Agile Architecture in Action (AGATA)[1]

Arquitectura Agile n acción (AGATA)[2]

Luis Freddy Muñoz Sanabria [3]
Julio Ariel Hurtado Alegría[4]
Francisco Javier Álvarez Rodríguez[5]

Abstract

Introduction: This work proposes Agile Architecture in Action (AGATA), a software process framework that scales Agile methods to larger teams. Methodology following a human interface model, several Extreme Programming (XP) development teams work together around a central team that takes advantage of the ability of architectural methods to define the solution at the architectural level, improving communication and maintaining Agile parameters. Results: AGATA was applied in a development project, involving software engineers and final year software engineering students, organized as a postgraduate practical course. In this case study we measured communication based in the architecture and face-to-face channels, taking into account the degree of distortion and quality of the channels. The main results show that communication levels in the whole team are reasonable and that the channels proposed by AGATA maintain Agile parameters as to intergroup relationship and client deliveries. There are reports indicating scaling problems as teams grow; particularly, communication worsens. Conclusion: It is necessary to propose clear channels of communication. AGATA practices managed to maintain Agile elements with a large team.

Keywords*:* software architecture, scaling, Agile method, software process.

Resumen

Introducción: Este trabajo propone Agile Architecture in Action (AGATA), un marco de proceso de software que escala métodos Agiles a equipos más grandes. Metodología: Siguiendo un modelo de interfaz humano, varios equipos de desarrollo de Extreme Programming (XP) trabajan juntos alrededor de un equipo central que aprovecha la capacidad de métodos arquitectónicos para definir la solución a nivel arquitectónico, mejorando la comunicación y manteniendo los parámetros ágiles. Resultados: AGATA se aplicó en un proyecto de desarrollo, en el que participaron ingenieros de software y estudiantes de último año en ingeniería de software, organizados como un curso práctico de posgrado. En este caso de estudio se midieron los canales de comunicación la arquitectura y el cara a cara, teniendo en cuenta el grado de distorsión y la calidad de los canales. Los principales resultados muestran que los niveles de comunicación en todo el equipo son razonables y que los canales propuestos por AGATA mantienen parámetros Agiles en cuanto a relaciones intergrupales y entregas de clientes. Hay informes que indican problemas de escala a medida que los equipos crecen; en particular, la comunicación. Conclusión, es necesario proponer canales claros de comunicación; las prácticas de AGATA lograron mantener los elementos Agiles con un equipo grande

Palabras clave: Arquitectura de Software, escalar, métodos ágiles, procesos software.

---

[1] Submitted on: September 11th,2016. Accepted on: January 25th, 2017 This article is derived from AGile/ArchiTecture in Action (ÁGATA): A Holistic and Managed Process for Architecture-Centered Agile Software Development for large Size Teams investigation project.
[2] Fecha de recepción: 11 de septiembre de 2016. Fecha de aceptación: 25 de enero de 2017. Este artículo se deriva de un proyecto de investigación denominado AGile/ArchiTecture in Action (ÁGATA): Un Proceso Holístico y Gestionado para el Desarrollo de Software Ágil con Arquitectura Orientado a Equipos de tamaño mediano.. Desarrollado por el grupo de investigación LOGICIEL -IDIS de la Universidad Fundación Universitaria de Popayán – Universidad del Cauca, Popayán, Cauca, Colombia.
[3] Ingeniero de Sistemas Universidad Antonio Nariño Magister en Computación Universidad del Cauca, Doctorado en Ciencias de la Electrónica Universidad del Cauca, Docente Investigador Fundación Universitaria de Popayán, Popayán, Colombia. lfreddyms@fup.edu.co.
[4] Ingeniero en electrónica Universidad del Cauca, Magister en Computación Universidad de Chile, Doctorado en Ciencias de la computación Universidad del Chile, Docente Universidad del Cauca, Popayán Colombia, ahurtado@unicauca.edu.co
[5] Licenciado en Informática Universidad Autónoma de Aguascalientes, Magister en administración Universidad Autónoma de Aguascalientes, Doctorado en Ingeniería Universidad Nacional de México, Docente Investigador Universidad Autónoma de Aguascalientes, Aguascalientes México, fjalvar@correo.uaa.mx.

I. Introduction

Organizations today require the automation of their processes due to the growing amounts of information they handle, the need to be competitive, and the desire for reliable and trustworthy results. The software industry has a responsibility to meet these expectations. It therefore seeks to rely on methodologies that meet the criteria, at the speed of the internet. Agile methodologies have gone some way to meeting these requirements [1]: as well as responding to rapid developments in environments of considerable uncertainty, they include basic quality practices. The biggest quality problems come from the specification of requirements [2, 3], which is addressed in the Agile context with short development cycles aimed at generating value and with direct participation of the client.

