ISSN: 2007-5448

RECIBE

Año 2 No. 1

Vol.2 No.1 | Dic 2012 - May 2013

Índice

Computación e Informática

ReCIBE, Año 2 No.1, Mayo 2013

Updating freeTribe to Support Efficient Synchronous Awareness in the Web Context

RicardoDelgadillo
University of Guadalajara (Mexico)
rdelgadi@cucsur.udg.mx

Eduardo Escofet
Gilogiq Internet Services Ltd.,17190Girona (Spain)
escofet@gilogiq.com

Humberto Rodríguez-Avila
Department of Informatics,
University of Holguín, 80100 Holguín(Cuba)
hrodriguez@acinf.uho.edu.cu

Julio C. Rodríguez-Cano
Department of Informatics,
University of Holguín, 80100 Holguín(Cuba)
jcrcano@acinf.uho.edu.cu

Abstract: **The research field of Computer-Supported Cooperative Work has been reflected fundamentally in theoretical contributions. This contributions have constitute the base to carry out several intents to facilitate the work of the collaborative systems developers, however, current tool-kits, APIs or class libraries only eliminate partially the gap between the technical aspects that impose the information technology and the stressed social character of the process of collaboration in the World Wide Web. In this paper is presented the framework freeTribe, which involve the domain of the distributed groupwares leaning on the Cooperative Model of the methodology AMENITIES, in the middleware platform ICE and in RIA technologies; freeTribe has been designed as a software framework, to maximize its reusability and adaptability with a minimal programming effort. Support for synchronous group tasks in the Web context is increasingly recognized as a desideratum for collaborative systems and several tools have emerged recently that help groups of people with the same goals to work together, but many issues for these collaborative systems remain under studied. We identified synchronous awareness as one of these issues in collaborative systems, and updated freeTribe with four well-accepted kinds of awareness (group awareness, workspace awareness, contextual awareness, and peripheral awareness) by the community focusing our interest in its synchronous mechanism for efficient interaction in Web contexts.**

Keywords: **computer-supported cooperative work, groupware, synchronous awareness.**

1. Introduction and motivation

"Collaboration" seems to be the buzzword this year,
just like "knowledge management" was last year.
-David Coleman

Reading these first lines maybe you asked yourself, why to update freeTribe with a Web-based infrastructure to support synchronous awareness? Why now? There is a simple two-fold answer to both these questions. Using technology to understand and support collaborative behavior has been around for a while, what is well known as Computer-Supported Cooperative Work (CSCW), but it is in the recent years that we have seen more specialized

attention given to applying CSCW methods and frameworks for explicit collaborative scenarios in the World Wide Web. On the other hand, the Web-based technologies for groupware development have found the importance of Rich Internet Applications (RIA) frameworks in order to support real-time interaction. This motivation has been getting pushed by the extended use of Internet applications like Social Networks (e.g., Facebook, Twitter, Google+ and LinkedIn) or Virtual Worlds (e.g., SecondLife, Wonderland and Qwaq) for instance (Coleman & Levine, 2008).

Attending to this situation we saw that a set of theories and models for understanding and providing awareness emerged in the early works reported in the CSCW literature. Gaver (1991) argued that an intense sharing of awareness characterizes focused collaboration in which people work closely together on a shared goal. He further claimed that less awareness is needed for division of labor, and that more casual awareness can head to serendipitous communication, which can turn into collaboration. Some lines of research focused on providing awareness using computational environments based on "event propagation mechanisms" for collecting, disseminating, and integrating information concerning collaborative activities in several groupwares that we can find available today.

In the CSCW context, the term groupware refers to an application that helps people work together collectively while located remotely (different place) or co-located (same place) from each other, and interacting synchronously (same time) or asynchronously (different time) (Ellis et al., 1991). One of the most general definitions was coined by Wells and Kurien (1996) *"Groupware is the software and hardware for shared interactive environments"*.

Our keystone to research around the groupwares and awareness is freeTribe (FRamework for dEvElopment of disTRIButed groupware)(Hurtado-Matos & Rodríguez-Cano, 2006), which has their design inspired in AMENITIES (MEthodology for aNalysis and desIgn of cooperative systEmS). AMENITIES is a methodology which allows addressing the analysis and design of CSCW systems systematically and which facilitates subsequent software development. It allows the realization of a conceptual model of cooperative systems and focuses on the group concept. It covers significant aspects of both group behavior (dynamism, synchronization, etc.) and structure (organization, laws, etc.). The resulting specification contains relevant information (cooperative tasks, domain elements, person-computer and person-person dialogues, etc.) to the creation of the user interface.

The objective of this paper is to present the current freeTribe design principles and characteristics, with an especial emphasis on the front-end extension to support synchronous awareness in the Web context. To do this, we have organized this paper as follows. In the next section we present a high-level overview of the *freeTribe* fundamentals. In the Section 3 is presented the explanation of RIA infrastructure that we use to update *freeTribe*. Then, in Sections 4 and 6 we include in this paper the awareness techniques and mechanisms supported by the new version of *freeTribe*. Finally, we conclude this paper in Section 6 by summarizing the exposed topics.

2. Overview of the freeTribe fundamentals

The development of collaborative systems is a complex task, which involves software technologies and cognitive sciences in different areassuch as distributed programming, human-computer interaction and many others. This situation is not ideal because it requires great programming efforts. Fortunately, design patterns, software frameworks, and middlewares are increasing their

popularity since they have a high reusability impact and suitable relationships (Schmidt & Buschmann, 2003) To address the freeTribe implementation problem, we have designed it as a groupware framework. Some methodologies for the development of a framework have been suggested that use domain analysis, software evolution, and design patterns. This section presents an overview of design patterns, frameworks, and middlewares and describes how these technologies complement each other to enhance their reuse and productivity.

2.1. Middlewares

As mentioned before, a groupware supports collaboration among group members that can be in different places at the same time. This capability requires a distributed architecture, usually Web applications with client/server architecture. This model is very useful when collaboration is asynchronous (e.g. the e-mail applications), but in synchronous interactive situations it is not very efficient. For that reason we consider the use of a middleware-based architecture as mechanisms of distributed communication, instead of a Web-based infrastructure. We wish to emphasize than with the middleware-based architecture it is also possible to surf the Web.

Over the past decade, a number of object-oriented middleware standards have emerged and matured, such as the Common Object Request Broker Architecture (CORBA) and Distributed Component Object Model (DCOM). Currently, the developers who are looking for an object-oriented middleware platform are offered some alternatives. Therefore, it is important to make careful selections. For example, .NET/WCF has the drawback that it supports only a limited number of languages and platforms. Java Remote Method Invocation (RMI) is a Java-only solution. CORBA has got the high degree of complexity of an aging platform, coupled with ongoing vendor attrition. Web Services have

severe inefficiencies and the need of using proprietary development platforms, as well as security issues (Henning & Spruiell, 2012).

For our purposes, we have selected the Internet Communications Engine (Ice) because its applications are open-source, suitable for use in heterogeneous environments: client and server can be written in different programming languages, run on different operating systems and machine architectures, and they communicate using a variety of networking technologies. The source code for these applications is portable regardless of the deployment environment (Henning & Spruiell, 2012).

