Using Scrum Together with UML Models: A Collaborative University-Industry R&D Software Project

Nuno Santos[1(✉)], João M. Fernandes[1,2], M. Sameiro Carvalho[1,3], Pedro V. Silva[4], Fábio A. Fernandes[1], Márcio P. Rebelo[1], Diogo Barbosa[1], Paulo Maia[1], Marco Couto[4], and Ricardo J. Machado[1,5]

[1] ALGORITMI Research Center, School of Engineering, University of Minho, Guimarães, Portugal
nuno.a.santos@algoritmi.uminho.pt
[2] Department of Informatics, University of Minho, Braga, Portugal
[3] Department of Production and Systems, University of Minho, Guimarães, Portugal
[4] Bosch Car Multimedia Portugal S.A., Braga, Portugal
[5] Department of Information Systems, University of Minho, Guimarães, Portugal

Abstract. Conducting research and development (R&D) software projects, in an environment where both industry and university collaborate, is challenging due to many factors. In fact, industrial companies and universities have generally different interests and objectives whenever they collaborate. For this reason, it is not easy to manage and negotiate the industrial companies' interests, namely schedules and their expectations. Conducting such projects in an agile framework is expected to decrease these risks, since partners have the opportunity to frequently interact with the development team in short iterations and are constantly aware of the characteristics of the system under development. However, in this type of collaborative R&D projects, it is often advantageous to include some waterfall practices, like upfront requirements modeling using UML models, which are not commonly used in agile processes like Scrum, in order to better prepare the implementation phase of the project. This paper presents some lessons learned that result from experience of the authors in adopting some Scrum practices in a R&D project, like short iterations, backlogs, and product increments, and simultaneously using UML models, namely use cases and components.

Keywords: Agile · Scrum · UML · Research projects

1 Introduction

When research and development (R&D) projects are executed within industrial environments, project management commonly follow plan-centric processes, like waterfall [1], the spiral [2] or the Rational Unified Process (RUP) [3]. All these processes allow research to performed and clearly refine the requirements before moving to the implementation phase.

However, requirements are not always clear at the beginning of projects from R&D nature, which, when using plan-centric processes, makes very difficult to define an early schedule to any software development project due to the high uncertainty that

O. Gervasi et al. (Eds.): ICCSA 2016, Part IV, LNCS 9789, pp. 480–495, 2016.
DOI: 10.1007/978-3-319-42089-9_34

characterizes these types of projects. This schedule uncertainty typically occur in early stages of the project when a high effort in domain and technological research tasks is required. When R&D projects are executed in an industrial environment, projects face many challenges as timeliness, addressing the needs of stakeholders, rigor and access [4]. In such industrial environment, agile software development processes bring many advantages, since they are characterized by frequent interactions and collaboration with clients [5]. Agile processes are based in self-organized teams that perform the development tasks. These processes are divided in small iterations, where software systems are periodically assessed, in order to detect/solve possible problems as soon as they emerge. This approach works fine when future requirements are largely unpredictable [6]. Within these processes, eXtreme Programming (XP) [7] and Scrum [8] are among the most popular methodologies [9].

This paper describes how Scrum was adapted by a newly-formed team to develop a software system in a R&D project, called Inbound Logistics Tracking System project (hereafter referred as iFloW project) [10], which is described in this paper as a demonstration case. The Scrum process adaptation was based by performing upfront requirements modelling before the implementation phase (more common in waterfall approaches), instead of starting directly in implementing the software within the small iterations. For the requirements modelling, UML diagrams were modelled, namely use cases and component diagrams. The choice of using UML models was based by their wide acceptance, but any other models that bring upfront knowledge of software requirements could be used. The iFloW project, as its name refers, relates to logistics domain, and was mainly focused in integration with third party logistics (3PL) service providers and integrating Radio Frequency Identification (RFID) technology [11, 12], Global Positioning System (GPS) technologies [13], and an integrated web-based RFID-Electronic Product Code (EPC) compliant logistics information system [14].

The R&D context required domain research tasks in an initialization phase, as well as technological-related research and third-party collaboration during the implementation phase. This phase was performed in form of Sprints, which are small cross-functional development cycles and are used in agile frameworks like XP and Scrum. Software development projects with important integration and interoperability issues require additional concerns when compared with typical agile processes, since implementation requires prior studies related to the technologies involved (except if the team has solid experience with those technologies) and collaboration with third-party entities.

This paper is structured as follows: Sect. 2 presents the related work; Sect. 3 describes the collaborative University-Industry context; Sect. 4 describes our proposed Scrum adaptation, which integrates typical agile practices together with UML models; in Sect. 5 is presented the lessons learned of the process adoption; conclusions are presented in Sect. 6.

