Comparing Agile Processes for Agent Oriented Software Engineering

Alma M. Gómez-Rodríguez and Juan C. González-Moreno

D. de Informática (University of Vigo)
Ed. Politécnico, Campus As Lagoas,
Ourense E-32004, Spain,
{alma,jcmoreno}@uvigo.es
http://gwai.ei.uvigo.es/

Abstract. Multi-agent Systems are at the moment an important new paradigm in software development. Several methodologies have been proposed for developing systems within this approach. Besides new agile process have been proposed to be used combined with the meta-models of such methodologies. This paper studies how the use of one of those Agent Oriented methodologies following an agile process such as Scrum produces improvements in the time consumed in the development that could shorten the learning time. This may have as outcome the possibility of using smaller groups in development.

1 Introduction

Agents represent a powerful abstraction tool in software development. The inherent characteristics of agents: autonomy, reactivity, proactivity, etc. provide a very good approach in the solution of distributed complex problems [1]. Therefore, Agent-Oriented Computing has become in the last decade a new Software Engineering paradigm [2]. The interest in software development with agents is focused in multi-agent systems (MAS) [3], [4], [5] that is, a set of autonomous agents which work cooperatively using high level communication languages and protocols.

There are many applications implemented using agents, nevertheless agent tools, methodologies and process have not yet a sufficient level of maturity for being used under warranty in commercial software development [2]. Two issues are essential if the agents are to be used in software industry: the availability of tools or frameworks which simplify multi-agent systems implementation and the use of suitable methodologies and development processes which guide the engineer during the system construction. At the moment much work is being carried out in all these fields. Many agent oriented methodologies have been proposed and applied to MAS development with good results [6], [7], [8], [9], [10], [11], [12], [13]. Some of these methodologies introduce a tool supporting the development, such as IDK [14], Metameth [15], etc.

This work focusses on the importance of processes in software development. So, we consider a software development process as a simple dependency graph with three basic components: the process participants (*roles or workers*), the consumed and generated products (*work products*) and the *activities* and *tasks* achieved during the process, which constitute particular instances (*work definitions*) of the works that must be done. Methodology, in contrast, defines the models to construct and the concepts and notation used in these models.

Among all the methodologies for Agent Oriented development, we have chosen INGENIAS for the case study proposed. INGENIAS methodology covers analysis and design of MAS, and it is intended for general use, with no restrictions on application domain [16], [6], [17]. It has two supporting tools: the INGENIAS Development Kit (IDK) and the INGENIAS Agent Framework (IAF) [17], [18]. The intended process of INGENIAS is Unified Development Process (UDP), but some previous works [19], [20] have shown that methodology and process can be considered independently. A previous work [19] has adapted INGENIAS to follow agile processes, in particular Scrum. The adaptation has been done by identifying common tasks in the different development processes and reordering them to construct a new process.

New processes for INGENIAS have been defined theoretically in previous works [19], [20]. So it is an important issue to When defining a process, it is very important to prove the applicability in practice of these definitions. We consider that a first step when approaching this verification is to apply the process to a particular development and consider its suitability. Following this idea, this paper focuses on the results obtained in the application of two different development processes to a MAS. The aim of this paper is to compare this two processes mainly in productivity. In this way, we try in this way to confirm the suitability of both of them.

The structure of the remaining of the paper follows. Section 2 introduces the methodology used in the development: INGENIAS while 3 details the process defined for the development: Scrum for INGENIAS. Section 4 explains the experiment done and show the results. Finally, Section 5 addresses the conclusions and future work.

2 INGENIAS Methodology

The original purpose of INGENIAS was the definition of a specific methodology for the development of Multi-agent Systems (MAS), by integrating results from research in the area of agent technology and from traditional Software Engineering Methodologies. Initially, the definition of the INGENIAS Methodology was based on the well-established Rational Unified Process (RUP) in order to define its lifecycle, and on the definition of a set of meta-models that describe the elements needed to specify and develop a MAS. These meta-models describe the system from five viewpoints: *agent*, *interactions*, *organization*, *environment* and *goals/tasks*.

