07-005

# Analysis of the agile methodologies applied in software engineering within the framework of the PMBOK knowledge areas

Manuel J. García Rodríguez; Vicente Rodríguez Montequín; Joaquín Manuel Villanueva Balsera; Ramiro Concepción Suárez

Universidad de Oviedo;

Software projects have a number of peculiarities that make them different from other projects. The most important ones are that they are labour intensive, the environment evolves rapidly and in many cases there is a poor definition of scope. Consequently, the methodologies to manage these projects will have to address these issues in order to get the project developed satisfactorily. In recent years, new trends have appeared in the management of software projects, most of them of "agile" type, as alternative of "traditional" or predictive methodologies. This paper compare and analyses the characteristics of the most used agile methodologies with traditional methodologies in the context of software engineering, using the knowledge areas of the PMBOK as a framework for the comparison. PRINCE2 and METRICAv3 are considered among the traditional methodologies, and PRINCE2 AGILE, SCRUM and Featured Driven Development (FDD) are considered among the agile methodologies. Results shows, among other things, that are important gaps not covered by any methodology.

***Keywords:*** Agile methodology;SCRUM;PRINCE2;PRINCE2 AGILE;FDD;METRICA

# Análisis de las metodologías ágiles aplicadas en ingeniería del software en el marco de las áreas de conocimiento del PMBoK

Los proyectos software tienen una serie de particularidades que los hacen distintos al resto de proyectos. Las más importantes son que son intensivos en mano de obra, el entorno evoluciona rápidamente y en muchos casos hay una pobre definición del alcance. Consecuentemente, las metodologías para gestionar estos proyectos tendrán que abordar estas problemáticas para conseguir que el proyecto se desarrolle satisfactoriamente. En los últimos años han aparecido nuevas tendencias en la gestión de proyectos de tipo software, la mayor parte de ellas de tipo “ágil”, como alternativa a las denominadas metodologías "tradicionales" o predictivas. En este trabajo compara y analiza las características de las metodologías ágiles más utilizadas con metodologías tradicionales en el contexto de la ingeniería del software, utilizando como patrón de comparación las áreas de conocimiento del PMBOK. Dentro de las metodologías tradicionales se han considerado PRINCE2 y METRICAv3, mientras que para las metodologías ágiles se han considerado PRINCE2 AGILE, SCRUM y Featured Driven Development (FDD). Los resultados muestran, entre otras cosas, que hay aspectos importantes que no está cubriendo ninguna metodología.

***Palabras clave:*** *Metodología ágil;SCRUM;PRINCE2;PMBoK*

Correspondencia: Vicente Rodríguez Montequín montequi@api.uniovi.es

# 1 Introduction

Efficient project management is very important for organizations for many reasons: high complexity of projects, strong competition between companies, cost reduction and inefficiencies, human resources management, quality assurance, a lot of stakeholders, monitor and control of project work and costs, etc... These are also key points for organizations delivering software projects. If fact, these organizations are putting an extra effort managing its projects due to the traditional problems within this field. In general, there is a perception that software project failures are high. This is not something recent, in 1968 there was the so-called "software crisis".

There were a series of events observed in software development projects:

- Project did not end on time.
- Project did not fit the initial budget.
- Poor quality of software.
- Software did not meet the specifications.
- Fixed code which made difficult the evolution of the project.

These points remain of great importance for the success of the project because they have not been resolved yet in an optimal way. This crisis provoked that the linear methods, also named as “traditional methods” (waterfall or cascade) were transformed into the evolutional methods. This suggests that software projects have a series of difficulties of their own, intrinsic to the field. These difficulties are very well summarized in Bourque and Fairley (2014). The Standish Group (2013) statistics show that less than 40% of software projects between 2004 and 2012 were successful, that is, they were completed fulfilling costs, deadlines and have all the functionalities. Therefore, the development of software projects has many different factors that prevent the success of the project, so management is a very important task for organizations.

During the past years, the use of agile methodologies appeared as a trend for overcoming the traditional issues. This paper makes a comparison between what are commonly referred to as traditional methodologies and agile methodologies and the new trends. The objective is not to make an exhaustive comparison between different methodologies, but rather to find the main characteristics of each philosophy within a common framework given by the PMBOK structure.