Most of the scientific reports on Agile methods show the methods are effective in small teams working on small, non-critical, totally new projects for the same organization, with stable architectures and simple working rules [4]; whereas in projects with other characteristics, problems arise. Among these problems is communication, as when teams increase in size, the complexity of communication among their members increases dramatically [5]. This can become a real problem, since the effective communication of a software development team is a critical factor in the success of a software project [6].

Hence, group size is an important consideration when making decisions about the structure of the teams and the eventual partition of projects into smaller sub-projects. This partition is a key practice with direct implications in the decision to distribute project teams [7]. The architecture becomes a communication channel, as an additional support at the technical level for each sub-project, and it aids team management in a large software project attempting to employ Agile methodologies [8].

The absence of an orientation toward management within Agile methodologies does not allow emphasizing early decisions that will have a profound impact on all software engineering work. Neither is a model built that, although relatively small and intellectually understandable, would make it possible to verify how the system is structured and how its components work together. The metaphor of the system [3] is an initial approximation of the architecture and the management that is useful for simple solutions and this can serve as a starting point for a description of a model that will enable scalability of the Agile methods.

One of the virtues of Agile methodologies is that they generate value quickly, for both client and development team. To do this, the methodologies establish rules such as the prioritization of requirements, with the idea of generating early delivery of functioning software. This competitiveness hardly ever benefits organizations at the development stage and in maintenance of the application. The idea is that the software industry strengthens at the same time it produces, i.e. the industry reuses its components, elements, structures and design decisions, as well as obtains products that can be easily maintained [9].

Therefore, for a market that demands quick solutions because their processes require it, absence of design in Agile methods can yield products that are scarcely competitive, and inflexible. This article presents a process framework for scaling XP and Scrum called *Agile Architecture in Action (AGATA)*, centered on team management, architecture and communication, oriented to large teams (2 to 9 sub-teams) [10]. The framework seeks to improve and enhance the productivity of development teams in the software industry.

AGATA introduces a holistic model for improving the coordination of a set of small development teams, working independently with XP and Scrum, synchronized by a team of Agile architecture. We validated the proposed model in a study involving the model's implementation in a software development project.

The remainder of this paper is organized as follows: Section II presents the main work related to scalability of agile methods; Section III presents the AGATA process, which involves specification of values, equipment, practices, and processes; Section IV presents the model's application in the case study along with analysis of the results. Finally, Section V presents conclusions, limitations, and future work.

II. Related work

Yang et al. [11] in their study analyzed the combination between architecture, team management, and Agile methods in the stages of exploration and analysis. They proposed the application of architectural designs in different Agile practices, taking account of costs, benefits, challenges, factors, tools, and lessons learned. The result of the study was that the application of architectural design is required, that makes it possible to establish communication criteria, quality, and maintainability even when the project abandons Agile principles.

A method based on the use of reference patterns and implementation of architecture in XP, called C3A and developed by Hadar and Silberman [12], presents a set of contracts for components and a methodology that aligns the timing and granularity of the tasks. This method, however, does not explicitly define practices regarding the development of requirements and architecture design. Neither does it report cases of application.

Erder and Pureur [13] described how to adopt an architectural approach for the overall process, sometimes called "DevOps", and how to use it for team management and communication, based on five components: feedback and continuous monitoring, continuous integration, continuous release and deployment, continuous testing, and hybrid cloud. The aim of continuous delivery is to respond quickly to business needs by delivering high quality software in rapid cycles. The method they propose does not present clearly the elimination of bottlenecks observed in the above steps; for this, they recommend systematic application in a disciplined architectural perspective. However, as teams grow in number they do not use these perspectives to define a communication model according to quality requirements, due to the size of the application.

Kazman, Bass and Klein [14] describe a new method for improving team management through architectural models within Agile methods, called APTIA (Analytical Principles and Tools for the Improvement of Architectures), which is used in the life cycle as a means for understanding the objectives of the business. Mapping the requirements leads to an architectural representation and evaluation of the risks associated with this assignment. The method proposed by Kazman et al developed into a series of techniques and shared components for both Agile methods and traditional methodologies that seek to clarify the requirements of the customer. There are no results proving the effectiveness and difference of APTIA when used in Agile and/or traditional methods.