2.2. Object Oriented Frameworks

A framework is a collection of classes that provides a set of services for a particular domain; a framework thus exports a number of individual classes and mechanisms that developers can reuse or adapt. It is characterized by three important features (Fayad et al., 1999):

First, a framework exhibits Inversion of Control (IoC) at runtime via callbacks to component hook methods after the occurrence of an event such as a mouse click or data arriving on a network connection. When an event occurs, the framework calls back to a virtual hook method in a pre-registered component which then performs application-defined processing in response to the event. The virtual hook method in the components decouples the application software from the reusable framework software, which makes it easier to extend and customize the applications as long as the interaction protocols and quality properties are not violated.

Second, a framework provides an integrated set of domain-specific structures and functionalities. Reuse of software depends largely on how well frameworks model the commonalities and variability in application domains. By leveraging the domain knowledge and prior efforts of experienced developers, frameworks embody common solutions to recurring application requirements and software design challenges that need not be recreated and re-validated for each new application.

Finally, a framework is a semi-complete application that programmers can customize to form complete applications by extending reusable components in the framework. In particular, frameworks help the canonical control flow of applications in a domain into architectures and families of related components. At runtime, these components can collaborate to integrate customizable application-independent reusable code with customized application-defined code.

2.3. Design patterns

The design of a groupware framework can be greatly improved by using design patterns. A design pattern is a description of communicating objects and classes which is customized to solve a general design problem in a specific context. Each pattern represents a common and recurring design solution which can be applied over and over again in different problem-specific contexts (Gamma et al., 1995). Patterns provide the designer with:

a. Abstract templates on how to make specific parts of a framework more flexible towards changes.
b. A mechanism to document the architecture of a framework using a high abstraction level vocabulary.
c. A mechanism to impose rules about how to reuse or extend the framework.

d. A higher level of documentation for a complex framework consisting of numerous heavily interconnected classes and objects.
e. Guidance on how to extend the framework with new variations and whether or not extensions can be made.

2.5. AMENITIES methodology

The natural complexity of CSCW systems demands great efforts in specifications and development. The development of groupware applications is more difficult than that of single-user applications given that social protocols and group activities must be considered in order to obtain a successful design. AMENITIES (Garrido et al., 2004) is a methodology which allows addressing the analysis and design of CSCW systems systematically and which facilitates subsequent software development. It allows the realization of a conceptual model of cooperative systems and focuses on the group concept. It covers significant aspects of both group and structure. The resulting specification contains relevant information (cooperative tasks, domain elements, person-computer and person-person dialogues, etc.) to the creation of the user interface.

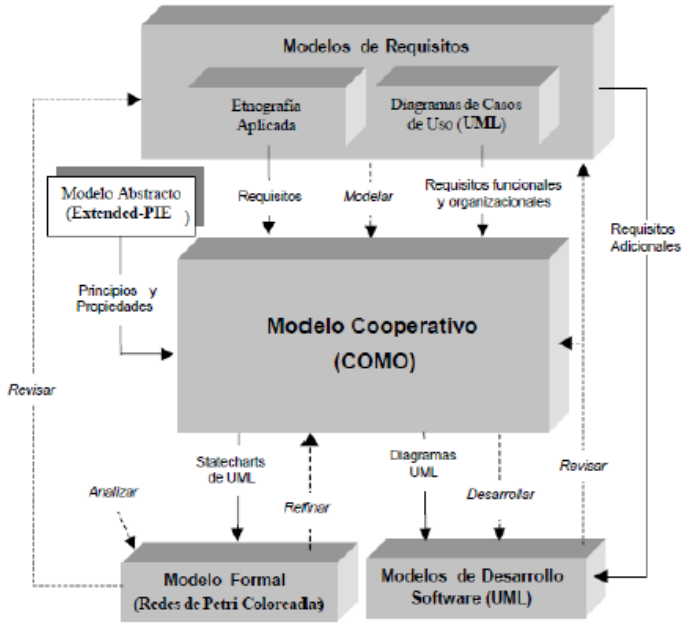

Figure 1. General diagram of AMENITIES methodology (Garrido, et al., 2004).

This methodology proposes the description of a cooperative system at two sets of models (Garrido, et al., 2004):

1. Models used in techniques for the capture and description of requirements. The requirements elicitation process is mainly accomplished by means of the application, mainly, of ethnography and use case techniques.
2. Cooperative model: It is a conceptual model that describes the basic structure and behavior of the complete cooperative system. This model is built hierarchically on the basis of other models, each one focused on providing a different view of the system. A structured method is proposed in order to build the cooperative model systematically. This method consists of the following stages: Specification of the organization, role definition, task definition and specification of interaction protocols.

The General diagram of AMENITIES methodology is presented at Figure 1, which shows the principal models and general stages. The general stages are (1) system analysis and obtaining of requirements; (2) modeling of the cooperative system; (3) analysis of the cooperative model; (4) system design; and (5) system developing. AMENITIES follow an iterative simple process, allowing carrying out a refining of the model as a consequence of the analysis of this, as well as a revision of the requirements from the start or of the cooperative model that could contribute news or different information to consider.

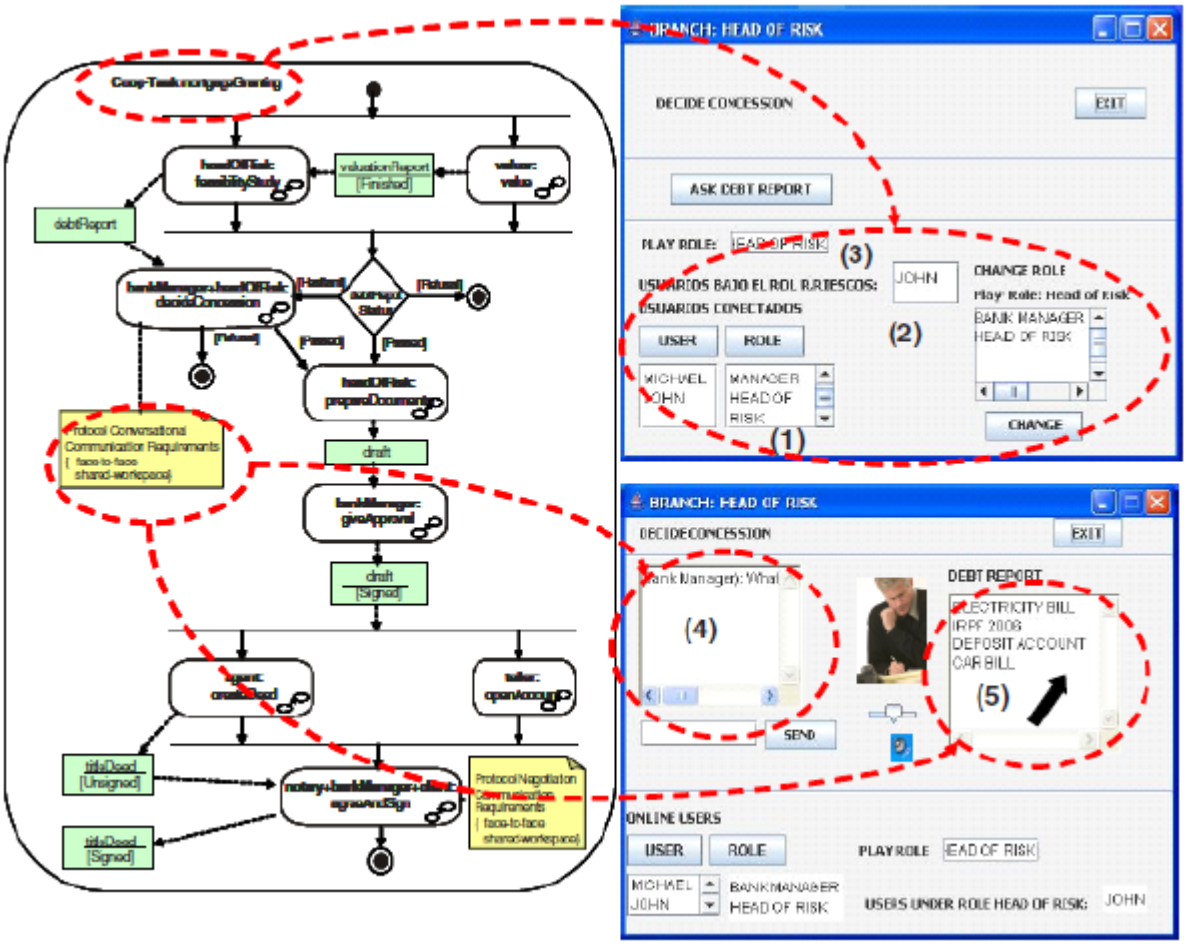

Figure 2. Mapping of cooperative task and interaction protocols to groupware components
(Rodríguez et al., 2007).