Traditional methodologies, also called predictive or cascade, seek to impose discipline on the software development process and thus make it predictable and efficient. To achieve this, they are based on a detailed process with emphasis on the planning, typical of other engineering. The main problem of this philosophy is that there are many tasks to follow, and this delays the stage of software development as well as not being easily adaptable to the changes (intrinsic characteristic to software projects).

This paper takes the project management structure of the *Software Extension to the PMBOK Guide* (2013) in order to have a metric and to be able to compare traditional and agile methodologies under analysis. Although PMBOK is not a methodology itself, it is the process standard for project management on which most of the methodologies are developed, so it can be used as comparison pattern. The *Software Extension* is the most recent guide and has allowed to PMBOK (2013) to adapt to the software projects, although it does not imply significant improvements with respect to it. The *ISO 21500* (2013) standard is based on PMBOK so it would be equivalent.

As a basis for the comparison, two methodologies (PRINCE2 and METRICA) will be studied within the group of traditional methodologies, which it does not exclude that they can be used as agile methodologies (in fact, it has been recently released a new PRINCE2 agile extension):

- **PRINCE2** (Projects in a Controlled Environment) (2009) derives from the PRINCE project management method, which was initially developed in 1989 as a UK government standard for IT project management systems. It soon came to be applied regularly outside the ICT environment, both in the British government and in the private sector. The current version is PRINCE2: 2009 Refresh. This methodology was chosen because its worldwide adoption.
- **METRICA** (2000) is a Spanish public methodology for the systematization of the activities that support the software life cycle. It supposes a set of rules, techniques and documents for the development of the software of diverse complexity, size and scope. It has been adapted to the evolution of the technologies that have been emerging, the last version being METRICA V3 (2001). METRICA is based on the *ISO/IEC 12207 Information Technology - Life Cycle Processes software development process model* and *ISO/IEC 15504 Software Process Improvement and Assurance Standards Capability Determination*. It has a structure of processes, interfaces, techniques and practices. In this study, it is being considered the project management interface, which is composed of tasks classified into 3 groups: project start-up (GPI), project tracking (GPS) and project completion (GPF). This methodology was chosen because its wide adoption in Spain. Even that it was developed in the nineteens, it is commonly required nowadays in most of the public administration contracts.

Agile methodologies are a heterogeneous set of methods with more or less rules, principles, recommendations and good practices. They emerged in the 90's and were first called “light” and then agile (Sommerville, 2011). They sought to reduce the probability of failure by not correctly estimating project costs, deadlines and scopes.

Agile software development encompasses software engineering methods based on iterative and incremental development, where requirements and solutions evolve through the collaboration of all stakeholders in the project. Although many times considered novel or revolutionary, it is convenient to remember that the veteran iterative and incremental lifecycle is even older than the cascade life cycle, beginning to be applied to software in the 60’s. There are many methods of Agile development, most minimizing risks by developing software in short periods (iterations).

This type of lifecycle is highly recommended in software projects because it is considered that changing requirements is a natural, inevitable and even desirable aspect of software development. Being able to adapt to changes in requirements at any point in the life of the project is a better and more realistic approach than trying to define all the requirements at the beginning of the project and then investing efforts in controlling any changes in requirements.

Among the many agile methodologies that exist, the following have been selected to be analysed because they have key characteristic elements of agile methodologies and they are popular in the present or in the recent past (Hoda et al., 2017).

- **PRINCE2 AGILE** (2015).
- **Scrum** in Pressman (2010) and Schwaber and Sutherland (2013).
- **FDD** (Feature Drive Development) in Pressman (2010).

A sort introduction of each methodology is included here.

### 1.1 PRINCE2 AGILE

PRINCE2 AGILE was published by AXELOS in 2015. It is a new concept which is a tailored form of PRINCE2, suitable for Agile environments such as Scrum. It does not contain an Agile delivery method, and supports the existing ones instead.