The integration of the INGENIAS MAS specification language with software engineering practices is achieved by defining a set of activities that guide the analysis and design phases, with the statement of the results that have to be produced by each activity. As in the rest of modern methodologies, the key point in INGENIAS is its meta-model language. As stated before, INGENIAS introduces five kinds of meta-models in order to define a MAS. The entities of the meta-models could appear in different diagrams, but are unique regarding the global system specification.

– **Organization meta-model.** It defines the global organization of the system, where organization is the equivalent to MAS architecture. An organization has a structure and a functionality. The structure is similar to the one stated in AALAADIN framework [21], and is defined attending to how agents should be grouped. Functionality is determined by defining the goals of the organization and the workflows it should execute.
– **Environment meta-model.** The environment model is composed of environment diagrams. The environment is what surrounds the MAS and what originates mainly agent perception and action. As a developer, one of the first tasks is to identify system resources, applications, and agents. System resources are represented using TAEMS [22] notation. Applications are wrappers of whatever is not an agent or a resource, and could be understood in INGENIAS as equivalent to objects in Object Orientation. Using these elements, a developer should be able to define how the MAS interact with the systems surrounding.
– **Tasks/Goals meta-model.** It describes how the mental state of agents change over the time, what is the result of executing a task over the agent the mental state, how to achieve goals, and what happens when a goal cannot be achieved.
– **Agent meta-model.** It defines the primitives to describe a single agent. It can be used to define the capabilities of an agent or its mental state. The mental state is an aggregate of mental entities that satisfy certain conditions. The initial or intermediate mental state is expressed in terms of mental entities such as those of AOP [23] and BDI [24].
– **Interaction meta-model.** These kind of diagrams show how two or more agents interact. The interaction behavior is described using different languages, such as UML collaboration diagrams, GRASIA interaction diagrams, or AUML protocol diagrams. An interaction has a purpose that has to be shared or partially pursued by interaction participants. Usually, this purpose is related with some organizational goal.

Recently in [25] the FAML meta-model has been proposed. Potentially, FAML is a an interesting candidate for future standardization of engineering agent modeling languages. FAML is composed by two layers: *design-time* and *runtime*, and each layer has two scopes: *an agent-external* and *an agent-internal*. In [25], INGENIAS meta-model (see Fig. 1) was considered by authors as one of the five more extant current agent-oriented approaches.

Originally the development process proposed for INGENIAS was an adaptation of the Rational Unified Process according to the modification showed in Table 1. Afterwards, in [26], [27] an agile version of INGENIAS based on

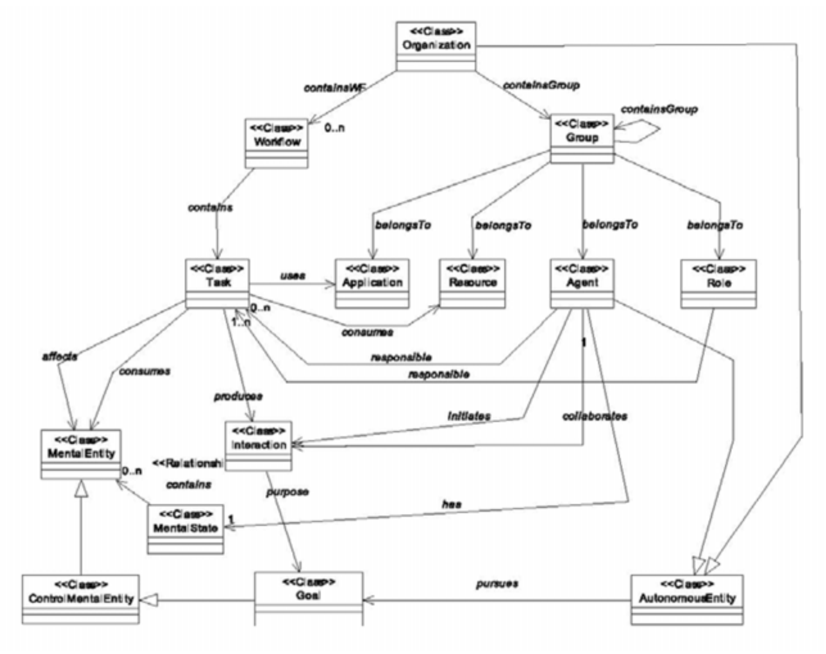

Fig. 1. INGENIAS Metamodel

OpenUp was presented. From the evidence that an agile process could be used with the INGENIAS metamodel language, a framework for deploying MAS over the JADE platform [28] appears in a natural way, this was the IAF [18], [29]. Besides, based on the use of this framework a modification of the Scrum process for INGENIAS was proposed in [19].