A case of study that apply this methodology was presented by Garrido (Garrido, et al., 2004), they considered a case study based on a help system for the decision of risky operations by financial institutions. In this study, they described a business process to grant a mortgage which a client has applied for in a branch office. The first step in a business process to grant a mortgage consists of realizing a feasibility study and making a report with all the information. The case study includes three organizations: branch, valuation office and notary office. The Branch organization has three roles: Bank Manager, Head of Risk and Teller. In Figure 3 they show an example of user interface of the *subactivity*

decideConcession, which presents a shared workspace (the Debt Report) and a *DIChat* (Chat for online users) component (4) to implement the interaction between the actor playing the *bankManager* role and the actor playing the *headOfRisk* role. Besides, the users can observe a *Telepointer* (Pointer movements of another user) component (5) on debt report corresponding to the action of the actor playing the *bankManager* role at that moment.

3. Making Web-based infrastructure with RIA

Today we save our information in Web sites, more than that, we have Web applications. That's where the RIA comes in. A RIA isn't a single specific thing; it's more of a paradigm, almost an approach to Web application development. RIA are characterized by appearing in many ways to look, feel, and function just like those native applications we left behind.

Ajax (Asynchronous JavaScript and XML) represents a paradigm shift for some people (even most people, given what most Web applications are today) because it can fundamentally change the way you develop a Web application (Ullman, 2012). The term AJAX is an overly-complicated acronym for saying that processes can take place in the background while the user is performing other tasks.

Ajax is a kind of next-generation DHTML; hence, it relies heavily on JavaScript to listen to events triggered by user activity and manipulates the visual representation of a page (that is, the document object model, or DOM) in the browser dynamically (Fränkel, 2011). Ajax is, at its core, an exceedingly simple, and by no stretch of the imagination original, concept: it is not necessary to refresh the entire contents of a Web page for each user interaction, or each

event, if you will. The server is no longer completely responsible for rendering what the user sees; some of this logic is now performed in the user's browser.

At the moment, we have literally thousands of JavaScript libraries to choose from, and many of them are rather good (others, not so much). Researchers have dedicated much time to the study of how CSCW technologies might create some level of awareness between workers. Systems have been designed to enhance collaboration through the provision of information to create or maintain awareness of the group members. Even though different approaches have been introduced to address awareness, its creation and maintenance, researchers agree that most collaboration demands knowledge of others' activities, and many have argued extensively that awareness is crucial for groups when performing their joint activities.

An important share of applications developed today uses thin-client paradigm (Fränkel, 2011), most of the time with a touch of Ajax augmentation. Unfortunately, there is no clear leader for web applications. Some reasons include the following:

- *Flex* would be a good candidate, as the technology is mature and Adobe a commercial force to be reckoned with, but Apple did not add the Flash player to its iOS platforms. Thus, surfing mobile with these devices cuts you from Flex content.
- *Ext JS* makes web application development simple by: providing easy-to-use cross-browser compatible widgets, interacting with the user and browser via the EventManager, and communicating with the server in the background without the need to refresh the page, but it still do not have a good integration with Java.
- Most developers know how to develop plain old web applications, with enough Ajax added in order to make them usable by users.

- *ZK* is an event-driven, component-based framework to enable rich user interfaces for web applications. ZK includes an Ajax-based event-driven engine, a rich set of XML User Interface Language and XHTML components, and a markup language called ZK User Interface Markup Language. ZK does not require you to have any knowledge of JavaScript to develop Ajax-based web applications, since the ZK engine auto-generates the JavaScript code, and offers a good integration with Java.
- *GWT*, although new and original, is still complex and needs seasoned developers in order to be effective .

3.1. Vaadin framework

Vaadin Framework is a Java web application development framework that is designed to make creation and maintenance of high quality web-based user interfaces easy. Vaadin supports two different programming models: server-side and client-side. The server-driven programming model is the more powerful one, and essentially lets you forget the web and program user interfaces much like you would program any Java desktop application with conventional toolkits such as AWT , Swing, or SWT, but easier (Fränkel, 2011).

While traditional web programming is a fun way to spend your time learning new web technologies, you probably want to be productive and concentrate on the application logic. The server-side Vaadin framework takes care of managing the user interface in the browser and the AJAX communications between the browser and the server. With the Vaadin approach, you do not need to learn and debug browser technologies, such as HTML or JavaScript.

For our purposes, we have selected this framework because its represent a unique framework in the current ecosystem in order to develop rich CSCW systems; its differentiating features include the following (Fränkel, 2011):

- There is no need to learn different technology stacks, as the coding is solely in Java. The only thing to know beside Java is Vaadin's own API, which is easy to learn. This means: the UI code is fully object-oriented, here's no spaghetti JavaScript to maintain, furthermore, the IDE's full power is in our hands with refactoring and code completion.
- No plugin to install on the client's browser, ensuring all users that browse our application will be able to use it "as is".
- As Vaadin uses GWT under the cover, it supports all browsers that GWT also supports. Therefore, we can develop a Vaadin application without paying attention to the browsers and let GWT handle the differences.
- Moreover, Vaadin uses an abstraction over GWT so that, in theory, you can use another rendering engine, even Swing! This architecture works toward alleviating risks of GWT becoming a closed source in the future and the Vaadin team is committed to open source.
- Finally, Vaadin conforms to standards such as HTML and CSS, making the technology future proof. For example, many applications created with Vaadin run seamlessly on mobile devices although they were not initially designed to do so.

4. Awareness techniques

Situation awareness research focuses on each individual’s capacity to perceive elements and the cognitive processes involved in maintaining awareness of the environment. Gutwin and Greenberg (1996) define workspace awareness as up-to-the-moment understanding of another person’s interaction with the shared workspace. It is knowledge about the group’s working environment, which creates an understanding of people within a workspace. In a collaborative environment, awareness involves knowledge about the people one is collaborating with (presence, identity, and authorship), the activities they are working on (actions, intentions and artifacts manipulated) and where (location of work, gaze direction, view and individual reach). Historical

awareness information also includes action, artifact, and event history and should be provided in asynchronous work situations.