PRINCE2 is one of the most commonly used project management approach in the world, and it is increasingly being used in conjunction with agile. As more organizations adopt agile, the need for specific guidance on how to use PRINCE2 in an agile context has grown accordingly. For this reason, it was developed this new approach.

PRINCE2 AGILE has the same 7 themes, 7 principles (Continued Business Justification, Learn from Experience, Defined Roles and Responsibilities, Manage by Stages, Manage by Exception, Focus on Products and Tailor to Suit the Project Environment) and 7 processes (Starting up a Project, Initiating a Project, Directing a Project, Controlling a Stage, Managing Product Delivery, Managing a Stage Boundary and Closing a Project) than PRINCE2 but they are reinterpreted using agile concepts and techniques. Tag clouds presented in Figures 1 and 2 can help understanding the differences between the PRINCE2 and PRINCE2 AGILE.

**Figure 1: PRINCE2 Tag Cloud (taken from "Understanding PRINCE2 Themes through tag cloud")**


**Figure 2: PRINCE2 AGILE Tag Cloud (figure prepared by the authors)**

Stages are still set based on the management needs of the project, rather than turning into iterations. Each Stage contains one or more "release", and each "release" contains one or more "iterations". Iterations are usually called "timeboxes" in PRINCE2 Agile. Plans are created as usual, with the default responsibilities. Then Work Packages would be the basis for creating the release plans and iteration plans (Team Plans), while their high-level aspects have been defined in the Project Plan and Stage Plans from the beginning. Delivery team members are empowered to decide on minor changes, as long as they do not affect the Category:Management Products directly. Otherwise, the usual change control process would be run, with escalations based on tolerances. Therefore, a limited level of adaptation exists in the delivery layer, and higher-level adaptation would happen in the higher layers, and specially in the Managing a Stage Boundary Process.

Historically, the competing constraints on a project have often been shown graphically as a shape such as a triangle with constraints of time, cost, scope, etc. pulling against each other. PRINCE2 AGILE does not have such a limited view of the variables on a project, as it identifies six “aspects” that need to be controlled and managed: date (time), resources (cost), requirements (scope), quality, risk and benefit. PRINCE2 AGILE does not place emphasis on any of these aspects over and above the others. They are considered as equally significant and to be managed according to the needs of a particular project.

Most of the heritage and thinking behind agile has come from IT and software development, but PRINCE2 AGILE does not assume an IT context. Although it can be used in an IT context, it is not an IT framework or an IT method.

### 1.2 SCRUM

The beginning of Scrum was in Hirotaka and Nonaka (1986). The authors wrote a comprehensive approach that increased the speed and flexibility of new product development. They compared this new approach, in which the phases overlap strongly and the entire process is carried out by a multifunctional team through the different phases. In 1995, Jef Sutherland and Ken Schwaber presented the conference "Scrum Development Process" at OOPSLA (Object-Oriented Programming Systems & Applications conference), their first public appearance. Nowadays it is one of the most used agile methodologies.

Scrum is a framework in which people can work on complex problems, while delivering products of maximum possible value productively and creatively. According to the authors, Scrum is light, easy to understand and extremely difficult to master. Scrum is not a process or a technique for building products but is a framework where various techniques and processes can be employed. The Scrum framework has the following components: scrum teams, roles, events, artifacts, and associated rules. Rules relate events, roles, and artifacts, governing relations and interactions between them. Specific strategies for using the framework are diverse and are not described in the methodology.

The Scrum operation is summarized. At each iteration (called sprint), typically a period of 2 to 4 weeks fixed by the team, the team creates a functional software release. The feature set of an iteration comes from the product backlog, which is a prioritized set of high-level job requirements (called user stories) with their estimate deadlines. Items of this iteration are determined during the iteration planning meeting. During this meeting, the Product Owner informs the team of the items in the product backlog that he wants to complete. The team determines how much of that it can commit to complete during the next iteration. During an iteration, no one can change the backlog, which means that the requirements are frozen for that iteration. The software is started by doing brief daily meetings (daily scrum), typically 15 minutes, for each person in the team to tell his progress and update the sprint backlog. When an iteration is completed, the team shows the software for validation by all stakeholders in the project. Finally, another new iteration would start.