Section 3 introduces a more detailed explanation of how to use Scrum Process for INGENIAS.

3 The Scrum Process for INGENIAS

A Scrum is a mechanism in the sport of rugby for getting an out-of-play ball back into play. The term was adopted in 1987 to describe hyper-productive development. Ken Schwaber formalized the process in the first published paper on Scrum at OOPSLA 1995 [30]. As pointed in [31], Scrum is an empirical Agile project management framework which is used to iteratively deliver to the customer software increments of high value. Scrum relies on self organizing, empowered teams to deliver the product increments. It also relies on a customer, the Product Owner, to provide the development team with a list of desired features using business value as the mechanism for prioritization. Scrum is a model for management of the process of software development. It is not a methodology, because it does not propose models or concepts to address, but a framework

Table 1. Results to be obtained in each phase of the INGENIAS Process

	PHASES		
	INCEPTION	ELABORATION	CONSTRUCTION
ANALYSIS	To generate use cases and identify actions of these use cases with the corresponding Interaction Model To outline the system architecture with an Organization Model To generate Environment Models which reflects Requirement elicitation	To refine use cases To generate Agent Models that detail the elements of the system architecture To continue with the Organization Models, identifying workflows and tasks To obtain Task and Goal Models to highlight control constraints (main goals, goal decomposition) To refine the Environment Model including new elements	To study the remaining use cases
DESIGN	To generate a prototype using RAD tools such as ZEUS or Agent-Tool	To focus the Organization Model on workflow To refine Tasks and Goal Models reflecting the dependencies and needs identified in workflows and the relationships with system's goals To show how tasks are executed using Interaction Models To generate Agent Models which show required mental state patterns	To generate new Agent models or refining existing ones To study social relationships in order to refine the organization

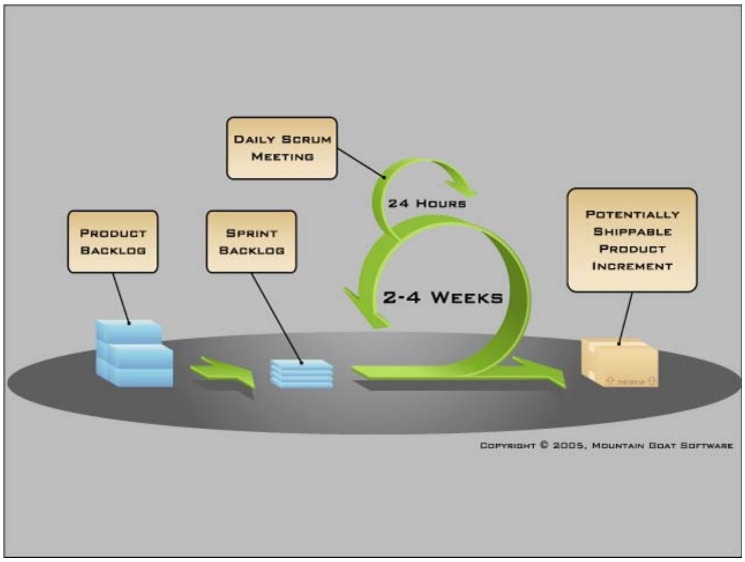
Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software

Fig. 2. Scrum lifecycle

where different methodologies can fit. The Scrum process is particularly suitable for Knowledge Engineering Developments based on the use of Multi-Agent Systems, because of the agile development and the user implication.

An initial view of Scrum process, as proposed by its authors in [31], can be seen in Fig. 2.