Ethnographic studies have determined that awareness allows group members to manage the process of working together and is necessary for coordination of group activities (Dourish & Bellotti, 1992). Being aware of others' activities in a workspace allows participants to better understand the boundaries of their actions, which in turn help them, fit their own actions into the collaborative activity stream. This also enables groups to better manage coupling levels between their activities, helping people decide who they need to work with and when to make the transitions from looser to tighter coupling (Heath & Luff, 1991). Furthermore, awareness simplifies communication by allowing individuals to artifacts, the workspace can be used as reference the shared environment and elements within it: When discussing shared a communication prop (Brinck & Gomez, 1992). This makes awareness an important building block for the construction of team cognition (Gutwin & Greenberg, 2004) and an enabler of shared understanding that allows individuals to get a better sense of the work that is being performed by others (Gutwin, Greenberg, Blum, & Dyck, 2005).

5. Strategic alienation to update freeTribe

There are several ways of defining and implementing awareness. Various research projects have used their own taxonomy and interpretation of awareness for creating frameworks and systems. For instance, Gutwin & Greenberg (2002) classified awareness in two types: situational, and workspace and they suggested that situational awareness underlies the idea of workspace awareness in groupware systems. Their definition of workspace awareness included how people interact with the workspace, rather than just awareness of the workspace itself. Simone and Bandini (2002) identified two

kinds of awareness: (a) by-product awareness that is generated in the course of the activities people must do to accomplish their collaborative tasks; (b) and add-on awareness that is the outcome of an additional activity, which is a cost for the collaborators to what they must do and is discretional in that it depends on collaborators' evaluation of the contingent situation. Chalmers (2002), likewise, divided the awareness in two kinds: awareness of people and of information artifacts. He suggested implementing activity centered awareness tool, in that it focuses on presenting the ongoing appearance and activity of people.

For the purpose of the work reported here, a more comprehensive and well-accepted taxonomy of awareness, which addresses four kinds of awareness (Liechti & Sumi, 2002) as listed below, will be used.

1. Group awareness. This kind of awareness includes providing information to each group member about the status and activities of the other collaborators at a given time.
2. Workspace awareness. This emphasizes the fact that awareness generally emerges when people share a space. In other words, this kind of awareness refers to a common space that the group members share and where they can bring and discuss their findings and create a common product.
3. Contextual awareness. This type of awareness relates to the application domain, rather than the users. Here, we want to identify what content is useful for the group and what the goals are for the current project.
4. Peripheral awareness. This refers to the human ability toprocess information at the periphery of the attention, witha very low overhead. In other words, peripheral awareness relates to the information that should be kept separate (on their periphery) from what a participant is currently viewing or doing.

freeTribe is a framework FLOSS (Free Open-Source Software) implemented using the Java Platform and ICE middleware. These technologies allow to freeTribe build CSCW systems based applications are open-source, suitable for use in heterogeneous environments: client and server can be written in different

programming languages, run on different operating systems and machine architectures, and they communicate using a variety of networking technologies.

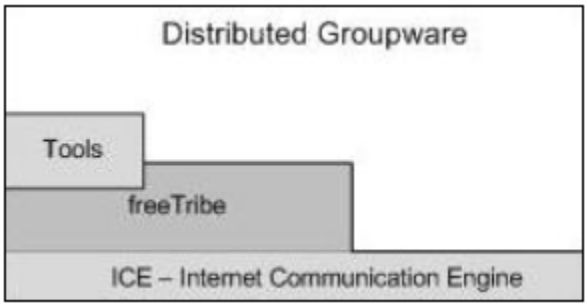

Figura 3. freeTribe software architecture

Figure 3 shows freeTribe software architecture based on three layers:

- First Level: the lower level is the base layer that manages the communications protocols.
- Second Level: it contains the necessary services to represent the concepts of AMENITIES (Garrido, et al., 2004), as well as the security and awareness services.
- Thi rd Level: at the upper level, there is a tool layer that represents systems developed over lower layers and a collaborative server.

The Figure 4 shows the relationship betwen Amenities, freetribe and distributed groupwares.

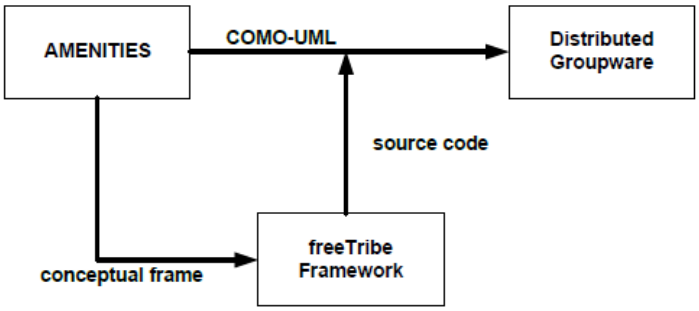

Figure 4. Relationship betwen Amenities, freetribe and distributed groupwares.

6. Conclusion

The CSCW community has been developing a lot of groupwares in the Web context through the last few years, but many of these developments address only specific problems or do not adequately support efficient synchronous awareness interaction. Therefore, it is convenient to see around the RIA frameworks and middlewares platforms as complements for traditional Web development technologies. In this paper we presented four kinds of synchronous awareness techniques through its implementation in *freeTribe* with the complements Vaadin and ICE.

Acknowledgements

We would like to thank Livan K. Badías-Ibarra from the University of Informatics Science (Cuba), for support and discussions concerning with freeTribe evaluation and for all of freeTribe community members for thei r valuable time and continues inputs.

References

Coleman, D., & Levine, S. (2008). *Collaboration 2.0: Technology and Best Practices for* Successful Collaboration in a Web 2.0 World. Silicon Valley, California, USA: HappyAbout.info.

Ellis, C. A., Gibbs, S., & Rein, G. (1991). Groupware: Some issues and experiences. Communications of the ACM, 34(1), 9-28.

Fayad, M., Schmidt, D., & Johnson, R. (1999). Building application frameworks: Object-oriented foundations of framework design.

Fränkel, N. (2011). Learning Vaadin powered by Vaadin-built RIAs.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*: Addison-Wesley.

Garrido, J., Gea, M., Noguera, M., González, M., & Ibáñez, J. (2004). Una Propuesta Arquitectónica para el Desarrollo de Aplicaciones Colaborativas. *Interacción 2004*, 164--171.

Henning, M., & Spruiell, M. (2012). *Distributed Programming with Ice*: ZeroC.

Hurtado-Matos, L. I., & Rodríguez-Cano, J. C. (2006). *Proposal of a Framework for Distributed/ Groupware Development*. Ingeniería Informática, Título de Grado, Universidad de Holguín, Holguín, Cuba.

Rodríguez, M. L., Garrido, J. L., Hurtado, M. V., & Noguera, M. (2007). An Approach to the Model-Based Design of Groupware.

Multi-user Interfaces. *CRIWG 2007*, 157–164.

Schmidt, D., & Buschmann, F. (2003). *Patterns, frameworks, and middleware: Their synergistic relationships*. Paper presented at the 25th International Conference on Software Engineering.

Ullman, L. (2012). *Modern JavaScript: Develop and Design*: Peachpit Press.

Biographical notes:

J. C. Rodríguez-Cano is PhD candidate from the University of Granada (Spain), Rodríguez-Cano received his Software Engineering degree in 2006 at the University of Holguín (UHO), Cuba. Since 2006, he has worked on CSCW projects within the Research Group in Distributed Systems at the UHO. Currently, he is professor at the Department of Informatics at the UHO, Cuba. His research interests include collaborative information retrieval (IR) techniques, search-driven software development, and P2P retrieval. He has been co-organizer of the Modern Applications of IR workshop held at V International Conference UHO 2011, co-organizer of the Collaborative Information Retrieval workshop held at CIKM 2011, and co-organizer of the Collaboration, Recommendation and Social Search workshop held at VI International Conference UHO 2013.