Zaychik and Regli [15] state that within the life cycle of the project, the initial stages for building the product are quite problematic from the perspective of communication. Hence, the teams have to

adopt cooperative work tools supported by computer to facilitate the processing of information, which in most cases has not been successful due to its complexity.

Marjaie and Rathod [16] introduce a method based on team communication with the client, based on inclusive and customized practices. They specify needs about qualitative aspects of the dynamics of systems, integrating existing methodologies to facilitate the iterative modeling process.

Reinhardt [17] states how some studies highlighted the fact that ad hoc informal communication in Agile methods is significant in group interaction; however, it is not possible to achieve any positive effect when the team grows.

III. Agile Architecture in Action (AGATA)

AGATA is a brand of software process based on the values and principles of Scrum process management. It adds the architecture practices of XP/Architecture (XA) [18] and Extreme Programming (XP) principles [19], and establishes channels and clear rules of communication with the aim of enabling the use of the Agile approach in projects with large teams (2 to 9 sub-teams) [10] where Agile methodologies have previously proven difficult to scale [20].

AGATA therefore becomes the stimulus motivating the fundamental organization of the development team, embodied in its components, in the relations between team members, in the working environment and in the principles orienting the design and evolution of the software system.

In AGATA, communication becomes a fundamental element for team synchronization, proposing an architecture team that will motivate sub-team members, controlling and proposing effective channels so that information relevant to the project flows automatically. AGATA is a holistic, managed process brand that seeks to continue obtaining results from Agile methods despite teams growing larger (2 to 9 sub-teams). Figure 1 illustrates the model.

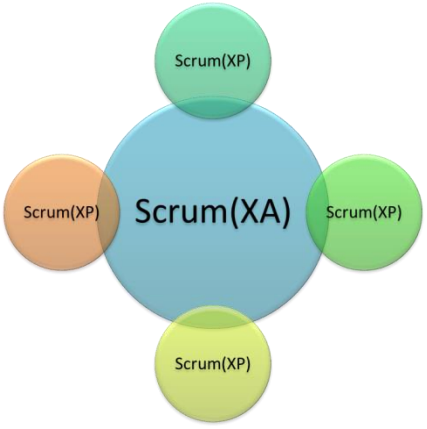

Figure 1. AGATA holistic model. (author's own elaboration)

3.1 AGATA elements

From the management perspective, AGATA proposes intergroup communication, architecture-centered development; the organization of tasks to be performed by the development team to achieve

the project objective in the time proposed by Agile methodologies. It is based on the following structure:

3.1.1 Values

AGATA is characterized by its intrinsic values that are reflected in each iteration, the events that take place within and the results obtained in these iterations:

Adaptiveness: It has the ability to change and learn from experience [21].
Empirical control: Control based on inspection and continuous adjustment depending on the results obtained after each iteration, and on the project context.
Transparency: In AGATA, it is important that the most relevant occurrences during the process are visible for all project managers.
Communication: This is crucial at all stages of the process and a fundamental proposition of AGATA. As such, it proposes two channels that will be responsible for directing the team for the success of the project.
Team management: The AGATA team self-organizes, depending on the interests and skills of each of its members, in sub-teams that revolve around the proposed model.

3.1.2 Principles

As Scrum [22], AGATA is based on practices focused on the results expected by the client:

- Take advantage of incremental development characteristics.
- Prioritize requirements based on value for the client.
- Take into account independent development variables of the product [23].
- Maintain empirical control of the project.
- Synchronizes and make daily adjustments to the team.
- Communication flowing around the project will become a vital element. This will generate in both client and development team a dynamic that will strengthen decision-making to obtain the desired results.

3.1.3 Events

AGATA proposes a series of events in order to strengthen and improve communication between teams and energize each of the AGATA actors:

Sprint planning meeting: This is the first meeting the customer has with AGATA for sharing of concerns about the system and the conditions for carrying out the project.
Product backlog organization: This is the visible means of monitoring the progress of the project. The client and teams discuss their tasks and prioritize their activities, as well as measure their results.
Iteration planning: AGATA prepares a list of tasks for iteration and estimates development effort. These tasks are the requirements the team has committed itself to carrying out.
Daily Sprint meeting: Also called daily synchronization meeting, lasting a maximum of 15 minutes, this is where the Agile team evaluate the progress of the tasks.
Sprint review: At the end of each Sprint, AGATA leaders hold a very informal meeting with the client to present the requirements completed.

Sprint retrospective: A meeting held by all AGATA actors once the Sprint is finished to analyze what went right, what processes could be improved, and how to improve them.

