# UML and Agile Methods: Looking for an Agreement

Jose A. Gallud[1] and Habib M. Fardoun[2]
[1]*Escuela Superior de Ingeniería Informática, University of Castilla-La Mancha, Albacete, Spain*
[2]*Ahlia University, Bahrein*

Keywords: UML, Agile methodologies, Xcrum.

Abstract: One of the effects of the nth software crisis is the great expansion of agile methodologies. Many agile methodologies have appeared since the publication of the Agile Manifesto. Agile methodologies are considered light in comparison to the traditional heavy ones. This paper describes (a) whether or not it is worth to use some UML diagrams as artefacts in Agile methods and, (b) what would be the set of UML diagrams selected as useful artefacts obtained in an agile process and why. The paper makes use of a particular agile method called Xcrum to show how and when the proposed UML diagrams would be employed.

## 1 INTRODUCTION

The field of Software Engineering has solved its nth growth crisis with the emergence of agile methodologies. These methodologies seem to free the heavy burden development teams were suffering while were applying documentation-based methodologies. The need to elaborate many UML artefacts has been considered a heavy task by the development team, with an unworthy impact in the final product.

The Agile Manifesto declares, among other ideas, that agile teams come to value working software over comprehensive documentation (Beck and several authors, 2001). Heavy documentation is not forbidden but it is not valued. Agile methods are introducing new ways of controlling the development process and reducing the need of documentation.

UML (J. Rumbaugh and Booch, 1998) has been playing an important role in the heavy methodologies we have been using during the last ten years. People involved in development teams, software architects, analysts, requirement engineers, software designers, database architects, and so on, have been spending hours elaborating long documents that were full of use cases, activity, class, deployment and all kinds of UML diagrams. The question that emerges now is what is the role of UML in this new agile world?

This article declares the validity of UML in this new agile age, and proposes a set of UML artefacts that are compatible with Agile methodologies. The proposal is based on Xcrum, a new agile proposal for the object-oriented development of software applications and services. The Object Oriented programming paradigm maintains its influence thanks to the new programming paradigms introduced by the Internet languages. Languages such as JavaScript, Python or Ruby are presented in many modern Web projects, and they are the appropriate technologies in projects where Xcrum is applied.

The article is organized in the following sections. Section II presents a brief overview of the current landscape of software development. The next two sections (Section III and IV) presents the basis of Xcrum. Section V presents a general discussion of the proposal. Finally, Section VI presents the conclusions and future work.

## 2 AGILE METHODS

One of the effects of this latest software crisis is the birth of agile methodologies. The appearance of the now famous Agile Manifesto (Beck and several authors, 2001) is the starting point of this new era. In its genesis is a group of software engineers who decided to rebel in the ”plan and document” style omnipresent in most development teams. At the time of the emergence of the Agile Manifesto, methodologies such as the cascade model, the Rational unified process (Rational Unified Process) (Kruchten, 2003), or Rapid Application Development (RAD) (Martin, 1990) dominated the development teams.

As noted by Fox and Patterson (Fox and Patterson, 2014), the crisis of the 1960s led engineers to try to develop methodologies that made it possible



Gallud, J. and Fardoun, H.
UML and Agile Methods: Looking for an Agreement.
DOI: 10.5220/0006940907800785
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 780-785
ISBN: 978-989-758-320-9

to develop quality software and predictable and controlled budget. The results of this effort were a series of development processes based on planning and documenting (Plan-and-Document). Some strategies (process models) on which many of these development processes are based are well known: Cascade model, Spiral model and Iterative and Incremental models. In this last group, the RUP process should be located, which is a combination of the previous models. Development processes based on Planning-Document are tedious since they put a lot of emphasis on the preparation of documents, such as templates, memories and diagrams, which sometimes diverts attention from the software product that is intended to be developed.

The agile proposal tries to recover the lost prominence for the software product in favour of documents and diagrams. Alternatively, the Agile Manifesto also gives importance to the motivation of the development team, and to maintain fluid contact with the client.

As a result, agile methodologies began to emerge that tried to apply the principles listed in the Agile Manifesto. Among the best known we can mention Scrum (Schwaber, 2004) and Extreme Programming (better known as XP) (Beck, 2004).

From Scrum and XP, there are numerous agile proposals that have been emerging, although these are general proposals that are based on applying the principles contained in the agile manifesto. For example, DSM (Bussiness, 2001) is based on a principle in the integrates several statements of the Agile Manifesto: the best business value emerges when projects are aligned with clear business objectives, often visible results are released and involve motivated people. In this sense, Xcrum also applies the principles of the Agile Manifesto, as it is an agile method based on XP and Scrum.