Previous works [27] have determined that when trying to map a well established methodology/process into a new process, it is necessary to define the steps to be done. In [27] several steps that must be followed in the definition of a new development process models for AOSE are defined, adopting SPEM [32] as model specification. These steps are:

1. *Identify the process model with an existent process model* if possible, if not define from zero the new one taking as basis the next steps.
2. *Define the lifecycle view.* Identify the phases, iterations and sequence of application. This step is essentially a temporal step in which other resources different from time are not considered.
3. *Define the disciplines.* Disciplines in SPEM determine process packages which take part in the process activities, related with a common *subject.* That is, disciplines represent a specialization of selected sub-activities, where these new sub-activities can not appear in other packages or disciplines. In this step, resources are the subject of the activities defined.
4. *Define the guidance and suggestion view.* The *Guidances* provide information about certain model elements. Each *Guidance* is associated to a *Guidance Kind.* This step is focused in exemplifying and documenting the activities previously defined.

The results of applying the previous steps to INGENIAS are shown in the next subsections.

3.1 Identify the Process Model

The methodology provides several pre-defined examples of development. These means that Multi-Agents Systems could be quickly constructed with INGENIAS by reusing previous developments. Recently, the INGENIAS Agent Framework (IAF) for JADE has been proposed and documented as a successful approach in this context [18].

3.2 Defining Lifecycle View

In Scrum, each release is produced within a number of iterations from 2 to 4 weeks called Sprints (see Fig. 2). Sprint goal is defined by the product owner, taking both priorities and team capabilities into consideration. At the end of each Sprint, the team produces a product increment which is potentially releasable. The evaluation of the product release drives to a backlog update before the next sprint starts. All the work is done in two basic phases: the *Preparation Phase* (before the first sprint) and the *Sprint Phases* (successive sprints leading to the release).

Although, Scrum does not describe engineering activities required for product development, INGENIAS-Scrum process must do it in order to adjust to IAF recommendations. IAF allows combining the classic approach of coding applications with modern techniques of automatic code generation.

IAF requires the use of the INGENIAS Development Kit (IDK), that contains a graphical editor for working with the specification model. Accordingly to IDK, the Scrum definition for the *Preparation Phase* comprises the tasks: *Initiate Product Backlog*, *Plan Release* and *Preparation Tasks*. The *INGENIAS Product Backlog* contains the product requirements established using the IDK. This process can be done by adapting a known model from a previous project (i.e. IDK-IAF distribution comes with a complete cinema project which can be used for other distributed e-commerce developments) or defining a completely new product backlog with the editor. After this initial model is completed, in the *Preparation Tasks*, the *Scrum Master* and the *Product Owner* establish the *Plan Release* in which the needed *Sprints* are defined.

From the Scrum perspective and taking into account that IAF bases on the automatic generation of code approach, the project team must be completely involved in getting the release within the planned sprints. So, the INGENIAS specification must be established with the IDK as the core of the development. From this core, the different scripts and sources will be automatically produced. Nevertheless, at the first stage, the generated code for the tasks may be incomplete and the programmer should add, if necessary, code in the tasks.

3.3 Define the Disciplines View

As previously pointed, in the INGENIAS-Scrum approach the disciplines are the tasks required in each sprint, so the intended meaning of each task, according to IAF, must be explained. But first, the roles and products involved in the development must be introduced [27]. The roles, in this case, are:*Product Owner*, this role must be play by an active *customer* as in eXtreme Programming (XP); *Scrum Master*, the *coach* and main of the development team; *Scrum Team*, this is a *collective role* that must be played by any of the team members; *Stakeholder*, anyone that does not directly participate on the project but can influence the product being developed, that is, an *interested party.*

The products or artifacts involved in a Scrum development process are: *Product backlog*, *Sprint backlog* and *Product increment.* The *product backlog* contains the product requirements and has the purpose of listing all the functionalities to implement from a customer perspective. The *sprint backlog* is the list of things to do from the development team point of view. It could be understood as a fine-grained planning on detailed tasks. At last, the *product increment* is a partial product obtained at the end of each sprint, which can be deployed in the production environment or simply made available to users. From the INGENIAS-Scrum perspective those artifacts are referred to the INGENIAS model and JADE code produced in each release. An INGENIAS model documented with the IDK can accomplish a low o high level of detail. Also, in the description of each model the *Scrum Master* can fix the work to be done in the next release, where release can be identified with the *package* entity of the INGENIAS model.

