An Advanced Collaborative Environment for Software Development

Gregory Moro Puppi Wanderley, Marie-Hélène Abel, Jean-Paul Barthès, Emerson Cabrera Paraiso

An Advanced Collaborative Environment for Software Development

Gregory Moro Puppi Wanderley, Marie-Hélène Abel, Jean-Paul Barthès
Sorbonne Universités, Université de Technologie de Compiègne
CNRS UMR 7253 Heudiasyc
Compiègne, France
Email: {gregory.wanderley, marie-helene.abel, barthes}@utc.fr

Emerson Cabrera Paraiso
Pontifícia Universidade Católica do Paraná
PPGIa - Graduate Program in Informatics
Curitiba, Brazil
Email: paraiso@ppgia.pucpr.br

***Abstract*—Collaborative software development is a complex activity. An important factor that needs to receive attention in collaborative software development is software quality. High quality software reduces the development and the maintenance; improves delivery schedules; and reduces repairs and rework. In order to measure, evaluate, control and improve the software quality, software metrics can be used. In this research we present an advanced collaborative environment for software development currently being built, called ACE4SD, which intends to support the improvement of the code quality during collaborative software development. ACE4SD is a system of systems composed of a software development environment, a multi-agent system and a platform to capitalize and manage knowledge, all of them being integrated in the same environment. ACE4SD can provide personalized support to team members to improve the code quality and encourage its reuse, it can answer questions or doubts arisen during the development, record document problems and solutions, and improve the awareness and collaboration between the participants.**

I. Introduction

Collaborative software development is a complex activity, composed of many steps like planning, analysis, design, implementation, or maintenance. During collaborative software development, participants have to work collaboratively in order to produce consistent results. One important factor that needs to receive attention in collaborative software development is *software quality*. According to Bonsignour and Jones [1] software quality is the critical path to achieve successful software development.

Software metrics allow measurement, evaluation, control and improvement of software products and processes (Fenton and Neil [2], Kitchenham [3]). They are essential resources to improve quality and control cost during software development (Wallace and Sheetz [4]). Dozens of metrics have been proposed since the mid-60s ([2] [3]).

In this research we are interested in the roles of the developers and of the managers in collaborative software development. The main goal of this work is to support them for improving code quality during the software development. To do this, we propose an advanced collaborative environment for software development, called ACE4SD, based on a software development environment, on multi-agent systems (MAS) and on a platform for capitalizing and managing the knowledge related to the code quality. Thus, ACE4SD is a system of systems integrating all elements in the same environment. The information concerning the code quality are capitalized through the use of software quality metrics.

ACE4SD can provide personalized support to team members to improve the code they are writing, to answer questions arising during the development, to record problems and solutions, to improve awareness and collaboration.

In this paper we state the problem, highlighting some of its aspects, and show how we address them. But first, in order to better understand the proposed approach, we give some background related to software quality metrics, multi-agent systems, and capitalization and management of knowledge in organizations. We then detail our approach and discuss some related work and the current state of the research.

II. Problem Definition

According to a study made by the University of Cambridge [5], the annual cost to correct source code defects is about US$312 billions. This cost could be reduced by improving the quality of software, which will also reduce the costs linked to the maintenance and support of the application. Having a high quality software also improves testing and delivery schedules. Furthermore, the repairs and rework can be reduced by more than 50%. On the other hand, low quality software makes delivery dates unpredictable. Repairs and rework become the major cost drivers of the project. Even more, low quality reduces customer satisfaction, can affect market share, and can even lead to criminal charges in some cases (Bettenburg and Hassan [6], Bonsignour and Jones [1]).

In order to handle and improve the quality of the source code, we propose an advanced collaborative environment for software development, ACE4SD. ACE4SD must provide personalized support to developers in: measuring the code quality and finding out solutions to improve it; giving answers to questions about quality that may arise during the development; documenting quality problems and solutions; and also increasing collaboration between team members. The improvement of code quality, reduces its complexity, facilitates its reuse (*i.e.* is easy to reuse) and code maintenance (*i.e.* readability).

ACE4SD is also concerned with the role of managers by improving *awareness*. Awareness is an understanding of the activities of others, which provides a context for your own activity. Furthermore, awareness information is always required to coordinate group activities (Dourish and Bellotti

[7]). Thus, in the context of collaborative software development, a manager must be aware of what is going on in the projects and with the developers. ACE4SD intends to improve managers awareness regarding the quality of project code and regarding each developer. It can be useful, for instance, to know if some developer encounters the same kind of quality problem frequently. If so, the manager could advise courses, formation or training, in order to improve the developer's ability to produce better code.