3.1.4 Roles

The management of a project in AGATA focuses on defining what are the characteristics the product must have (what to build, what not to build, and in what order) and overcoming any obstacles that might hinder the work of the development team. The AGATA actors are:

Scrum (XA) team: Also known as architecture team, coordinates the project with the AGATA actors, organizes the product backlog, reviews the Sprint results, manages the project through the architecture as a communication channel.
Scrum (XP) team: Also known as development team, follows Extreme Programming practices while keeping Scrum management in mind.
AGATA master: Maintains team synergy, while ensuring adherence to AGATA principles and values.
Product owner: The representative of the client in AGATA, serves to focus team vision and is further responsible for the ROI of the project.

3.2 AGATA life cycle

AGATA follows its phases and iterations as shown in Figure 2.

Figure 2. Phases of the AGATA method. (author's own elaboration)

***Exploration*:** In AGATA, Scrum (XA) team members meet with the client (XA) to set out the initial requirements of the system, establishing system limits.
***Planning*:** This is the initial stage of all projects. It is ongoing in nature, beginning a continuous relationship between client and development team to find the system's requirements. Here, the AGATA team plan the number and size of the project iterations and make adjustments to the practices of the Scrum (XP) teams based on the characteristics of the product [24].
In planning, the AGATA team should take into account the following aspects :
***History of architecture*:** Comprising the results of the conversation between client (XA) and Scrum (XA) team, it will be the best means of communication for managing the Scrum (XP) teams.
***User stories*:** The Scrum (XA) leader is the one who decides what to do, based on the meeting of Scrum team (XA) and client. As a first step, the leader provides a clear idea of what the project will be [25]. User stories serve as a tool to let the scrum (XP) development team know the requirements of the system. The stories are small texts in which an activity to be carried out by the system is described; writing these is done with the client in mind, not the developer, so that the terminology is clear and simple, without going into detail. The user stories allow the estimation of delivery time.
***Speed of project*:** This is a measure of the ability of the development team to evacuate user stories in a given iteration. The AGATA team calculates this by totaling the number of user stories resolved in a single iteration.

Daily meeting or "stand-up meeting": Both the Scrum (XA) team and Scrum (XP) teams require a continuous review of the work plan. Therefore, the AGATA team schedule very short daily meetings to discuss the point where everyone got to the previous day, what the problems were, and what work the teams are planning for the day ahead [26].
***Delivery plan*:** At the start of every iteration, the Scrum (XA) team and the client meet. At that meeting, they define the time frame for implementing the system. The client sets out his requirements to the group members and they verify the tasks that are broken down from each architectural story and estimate the degree of difficulty of implementing each one. Tools help to show which jobs have still to be done, which ones are in process, and which ones have been done.
Building with architecture: Based on the previous phase, the AGATA team prioritizes the requirements of the system, and develops a draft of the architecture proposed by the Scrum (XA) architectural team for those requirements. The system is modularized and the requirements distributed to the Scrum (XP) development teams involved in the project. The AGATA team works on defining the architecture alongside the development of the Scrum (XP) teams. The system undergoes continual integration. As in Extreme Programming, this construction is iterative and incremental ensuring continued customer feedback.

3.3 AGATA architecture practices

AGATA maintains the management of Scrum, and XP the simplicity, the metaphor of the system, and the refactoring. Furthermore, AGATA applies adapted forms of the Attribute-Driven Design ADD [27] and Quality Attribute Workshops (QAW) [28] architecture methods. Previous studies have combined and compacted these practices with the base practices of XP [12, 29] and this research takes them as a reference point for obtaining the fragments of process concerning architectural themes as shown in the following:

***The planning game*:** A continuous practice, here the client comes to agreement with the AGATA team. From the beginning of product development, the group and the client must have an overall, clear picture of what tasks they want and which of these tasks they will do, i.e. they need to understand and agree with what the "other party" puts forward. During the project, a number of meetings take place in order to organize the tasks and new ideas arising from the customer and the team [30].
***In-situ client*:** The client must be available to the architecture team in order to resolve any questions or concerns that may arise in the course of the project. He himself represents the requirements in real life and validates whether or not the delivery is useful [31].
***Simple design*:** Complex designs have no place in this discipline, because they generally do not provide clear solutions to product development. This does not mean forgoing design: AGATA instead calls for a simple and flexible design understandable to all team members [32].
***System metaphors*:** With the aim of establishing a unified language in AGATA, the idea is to improve communication and make very simple stories of the requirements [9].
***Refactoring*:** Architecturally, facilitates the rapid formulation of solutions; while in parallel ADD, provides architectural tactics and strategies for a longer-term solution, firing architectural refactoring requirements. This makes it possible for the technical requirements to arise incrementally, backed by the capacity of the requirements negotiation and refactoring practices, which are architectural in nature in order to locate the potential changes of greatest impact as early as possible in the project.
***Collective ownership of code*:** All group members know and handle the code. This even means that the group has to apply programming standards.