3.4 Define Guidances View

Developing a system with code generation facilities requires some guidance. In IAF documentation [18] several guidelines for development are proposed. In multi-agent systems, we recommend specially the use of two kinds of guidance: *Technique* and *Guideline.* The *technique* provides an algorithm to create a work product. The *guideline* is a set of rules and recommendations about a work product organization.

4 Case Study and Results

This section addresses the comparison of INGENIAS following the Scrum Process using the IAF on the IDK and the RUP for INGENIAS using the full version of the IDK tool. This comparison is based on the development of the same MAS by different groups of students and in measuring during the development some variables that can determine the productivity and performance of the processes selected. As authors teach Software Engineering Courses at Computer Science Faculty on the Vigo University, it was decided to use the students for this study. The students were divided into groups of 4-6 people and were asked to develop a complete agent oriented system. The groups were homogeneous in the number of members, but defer among them in the knowledge of development processes, in particular RUP. Three of the groups have a good knowledge of RUP, another three have a superficial knowledge and the rest do not know anything of RUP. None of the groups have notions about what was Scrum process of development.

Moreover, people that conforms the teams do not know the INGENIAS methodology, neither its meta-modeling language. So the challenge for the teams was twofold: to learn INGENIAS methodology and meta-model and to understand Scrum process.

With this set of students, we plan an experiment, which tries to constitute a first approach in the study of the influence of the development process selected over productivity. Data of the experiment were collected along the Software Engineering Course, while the students develop the system that was mandatory. The requirements of the system to obtain are explained next.

4.1 Case Study Description

As stated before, all the groups were asked to develop the same agent-oriented system. In particular, every team must develop a web site for managing software development projects based on the Scrum Development Process. This selection was made to increase the level of knowledge about Scrum in the groups in the first steps and phases of the development. The portal must allow:

1. *The Creation, Modification, Project Monitoring and Closure* of a Software Project that is being constructed using Scrum Process.
2. *The Establishment of the teams and the assignation of the roles* that each member team will assume along the project. The system must also allow changes in team composition or roles assignation. These changes may be done dynamically during development.
3. *The Management of each Project Sprint.*
4. *The Creation, Identification, Modification and Monitoring* of every meeting held during the Sprints.
5. *The Storage, Retrieval and Collaborative modification* of any work product obtained along the development.

The different teams were asked to provide some intermediate deliverables along the development, in order to have some control on their evolution with independency of the process selected. The first deliverable demanded was a simulation of the proposed solution to the portal modeled using Alice [33]. This first deliverable had a double utility. In the one hand, it served as a control point in the degree of knowledge of Scrum achieved by the teams. On the other, it was used as the way of establishing the system functional requirements for each group. Obviously, if the prototype was wrong, the students had to modify it until considered correct.

Besides, three more deliverables were established one of them each two months. Excluding the first one described that must be finished one month after the beginning.

These deliverables were used to measure the degree of functionality accomplished, according to the functionality proposed by each team in the prototype provided.

4.2 Experiment and Results

Several variables were taken into account in the study, as we considered that they may influence the final productivity of the teams developing the system.

Each team is identified by a number (it appears in first column of Table 2). This is just a way of having the possibility of referring to a particular team, if needed.

The experience column expresses the background of each team regarding the knowledge of RUP or any other development process. We consider that previous knowledge of processes can influence in learning new ones. Nevertheless, as said before, none of the groups have previous knowledge of Scrum.

The process column shows what kind of process was used by the team during the development. The process that they have to follow was assigned randomly by the teachers and independently from previous background of the teams.

The prototype column indicate the kind of solution chosen by each team. Three are the possibilities in this column:

- *Server-oriented solution.* A server-oriented solution is a classic client-server solution in which features are provided by a single central server to different kind of users. Each user has its own privileges to operate in the web site.
- *Service-oriented solution.* A service-oriented solution describes the site as a set of distributed services that may be requested concurrently by the user. It is also possible in this kind of solution that certain services are automatically offered to selected users that satisfy some condition.
- *User-oriented solution.* An user-oriented solution prioritizes a centralized and sequential management of the development process. In this kind of solution users interact with the system and decide when and which data should be entered.