ACE4SD is being built as a system of systems integrating a software development environment, an MAS and a platform to capitalize and manage knowledge. A system of systems is a super system composed of other systems, which are themselves independent complex operational systems, interacting to achieve a common goal (Saleh and Abel [8]).

To support collaborative software development, researchers have proposed different approaches using MAS (Tacla et al. [9], Ramos et al. [10]). The software agents have been found to be useful in environments where the work load is high, the goal is difficult to achieve and the environment is highly collaborative (Sethuraman et al. [11]). According to Tacla et al. [9], an MAS can potentially improve the exchange of information among the participants, provide support, improve workflows and procedures controls, and provide convenient user interfaces systems (Paraiso and Barthès [12]). Moreover, agents can also learn the behavior of the participants and give them a customized support.

The multi-agent part of the ACE4SD connects the two other parts of the environment, the software development environment and platform to capitalize and manage the knowledge. The platform to capitalize and manage knowledge is used to store and recover the data concerning the code quality, such as the problems, the users related to them, and the solutions associated with the problems.

III. Background

This section introduces software quality metrics, multi-agent systems, and the capitalization and management of knowledge in organizations.

A. Software Quality Metrics

The quality of the source code can be measured by software metrics regarding complexity and maintainability. The goal of quality software metrics is to manage and to reduce the complexity of the structures, to improve maintainability and to improve the development of source code and consequently to improve the software itself as mentioned by Yu and Zhou [13].

Studies by Olague et al.[14], Sellers [15], or Martin [16] report that source code metrics may indicate situations of increasing or decreasing code quality related to maintainability, reusability, complexity and understandability. Regarding object-oriented code quality metrics, table I shows some metrics related to complexity; inheritance; size; and coupling.

TABLE I. Excerpt of Quality Metrics.

Metrics	
McCabe's Cyclomatic complexity metric - MCC [17]	Weighted Methods per Class metric - WMC [18]
Lack of Cohesion in Method metric - LCOM* [18], [15]	Nested Block Depth - NBD [16]
Depth of Inheritance Tree - DIT [18]	Number of Children - NOC [18]
Number of Overridden Methods - NORM [15]	Specialization Index - SIX [15]
Method Lines of Code - MLOC [15]	Number of Attributes per Class - NOA [15]
Number of Static Attribute - NSF [19]	Number of Static Methods - NSM [19]
Number of Parameter - NOP [19]	Number of Interfaces - NOI [19]
Number of Package - NOP [19]	Afferent Coupling - Ca [16]
Efferent Coupling - Ce [16]	Instability metric - I, named here as RMI [16]
Abstractness - A, named here as RMA [16]	Normalize Distance from Main Sequence - D [16]

B. Multi-agent Systems

Multi-agent systems are sets of agents that interact to coordinate their behavior and often cooperate to achieve challenging tasks (Ren and Chen [20]).

Many platforms exist to create and support multi-agent systems. We consider the OMAS (Open Multi-Agent Systems) developed in our laboratory [21]. Indeed, in addition to be readily available, OMAS offers a rich environment for developing multi-agent applications. OMAS agents are cognitive, persistent, multi-threaded; they have skills (what they can do) and goals (what they plan to do). Each agent can have its own ontology to understand the expressions of the content language, to build a knowledge base, or to interpret the utterances from the user. The ontologies are expressed with the MOSS[1] representation language. The OMAS platform has three kinds of agents: *Service Agent*(SA), *Transfer Agent* (XA) and *Personal Assistant* (PA). SAs provide a particular type of service corresponding to specific skills; XAs are gateways that communicate with platforms having different structures, implementing their own protocol, and translating communication and content languages; PAs are in charge of interfacing humans to the system. The PA acts autonomously and is built to be a real assistant or surrogate of its master (Wanderley et al. [22]), or even to act as a "digital butler" (Negroponte [23]).

A PA has a crucial function, being dedicated to:

- understanding its master's needs (i.e. the needs of the user owning the agent);
- acting pro-actively to anticipate its master's needs;
- mobilizing SAs to execute a command or demand from its master;
- mediating all information exchanges among team members (who are considered information sources);
- organizing the documentation of its masters with the help of an SA;
- capturing and representing the team members' operations, helping them in the process of preserving and creating knowledge (Paraiso and Barthès [12]).