***Code conventions*:** The idea is that all team members know the code and have access to it so they can make changes. For this reason, the aforementioned programming standards are applied.
***No overtime*:** This practice seeks to maximize the performance of the teams without resorting to punishing working hours in the planning of more than 40 hours a week. Instead, the AGATA team plans a "steady pace" in the development of each story [33].
***Test directed development*:** "When we know what we are going to test, then we will know what we are going to develop" [34]. Before making any unit of code, it is necessary to have the respective test unit. The programmers conduct tests directed toward the operation of the code. The client and test engineer are responsible for designing acceptance tests, the purpose of which is to verify that the user stories are implemented correctly [35].

IV. Case study

We applied AGATA in the framework of a project for the software industry, adhering to the guidelines of empirical research from case studies of a phenomenon studied within its real context [11].

4.1 Case study design

Based on the hypothesis that *AGATA as an integrated mechanism for Agile project management could improve the communication challenges for large teams (of 2 to 9 sub-teams),* this case study focuses on evaluating the ability of AGATA to establish communication mechanisms that enable scaling to large groups within a development project framework. We followed the proposal of Runenson and Host [22] to design this research, the starting point being the main research questions to be solved: how to scale agile methods in software projects with large teams (of 2 to 9 sub-teams) and which practices could be introduced to overcome the communication challenges.

To support this research, we defined indicators, metrics and data collection instruments as presented in Table 1. For the collection of information in this case study we used and designed the following instruments:

Surveys: We surveyed the client, Scrum (XP) teams, and Scrum (XA) team in order to

- investigate the degree of distortion of the requirements of the system, as well as thesatisfaction of each actor with the method and product;
- understand the degree of acceptance of the proposed model among team members and the degree of responsiveness of the model and its dynamism relating to the project and the other actors.

Artifacts: Checklists for the team and the product to evaluate coverage of the requirements, the defects plan to assess quality, and evaluation of the proposed architecture.

Table 1. Indicators, metrics, information sources, and instruments defined for research support. (author's own elaboration)

Indicator	Measurements	Information sources	Instruments
Distortion (D)	Measures degree of distortion of message	Client and teams	Survey
Effectiveness of Channel (EC)	Functionality: ease of proposed channels	Client, teams and product	Observation Checklist

4.2 Project

When development teams grow, some variables are difficult to control and therefore may adversely affect the product and the client because of possible lack of quality in the results. Accordingly, we validated AGATA in an industrial environment with a team that surpassed the possibilities of Agile methods, to develop software with all the requirements and commercial complexity. We based the object of study on the development of a Kanban board in order to allow a better production process workflow in the development of software. To this end, we observed the following principles: Visualize workflow, Limit work in progress, Manage and measure workflow, Implement feedback cycles, Clarify policies and procedures, Continuous collaborative evolution.

4.3 Project development

We delegated a team of 24 participants to develop the application, which would allow automated tracking of Scrum (XP) and Scrum (XA) teamwork. The 24 project participants organized autonomously for the development of the proposed project.

The 24 participants were unaware of the AGATA methodology. We gave them training over an extended 8-hour day. On the same day, we also addressed issues relating to Scrum and Extreme Programming (XP), in order to unify terms of reference. Finally, we also trained them in architecture topics and methods, since the group required to know basic concepts, views, styles, and methods proposed by SEI (Software Engineering Institute), placing an emphasis on ADD (Attribute-Driven Design) and QAW (Quality Attribute Workshop). Image 1 illustrates.

Case context: the developers organized themselves into four Scrum (XP) teams of five members and a Scrum (XA) team of four members. Each team chose their client/architect representative to Scrum (XA).

Image 1. AGATA members in training. (author's own elaboration)

In the dynamic of the project, while Scrum (XA) met with the client of the project, the Scrum (XP) teams organized their information displays (whiteboards) in a place visible to all (see Image 2). At the beginning of every one of the six 8-hour sessions, the Scrum (XA) team held a 15-minute meeting with the client to receive feedback.