From a theoretical perspective, the more suitable solution from MAS point of view is the second one, because a distributed net could be easier established over an agent platform. Each service could be offered by a particular agent or by a group of agents. Each agent could have their own goals, that have to be satisfied by a set of tasks and that require some interaction with other agents. So that, this drives to a collaboration or coordination on the tasks to satisfy agent's goals. The rest of solutions present some drawbacks in the deployment, because the methodology is not so suitable for them. Nevertheless, all solutions are still feasible to be adopted in a MAS approach, although some of them will imply an extra cost of time.

The four latest columns measure the rate of functionality achieved for each of the deliverables provided. As stated before, this rate is calculated referring to those proposed by each team in their prototype and are ordered by their final achievement.

Table 2 presents the results obtained from the case study.

Table 2. Teams' performing results

Team	Experience	Process	Prototype	1st Del.	2nd Del.	3rd Del.	Final
6	High	Scrum	Service Oriented	15%	30%	55%	85%
2	High	Scrum	Server Oriented	15%	25%	50%	80%
1	High	RUP	Service Oriented	20%	35%	50%	75%
5	Medium	Scrum	User Oriented	15%	25%	45%	75%
8	Low	Scrum	Service Oriented	15%	22%	45%	75%
3	Medium	RUP	Server Oriented	15%	30%	40%	70%
7	Low	Scrum	Server Oriented	15%	30%	50%	70%
4	Medium	RUP	Server Oriented	15%	25%	45%	65%
9	Low	RUP	User Oriented	10%	20%	40%	60%

4.3 Discussion

An initial analysis of the results tries to address productivity of the teams. We consider that teams are more productive when they achieve a higher rate of functionality for the same increment. This consideration could make sense as far as all the teams have the same time available for delivering the increment. The results show that the productivity on the teams using Scrum as process was higher in average, even when taking into consideration the kind of solution adopted. The study shows also that the learning time is shorter for members of the development group that follow an Scrum processes, because the increments in the rate of functionality are higher. This could allow the use of smaller groups of development, due to the higher productivity.

A second issue to consider in results shows that the ratio of evolution in the productivity is higher in the last deliverables for Scrum teams. This suggests, in our opinion, that Scrum process is a better solution to manage the lifecycle for INGENIAS methodology than the RUP originally proposed.

Other factor that can influence the results, is the previous knowledge or background of the teams. We do not have a quantitative measure of the background, instead we consider a rough measure of it, idenfying three possible values: High, Medium or Low. Nevertheless, in order to improve the study in the future a quantitative measure will be incorporated. Moreover, although experience must have an impact on productivity, we thought that the process has a more important influence. This is justified by the fact that less experienced groups have a better productivity using Scrum than more experienced groups using RUP.

The kind of solution chosen (column Prototype of Table 2) can also influence productivity of teams. Nevertheless we consider that it is not so important, because teams using *ServerOriented* Prototype have good productivity results.

As said in previous subsection, the selection of what process must follow each group was assigned randomly by the teachers of Software Engineering. Productivity may be affected if the choice were free. Other studies in the future may address the changes in productivity when groups can choose the process to follow. At present, this objective was not in the scope of the experiment.

The results obtained have exceeded expectations. For instance, a medium experienced and a novice team got the same productivity results using Scrum than one of the expert teams using the RUP. Moreover, the worst results were obtained by a novice team and a medium experience team using the RUP approach. These results could be interpreted as an evidence that the RUP approach needs much more time to be learned and used with solvency.

However, some results not expected were found. First, one of the worst results was gotten using RUP with a server oriented solution. Secondly, a good rate of productivity was obtained by team with an user oriented solution using the Scrum development Process.