C. Capitalization and Management of Knowledge in Organizations

Different approaches intend to handle the question of capitalizing and managing the knowledge produced in organizations. One of them is the MEMORAe platform (Abel

[1] See http://www.utc.fr/∼barthes/MOSS/ for full documentation.

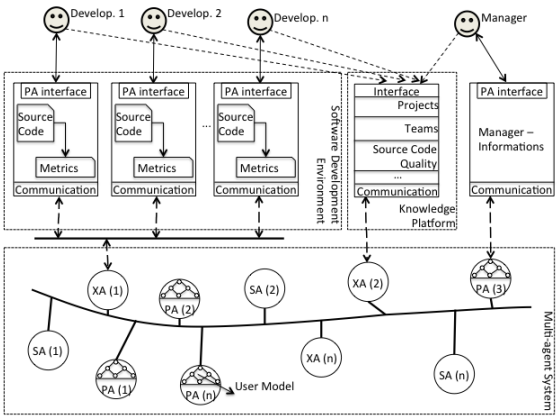

Fig. 1. The ACE4SD Architecture.

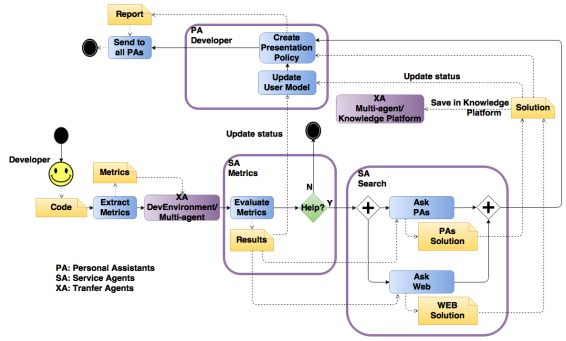

Fig. 2. Diagram representing the flow of assistance provided by a PA, concerning the quality of developers' source code.

[24]) also developed in our laboratory. It aims at capitalizing knowledge and skills in the context of organizations, and more precisely the resources associated with this knowledge. The MEMORAe platform lets its users make annotations, comments, descriptions, recommendations, etc., and associates them with resources, like documents, pictures, events, or forums. Furthermore, the user can vote about resources, expressing his opinion about them. Moreover, the user is also able to index the resources using a semantic map (an ontology) visualized as concepts and instances. Besides, MEMORAe gives users the ability to annotate resources and share them within different sharing spaces, in addition to tracing users' activities within the platform.

IV. The ACE4SD Environment

In this section we present ACE4SD. First, we introduce the architecture of the environment, and then we introduce a prototype.

A. Architecture

Figure 1 shows the proposed architecture of ACE4SD. As mentioned earlier, the architecture forms a system of systems composed of three main parts: a software development environment, a multi-agent system and a platform to capitalize and manage knowledge. In addition, managers have an interface, named "Manager - Information," with which they can interact with the environment. It is important to highlight that in the architecture, both developers and managers can interact directly with the knowledge platform.

The knowledge platform aims at documenting the code quality problems detected during software development and the solutions that were found. Moreover, it allows developers and managers to contribute with and improve the solutions, making annotations, comments, descriptions, recommendations; adding resources such as excerpt of codes; sharing ideas; and also voting for the best solutions giving them a score.

In ACE4SD, each participant has its own Personal Assistant Agent (PA), and the communication uses natural language. Furthermore, all of the agents (SA, XA and PA) are in the multi-agent platform. It is important to highlight that each agent has its own ontology, which is used for understanding the expressions of the content language (*e.g.* messages between agents and systems), to interpret the users' utterances (*e.g.* dialogues between developers/managers and their PAs), and to work as a knowledge base. In the latter case, a PA ontology can be used to store in a *User Model* personal information (name, address, email, etc.) of its master, as well as information concerning the quality of the developer's code. The agent can be aware of the experiences, skills and competencies of the participants, and then give them personalized support. That is, for developers, the ontologies can store information that keeps track of the necessary quality improvements and their respective solutions. It is thus possible to better understand the developers' needs.

Managers on the other hand, interface with their PAs through the "Manager - Information." ACE4SD intends to make managers aware of the quality of the code being developed. Managers are able to assess the quality of the projects, packages, classes or methods, and also assess the quality of the code produced by each developer.