Ricardo_Delgadillo_Lizaola is PhD candidate from the University of Granada (Spain), professor in the engineering department of CUCSUR (Centro Universitario de la Costa Sur) at the University of Guadalajara, since 1993. He has taught at several universities and has worked as a consultant in the implementation of new technologies in different companies. Currently working on several projects related to CSCL: Computer-supported collaborative learning with professors from the University of Olguin (CUBA) and Pontifical Catholic University of Peru (PERU). His research interests include Soft computing applications for database technologies, collaborative information retrieval (IR) techniques, retrieval and Cognitive Load Measurement in Web Search, Computer-supported Cooperative Work (CSCW) and Computer-Supported Collaborative Learning (CSCL).

Humberto Rodríguez-Avila is PhD candidate from the University of Granada (Spain), Rodríguez-Avila received his Software Engineering degree in 2011 at the University of Holguín (UHO), Cuba. Since 2011, he has worked on CSCW & CIS projects within the Research Group in Distributed Systems at the UHO. Currently, he is professor of Object-Oriented Programming, Data Structures and Algorithms at the Department of Informatics at the UHO, Cuba. His research interests include Collaborative Information Seeking techniques, P2P retrieval and Cognitive Load Measurement.

Eduardo Escofet got his B.Sc. from the Central University of Las Villas, Cuba in Computer Sciences Specialized in Artificial Intelligence in 1994. He obtained a M.Sc. degree in Business Information Technology specialized in Business Executive Decision Making in 1997 from the University of Holguín, Cuba and

another M. Sc. in Information Technologies and Soft-computing in 2003 from the Univeersity of Granada, Spain. He has lectured at several universities, developed multiple software products and advised various theses and software projects. Nowadays, he is working in software-driven innovation companies as project manager, software developer and quality engineer. His main areas of interests are application framework development, service-oriented architecture integration, business intelligence engineering, Web and mobile-based business strategies and software quality improvement methods.

ReCIBE, Año 2 No.1, Mayo 2013

Influencia de los Roles de Equipo en las Actividades del Desarrollador de Software

Elsa Estrada Guzmán
Computer Science Department
CUCEI – Universidad de Guadalajara
elsa.estrada@red.cucei.udg.mx

Adriana Peña Pérez Negrón
Computer Science Department
CUCEI – Universidad de Guadalajara
adriana.ppn@red.cucei.udg.mx

Resumen: Uno de los roles básicos en el proceso del software es precisamente el de desarrollador, también denominado ingeniero de software, cuyas actividades principales son: el análisis, diseño, programación y pruebas del producto a desarrollar. Estas actividades, dependiendo generalmente del tamaño del proyecto y de la metodología, pueden estar a cargo de diferentes personas o bien de un grupo de desarrolladores que en conjunto las llevan todas a cabo; en este último caso, estaríamos hablando de trabajo en equipo entre iguales o pares. Por otro lado, de acuerdo con la teoría de roles de equipo, las personas tienden a comportarse de manera regular en forma distintiva cuando colaboran, estas formas particulares de colaborar es

probable que influyan en el desempeño del equipo de desarrolladores de software. En este documento se presenta un caso de estudio con la finalidad de entender la influencia de los roles de equipo en ciertas actividades involucradas en el proceso de desarrollo de software.

Palabras clave: **desarrollador de software, ingeniero de software, roles en la ingeniería de software, roles de equipo de Belbin**

Team role influences in software development activities

Abstract: **One of the basic roles in the software process is precisely that of a developer, also called software engineer, whose main activities are: analysis, design, programming and product testing for said product. These activities, usually based on the project size and methodology, they can be assigned to different people or to a group of developers to take care of them; in this former case we would be talking about group work among peers. On the other hand, according to the team role theory, people tend to behave in a specific way when they collaborate, these particular collaborative behaviors probably have an influence on the software developer team's performance. This document presents a case of study, with the intention of understanding the influence of team roles in certain activities involved in the software development process.**

Keywords: **software developer, software engineer, software engineer roles, Belbin team roles.**

1. Introducción

La ingeniería de software entendida como la aplicación sistemática, disciplinada y cuantificable para el desarrollo, operación y mantenimiento de software (Computer Society of the IEEE, 1990), es considerada como portadora de paradigmas o patrones de trabajo. Las metodologías del software especifican la serie de pasos o actividades a seguir para el desarrollo de un

sistema, que van desde la identificación de las necesidades hasta su ejecución a través de su construcción, su operación y su retiro (Dorfman, 1997).

Para llevar a cabo un proyecto, sus ejecutores suelen tener diferentes responsabilidades y derechos a fin de facilitar la organización de las actividades encaminadas al cumplimiento de los objetivos. De igual manera, para el proceso de software las actividades son asignadas con base a la ingeniería del software, de acuerdo a las funciones requeridas en el modelo implementado, estos son los llamados roles en la ingeniería de software.

Así pues, los patrones que conforman los diferentes paradigmas de la ingeniería del software se rigen por ciertos criterios esenciales que los distinguen y clasifican de acuerdo a su enfoque, sus metodologías y sus modelos. En este contexto, es importante establecer que independientemente del modelo y los roles de ingeniería del software que maneje un enfoque, el rol de desarrollador es imprescindible.

Por ejemplo, en lo que se refiere a métodos ágiles, para el modelo XP los roles de ingeniería básicos reconocidos son (Chromatic, 2003): el cliente que representa a los usuarios finales; *el rastreador* que verifica si el proyecto va en iempo; *el entrenador* que guía al equipo para entender XP y *el desarrollo* de software sugiriendo cambios en la implementación; y el desarrollador. Mientras que en el modelo SCRUM orientado al desarrollo de software en equipo existen tres roles: *el propietario del producto* que decide lo que será construido; *el gerente de SCRUM o facilitador*, que es el líder del equipo y *el equipo*, quienes desarrollan el producto a entregar, los desarrolladores (ScrumAlliance, 2008).

Aún más, de acuerdo con un estudio para determinar los requisitos de capacitación del perfil profesional para los ingenieros de software encaminado a las necesidades de las micro y pequeñas empresas, los roles de ingeniería del software requeridos son (Montilva, Barrios, & Rivero, 2004): *líder de proyectos de software* que ejecuta la planificación y control de los proyectos; *el*

ingeniero de soporte que asegura la calidad y valida el software; *el ingeniero de operación y mantenimiento de software* que administra las aplicaciones y las bases de datos e implementa los cambios; y *el desarrollador de software*.

De tal forma que independientemente de las diferentes modalidades de los roles de la ingeniería del software, la figura del desarrollador deberá estar presente, aunado a sus actividades principales que comprenden el análisis, diseño, codificación o implementación y pruebas del producto a desarrollar.

A este respecto, el estándar 1074 de la IEEE muestra las actividades del ciclo de vida del software agrupadas en las fases: Análisis, Diseño, e Implementación. Esta última actividad se subdivide en codificación, pruebas, e instalación (IEEE Computer Society, 1997). Es importante hacer notar que aunque existe una revisión posterior de este estándar hecha en el 2006, aún no está aprobada. Utilizaremos entonces la siguiente clasificación considerando el trabajo del programador independiente del de probador, como se muestra en la Tabla 1.

Tabla 1 Responsabilidades del Ingeniero de Software de acuerdo al estándar 1074 de la IEEE

Ingeniero de Software	Responsabilidades
Analista	Definición de los requerimientos del software. Análisis de las funciones requeridas por el sistema. Desarrollo de la arquitectura funcional. Análisis de la descomposición de los requerimientos del sistema.
Diseñador	Diseño de la arquitectura de los datos. Diseño de la base de datos. Diseño de interfaces. Diseño de detalle de la arquitectura del software.
Programador	Creación código ejecutable. Creación documentación de operación.

