al., "The Nepomuk Project – On the Way to the Social Semantic Desktop", *In Proceedings of I-Semantic 07',* 2007, pp. 201-211.

[6] Margaret E. Morris, "Social Networks as Health Feedback Displays" *IEEE Internet Computing* ,vol. 9, no. 5, pp. 29-37, September/October, 2005.

[7] McDonald, D. W. 2003. Recommending collaboration with social networks: a comparative evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA, April 05 - 10, 2003). CHI '03. ACM, New York, NY, 593-600.

[8] Heer, J.; Boyd, D., "Vizster: visualizing online social networks" *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on* , vol., no., pp.32-39, 23-25 Oct. 2005.

[9] Cross R., Borgatti S. and Parker A., A bird's eye view. Using social network analysis to improve knowledge creation and sharing *IBM Institute for Business Value ,* 2002.

[10] Ogata, H., Yano, Y., Furugori, N., and Jin, Q. 2001. Computer Supported Social Networking For Augmenting Cooperation. *Comput. Supported Coop. Work* 10, 2 (Jan. 2001), 189-209.

[11] Cross Rob, Borgatti Stephen, Parker Andrew, Prusak Laurence, Making Invisible Work Visible: Using Social Network Analysis to Support Strategic Collaboration Creating Value with Knowledge, January 2004 , pp. 82-103(22)**:** Oxford Scholarship Online Monographs

[12] A web Introduction to Social Network Methods http://faculty.ucr.edu/~hanneman/nettext/

[13]A web introduction on Scale free network http://en.wikipedia.org/wiki/Scale-free_network

[14] A web Introduction on Nash Equilibrium http://en.wikipedia.org/wiki/Nash_equilibrium

[15] Pablo M Gleiser, How to become a superhero, Journal of Statistical Mechanics: Theory and Experiment September 2007, Start Page: P09020

[16] Borgatti, S.P. 2-Mode Concepts in Social Network Analysis. *Encyclopedia of Complexity and System Science*.

[17] Guido Caldarelli Scale-Free Networks: Complex Webs in Nature and Technology Oxford University Press; 1 edition (Jun 28 2007)

[18] Stanley Wasserman, Katherine Faust, Social Network Analysis: Methods and Applications, Cambridge University Press; 1 edition (November 25, 1994)

[19] John P Scott Social Network Analysis: A Handbook Sage Publications Ltd; Second Edition (March 2000)

[20] Mehlmann, A. *The Game's Afoot! Game Theory in Myth and Paradox,* American Mathematical Society (2000).

[21] Gerald M. Weinberg, Becoming a Technical Leader ISBN: 0-932633-02-1 Dorset House Publishing

[22] Gerald M. Weinberg, Quality Software Management Volume 1 to 4 (Systems Thinking, First Order Measurement, Congruent Action, Anticipating Change) ISBN: 0-932633-22-6 Dorset House Publishing

[23] D. Cartwright and A. Zander, Group Dynamichs, research and theory, Tavistok London 1960.

[24] R.B. Cattel New Concepts foe Measuring Leadership in Term of Group syntality, journal on human relation, 1951, 4, 161-184

[25] Forsyth, D.R. 2006. *Group Dynamics, 4th Edition*. Belmont, CA: Thomson Wadsworth. ISBN 0-534-36822-0

[26] Freud, Sigmund (1922) *Group Psychology and the Analysis of the Ego.* New York: Liveright Publishing.

[27] Kim Man Lui, Keith C. C. Chan Software Development Rhythms: Harmonizing Agile Practices for Synergy ISBN: 978-0-470-07386-5 April 2008

[28] Edward Yourdon, Death March: The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects Prentice Hall PTR (June 15, 1999)

[29] B.J. Fogg Persuasive Technology: Using Computers to Change What We Think and Do Morgan Kaufmann Publishers (December 2002)