One of the ways that PAs can help developers is by measuring the code quality and finding possibilities of improving it. Figure 2 shows the flow of assistance provided by a PA, regarding the metrics and quality of the developer's source code.

During collaborative software development, while a developer is developing code, the software quality metrics of Table I (Section III) are calculated and extracted from the code each time the code is committed. Then, the resulting metrics are sent to the multi-agent platform, through the transfer agent "DevEnvironment/Multi-agent" that bridges the two platforms. After that, the service agent "Metrics Agent" receives and evaluates the metrics, verifying whether the code needs quality improvement. To evaluate the metrics, the agent compares them with thresholds that can be found in the literature (Filo et al. [25], Martin [26]). If the evaluation concludes that the code needs some improvement, them the Metrics Agent forwards it to the agent "SA Search," that will try to find solutions in the following way: by asking the other developers' PAs or by searching the Web. When the agent "SA Search" asks other PAs, it will try to find out whether their masters (developers) have knowledge about this quality problem.

An agent is able to find solutions in documentation, excerpt of codes, answers from forums, recommendations and hints given by other development participants. As it goes through the Web, and given that the team members can contribute to

the solutions in the knowledge platform, it is expected to find and propose new solutions frequently.

Once a solution is found, it is synthesized and documented in the knowledge platform, with information about the user; the metrics and their values regarding the methods, classes, packages, projects needed to be improved; and solutions found for each quality problem. After that, the solution is sent to the developer's PA. The PA updates the User Model, saving information about the quality improvement needed and about the solutions that were found. After that, the PA sends a report to all other PAs, making the collaborative development team aware of the quality problems in this part of the code.

For example, if the Metrics Agent found, for a given developer, that the value of "Depth of Inheritance Tree (DIT)" Table I (Section III) has increased abruptly since the last evaluation, then it can indicate that this part of the code is complex and needs quality improvement. For the DIT metric, the deeper a class will be in the hierarchy, the greater the possibility to inherit more methods, making it more complex for predicting its behavior as shown by Suresh et al. [27]. Then, the agents can search for solutions to reduce this type of complexity.

In addition, ACE4SD must support team members, trying to provide answers to questions about quality that may arise during development. For example, a developer can directly ask his PA if the use of the repetition structure "for" one inside the other could affect the quality. Then in order to find out the answer, the PA would ask the agent "SA Search," that exchanges information with other PAs, and also search the Web. The information and the results of the answers are stored in the knowledge platform, and we also keep track in the ontologies of the PAs. When the PA interacts, exchanging information with other users' PAs, it improves the collaboration between team members. Moreover, the use of the knowledge platform that capitalizes and shares knowledge, also improves the collaboration as well as the documentation.

B. Prototype

This section presents a mock up of the ACE4SD environment from the point of view of the developers and managers. In order to build the prototype, the following tools were used: the Eclipse IDE, the OMAS platform and the MEMORAe platform. The Eclipse IDE is a Java environment, in which developers write their codes. The OMAS platform provides the multi-agent environment, and the MEMORAe platform the environment to capitalize and share knowledge.

The Eclipse IDE contains two plugins: (i) Metrics [28]: and (ii) ACE4SD.

- The Metrics plugin calculates, automatically and in a non-intrusive way, quality metrics of Java source code, providing the measured values, the mean and standard deviation for resources (project, class and method). Some of the quality metrics that the plugin calculates are available in Table I (Section III). The goal is to extract the quality metrics of the source code that is being produced by a developer and to send it to the OMAS plugin.

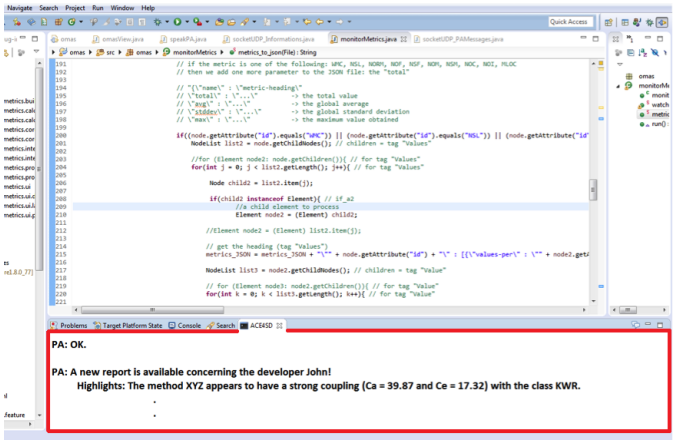

Fig. 3. The ACE4SD developers' interface inside the Eclipse IDE.