Letelier, in (Letelier and Penades, 2017), presents a catalogue (AgileRoadmap) to implement agile practices in development teams, without suggesting a particular agile method. The Xcrum proposal is similar to AgileRoadmap in that it promotes agile practices, although Xcrum is based on principles that come from Scrum and XP, to which the object-oriented vision of the solution joins.

In (Mekni et al., 2017), the authors summarize the best-known agile methods to point out the little attention paid to software architecture. The authors then propose a methodology to define software architecture in agile environments. In Xcrum, the software architecture is an essential element, since it is based on starting defining the solution as an object-oriented solution.

To conclude this section, Xcrum is related to most agile methods insofar as it applies the principles of the Agile Manifesto. However, it is novel in that it takes as reference two of them, Scrum and XP, to propose a synthesis of both.

# 3 Xcrum

This section briefly describes the main elements of Xcrum (Gallud, 2018). The section is organized in the following sections: roles, iteration, artefacts and meetings.

Like any good agile method, the most important artefact in Xcrum is the code tested and working. All other elements are means to get the software product.

## 3.1 Roles of Xcrum

The roles of Xcrum take into account the separation of responsibilities, business and techniques, suggested by the Agile Manifesto. There is separation and also complementarity since it is about that both work together in obtaining the solution.

Thus, Xcrum uses Business and Development as terms to define the roles that identify two responsibilities. Business defines the user stories and the priority of them. Development is responsible for estimating stories and converting them into code.

The Business role of Xcrum is equivalent to the Scrum Product Owner. The Development role is equivalent to the Scrum Team role. The ScrumMaster of Scrum is the leader of the development team at Xcrum.

## 3.2 Iteration in Xcrum

The heart of Xcrum is the Iteration in the same sense that the Sprint is for Scrum. The Iteration in Xcrum lasts from 1 to 4 weeks. At the end of the iteration, the team must provide an increase in value in the form of code tested and functioning.

The Xcrum Iteration is equivalent to the Scrum Sprint.

## 3.3 Xcrum Artefacts

User stories: Xcrum is based on defining the system requirements (functional, non-functional and information) as user stories, in a similar way to other agile methodologies.

- List of System's Histories: is the list of user stories defined by Business to describe a system.

This list is equivalent to the Scrum Product Backlog.

- List of Iteration's Stories: is the subset of stories that are assigned to an iteration. This list is equivalent to the Sprint Backlog.
- Progress Chart: is the diagram that in Scrum is called Sprint Burndown and that serves to measure the progress of the iteration (sprint).

As it can be noted, the Xcrum artefacts are mainly those of Scrum.

## 3.4 Xcrum Meetings

The Xcrum meetings follow the structure of Scrum, with incorporation of some XP activities. The following sections detail how XP activities are combined in Scrum meetings.

### 3.4.1 Iteration Preparation Meeting

The two roles, Business and Development, participate in this meeting. In Scrum the Sprint Preparation Meeting has two parts, in the first one it is carried out between Business and Development and its objective is to choose the functionality of the next Sprint. In Xcrum this meeting has the same objective, although it is proposed to incorporate the activities of the Exploration and Commitment phases of XP, namely:

- Write a story: Business writes a functionality
- Estimate a story: Development estimates the time
- Divide a story: If you can not estimate
- Sort the stories: Business order by value and Development by risk
- Choose field: Business chooses the end date of the Iteration or functionality (and Development date)

As you can see, this part of the preparation meeting of the Iteration takes advantage of the detail of activities that XP provides, while, in Scrum, it is left undefined.

The second part of the meeting corresponds to the team and consists of detailing the tasks in which each story is broken down.

### 3.4.2 Xcrum Daily Meeting

On a daily basis, the team reviews the status of the project following the same scheme proposed by the Daily Scrum. At this point some XP activities are proposed to obtain the increment:

- Accept a task: a developer chooses a pending task
- Implement a task: define the test cases, implement the task and integrate the code
- Recovery and re-estimation: these are activities to readjust the load, or the dates, with respect to the estimate.

These XP activities, incorporated into Xcrum, serve to give content to the daily task of the team. The only difference in relation to XP is that in Xcrum it is not necessary to implement the task through the technique of programming in pairs.

An important aspect of Xcrum are the tests. The team must write test cases so that the meaning of "finished" is demanding, not thinking about Business, but internally, thinking about the team.

In the day to day of development the third principle of Xcrum is put into play: Object Oriented Solution. We will deal with this aspect in a later section (section VI).

### 3.4.3 Iteration Review Meeting

In Scrum, at the end of the Sprint, two meetings are held: the Review meeting and the Retrospective meeting. The first is done with the Product Owner. The second is internal to the team.