Image 2. Scrum (XA) team members. (author's own elaboration)

The client showed very high commitment, was available, and actively participated in the development of requirements. In addition, the Scrum (XA) and Scrum (XP) teams continued iteration planning and daily work practices as established by AGATA. In the course of the project, the AGATA team collected information as planned. The dynamic corresponds to the proposed model, since during the course of the case study there was commitment by each of the team members to follow the practices proposed by the model, achieving an adherence of 82%, which is acceptable for evaluating the results of the case study.

4.4 Results

The productivity of the Scrum (XP) teams was between 0.010 and 0.017 US-P-H (User Stories per Person per Hour), while for the Scrum (XA) team it was 0.007 HA-P-H (Histories of Architecture per Person per Hour). In this case study, the AGATA team developed 12600 LOC, reaching a productivity of 14.58. In the project, we evaluated other important variables. Results reported 95% level of acceptance of the proposed model and 90% level of client participation in the course of development of the project. Once the product had been delivered, customer satisfaction was assessed at 95%. With 95% meeting of the requirements, the observed level of adherence to the method by teams and client was 95%. These results provide preliminary validation of the productivity of AGATA. However, further empirical evaluation should be carried out by the software industry.

Meanwhile, we measured distortion from the context or semantics of the messages transmitted by the teams, since this is one of the main barriers to communication. We therefore took into account conceptualization depending on the message or the status of each project actor, degree of significance of the messages understood, information expressed poorly, incongruities, encoding of messages. To measure distortion, we took into account the following variables: number of defects found, number

of messages transmitted, messages and defects in functional requirements, messages and information on restrictions, and the messages and defects in architectonic information (Drivers, Tactics and Strategies). We applied the following equation:

D = {(NDRF/NMERF),(NDR/NMER), (NDA/NMEA)}

Where ND is the number of defects found. NME is the number of messages transmitted. RF refers to messages and defects with functional requirements. R are messages and defects with constraint information. A refers to messages and defects with Architectonic information (Drivers, Tactics and Strategies)

We found distortion was 0.01%, giving a reasonably high degree of reliability for the project (see Figure 3).

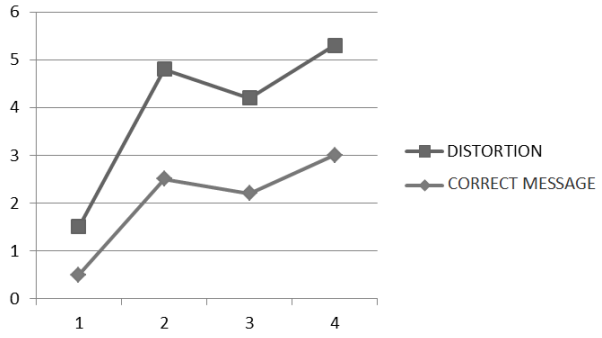

Figure 3. Distortion. (author's own elaboration)

With these results, it was necessary to measure the quality of the channels used - in this case the architecture proposed by Scrum (XA) and the face-to-face communication held in each of the events proposed by AGATA. For this, we took into account the following equation:

MC = (1 - Messages Incorrectly Transmitted Due to Channel/Total Transmitted Messages) * 100,

resulting in a degree of acceptance and efficiency of 95%, intimating that the channels used achieved a high degree of confidence on sharing information among the AGATA actors.

V. Conclusions, limitations and future work

This article has presented AGATA, a development method that extends the Agile practices of XP and the management of Scrum for large teams (2 to 9 sub-teams).

- In a preliminary way, the method scales when applied in an industrial case. The main qualitative findings of this case study include:
    - Due to size of the team, thought should be given to management of its members; a method should be sought that holistically coordinates the activities to be carried out by each team member.

- As the team grows, communication is a factor that must be controlled; the bigger the team, the greater the information load and the greater the number of communication messages.
- The AGATA members understand their role in achieving the integrity of the holistic model, to generate the histories architecture as a communication mechanism for establishing the requirements.
- It is necessary to propose a preliminary meeting between the architectural team and the client, or iteration 0. This will ensure that when the sub-teams of the project are integrated, they already have clear tasks to carry out.

In the literature, most cases do not report the measurements made, or report without characterizing the context of the case studies, and very rarely is communication taken into account as a relevant, measurable element in development teams. That given, this research suffers from certain limitations due to the type of comparisons of the measures performed and those found in the literature. We therefore recommend as future work to conduct controlled experiments to be able to establish a more normalized comparison. Team management and communication are two aspects that need to be taken into account to achieve scaling.