- The ACE4SD plugin creates the interface between developers and their PAs (which are inside the OMAS platform). In this way, the developers can interact with their PA's directly from Eclipse, avoiding switching between different windows from different tools. Moreover, the ACE4SD plugin receives the calculated metrics from the Metrics plugin and sends it to the service agent "Metrics Agent" (Figure 1). This agent is responsible for analyzing the metrics.

Figure 3 shows a developer's interface of the Eclipse IDE. The interface provides a communication channel between developers and their PAs. In the interface, a developer can communicate and exchange information with his PA in natural language directly, avoiding changing windows or tool. For instance, the developer can tell his PA that he wants to access the MEMORAe platform. Besides, the PA is proactive and it is always monitoring its master. In the example of Figure 3, the PA alerts the developer that a new report with information about problems related to code quality and possible solutions is available. It also shows directly in the interface some highlights of the report.

A manager's interface of ACE4SD is shown in Figure 4. The interface enables a manager to be aware of the quality of the projects and the developers. She can see the quality of the whole project, the resources (.java) with the worst quality, and what are their main defects, such as inheritance, coupling, complexity, etc. If the quality of some project is under a defined threshold, it is highlighted in red. Besides, in her interface, the manager is able to interact and exchange information in natural language with her PA.

For instance, in Figure 4 the manager asks more details about a specific developer, and the PA brings her the results showed in Figure 5. In this figure, the manager has a quality chart of the developer, being aware of the quality of the code he produced as well as its problems, such as it is difficult to test, *i.e.* the code contains a high number of decisions paths, and more tests are needed.

V. Related Work

In this section we present other approaches related to supporting the collaborative software development teams for

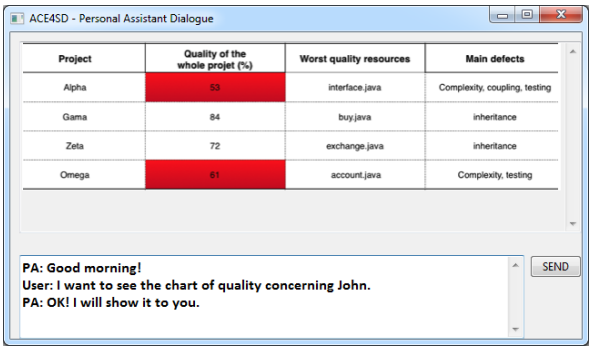

Fig. 4. The ACE4SD managers' interface.

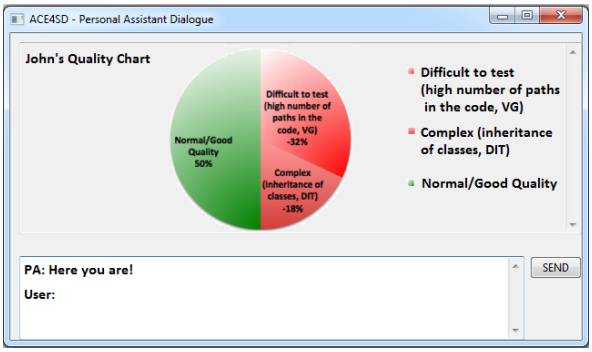

Fig. 5. Details and informations of some developer in the manager's view.

improving the code quality.

The work of Haderer et al. [29] presents a framework for measuring the quality of the code after each commit made by a developer. After that, it sends an email to the development team, comparing the last evaluation with the new one, and giving advice for improving the code. However, the approach is reactive, and it does not provide proactive nor personalized support to the team members, as agents can do. Furthermore, as far as we can see from the paper, advice proposed by the framework to improve the quality appears to be limited to predefined strings. That is, we do not know whether the solutions improve over time, whether they come from different sources, or whether the system provides other kinds of resources like excerpt of codes or answers from forums.