### 1.3 FDD (Feature Drive Development)

FDD was originally conceived by Coad, Lefebvre and De Luca (1999) as a process model for object-oriented programming for Software Engineering. Subsequently, it is published by Palmer and Felsin (2002) where the previous work is expanded and improved, describing adaptive, agile processes that can be applied to medium and large projects. Like other agile methodologies, FDD adopts the following philosophy:

- Emphasizes collaboration between team members.
- The complexity and problems of the project are managed using a decomposition based on features (or functions) which are integrated in successive increments of the software.
- Communication of technical details using verbal, graphic, and written resources.

The development cycle is incremental and it is divided into 5 phases. Each increment (iteration) has two phases: design and construction of one characteristic. A feature is functionality that brings value to the customer that can be done in 2 weeks or less. Each step is explained below.

1. Develop a global model. At the beginning of development, a model is constructed taking into account the vision, context and requirements that the system must have.
2. Build list of features. A list is written summarizing the functionalities that the system must have and this list is evaluated by the client.
3. Planning. The sets of functionalities are sorted according to their priority and dependency, and assigned to the master programmers.
4. Designing. A set of features are selected from the list. It proceeds to design and build the functionality through an iterative process, deciding which functionality will be performed in each iteration.
5. Build. The total construction of the project is proceeded.

## 2 Traditional vs agile approach

Agile and traditional methodologies have two fundamental differences: agile are adaptive (not predictive) and people oriented (not processes oriented). They are two different philosophies of how to manage and develop software projects, although they are not clearly delimited. To get an idea of both conceptions see table 1.

**Table 1: Comparison between tradition and agile vision.**

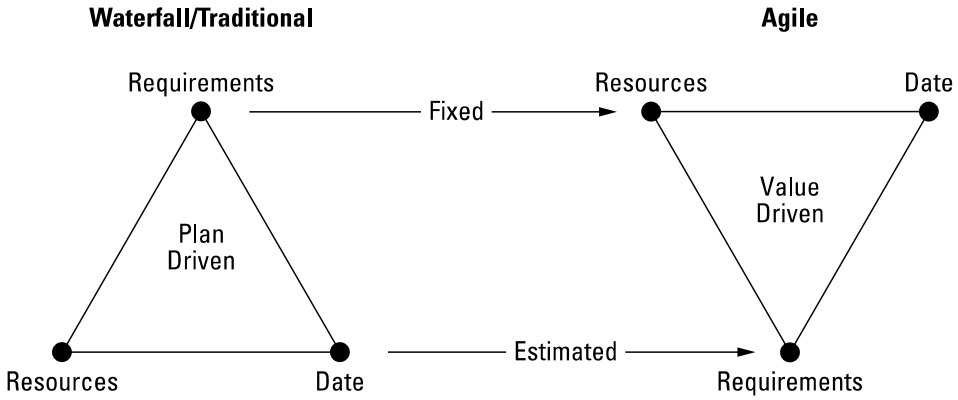
| Issues | Traditional view | Agile view |
|---|---|---|
| **Development life cycle** | Waterfall, spiral, ... | Iterative, evolutional, ... |
| **Style of development** | Anticipatory. | Adaptive. |
| **Requirements** | Knowable, stable and clearly defined and documented | Unknown at first, defined during the Project. |
| **Architecture** | Heavy weight architecture for current and future requirements. | Philosophy You Aren't Gonna Need It (YAGNI) |
| **Management** | Process-centric: command and control. | People-centric: collaboration and leadership. |
| **Documentation** | Detailed, explicit knowledge. | Light (face to face communication), tacit knowledge. |
| **Goal** | Predictability and optimization. | Exploration or adaptation. |
| **Change** | Aversion to change | Embrace change. |
| **Team** | Pre-structured teams. | Self-organizing teams. |
| **Client** | Passive, low involvement. | Proactive as a team member. |

| Issues | Traditional view | Agile view |
|---|---|---|
| **Software development process** | Universal approach to provide predictability and high assurance. | Flexible approach adapted to the particular needs of the project to provide faster development. |
| **Measure of success** | Conformance to plan. | Business value delivered. |

In The Standish Group (2015) statistics are shown in which 39% of project are successful with agile approach and only 11% of project are successful with waterfall. Nevertheless, this study does not have enough projects to be conclusive. Serrador and Pinto (2015) wrote a specific paper of a quantitative analysis of agile project success. Both approaches working together is not unusual such as is mentioned in Špundak (2014) and Binder, Aillaud and Schilli (2014).