Ingeniero de Software	Responsabilidades
	Integración del entorno. Conducir revisiones del software.
Probador	Definición de la arquitectura de las pruebas. Diseño de casos de prueba. Ejecutar pruebas del software.

Siendo que las micro y pequeñas empresas dedicadas al desarrollo de software representan un porcentaje mayoritario de este importante sector de la economía mundial (Montilva, Barrios, & Rivero, 2004), la tendencia parece moverse hacia el uso de grupos de desarrolladores con la misma jerarquía que desempeñan las diferentes responsabilidades del desarrollador, esto es el trabajo en equipo entre pares, relación en la que nos enfocamos en este documento.

Una importante propuesta a este respecto es el Team Software Process o TSP basado en el modelo Personal Software Process o PSP (Humphrey, The Personal Software Process (PSP), 2000), para el que los roles de ingeniería transforman a los miembros del equipo en co-administradores, manejando dos tipos de roles para cada miembro (Humphrey, Chick, Nichols, & Pomeroy-Huff, 2010): un *rol administrativo* con responsabilidades que se distribuyen entre los miembros del equipo y que comprenden a los diferentes administradores de planeación, del proceso, de calidad, de soporte, de interfaz del cliente, de diseño, de codificación y de pruebas; y un *rol de miembro de equipo* o ingeniero de software, el desarrollador.

Sin embargo, aún cuando hay equipos que parecen haberse hecho a medida, existen otros en los que sus miembros batallan para adaptarse o incluso les es imposible hacerlo. Esta diferencia podría explicarse por medio de la teoría de roles de equipo.

1.1 Roles de equipo de Belbin (REB)

La teoría conocida como “Belbin Team Role” o Roles de Equipo de Belbin (REB) del Dr. Meredith Belbin, basada en el estudio de los roles y responsabilidades de los miembros integrantes de un equipo, sostiene que cada individuo posee un comportamiento en el entorno laboral, al mismo tiempo que desempeña de manera más eficaz aquellas funciones que le son más naturales (Belbin M. , 2013). En esta teoría se han identificando nueve roles cuya breve descripción se muestra en la Tabla 2.

Tabla 2. Descripción de los Roles de Equipo de Belbin

REB	Descripción
Planeador/Cerebro	Creativo, imaginativo, no ortodoxo resuelve problemas difíciles.
Investigador de recursos	Extrovertido, entusiasta, comunicativo. Explora oportunidades. Hace contactos.
Coordinador	Maduro, confidente. Un buen líder, clarifica metas, promueve la toma de decisiones, delega.
Impulsor	Retador, dinámico, trabaja bajo presión, tiene el coraje para manejar y superar obstáculos.
Cohesionador	Cooperador, apacible, perceptivo y diplomático. Escucha e impide los enfrentamientos.
Monitor/Evaluador	Sobrio, estratégico y perspicaz, ve todas las opciones. Juzga con precisión.
Implementador	Disciplinado, confiable, conservador. Cambia ideas en acciones prácticas.
Finalizador	Esmerado, consciente, ansioso. Busca errores y omisiones, entrega en tiempo.
Especialista	Dedicado, aporta cualidades y conocimientos específicos.

De acuerdo con Belbin, la clave para la efectividad de los equipos está en la combinación y equilibrio entre los diferentes REB dentro del equipo. Mientras que la combinación se refiere a que un equipo debe estar formado preferentemente por REB compatibles, el equilibrio es la importancia de reconocer a todos los comportamientos como necesarios, pero sin excederse en la cantidad de miembros que se identifiquen con el mismo REB (Belbin M. , 2012).

En las relaciones colaborativas de trabajo de equipo entonces, se llegan a presentar casos de compatibilidad e incompatibilidad de REB cuando se trabaja entre pares o iguales, tal como se muestra en la Tabla 3 (Aguilar, 2008 pp. 35).

Tabla 3. Relaciones entre roles de equipo de Belbin

Rol	Compatible con	Incompatible con
Impulsor	Investigador de Recursos	Coordinador Cohesionador
Implementador	Coordinador Investigador de Recursos Monitor-Evaluador Especialista Finalizador	Implementador Planeador/Cerebro
Finalizador	Implementador	Investigador de Recursos Monitor- Evaluador
Coordinador	Planeador/Cerebro Implementador	Impulsor
Cohesionador	Cohesionador Planeador/Cerebro	Impulsor
Investigador de Recursos	Cohesionador Implementador	Finalizador Especialista

Rol	Compatible con	Incompatible con
Planeador/Cerebro	Coordinador Investigador de Recursos Cohesionador	Monitor-Evaluador Planeador/Cerebro Especialista Implementador
Monitor-Evaluador	Coordinador Implementador	Finalizador Monitor-Evaluador Planeador/Cerebro
Especialista	Implementador Cohesionador	Planeador/Cerebro Investigador de Recursos

Con la intención de mejorar la calidad en el trabajo de equipo, que a su vez se verá reflejado en la calidad del producto de software, se realizó un estudio en el que se observaron los REB para el trabajo colaborativo durante el desarrollo de software. Las preguntas que condujeron el estudio son:

1. ***¿Qué combinaciones de REB producen mejores resultados en la práctica de los equipos de desarrolladores de software?***
2. ***¿Existe un REB asociado a las diferentes actividades del desarrollador de software?***

2. Caso de estudio

2.1 Método experimental

Este estudio se aplicó a cuatro grupos de alumnos que llevan el curso: Taller de Estructuras de Archivos. Los 56 estudiantes que conforman los cuatro grupos pertenecen o bien a la carrera de Ingeniería en Computación o a la de Licenciatura en Informática.

Tres de estos grupos fueron experimentales y uno de control. Para los experimentales se formaron los equipos con miembros de la misma clase de manera aleatoria, al grupo control se les permitió formar los equipos con los

miembros de su elección. Se conformaron 18 equipos, doce en los grupos experimentales: once de tres integrantes y uno de cuatro; en el grupo control seis: cinco de tres integrantes y uno de cuatro.

Cada equipo recibió la descripción de un problema a ser resuelto mediante el desarrollo de una aplicación que debería incluir el manejo de archivos. El tiempo para llevar a cabo la práctica se fijó en 2 horas.

Se hicieron dos propuestas de ejercicio de entre las cuales los equipos podían elegir:

1. Hacer un programa para el registro de marcas y registro de productos utilizando multilistas en archivos.
2. Hacer un programa para el registro de derechohabientes del seguro social de salud, utilizando archivos de dispersión para resolver el problema de códigos duplicados.

Se entregó a los equipos la lista de requerimientos funcionales para cada sistema:

> Caso 1: Registrar marcas de productos, registrar productos y consultar productos de una marca.
>
> Caso 2: Registro de derechohabientes y consulta de derechohabiente.

Finalmente, para la entrega se les pidió presentar tres archivos que deberían contener:

1. El modelado de los archivos y los datos para su manipulación;
2. El modelado de las funciones/operaciones utilizando, e.g. diagramas o pseudocódigo;
3. El código fuente.

El ejercicio consistió en la aplicación de técnicas de ingeniería de software para las fases: 1) modelado del análisis, 2) modelado del diseño y 3) codificación y ejecución. Cada una de estas fases se calificó en una escala de 0 a 5.Los puntos a evaluar en cada fase se muestran en la Tabla 4.