Results obtained by team number five seem to be a surprise because the kind of solution chosen is not the most suitable for the problem. This can be explained by the fact that the team has dynamically chosen to apply a shorter cycle of deliverables and to associate an agent to each kind of user. This has increased the interaction between agents, and in fact, the architecture proposed by the group represents each of the Scrum roles by an agent. This change has increased the productivity of the final two deliverables.

5 Conclusions and Future Work

In previous sections the MAS methodology INGENIAS was presented jointly with two different approaches for their process development. In order to compare productivity and suitability of both a case study has been also presented. The case study was solved by nine teams that have no knowledge about the Scrum process, that was *the object of the study.* Teams have a different preparation on Process Development: *three are experts, three are novices, and the rest have an average training.* This selection was chosen in order to get a complete information about the real productivity of the Scrum approach when compared with the OpenUp approach.

Summarizing results, the conclusion is that the ratio of evolution in the productivity is higher in the last deliverables for all the Scrum teams. At first glance, at attending these results, Scrum process seems a better solution to manage the lifecycle of the INGENIAS approach that the OpenUp one. Of course these results constitute an initial approach in the comparison of the processes (OpenUp and SCRUM) for INGENIAS methodology.

This first approach suggest that SCRUM is better suited but more studies have to be done. In particular, we consider that more data are needed, so that we will go on with the experiment next years. Other studies will be also valuable, such as fixing as object of the study the OpenUp approach to compare the results, or doing the comparative with teams with the same level of capacitation on the two development processes. Other possibility, considered as a future work, is to have a new structure for teams (for instance, 4 or 6 people working by pairs as suggested by XP). Besides, it is worth considering in the future other variables that may influence the results such as the kind of system to construct, the size of groups, etc.

In any case the results obtained are very promising in order to experiment with new agile process for AOSE development with INGENIAS.

Acknowledgement. This work has been supported by the project *Novos entornos colaborativos para o ensino* supported by Xunta de Galicia with grant 08SIN009305PR.

References

1. Jennings, N.R., Wooldridge, M.: Agent-Oriented Software Engineering. In: Bradshaw, J. (ed.) Handbook of Agent Technology. AAAI/MIT Press (to appear, 2001)
2. Wooldridge, M., Ciancarini, P.: Agent-Oriented Software Engineering: The State of the Art. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 1–28. Springer, Heidelberg (2001)
3. DeLoach, S., Wood, M., Sparkman, C.: Multiagent Systems Engineering. International Journal of Software Engineering and Knowledge Engineering 11(3), 231–258 (2001)
4. Henderson-Sellers, B., Giorgini, P.: Agent-Oriented Methodologies. Idea Group Inc., USA (2005)
5. Ricordel, P.M.: Programmation Orientée Multi-Agents: Développement et Déploiement de Systemes Multi-Agents Voyelles. PhD thesis, Institut National Polytechnique De Grenoble (2001)
6. Pavón, J., Gómez-Sanz, J.: Agent Oriented Software Engineering with INGENIAS. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 394–403. Springer, Heidelberg (2003)
7. O'Malley, S.A., DeLoach, S.A.: Determining when to use an agent-oriented software engineering pradigm. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, p. 188. Springer, Heidelberg (2002)
8. Cuesta, P., Gómez, A., González, J., Rodríguez, F.J.: The MESMA methodology for agent-oriented software engineering. In: Proceedings of First International Workshop on Practical Applications of Agents and Multiagent Systems (IWPAAMS 2002), pp. 87–98 (2002)
9. Bernon, C., Cossentino, M., Pavón, J.: Agent-oriented software engineering. Knowl. Eng. Rev. 20(2), 99–116 (2005)
10. Mas, A.: Agentes Software y Sistemas Multi-Agentes. Pearson Prentice Hall, London (2004)
11. Padgham, L., Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents. In: Proceedings of the Third International Workshop on Agent Oriented Software Engineering, at AAMAS (2002)
12. Chella, A., Cossentino, M., Sabatucci, L., Seidita, V.: Agile PASSI: An Agile Process for Designing Agents. International Journal of Computer Systems Science & Engineering. Special issue on Software Engineering for Multi-Agent Systems (May 2006)
13. Cossentino, M., Sabatucci, L.: Agent System Implementation. In: Agent-Based Manufacturing and Control Systems: New Agile Manufacturing Solutions for Achieving Peak Performance. CRC Press, Boca Raton (2004)
14. INGENIAS Development Kit, `http://ingenias.sourceforge.net/`
15. Cossentino, M., Sabatucci, L., Seidita, V., Gaglio, S.: An Agent Oriented Tool for New Design Processes. In: Proceedings of the Fourth European Workshop on Multi-Agent Systems (2006)