In a graphical way, the project management triangle has different meanings for both visions (Figure 3). In the traditional view the requirements are fixed following a plan-driven in which the date and resources are estimated to meet the plan. By contrast, in the agile vision the date and resources are fixed following a value-driven and the project is developed according to changing requirements agreed upon by all stakeholders periodically.

**Figure 3: The project management triangle.**




## 3 Comparison of methodologies

This section compares the methodologies introduced in first section. As already mentioned, all are called methodologies when some of them (especially the agile ones) are rather a dispersed set of principles, values and good practices. However, the comparison has interest in explaining how they approach the different areas, if they do, of project management.

Tables 2 to 5 compare the methodologies described using PMBOK as a reference for comprehensively cover project management. As previously stated, PMBOK is not a methodology but it is the process standard for project management on which most of the methodologies are developed and it is the most recognized process set. It has 10 areas of knowledge and 47 processes. PRINCE2 and METRICAV3 have more specific rules than PRINCE2 AGILE, SCRUM and FDD. In general, the strictest methodology is PRINCE2 and the less Scrum. PRINCE2 AGILE has a good balance between both approaches.

Stakeholders and procurement areas are not important formally for all methodologies and it is significant that the cost is irrelevant for FDD.

**Table 2: Table comparative of PRINCE2, METRICAV3, PRINCE2 AGILE, Scrum and FDD. (Integration & Stakeholders Knowledge Areas)**

| AREAS AND PROCESS OF PMBOK | | PRINCE2 | METRICAV3 | PRINCE2 AGILE | SCRUM | FDD |
|---|---|---|---|---|---|---|
| **Integration** | Develop project charter.<br>Develop project management plan<br>Direct and manage project work.<br>Monitor and control project work.<br>Perform integrated change control.<br>Close project. | Starting up a project: focus the project and make a summary.<br>Starting up a project: preliminary business case.<br>Initiating a project: create the project plan.<br>Initiating a project: strategy configuration management.<br>Directing a project: to authorize the opening.<br>Directing a project: authorize the project.<br>Directing a project: authorize closure project.<br>Closing a project. | GPI 2.5: Submission and acceptance of the overall project planning.<br>GPS 3.1: Tracing of tasks.<br>GPS 7.1: Approval of the solution.<br>GPS 9.1: Changing registration requirements.<br>GPS 11.3: Development of the monitoring report.<br>GPS 13.1: Verification of internal acceptance.<br>GPF 1.1: Inclusion in historic projects.<br>GPF 1.2: Archive documentation project management. | Starting up a project [Chapter 17]: vision, product roadmap.<br>Initiating a project [Chapter 17]: product backlog, The Cynefin framework.<br>Directing a project [Chapter 18].<br>Controlling a stage [Chapter 19]: release, release backlog, release retrospective.<br>Managing a stage boundary [Chapter 21]: as for Controlling a stage.<br>Closing a project [Chapter 22]: project retrospective | Verification of management approval and funding during planning phase.<br>Validation of development tools and infrastructure during planning phase.<br>Strong change management procedure with product and sprint backlog.<br>Refinement of system architecture to support changes. | Development of the overall system model. |
| **Stakeholders** | Identify stakeholders.<br>Plan stakeholder management.<br>Manage stakeholder engagement.<br>Control stakeholder engagement. | | | | | |