Tabla 4. Funciones a evaluar por equipo

Análisis	Diseño	Codificación y Depuración
Definición Estructuras de Archivos	Definición de Nombres de funciones	Los tipos de datos declarados se identifican con la actividad 1 o 2
Definición de Estructuras de datos	La o las funciones utilizan las estructuras de datos modeladas	Número de funciones que se ejecutan exitosamente.
Claridad (Nombres y tipos)	La o las funciones utilizan las estructuras de archivos modeladas %	
Utilización de alguna técnica de modelado	La o las funciones usan el o los nombres de archivos identificados en la actividad 1	
	Las funciones se identifican con los requerimientos y opciones del sistema.	

Recursos

Todos los equipos contaron con al menos una computadora con la aplicación Moodle instalada, esta aplicación es una plataforma para la administración de recursos pedagógicos. En el Moodle se colocaron dos archivos con los temas: hash, técnica utilizada en archivos para el almacenamiento y recuperación de datos; y multilistas, conjunto de registros almacenados relacionados entre sí

por direcciones o cursores que los ligan. Este material se publicó con una semana de anticipación a la fecha en que se llevó a cabo la práctica.

Datos

Se aplicó un cuestionario de autoevaluación de Belbin (Lindgren & Rolf, 1997) traducido al español, para identificar el REB de cada estudiante. Dos estudiantes no contestaron el cuestionario, uno perteneciente a los grupos experimentales y uno al grupo de control, ocasionando datos faltantes. La sumatoria de los REB de los 54 alumnos en cada grupo se presenta en la Tabla 5.

Tabla 5. Frecuencia de REB en los estudiantes

	Experimentales			**Control**	
REB	**Grupo 1**	**Grupo 2**	**Grupo 4**	**Grupo 3**	**Frecuencia**
Total	15	14	8	17	54
Investigador	0	0	0	1	1
Monitor	1	0	2	0	3
Impulsor	2	2	1	3	8
Coordinador	2	3	1	3	9
Especialista	3	1	0	1	5
Cohesionador	5	2	0	2	9
Cerebro	0	1	1	2	4
Finalizador	0	2	0	1	3
Implementador	2	2	3	5	12

Los equipos quedaron integrados de acuerdo a su REB como se muestra en la Tabla 6.

Tabla 6 Combinaciones de REB por equipo

Equipo	REB de los integrantes			
1	Coordinador	Finalizador	Finalizador	
2	Especialista	Cohesionador	Implementador	
3	Impulsor	Impulsor	Cohesionador	Cerebro
4	Coordinador	Coordinador	Implementador	
5	Coordinador	Implementador	Implementador	
6	Monitor	Impulsor	Implementador	
7	Monitor	Cerebro	*sin dato*	
8	Coordinador	Cohesionador	Cerebro	
9	Coordinador	Cerebro	Implementador	
10	Especialista	Finalizador	Implementador	
11	Coordinador	Cohesionador	Implementador	
12	Impulsor	Implementador	*sin dato*	
13	Investigador	Impulsor	Impulsor	Implementador
14	Monitor	Cohesionador	Cohesionador	
15	Especialista	Cohesionador	Implementador	
16	Coordinador	Cohesionador	Cohesionador	
17	Impulsor	Especialista	Implementador	
18	Impulsor	Coordinador	Especialista	

Equipos del grupo de control

También se aplicó un post-cuestionario impreso en el que se pidió a cada estudiante asignar un porcentaje de participación a cada miembro de su equipo (incluido él mismo) de acuerdo al desempeño en cada una de las actividades de desarrollo de software que se llevaron a cabo en el ejercicio, entre otras preguntas (ver Anexo). Debido a la tendencia de los alumnos a estandarizar los resultados, esto es a no utilizar muy bajos o muy altos porcentajes, además de que “los porcentajes” en ocasiones no daban el entero, se decidió que cuando al menos dos integrantes del equipo asignaron el mayor porcentaje o la más alta calificación en alguna de las actividades a un alumno, a éste se le sumaría un punto para esa actividad. En caso de que dos alumnos cumplieran con este criterio, a ambos se les sumó un punto. Los resultados asociados a cada REB se presentan en la siguiente Tabla 7.

Tabla 7 Puntos acumulados del desempeño en las actividades para cada REB

Frecuencia	REB	*Actividades del Desarrollador de Software*		
		Análisis	Diseño	Codificación y Depuración
1	Investigador	0	2	1
3	Monitor	1	5	1
8	Impulsor	4	2	2
9	Coordinador	1	3	3
5	Especialista	3	5	2
9	Cohesionador	3	0	2
4	Cerebro	0	1	2
3	Finalizador	3	4	3
12	Implementador	0	0	9

Los resultados de la evaluación por equipo para cada una de las fases se muestran en la Tabla 8. En la primera columna está el número del equipo, seguido del número de integrantes. En las siguientes tres columnas se encuentra el número de roles compatibles, incompatibles y los neutrales dentro del equipo. Posteriormente está la calificación para cada una de las actividades, en la penúltima columna el promedio y la calificación final redondeada en la última columna. Como se puede observar, de los 18 equipos: 6 equipos obtuvieron la calificación máxima, 10 equipos obtuvieron cuatro puntos y 2 equipos la mínima calificación obtenida de tres puntos.

Tabla 8. Combinación de roles por equipo y calificación obtenida

Equipo		No. de roles			Calificación de cada Actividad				Calificación Total	
Número	Integrantes	compatibles	Incompatible	neutrales	análisis	diseño	codificación	ejecución	Promedio	redondeada
1	3	0	0	6	4	4	5	5	4.5	5
2	3	3	0	3	5	5	3	3	4.0	4
3	4	2	4	6	4	5	5	4.5	4.6	5
4	3	4	0	2	5	5	3	3	4.0	4
5	3	4	2	0	5	5	3	3	4.0	4
6	3	2	0	4	5	5	3	3	4.0	4
7	3*	0	2	0	5	5	3	3	4.0	4
8	3	4	0	2	3	5	5	4	4.3	4
9	3	3	2	1	3	3	4	3	3.3	3
10	3	4	0	2	4	5	4	4	4.3	4
11	3	3	0	3	4	4	5	5	4.5	5
12	3*	0	0	2	4	4	5	5	4.5	5
13	4	5	0	7	3	5	5	5	4.5	5
14	3	2	0	4	3	3	3	3	3.0	3
15	3	3	0	3	5	5	5	5	5.0	5
16	3	4	0	2	5	5	4	3	4.3	4
17	3	2	0	4	5	5	4	3	4.3	4
18	3	0	2	4	5	5	4	3	4.3	4
.	Equipos del grupo de control				*	Grupos con un rol sin determinar				

Promedio por REB

Los datos de calificación por REB para cada una de las actividades y el promedio final se presentan en la Tabla 9.

Tabla 9 Calificaciones al desempeño por actividad por REB y su desviación estándar

		Calificación				
REB	Frecuencia	Análisis	Diseño	Codificación	Ejecución	**Total**
Investigador	1	3.00	5.00	5.00	5.00	4.50
Monitor	3	4.33	4.33	3.00	3.00	3.67
Impulsor	8	4.13	4.88	4.50	4.13	4.41
Coordinador	9	4.33	4.56	4.00	3.56	4.13
Especialista	5	4.80	5.00	4.00	3.60	4.38
Finalizador	9	4.11	4.44	4.11	3.72	4.11
Cerebro	4	3.75	4.5	4.25	3.625	4.05
Finalizador	3	4.00	4.33	4.67	4.67	4.43
Implementador	12	4.42	4.67	3.92	3.75	4.20
	54	4.10	4.63	4.16	3.89	4.21

3. Resultados

Es importante hacer notar que la frecuencia en el número de perfiles debido al tamaño de la población generó un total poco representativo en varios casos, por ejemplo, hubo un solo caso con perfil de Investigador.