In Xcrum, a similar scheme to Scrum is proposed. In the first meeting (Review) the increase in value of this Business Iteration is shown to check if it is what you requested. The second part is that Development reviews the realization of the iteration.

# 4 THE OBJECT-ORIENTED APPROACH IN Xcrum

Since the appearance of the book Design Patterns (E. Gamma and Vlissides, 2011), the paradigm of Object Oriented programming has experienced a growing development that continues to influence many of the software solutions regardless of the technology used.

In recent years we have witnessed, in Web programming, a renaissance of dynamic languages, some of which have appeared for some time, such as Javascript, Python, Ruby and PHP. While most dynamic languages can be said to be object oriented and class oriented. However, languages such as Javascript (at least until version 7), omit the definition of classes, since they are languages oriented to prototypes and functions, although it is possible to use them with the same approach used in the object-oriented and class-oriented paradigm.

There are many advantages to using the object-oriented and class-oriented paradigm, as opposed to prototype-oriented. However, this discussion is outside the scope of this article.

What interests us here is to define how to obtain an object-oriented solution, and how it is integrated into the Xcrum methodology.

Obtaining an object-oriented solution is based on a single principle:

*Object model should come first*

This principle establishes as a priority to first define the object model of our application. This means that both the user interface and persistence must be left for later. In this way, and in broad strokes, the proposal for the content of the iterations follows this sequence:

- Examine functionality based on user stories
- Define the object model (with the tests)
- Define the service layer
- Define the user interface
- Define the persistence layer
- Final deployment

This principle applies more easily to Web-based solutions. In these environments, the first phase (define the object model), can be detailed as follows:

- Develop the model objects in the client: using the tools provided by most browsers we can validate our object model
- Define the tests using some testing framework
- Move the model to the server: the ideal solution is for the client's own object solution to be the one we use on the server. This depends on the technology chosen
- Adapt the tests to the model on the server

The first step (develop the object model) is also broken down into activities that are repeated until we get the complete object model. Here we apply an XP principle by which we design (and implement) only what is needed, avoiding including everything we know in advance that we will need. So, the object model is obtained with the following steps:

1. Choose a functionality (from the list of user stories)
2. Draw the class diagram with the minimum that is needed to implement that functionality
3. The team conducts a discussion of the model
4. Implement the classes following the diagram
5. Validation in the browser
6. Optional tests are implemented: It is not convenient to write the tests too early to avoid rewriting. This contradicts an XP principle but is more practical.
7. Go back to step 1

The previous steps are completed when the developed model contemplates a sufficient set of functions.

This procedure is especially useful when using the same technology for client and server (such as Javascript and NodeJS). The benefit of using the same technology is applied to the tests, since the same set of test cases, with minimal changes, is valid on the server side.

# 5 UML AND AGILE METHODOLOGIES

A true agile team member knows that the most important artefact obtained in an agile methodology is the tested and working code. There is nothing more important than the working code. All the other agile artefacts (list of user stories, iteration user stories and progress chart) are only secondary tools to help developers to control the agile process.

Regarding documentation and UML artefacts, one thing is to force the development team to follow a heavy and strict set of documentation as part of the development process, as it happens in the Unified Process, and a different thing is to let them decide what artefacts they want to use and which ones to avoid.

An important Agile Manifesto principle says "Build projects around motivated individuals" (Beck and several authors, 2001), which can be applied to UML artefacts. Provided that the development team understand how UML can help to develop a modular and extensible solution, they will be the first to propose use one UML diagram.

The following set of UML artefacts are highly recommended to use in Agile methodology:

- Class diagrams
- Deployment diagrams

UML Sequence diagrams can be useful for the development team. However, they could be considered as optional artefacts.

The following sections describe the role of each diagram in Xcrum. The fact we illustrate how to use these UML artefacts in Xcrum does not mean they are restricted to this agile method. They can be used in any other agile method in the same way they are used in Xcrum. At least it is sure they can be used in Scrum and XP, since these agile methods are the foundation of Xcrum.

### 5.1 UML Class Diagrams

UML class diagrams provides a structural object-oriented view of our solution. This artefact is one of the most important artefacts a development team can use.

The team's goal is to build an object-oriented solution. One of the best means to reach this goal is by using UML class diagram. Class diagram allows the development team to discuss the best design that will implement the required functionality.

The previous sections shows how Xcrum promotes the definition of the object model at first. As you can see in that section, the step 2 is "Draw the class diagram with the minimum that is needed to implement that functionality". Xcrum recommends the team to design only those entities and relationships that are needed to implement the selected user story.

As the solution is built by increments defined by the iterations, this artefact will be evolving as the same pace as the software. First iterations will show a few entities and relationships and, as long as the solution grows, the class diagrams grows as well.