**Table 3: Table comparative of PRINCE2, METRICAV3, PRINCE2 AGILE, Scrum and FDD. (Scope & Time Knowledge Areas)**

| AREAS AND PROCESS OF PMBOK | | PRINCE2 | METRICAV3 | PRINCE2 AGILE | SCRUM | FDD |
|---|---|---|---|---|---|---|
| **Scope** | Plan scope management.<br>Collect requirements.<br>Define scope.<br>Create Work Breakdown Structure (WBS).<br>Validate Scope.<br>Control Scope. | Initiating a project: initial project documentation.<br>Initiating a project: refine the business case.<br>Managing a boundary stage: project plan update.<br>Managing a boundary stage: upgrade business case.<br>Controlling a stage: review state of the stage.<br>Controlling a stage: check package status job. | GPI 1.1: Identification of elements to develop.<br>GPI 1.2: Calculation of effort.<br>GPI 2.1: Selection of development strategy.<br>GPI 2.2: Selecting the structure activities, tasks and products.<br>GPS 1.1: Assignment task.<br>GPS 5.1: Registering the change request requirements.<br>GPS 6.1: Study requirements change request.<br>GPS 6.3: Study of alternatives and proposed solution. | Change [Chapter 14]: the feedback loop<br>Managing product delivery [Chapter 20]: sprint, sprint backlog, sprint review, retrospective, Kanban, Lean Startup. | Perform domain analysis for building domain model.<br>Development of a comprehensive product backlog list.<br>Development of a comprehensive product sprint backlog.<br>Definition of the functionality that will be included in each release.<br>Selection of the release most appropriate for immediate development.<br>Review of progress for assigned backlog items. | Perform domain analysis for building domain model (step 1).<br>Build features list, subject areas (step 2). |
| **Time** | Plan schedule management.<br>Define activities.<br>Sequence activities.<br>Estimate activity resources.<br>Estimate activity durations.<br>Develop schedule.<br>Control schedule. | Managing a boundary stage: planning next stage.<br>Managing a boundary stage: final report stage.<br>Controlling a stage: take corrective action. | GPI 2.3: Setting the schedule milestones and releases.<br>GPI 2.4: Detailed planning of activities and resources.<br>GPS 11.1: Update tasks. | Plans [Chapter 12]: agile estimation.<br>Progress [Chapter 15]: burn charts, information radiators. | Definition of the delivery date and functionality for each release.<br>Monthly iterations. | Determine development sequence (step 3).<br>Assign business activities to chief programmers (step 3).<br>Assign classes to developers (step 3).<br>Chief programmer work package. |

**Table 4: Table comparative of PRINCE2, METRICAV3, PRINCE2 AGILE, Scrum and FDD. (Cost, Quality, Human Resources & Communications Knowledge Areas)**

| AREAS AND PROCESS OF PMBOK | | PRINCE2 | METRICAV3 | PRINCE2 AGILE | SCRUM | FDD |
|---|---|---|---|---|---|---|
| **Cost** | Plan cost management.<br>Estimate costs.<br>Determine budget.<br>Control costs. | Initiating a project: define controls for the project.<br>Controlling a stage: report important aspects. | GPS 6.2: Impact of change request requirements.<br>GPS 8.1: Estimated effort for change.<br>GPS 8.2: Planning changes.<br>GPS 10.1: Checking the task.<br>GPS 11.2: Getting extrapolation. | Business case [Chapter 9]: value and benefits. | Estimation of release cost, during planning phase. | |
| **Quality** | Plan quality management.<br>Perform quality assurance.<br>Control quality. | Initiating a project: quality management strategy.<br>Managing product delivery: delivery work package.<br>Managing product delivery: acceptance work package. | Interface quality assurance. | Quality [Chapter 11]: planning and control, Test-Driven Development (TDD), Behaviour-Driven Development (BDD), refactoring, … | Distribution, review and adjustment of the standards with which the product will conform.<br>Design review meeting.<br>Sprint planning meeting.<br>Sprint review meeting.<br>Daily scrum. | Emphasis on quality with an incremental strategy of development.<br>Review meetings (all steps).<br>Code inspection and unit test (Step 5). |
| **Human Resources** | Plan human resource management.<br>Acquire project team.<br>Develop project team.<br>Manage project team. | Starting up a project: appoint the executive and the project manager.<br>Starting up a project: design and appoint the project team. | | Organization [Chapter 10]: servant leadership and incorporate the wider customer view and the product owner role. | Appointment of project team per release.<br>Team participation in sprint meetings.<br>Team participation in daily scrums. | Appoint modelling team (step 1).<br>Appoint feature list team (step 2).<br>Appoint planning team (step 3).<br>Appoint feature team (step 3). |
| **Communications** | Plan communications management.<br>Manage communications.<br>Control communications. | Initiating a project: management strategy communication. | GPS 2.1: Report to the project team.<br>12.1 GPS: Internal tracing meeting. | Rich communication [Chapter 26]: workshops. | Design review meeting.<br>Scrum meeting.<br>Sprint planning meeting.<br>Sprint review meeting.<br>Communication of standards to the project team. | Review meetings (all steps). |