[30] Michael Hartl, Aurelius Prochazka Building a Social Networking Website with Ruby on Rails ISBN 13: 978-0-321-48079-8 (July 2007)
[31] Alan Bradburne Practical Rails Social Networking SitesISBN-13 (pbk): 978-1-59059-841-2 Apres 2007
[32] Robert B. Cialdini, Influence: The Psychology of Persuasion Collins; Revised edition (October 7, 2005).
[33] Spaltro Enzo Pluralità. Psicologia dei piccoli gruppi, Pàtron (1993)
[34] Gerald M. Weinberg The Psychology of Computer Programming: Silver Anniversary Edition Dorset House Publishing Company, Incorporated; Anl Sub edition (September 1998)
[35] Daniel Goleman Working with Emotional Intelligence Bloomsbury Publishing PLC; New Ed edition (29 Jun 1999)
[36] Aguiar, A., & David, G. (2005). WikiWiki weaving heterogeneous software artifacts. *WikiSym '05: Proceedings of the 2005 international symposium on Wikis* (pp. 67--74). San Diego, California: ACM.
[37] Anderson, K. M. (n.d.). Supporting software engineering with open hypermedia. *ACM Comput. Surv.* , 20.
[38] Boldyreff, C., Nutter, D., Rank, S., Smith, M., Wilcox, P., Dewar, R., et al. (2003). Environments to Support Collaborative Software Engineering. *In 2nd Workshop on Cooperative Supports for Distributed Software Engineering Processes*, (pp. 25-28).
[39] Booch, G., & Brown, A. W. (2003). Collaborative Development Environments.*Advances in Computers,59*,2-29.
[40] Cheyer, A., & Levy, J. (2006). A Collaborative Programming Environment for Web Interoperability. *Proceedings of the First Workshop on Semantic Wikis -- From Wiki To Semantics*. ESWC2006.
[41] DeMarco, T., & Lister, T. (1987). *Peopleware: productive projects and teams*. New York, NY, USA: Dorset House Publishing Co., Inc.
[42] Dossick, S. E., & Kaiser, G. E. (1999). CHIME: a metadata-based distributed software development environment. *ESEC/FSE-7: Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering* (pp. 464--475). Toulouse, France: Springer-Verlag.
[43] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* PhD Thesis, UNIVERSITY OF CALIFORNIA, IRVINE.
[44] Fournier, R. (2001). Teamwork is the Key to Remote Development. *Infoworld.*
[45] Grudin, J. (1994). Computer-Supported Cooperative Work: History and Focus. *Computer , 27*, 19--26.
[46] Jazayeri, M. (2007). Some Trends in Web Application Development. *FOSE '07: 2007 Future of Software Engineering* (pp. 199--213). IEEE Computer Society.
[47] Jones, C. (1986). *Programming productivity.* New York, NY, USA: McGraw-Hill, Inc.
[48] Louridas, P. (2006). Using Wikis in Software Development. *IEEE Softw. , 23*, 88--91.
[49] Meyrowitz, N. (1989). The missing link: why we're all doing hypertext wrong. 107--114.
[50] Oppenheimer, H. L. (2002). Project Management Issues in Globally Distributed Development. *in the Proceedings of the Global Software Development Workshop.*
[51] O'Reilley, T. (2005). *What is Web 2.0---Design Patterns and Business Models for the Next Generation of Software*. Retrieved from http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what- is-web-20.html
[52] Oreizy, P., & Kaiser, G. (1997). The Web as Enabling Technology for Software Development and Distribution. *IEEE Internet Computing , 1*, 84--87.
[53] Rech, J., Bogner, C., & Haas, V. (2007). Using Wikis to Tackle Reuse in Software Projects. *IEEE Softw. , 24*, 99--104.
[54] Sarma, A. (2005). *A survey of collaborative tools in software development.* Institute for software research, Donald Bren school of information and computer science, University of California, Irvine.
[55] Sawyer, S. (2004). Software development teams. *Commun. ACM , 47*, 95--99.
[56] Schuster, N., Zimmerman, O., & Pautasso, C. (2007). ADkwik: Web 2.0 Collaboration System for Architectural Decision Engineering. *In Proc. of the 19th International Conference on Software Engineering and Knowledge Engineering (SEKE 2007)*.
[57] Strubing, J. (1994). Designing the Working Process: What Programmers Do Besides Programming. *User-Centered Requirements for Software Engineering Environments.*
[58] Whitehead, J. (2007). Collaboration in Software Engineering: A Roadmap. *FOSE '07: 2007 Future of Software Engineering* (pp. 214--225). IEEE Computer Society.
[59] Xiao, W., Chi, C., & Yang, M. (2007). On-line collaborative software development via wiki. *WikiSym '07: Proceedings of the 2007 international symposium on Wikis* (pp. 177--183). Montreal, Quebec, Canada: ACM.
[60] Østerbye, K., & Wiil, U. K. (1996). The flag taxonomy of open hypermedia systems. *HYPERTEXT '96: Proceedings of the the seventh ACM conference on Hypertext* (pp. 129--139). Bethesda, Maryland, United States: ACM.
nited States: ACM.
[61] Pankaj K. Garg, Mehdi Jazayeri (1995) Process-Centered Software Engineering Environments Institute of Electrical & Electronics Enginee ISBN 0818671033
[62] Dresher, M. (1961). *The Mathematics of Games of Strategy: Theory and Applications* Prentice-Hall, Englewood Cliffs, NJ.
[63] Boyd, d. m., Ellison, N. B. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, *13*(1), article 11 (2007)
[64] Leo Sauermann, Ludger van Elst, Andreas Dengel: PIMO - a Framework for Representing Personal Information Models. In Proceedings of I-Semantics' 07