One of the hard aspects of using UML diagrams is to maintain the coherence between the code and the class diagram. This is not a problem since this review can provoke the development team re-think the solution, which could derive in some kind of refactoring or redesigning. Some developers find better design solutions after analysing the class diagram.

### 5.2 UML Deployment Diagrams

UML deployment diagrams can play an important role in any software development process. When developing software using an agile methods, it is possible to use UML deployment diagrams for the exact same purpose.

One of the first step in the development of any software is to define the architecture of the solution. Special attention deserves the software architecture of the solution. UML deployment diagrams are one of the best means to represent the designed software architecture.

UML deployment diagrams show the nodes and artefacts that are involved in the solution, and how they are interconnected. The different artefacts change through the development of the software. At the end of each iteration, the team can show the artefact that has been included in that iteration.

In Xcrum we can use UML deployment diagrams from the first iteration, since this first diagram provides the foundation elements of the software architecture. UML deployment diagrams allow the development team to have a global idea of the important building blocks of the solution.

### 5.3 UML Sequence Diagrams

UML sequence diagrams can play an important role when the development team is discussing some complex interaction among objects. The development team can learn from the designed solution and find better ways to implement the desired functionality.

However, UML sequence diagrams have been declared as optional artefacts due to different reasons. There are two main reasons:

- An object-oriented solution well designed consists on many small objects. In this solutions the use of delegation and composition makes sequence diagrams hard to draw (and many times useless)
- The wide use of dynamic languages (as Javascript or Python) makes almost impossible to represent the complex behaviour they are able to implement. Important aspects of the language as are anonymous functions, cannot be easily represented using any of the UML interactions diagrams.

When using Xcrum, the development team is kindly invited to design and build an object-oriented solution. Therefore, we can experience the problem addressed before about the number of small collaborative objects. However, as stated before, the development team can use a sequence diagram to discuss a particular design or to explain the rest of the team, how works a complex interaction among objects.

In summary, some UML diagrams, as class and deployment diagrams, can be useful for an agile development team since they can contribute to design and build the best object-oriented solution. An object-oriented solution provides many of the best qualities any agile development team seeks, as are extensibility, modularity, robustness and correction.

## 6 CONCLUSIONS

This article argue in favour to asign a role to UML artefacts in this new agile age. The proposal includes a set of UML artefacts that are compatible with Agile methodologies.

UML class diagrams and deployment diagrams have been proposed as essential diagrams to be used in any agile method. Sequence diagrams have been declared as recommendable optional diagrams.

The proposal is explained using Xcrum, a new agile method for the object-oriented development of software applications and services. Xcrum has been inspired by Scrum and eXtreme Programming.

This article shows that UML diagrams may be useful to reach the goal of designing and building the best software in the shortest time.

## ACKNOWLEDGEMENTS

There is always someone whose ideas have been source of inspiration. In our case we would like to thank the small but bold group of Smalltalk developers since they are always promoting object-oriented creative software.

## REFERENCES

Beck, K. (2004). *Extreme Programming Explained*. Addison-Wesley; 2nd edition.

Beck, K. and several authors (2001). Agile Manifesto. Website. http://www.agilemanifesto.org.

Bussiness, A. (2001). The Dsm Agile Project Framework. Website. https://www.agilebusiness.org/resources/dsdm-handbooks/the-dsdm-agile-project-framework-2014-onwards.

E. Gamma, R. Helm, R. J. and Vlissides, J. (2011). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Fox, A. and Patterson, D. (2014). *Engineering Software as a Service. An Agile Approach Using Cloud Computing*. Strawberry Canyon LLC.

Gallud, J. A. (2018). An agile object-oriented method to develop modern software applications. In *Technical Report DIAB-18-05-2, DSI, UCLM*.

J. Rumbaugh, I. J. and Booch, G. (1998). *The Unified Modeling Language Reference Manual*. Addison Wesley.

Kruchten, P. (2003). *The Rational Unified Process: An Introduction*. Addison-Wesley Professional.

Letelier, P. and Penades, M. C. (2017). Agileroadmap: An approach to implement agile practices in teams. In *IEEE Latin America Transactions, VOL. 15, NO. 7*.

Martin, J. (1990). *Rapid Application Development*. MacMillan Publishing Co. Ed.

Mekni, M., Mounika, G., Sandeep, C., and Gayathri, B. (2017). Software architecture methodology in agile environments. In *J Inform Tech Softw Eng 7: 195. doi: 10.4172/2165-7866.1000195*.

Schwaber, K. (2004). *Agile Project Management with Scrum*. Microsoft Press.