2 Related Work

The adoption of agile frameworks commonly found in literature is basically grounded in practical experience. There has been an increased interest in performing some empirical studies in agile software development [15]. Agile principles (namely those from XP) are

introduced within a research work related to enterprise architecture development projects and software development projects [16]. Abrahamsson [17] notes that while more research was being done, agile methods were still driven by consultants and practitioners, that there was a lack of research rigour, and that researchers needed to address core questions such as what constitutes agility, how agile methods can be extended, and how mature teams use agile methods. Barroca *et al.* [4] state that collaboration between industry and research in agile methods allowed building trust and regular feedbacks, appropriate contracts at the beginning of the project, and learning experience for both teams. However, to the best of our knowledge, there are no research works in literature that describe the adoption of agile frameworks within R&D projects.

At first glance, agile processes seem to be only suitable for small teams operating in a local environment. The iFloW software system (the main deliverable of the iFloW project) was designed to perform in an environment where the main inputs are provided by integrated systems from suppliers or forwarders. Therefore, agile processes must consider implementations that refer specifically to integration, like the research from Niemelä and Vaskivuo for developing a middleware [18]. Välimäki and Kääriäinen propose an organizational pattern for these cases [19].

Projects where significant interoperability issues strongly rely upon dependency and communication with third-parties, like the case of the iFloW project, commonly involve development effort by distributed teams, which can be highly challenging. In fact, he coordination between different teams in agile development is one of the top concerns identified in [20]. In agile development, some techniques for adapting events, actors and artifacts arise so that both distributed and dispersed teams may work in the same product development [21]. Some examples within Scrum are Isolated Scrums, Scrum of Scrums [22] and Totally Integrated Scrum [23]. In these distributed environments, good requirements modeling, including using UML models, can be advantageous. It is the case of [24], where the identification of contact points where there is a need for synchronizing efforts within distributed Scrums and effort dependencies can be performed using artifacts like an architecture. Managing the distributed work is not an exclusive concern within Scrum framework. An example for these practices within the XP framework is Industrial XP [25].

3 The Collaborative University-Industry R&D Software Project

3.1 The HMI-Excel Program

The Human-Machine Interface Excellence (HMIExcel) project [26] was sponsored by the consortium between University of Minho (UMinho) and Bosch Car Multimedia Portugal (Bosch). It was designed in order to tackle scientific and technological challenges of Bosch Braga and to obtain recognition of this unit as an International Competence Centre in Human-Machine Interface (HMI). The project was 28 months long, starting in March 2013 and ending in June 2015. Although HMIExcel for the funding body is seen as a project, its complexity and uncertainty led the consortium (UMinho and Bosch) to manage it as a program, *i.e.*, a set of projects that are somehow related and contribute to the same goal [27]. The main goal for the HMIExcel was to promote

the investment in R&D, for developing and producing future mobility concepts in the automotive domain, where academia and industry worked together aiming innovative products and processes. These solutions were developed to meet production needs and preparing Bosch to respond to the challenges from the Fourth Industrial Revolution (Industry 4.0).

The HMIExcel project divided itself into thirteen research lines, each one tackling one specific challenge. Each research line followed an independent course, as all projects were coordinated by one team focused on the objectives and deliverables expected in each individual line. One of those research lines, iFloW, is used in this paper as a case study analysis for purpose of validation of our approach.

3.2 The iFloW Project

iFloW is an R&D project that aims at developing an integrated logistics software system for inbound supply chain traceability. iFloW is a real-time tracking software system of freights in transit from the suppliers to the Bosch plant, located in Braga. The main goal of the project is to develop a tracking platform that by integrating information from freight forwarders and on-vehicle GPS devices allows to control the raw material flow from remote (Asian) and local (European) suppliers to the Bosch's warehouse, alerts users in case of any deviation to the Estimated Time of Arrival (ETA) and anticipates deviations of the delivery time window.

Figure 1 illustrates the architecture of the iFloW software system, namely the integration of the iFloW main server with other systems. The architecture allows depicting the significant interface protocols and systems that were involved. The system uses information that is provided by different sources, like GPS, Personal Digital Assistant