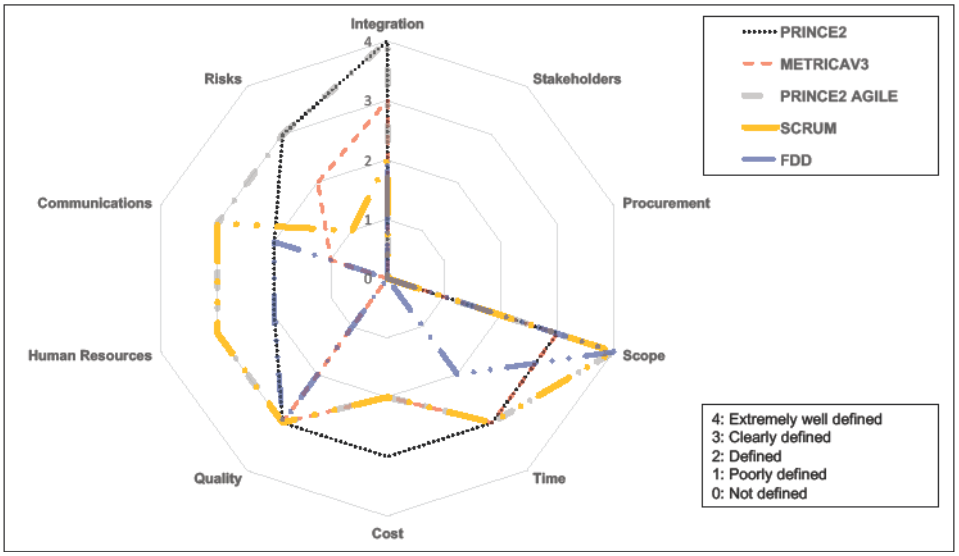
**Table 5: Table comparative of PRINCE2, METRICAV3, PRINCE2 AGILE, Scrum and FDD. (Risk & Procurement Knowledge Areas)**

| AREAS AND PROCESS OF PMBOK | | PRINCE2 | METRICAV3 | PRINCE2 AGILE | SCRUM | FDD |
|---|---|---|---|---|---|---|
| **Risks** | Plan risk management.<br>Identify risks.<br>Perform qualitative risk analysis.<br>Perform quantitative risk analysis.<br>Plan risk responses.<br>Control risks. | Initiating a project: risk management strategy.<br>Controlling a stage: capture and examine problems and risks.<br>Controlling a stage: report problems and risks. | GPS 4.1: Analyse impact.<br>GPS 4.2: Proposed solution of the problem.<br>GPS 4.3: Record the incidence. | Risk [Chapter 13]: risk management procedure, Management of Risk, Risk burn-down charts, Spiking, prototyping, proof of concepts, experiments. | Initial assessment of risks during pregame.<br>Risk review during review meetings. | |
| **Procurement** | Plan procurement management.<br>Conduct procurements.<br>Control procurements.<br>Close procurements. | | | | | |

Figure 4 is a radar chart which simplifies graphically the comparative table. It is a graphical method of displaying the degree of detail in the definition of the methodologies for each area of PMBOK. Each methodology is assigned an integer value between 0 and 4 for each area. If the methodology does not treat the area in question, it is assigned to 0. On the contrary, if the methodology exhaustively defines the area with processes, techniques, concepts, descriptions, examples or any other explanatory element, is assigned to 4. Note that it is an assessment with a subjective degree.