The goal of the work of Mishra and Srivastava [30] and Santos et al. [31] is to develop a multi-agent architecture to handle software maintenance. Given a requirement made by some user, such as a change or an enhancement in the code, the agents compute some metrics in order to assess and estimate the complexity of the modifications. In the work of Santos et al. [31], the developer chooses the quality attribute she wants to improve, and then the agents search the code for opportunities to apply the maintenance tasks and evaluate whether the source code quality has been improved. Nevertheless, in this works the only kind of support provided by the authors to the users is related to code maintenance. The approach does not support the software development team providing solutions, answering question/doubts, or giving personalized support for each team members during the development.

The work of Heinemann et al. [32] describes a tool for providing feedback to the developer using incremental code quality analysis. The approach aims at evaluating the quality of a piece of code (through the use of quality metrics) when it is committed by a developer, revealing the impact of this new code onto the quality of the entire system. The tool also annotates the source code with quality defects. However, it does not provide support to the collaboration between the team members, that is, there is no exchange of information, contributions, messages or help, among users.

The fact is that none of the cited works proposes a system of systems approach, that takes advantages of a software development environment, a multi-agent part and a platform to capitalize and manage knowledge. We did not find an approach that: measures the code quality; tries to find out solutions (coming from different sources) to improve the quality; provides answers to questions that may arise during the development; documents quality problems and solutions; and also increases collaboration between team members.

VI. Discussion and Current State of the Research

Collaborative software development is a complex activity. An important factor that needs to receive attention in order to achieve successful software development is software quality. Low quality code makes delivery dates unpredictable and repairs and rework expensive.

Software metrics can be used to measure, evaluate, control and improve software products and processes. They are essential for improving code quality.

In our research, we plan to support developers and managers for improving the code quality. To do that, we started to develop an advanced collaborative environment for software development, ACE4SD, integrating a software development environment, an MAS platform and a platform to capitalize and manage the knowledge related to the code quality..

ACE4SD will provide personalized support to developers by measuring the code quality and finding out solutions to improve it; attempting to provide answers to questions that may arise during the development; documenting the problems encountered, and also increasing the collaboration. In the case of managers, ACE4SD will increase their awareness of the developers.

In this paper, we presented the general architecture of the ACE4SD, discussed the help that a PA can provide related to the quality of the developers source code. Besides, we described a first prototype that shows the views of a developer and a manager in ACE4SD. In the prototype the software development environment was handled by the Eclipse IDE, the multi-agent platform OMAS, and the knowledge platform MEMORAe.

In the current state of this research, the interface between Eclipse and OMAS is handled by a transfer agent, "DevEnvironment/Multi-agent." The quality metrics are extracted and the results are sent to OMAS. The results are also analyzed and evaluated by a service agent, "Metrics Agent" that can detect quality problems. Moreover, the interface to the MEMORAe platform is done by an OMAS transfer agent,

"Multi-agent/Knowledge Platform" agent, allowing to store the data concerning the metrics and the results of evaluating each developer. Now, we are working on better ways to recover information stored in MEMORAe to help the developer efficiently.

VII. Acknowledgment

Gregory Moro Puppi Wanderley would like to thank CNPq-Brazil for its support in this research.

References

[1] O. Bonsignour and C. Jones, *The Economics of Software Quality*. Addison Wesley, 2011.

[2] N. E. Fenton and M. Neil, "Software metrics: roadmap," in *Proceedings of the Conference on the Future of Software Engineering*. ACM, 2000, pp. 357–370.

[3] B. Kitchenham, "Whats up with software metrics?–a preliminary mapping study," *Journal of systems and software*, vol. 83, no. 1, pp. 37–51, 2010.

[4] L. G. Wallace and S. D. Sheetz, "The adoption of software measures: A technology acceptance model (tam) perspective," *Information & Management*, vol. 51, no. 2, pp. 249–259, 2014.

[5] Cambridge University, "Cambridge University study," http://insight.jbs.cam.ac.uk/2013/financial-content-cambridge-university-study-states-software-bugs-cost-economy-312-billion-per-year/, 2013, online; accessed 09 June 2016.

[6] N. Bettenburg and A. E. Hassan, "Studying the impact of social interactions on software quality," *Empirical Software Engineering*, vol. 18, no. 2, pp. 375–431, 2013.

[7] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*. ACM, 1992, pp. 107–114.

[8] M. Saleh and M.-H. Abel, "Information systems: Towards a system of information systems," in *KMIS 2015 7th International Conference on Knowledge Management and Information Sharing*, 2015, pp. 193–200.