En referencia a la pregunta 1, sobre la combinación de REB con mejores resultados, en el caso del grupo de control es interesante observar que solo hubo un equipo con roles incompatibles y que es el equipo con menor calificación. Para el resto de los equipos no existe correlación entre el número de REB compatibles o incompatibles y la calificación obtenida. Sin embargo, se encontró correlación estadísticamente significativa (.567 a nivel .01 con dos colas) entre el número de miembros con relación neutral y la calificación obtenida, esto es, a mayor número de relaciones neutrales, mayor calificación.

Respecto a la segunda pregunta sobre los REB asociados a las distintas actividades del desarrollador, conforme con la autoevaluación de los

compañeros de equipo acerca de su aportación, ver Tabla 7, parece claro que el Implementador (9 estudiantes) suele tener una clara aportación para el trabajo de codificación. Mientras que el Especialista (5 estudiantes), el Monitor (3 estudiantes) y el Finalizador (4 estudiantes) hicieron una buena aportación durante el diseño. En el caso de la actividad de análisis, los alumnos con REB de Impulsor (4 estudiantes) fueron los que se distinguieron por su aportación.

La frecuencia en los roles de equipo también podría ser un indicador en la tendencia de ciertas formas características de colaborar que adoptan los desarrolladores de software; los REB con mayor frecuencia fueron los de Implementador, Coordinador y Finalizador.

4. Conclusiones y Trabajo Futuro

Particularmente en las micro y pequeñas empresas que constituyen un importante porcentaje de la industria desarrolladora de software, las actividades del desarrollador de software tienden a realizarse en equipos de trabajo con miembros de la misma jerarquía, esto es, en colaboración entre pares. En este contexto, es importante que además de las capacidades de los desarrolladores se estudien las características que deban tener los miembros del equipo para que éste funcione adecuadamente.
De acuerdo con la teoría de los roles de equipo de Belbin, las personas desempeñan de manera más eficaz aquellas funciones que le son más naturales, mismas que hacen más eficiente el trabajo en equipo. En este documento se presentó un caso de estudio en el que se observaron los roles de equipo en relación a algunas de las actividades que desempeñan los desarrolladores de software: análisis, diseño y codificación de software.

A pesar de una muestra poblacional relativamente pequeña y considerando que existen 9 diferentes roles de equipo, se establecieron algunas

observaciones como el hecho de que un mayor número de miembros con roles neutrales y no el número de relaciones compatibles o incompatibles, influyeron en el desempeño del equipo. También se encontró que algunos roles corresponden a una mayor aportación con ciertas actividades como el caso del Implementador con la codificación.

Trabajos futuros

La formación de equipos de desarrollo de software es una tarea en la que aún se sigue investigando. Consideramos gratificantes los logros del trabajo en equipo y cuyos beneficios permiten alcanzar altos objetivos en la práctica de la ingeniería de software. Es por esto que es importante dar continuidad en este tipo de estudios que deseamos extender por ejemplo: realizando el experimento con alumnos de grados de estudio más avanzados para tener una mejor visión en los resultados; revisando la forma de evaluar las actividades por equipo en el ciclo de vida del software y estudiando el mecanismo que utilizan las personas para identificar sus propias preferencias en el desarrollo de software.

Bibliografía

Aguilar, R.A. (2008). Entrenamiento de Grupos: Una Estrategia Asistida por Entornos Virtuales Inteligentes. Tesis doctoral Universidad Polit&eacutecnica de Madrid.

Belbin , M. (2013). *Method, Reliability & Validity,A Comprehensive Review of Belbin Team Roles*. Retrieved 04 10, 2013, from Belbin Team Roles: http://www.belbin.com/content/page/5599/BELBIN(uk)-2013-A%20Comprehensive%20Review.pdf

Belbin, M. (2012). *BELVIN*. Recuperado el 30 de 01 de 2013, de BELVIN: http://www.belbin.com/rte.asp?id=8

Chromatic. (2003). Team-Based Software Development Extreme Programming. In Chromatic, *Extreme Programming* (p. 108). O'Reilly Media.

Computer Society of the IEEE. (1990, 09 28). IEEE Standar Glossary of Software Engineering Terminology. *Std 610.12-1990*. New York, New York, USA: IEEE Standars Board.

Dorfman, M. (1997). Requirements Engineering. *IEEE*, 7-22.

Humphrey, W. S. (2000). *The Personal Software Process (PSP)*. Pittsburgh: Carnegie Mellon Software Engineering Institute.

Humphrey, W. S., Chick, T. A., Nichols, W., & Pomeroy-Huff, M. (2010). *Team Software Process (TSP) Body of Knowledge (BOK)*. Massachusetts: Carnegie Mellon University.

IEEE Computer Society. (1997). *IEEE Standar for Developing Software Life Cycle Processes IEEE std 1074-1997*. New York USA: John W. Horch.

Lindgren, B., & Rolf, M. (1997). *Rants from Rolf Marvin Bøe Lindgren*. Retrieved 01 30, 2013, from R. Meredith Belbin's Team Roles Viewed From the Perspective of The Big 5: http://blog.grendel.no/wp-content/uploads/2002/02/hovedoppgave.pdf

Montilva, J., Barrios, J., & Rivero, M. (2004). *Requisitos de capacitación y perfiles para ingenieros de software en micro y pequeñas empresas*. ecuperado el 10 de 04 de 2013, de CEISOFT: http://www.ceisoft.org/DB/Methodius/EDOCS/SRed/2009/07/T022000000150-0-Requisitos_y_perfiles_IS_para_MyPEs.pdf

ScrumAlliance. (2008). ScrumAlliance transforming the world of work. Retrieved 04 09, 2013, from http://www.scrumalliance.org/pages/scrum_101

Anexo

Cuestionario final

Nombre de cada uno de los miembros del equipo:_____
Nombre de la práctica que podría ser hash o multilistas:_____
Sección o grupo:_____

1. ¿Crees que realmente se trabajó en equipo?_____
2. ¿Cuál crees que fue la mayor falla en el equipo?_____
3. De las tres actividades ¿crees que se puede omitir alguna en el desarrollo de software? ¿cuál o cuáles? Y ¿por qué? _____
4. Dividir el 100% de participación entre las 3 actividades realizadas, de acuerdo al esquema siguiente:

	Análisis	Diseño	Codificación
5. % De tu participación:			
6. % De participación de compañero 1:			
7. % De participación de compañero 2:			

Notas biográficas:

Elsa Estrada Guzmán recibió el grado de Maestría en Sistemas de Información en la Universidad de Guadalajara en el 2007. Actualmente es profesora en esa Universidad en el Departamento de Ciencias Computacionales del Centro Universitario de Ciencias Exactas e Ingenierías. Sus intereses son en ingeniería de software y estructuras de archivos.

Adriana Peña Pérez Negrón recibió el grado de Doctora en Informática de la Universidad Politécnica de Madrid haciendo una estancia de investigación en la Universidad de Salford del Reino Unido para obtener la mención de “Doctor Europeo”. Su interés es en la interacción del usuario y el desarrollo de agentes en los entornos virtuales colaborativos.