The case study presented here establishes that to corroborate the results obtained in this research it is necessary to consider other variables of context such as those established by Layman et al. [36] to make a more normalized comparison - for example, the expertise of the participants. The software industry is currently adopting all these considerations to improve the design of future case studies.

Acknowledgments
The authors would like to thank the IDIS Research groups of the Universidad del Cauca and LOGICIEL of the Fundación Universitaria de Popayán, which made possible the development of this study.

VI. References

[1] BECK Kent and CYNTHIA Andres. Extreme Programming explained. Edition 1, *Pearson Education*, 2004. ISBN: 978-0321278654.
[2] NORD, Robert and TOMAYKO, James. Software Architecture Centric Methods and Agile Development. *IEEE Software*, 2006, Volumen 23, Number 2, Pages 47- 53.
[3] DONALD, Reifer and HAKAN Erdogmus, Scaling Agile Methods, *Diario IEEE Software archive*, 2003,Volumen 20, Number 4, Pages 12-14
[4] BECK Kent and FOWLER Martin. 2000. Planning Extreme Programming. Edition 1, *Addison-Wesley*, Longman Publishing Co., Inc., Boston, MA, USA.
[5] AMBLER, Scott. The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments, *IBM Corporation*, 2009. AvalaibleFrom: https://www.researchgate.net/profile/Scott_Ambler/publication/268424579_Adapting_Agile_Methods_for_Complex_The_Agile_Scaling_Model_ASM_Adapting_Agile_Methods_for_Complex_Environments/links/55003e780cf28e4ac347ee34.pdf?origin=publication_detail.
[6] MACKELLAR, Bonnie. A Case Study of Group Communication Patterns in a Large Project Software Engineering Course. *Software Engineering Education and Training (CSEE&T),* 2012. ISBN: 978-1-4673-1592-0.
[7] RODRÍGUEZ, Daniel, SICILIA Miguel Angel, GARCÍA, Elena and HARRISON, Rachel. Empirical findings on team size and productivity in software development. *The Journal of Systems and Software*, 2012, Volumen 85,Number 3, Pages 562–570.

[8] Extreme Programming: A gentle introduction, Date of consultation: 8 October 2013. Available from: http://www.extremeprogramming.org/.
[9] Beck Kent and Andres Charles. Extreme programming explained: embrace change. *Addison-Wesley*, pp. 224, Enero 2012.
[10] PENDHARKAR, Parag and RODGER, James. The Relationship Between Software Development Team Size and Software Development Cost. *Communications of the ACM - Rural development engineering MCCA*, 2009, Volumen 52, Number 1, Pages 141-144.
[11] CHEN, Yang and PENG, Liang, A systematic mapping study on the combination of software architecture and agile development. *Journal of Systems and Software*, 2016, Volumen 111, Pages 157–184.
[12] HADAR, Ethan and SILBERMAN, Gabriel, Agile architecture Methodology: Long Term Strategy Interleaved with Short Term Practics. *OOPSLA Companion '08 Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications* , 2008, Pages 641-652. ISBN: 978-1-60558-220-7.
[13] ERDER Murat and PUREUR Pierre. Continuous Architecture: Sustainable Architecture in a Agile and Cloud- Centric World, *Elsevier*, 2006, ISBN: 978-0-12-803284-8.
[14] KAZMAN Rick, BASS Lean, KLEIN Marcos. The essential components of software architecture design and analysis, *Elsevier*, 2006, Volumen 79, Number 8, Pages 1207-1216.
[15] ZAYCHIK Vera and REGLI William. Capturing communication and context in the software project lifecycle. *Journal Research in Engineering*, 2003, Volume 14, Number 2, Pages 75-88.
[16] MARJAIE Ali, RATHOD Urvashi, Communication in Agile Software Projects: Qualitative Analysis using Grounded Theory in System Dynamics, *Symbiosis Centre for Information Technology (SCIT)*, 2014.
Available from: http://www.systemdynamics.org/conferences/2011/proceed/papers/P1353.pdf.
[17] REINHARDT Wolfgang. Communication is the key – Support Durable Knowledge Sharing in Software Engineering by Microblogging. *Proceedings of the 1st International Workshop on Software Engineering within Social software Environments (SENSE09)*, 2009.
[18] MUÑOZ SANABRIA Luis and HURTADO ALEGRIA Julio, *XA: Una extensión XP para apoyar estudios de arquitectura*. IEEExplore. XI Computer Congress (CCC). Colombia, 2015.
[19] MARCHESI Michelle, SUCCI Giancarlo, WELLS Don, and WILLIAMS Laurie, Extreme programming perspectives. *Addison Wesley*, pp. 624, Agosto 2002.
[20] PENDHARKAR Parag, and RODGER James, *The relationship Between Software Development Team Size and Software Development Cost.* ACM, vol 52, 1, pp. 141-144, Enero 2009.
[21] SCHWABER, Ken and SUTHERLAND, Jeff, The scrum Guide: The definitive guide to scrum: the rules of the game. *Scrum.Org and ScrumInc*, 2013. Available from: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf.
[22] RUNESON, Per and HOST Martin, Guidelines for Conducting and ReportingCase Study Research in Software engineering, *Empirical Software Engineering*, 2009, Volumen 14, Number 2, Pages 131–164.
[23] ROLF NJOR Jensen, MALLER Thomas and TJORNEHOJ Gitte, Architecture and Design in extreme Programming: Introducing Developer Stories, *Lecture Notes in Computer Science*, 2006, Volumen 4044, Pages 133-142.
[24] KUPPUSWAMI, S, Vivekanandan, k, RAMASWAMY, Prakash y RODRIGUES, Paul, The Effects of Individual XP Practices on Software Development Effort, *Newsletter ACM SIGSOFT Software Engineering Notes*, 2003, Volumen 28, Numero 6, Pages 6-6.
[25] ECHEVERRY TOBÓN Luis and CARMONA DELGADO Elena, *Caso práctico de la metodología ágil XP al desarrollo de software*, tesis pregrado inédita, Universidad tecnológica de Pereira, Colombia, 2007.