[9] C. A. Tacla, A. R. Freddo, E. C. Paraiso, M. P. Ramos, and G. Y. Sato, "Supporting small teams in cooperatively building application domain models," *Expert Systems with Applications*, vol. 38, no. 2, pp. 1160–1170, 2011.

[10] M. Ramos, C. Tacla, G. Sato, E. Paraiso, and J.-P. Barthès, "Cscw in software development: Collaboration among humans and artificial agents through dialogs," *International Journal of Energy, Information and Communications*, vol. 2, no. 4, pp. 31–45, 2011.

[11] A. Sethuraman, K. K. Yalla, A. Sarin, and R. P. Gorthi, "Agents assisted software project management," in *Proceedings of the 1st Bangalore Annual Compute Conference*. ACM, 2008, p. 5.

[12] E. C. Paraiso and J.-P. A. Barthès, "An intelligent speech interface for personal assistants in r&d projects," *Expert Systems with Applications*, vol. 31, no. 4, pp. 673–683, 2006.

[13] S. Yu and S. Zhou, "A survey on metric of software complexity," in *Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on*. IEEE, 2010, pp. 352–356.

[14] H. M. Olague, L. H. Etzkorn, and G. W. Cox, "An entropy-based approach to assessing object-oriented software maintainability and degradation-a method and case study." in *Software Engineering Research and Practice*. Citeseer, 2006, pp. 442–452.

[15] B. H. Sellers, "Ojbect-oriented metrics. measures of complexity," 1996.

[16] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.

[17] T. J. McCabe, "A complexity measure," *Software Engineering, IEEE Transactions on*, no. 4, pp. 308–320, 1976.

[18] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *Software Engineering, IEEE Transactions on*, vol. 20, no. 6, pp. 476–493, 1994.

[19] R. Harrison, S. Counsell, and R. Nithi, "An overview of object-oriented design metrics," in *Software Technology and Engineering Practice, 1997. Proceedings., Eighth IEEE International Workshop on [incorporating Computer Aided Software Engineering]*. IEEE, 1997, pp. 230–235.

[20] C. Ren and C. P. Chen, "Decentralized control for second-order uncertain nonlinear multi-agent systems consensus problem based on fuzzy adaptive high-gain observer," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4935–4940.

[21] J.-P. A. Barthès, "Omasa flexible multi-agent environment for cscwd," *Future Generation Computer Systems*, vol. 27, no. 1, pp. 78–87, 2011.

[22] G. M. P. Wanderley, M. P. Ramos, C. Tacla, G. Y. Sato, E. J. d. Silva, E. C. Paraiso *et al.*, "Modus-sd: User modeling in collaborative software development," in *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*. IEEE, 2012, pp. 372–378.

[23] N. Negroponte, *Being digital*. Vintage, 1996.

[24] M.-H. Abel, "Knowledge map-based web platform to facilitate organizational learning return of experiences," *Computers in Human Behavior*, 2014.

[25] T. Filó, M. Bigonha, and K. Ferreira, "A catalogue of thresholds for object-oriented software metrics," in *Advances and Trends in Software Engineering, 2015 The First International Conference on*. IARIA, 2015, pp. 48–55.

[26] R. Martin, "Oo design quality metrics," *An analysis of dependencies*, vol. 12, pp. 151–170, 1994.

[27] Y. Suresh, J. Pati, and S. K. Rath, "Effectiveness of software metrics for object-oriented system," *Procedia Technology*, vol. 6, pp. 420–427, 2012.

[28] Metrics, "Eclipse Metrics Plugin," http://metrics2.sourceforge.net, 2016, online; accessed 15 April 2016.

[29] N. Haderer, F. Khomh, and G. Antoniol, "Squaner: A framework for monitoring the quality of software systems," in *Software Maintenance (ICSM), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–4.

[30] A. Mishra and V. Srivastava, "Multi agent paradigm used to complexity measure for perfective software maintenance," in *Computer Science and Engineering (APWC on CSE), 2014 Asia-Pacific World Congress on*. IEEE, 2014, pp. 1–9.

[31] H. Santos, J. F. Pimentel, V. T. Da Silva, and L. Murta, "Software rejuvenation via a multi-agent approach," *Journal of Systems and Software*, vol. 104, pp. 41–59, 2015.

[32] L. Heinemann, B. Hummel, and D. Steidl, "Teamscale: software quality control in real-time," in *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 592–595.