It is observed as PRINCE2 AGILE have a good balance of each area and possibly METRICAV3 is the worst of them because human resources and communications are not important for it.

**Figure 4: Degree of detail in the definition of the methodologies for each area of PMBOK.**



## 4 Conclusions

Difficulties have been encountered in comparing the different methodologies. This is because it is not trivial to find common elements to all methodologies and, in addition, to define them completely. It is complex to find a basis when comparing different methodologies.

The boundaries between methodologies are not clearly delimited. Traditional methodologies have characteristics of agile and vice versa. However, it is seen how PRINCE2 and METRICAV3 emphasize the area of integration. PRINCE2 AGILE, Scrum and FDD have in mind the importance of the scope. PRINCE2 AGILE and Scrum emphasize human resources and communications.

Maybe PRINCE2 AGILE is the border between PRINCE2 and METRICAV3 and Scrum and FDD because it is the most recent methodology and it has features of all of them.

It is remarkable that stakeholders and procurement management are not covered by any methodology. Developing and extension for some of these methodologies including processes for managing these points could be performed as future work.

In conclusion, there is no single and perfect methodology that guarantees success to any type of project. In the best case, there will be a methodology that best suits the type of project, client, company and team. The complexity is to get the midpoint recommended for each project between agility and planning.

## References

*A guide to the project management body of knowledge (PMBOK guide)* (2013). Project Management Institute. 5th edition. ISBN-13: 978-1-935589-67-9.

Binder, J., Aillaud, L., & Schilli, L. (2014). The project management cocktail model: An approach for balancing agile and ISO 2150, 27th IPMA World Congress, *Procedia - Social and Behavioral Sciences 119*, 182-191.

Bourque, P., & Fairley, R.E (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, version 3.0. IEEE Computer Society. ISBN-13: 978-0-7695-5166-1.

Coad, P., Lefebvre, Eric, & De Luca, J. (1999). *Java Modeling in Color with UML*, Prentice Hall. ISBN-10: 0-13-011510-X.

Hirotaka, T., & Nonaka, I. (1986). The New New Product Development Game. *Harvard Business Review 64*, no. 1.

Hoda, R., Salleh, N., Grundy, J., & Tee, H. M. (2017). Systematic literature reviews in agile software development: A tertiary study, *Information and Software Technology 85*, 60-70.

*Managing Successful Projects with PRINCE2* (2009). The Stationery Office. ISBN: 978-0-11-331059-3.

*Methodological Guidelines for the Development of ICT Projects, Métrica Version 3*, (2000). Ministry of Public Administration of Spain.

Palmer, S., & Felsin, J. (2002). *A Practical Guide to Feature Driven Development*, Prentice Hall. ISBN.13: 978-0-13-067615-3.

Pressman, Roger (2010). *Software engineering: a practitioner's approach*, 7th edition. Mc Graw Hill. ISBN: 978-0-07-337597-7.

*PRINCE2 AGILE* (2015). The Stationery Office. ISBN: 978-0-11-331467-6.

Schwaber, K., & Sutherland, J. (2013). *The Definitive Guide to Scrum: The Rules of the Game*.

Serrador, P., & Pinto, J. (2015). Does Agile work? - A quantitative analysis of agile project success, *International Journal of Project Management 33*, 1040-1051.

*Software Extension to the PMBOK Guide* (2013). Project Management Institute. ISBN-13: 978-1-6282-5013-8.

Sommerville, I. (2011). *Software Engineering*, 9th edition. Addison-Wesley. ISBN-13: 978-0-13-703515-1.

Špundak, M. (2014). Mixed agile/traditional project management methodology – reality or illusion?, 27th IPMA World Congress, *Procedia - Social and Behavioral Sciences 119*, 939-948.

*The Standish Group, CHAOS Manifesto 2013* (2013).

*The Standish Group, CHAOS Manifesto 2015* (2015).

*UNE-ISO 21500, Guidance on project management* (2013). AENOR.

*Understanding PRINCE2 Themes through tag cloud* (n.d.), accessed 20/04/2017 from https://es.pinterest.com/pin/518758450797252773/