[26] MARTIN, Angela, NOBLE, James, BIDDLE, Robert, Programmers are from Mars, Customers are from Venus: A practical guide for customers on XP Projects, *PLoP '06 Proceedings of the 2006 conference on Pattern languages of programs*, 2006, *Number 20*. ISBN: 978-1-60558-372-3.
[27] WOJCIK, Rob, BACHMANN, Felix, BASS, Len and Others, Attribute-Driven Design (ADD) *Version 2*. *Software Architecture Technology Initiative*, 2006. Available from: http://www.sei.cmu.edu/reports/06tr023.pdf.
[28] BARBACCI, Mario, ELLISON, Robert and Others, Quality Attribute Workshops QAW -Third Edition, *Architecture Tradeoff Analysis Initiative*, 2002. Available from: https://pdfs.semanticscholar.org/da24/7e22910e7e4c98071e90fcdc419245ea45bd.pdf.
[29] ABRAHAMSSON, Pekka, SALO, Outi, RONKAINE, Jussi, and WARSTA Juhani. Agile Software Development Methods: Review and Analysis, *VTT Electronics*, 2002, Pages 107. Available from: http://www.pss-europe.com/P478.pdf.
[30] REIFER Donald, MAURER Frank, ERDOGMUS Hakan. Scaling agile methods. *IEEE,* vol 20, 4, pp. 12-14, Jul 2003.
[31] SILLITTI Alberto. and SUCCI Giancarlo. Requirements engineering for agile methods, *Springer*, pp. 478, 2005. Available from: www.springer.com/987-3-540-25043-2.
[32] GITTINS, Robert, HOPE, Sian, A study of Human Solutions in Extreme Programming. University of Wales Bangor, 2001, Pages 41-51. Available from: https://pdfs.semanticscholar.org/6117/7dcb504e5eab9e9e3f5c2e1335f115c5b208.pdf.
[33] BECK Kent, BEEDLE Mike, BENNEKUM Arie and Others, Manifesto for Agile Software Development, 2001. Available from: https://www.agilealliance.org/.
[34] BIOLCHINI, Jorge, GOMES, Paula, CRUZ Ana and TRAVASSOS Gilherme, Systematic Review in Software Engineering, *system engineering and computer science department*, 2005. Available from: ftp://161.24.19.221/ele/ivo/Leitura/biolchini_2005.pdf.
[35] KITCHENHAM Barbara, Procedures for Performing Systematic Reviews, *Software Engineering Group Department of Computer Science Keele University*, 2004. Available from: http://www.inf.ufsc.br/~aldo.vw/kitchenham.pdf.
[36] LAYMAN, Lucas, WILLIAMS, Laurie, and CUNNINGHAM Lynn. Exploring ExtremeProgramming in Context: An Industrial Case Study, *Proceedingsof the Agile Development Conference*, 2004, Pages 32 – 41. Available from: https://collaboration.csc.ncsu.edu/laurie/Papers/ADC.pdf.