16. Pavón, J., Gómez-Sanz, J.: Agent Oriented Software Engineering with INGENIAS. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 394–403. Springer, Heidelberg (2003)
17. Pavón, J., Gómez-Sanz, J.J., Fuentes-Fernández, R.: IX. In: The INGENIAS Methodology and Tools, pp. 236–276. Idea Group Publishing, USA (2005)
18. Gómez-Sanz, J.: Ingenias Agent Framework. Development Guide V. 1.0. Technical report, Universidad Complutense de Madrid (2008)
19. García-Magariño, I., Gómez-Rodríguez, A., Gómez-Sanz, J., González-Moreno, J.C.: INGENIAS-SCRUM Development Process for Multi-Agent Development. In: International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2008), Advances in Software Computing (2008)
20. Fuentes-Fernández, R., García-Magariño, I., Gómez-Rodríguez, A.M., González-Moreno, J.C.: A technique fordefining agent-oriented engineering processes with tool support. Engineering Applications of Artificial Intelligence 23(3), 432–444 (2010)
21. Ferber, J.: Multi-Agent Systems. Addison-Wesley, Reading (1999)
22. Wagner, T., Horling, B.: The struggle for reuse and domain independence: Research with taems, dtc and jaf. In: Proceedings of the 2nd Workshop on Infrastrucutre for Agents, MAS and Scalable MAS (2001)
23. Shoham, Y.: Agent-oriented programming. Artificial Intelligence 60, 51–92 (1993)
24. Kinny, D., Georgeff, M.: Modelling and design of multi-agent systems. In: Jennings, N.R., Wooldridge, M.J., Müller, J.P. (eds.) ECAI-WS 1996. LNCS (LNAI), vol. 1193, pp. 1–20. Springer, Heidelberg (1997)
25. Beydoun, G., Low, G.C., Henderson-Sellers, B., Mouratidis, H., Gómez-Sanz, J.J., Pavón, J., Gonzalez-Perez, C.: Faml: A generic metamodel for mas development. IEEE Trans. Software Eng. 35(6), 841–863 (2009)
26. García-Magariño, I., Gómez-Rodríguez, A., González, J.C.: Modeling INGENIAS development process using EMF. In: 6th International Workshop on Practical Applications on Agents and Multi-agent Systems, IWPAAMS 2007, Salamanca Spain, November 12-13, pp. 369–378 (2007) (in Spanish)
27. García-Magariño, I., Gómez-Rodríguez, A., González-Moreno, J.C.: Definition of process models for agent-based development. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2009. LNCS, vol. 5386, pp. 60–73. Springer, Heidelberg (2009)
28. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: JADE - A Java Agent Development Framework. Multiagent Systems, Artificial Societies, and Simulated Organizations 15(2), 125–147 (2005)
29. García-Magariño, I., Gómez-Sanz, J.J., Fuentes-Fernández, R.: Model transformations for improving multi-agent systems development in ingenias. In: The 10th International Workshop on Agent-Oriented Software Engineering, AOSE 2009, Budapest, Hungary (with the annex), May 11 (2009)
30. Sutherland, J.: Business object design and implementation workshop. In: OOPSLA 1995: Addendum to the proceedings of the 10th annual conference on Object-oriented programming systems, languages, and applications, pp. 170–175. ACM, New York (1995)
31. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall PTR, Upper Saddle River (2001)
32. OMG.: Software Process Engineering Metamodel Specification. Version 2.0, formal/2008-04-01 (2008), `http://www.omg.org/`
33. Dann, W.P., Cooper, S., Pausch, R.: Learning To Program with Alice, 2/E. Prentice Hall, Englewood Cliffs (2009)