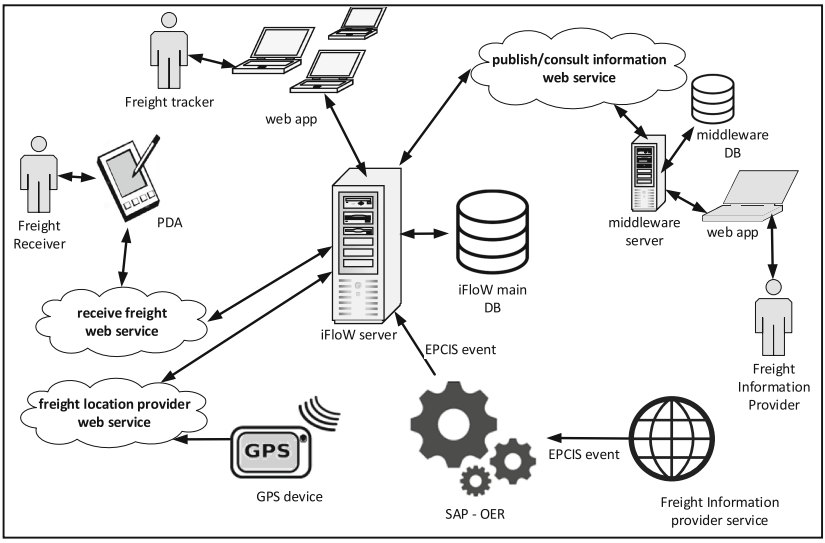

Fig. 1. The architecture of the iFlow software system

(PDA) devices or SAP-OER (Object Event Repository). The '*middleware server*' was developed aiming to standardize the way information between Bosch and providers is exchanged. Then, the '*iFloW server*' executes all business logic, where Bosch employees access all features via a web-based user-interface (web app).

In the case of the iFloW project, this collaboration UMinho and Bosch was based in the premise where Bosch mainly performed as a software customer and UMinho as a contracted software development entity. The core iFloW team was composed of nine collaborators with multidisciplinary backgrounds:

- Bosch:
  - one Product Owner, that was representing other eight elements from the Logistics department, that formally dictated the requirements.
  - one member of the IT department, responsible for validating that each developed product increment could be easily integrated within Bosch information system;
- UMinho:
  - three R&D coordinators, with the role of assuring that the scientific rigor (from both the system and the software development process) and deadlines of the project are met;
  - four software developers with methodological and technological competences (like analysis, requirements, design, database modeling, programming, testing, deployment, etc.).

The entire software development was performed within Bosch's premises, where the iFloW team elements (in exception of R&D Coordinators) were located on a daily basis. The elements from UMinho had no previous knowledge of the domain (in this case, logistics), so the team decided that the project kicked-off by gathering and documenting requirements in a waterfall-based approach.

After the requirements engineering was performed, and since iFloW aimed developing a software system for an industrial context, the team decided to follow the Scrum framework as the iterative approach for the implementation phase. This phase was performed by development iterative cycles in form of Scrum sprints. Based in incremental software deliveries, both UMinho and Bosch could manage their project's expectations.

As a collaborative University-Industry R&D software project, the previously presented roles are slightly different from the roles defined by the Scrum framework (namely, Product Owner, Scrum Master and Development Team) [28], however easily mapped, as depicted in Table 1.

The Product Owner (PO) was the element responsible for dictating the requirements, as previously stated, but also participated in meetings with UMinho R&D coordinators and Software Developers for project monitoring (the Project Manager role is not included within a Scrum team). Thus, it is directly mapped with a typical Scrum PO. Additionally, the PO performed as a contac point for distributed development with third-party service provider entities that composed the ecossystem. The fact that Bosch's information system (not only from Braga plant, but from all Bosch Car Multimedia worldwide plants) is managed by the Bosch Car Multimedia Group, in Germany, and that the system under development will be included in Bosch's information system, also

includes Bosch's IT department from Germany in the project's ecossystem. Thus, the PO also performed as a contact point in the negotiation between the iFloW team and the IT department form Germany, regarding the compliance requirements from security and policy issues of Bosch Car Multimedia Group.

The Bosch IT element was responsible for validating the code developed during sprints, in order to be integrated within the existing information systems. He interacts directly with the software developers from UMinho. For this reason, he is considered as part of the development team. However, he was responsible for representing the IT department from Braga during the negotiations with IT department form Germany and with third-party service provider entities. So, within these tasks he also performed the role of a typical Scrum PO.

The R&D Coordination was composed by three professors from UMinho, from the fields of software engineering and also logistics. They were responsible for assuring that the academic concerns of these types of R&D projects were met, where their main concerns was to assure development beyond the state of the art, and to coordinate and monitor the development of the project's deliverables and scientific papers. This responsibility is not mapped to any role in Scrum. Since Bosch had never adopted agile software development processes, they were responsible for training and control the adoption of Scrum practices by the project team, when the project entered the implementation phase. For this responsibility in this phase, these elements are mapped as typical Scrum Masters.

The Software Developers performed tasks related with software engineering (e.g., analysis, requirements, architecture, layout design, coding, testing, deployment, etc.) thus, their responsibility was directly mapped with a Scrum development team.

Table 1. Mapping between iFloW roles and typical Scrum roles

Scrum Role / iFloW Role	Product Owner	Scrum Master	Development Team
Bosch			
Product Owner	✓		
Bosch IT	✓		✓
UMinho			
R&D Coordinators		✓	
Software Developers			✓

4 Using Scrum with UML Models

This section presents the software process and the required adaptations that were due to the research nature of the project and the required integration with third-party services. An overview of the process is depicted in Fig. 2. The process is composed of three phases: Initialization, Implementation, and Deployment.

The initialization phase includes typical activities from domain engineering, requirements engineering and design. In this phase, use cases diagrams and the component diagram were modelled. The implementation phase was performed in small iterations

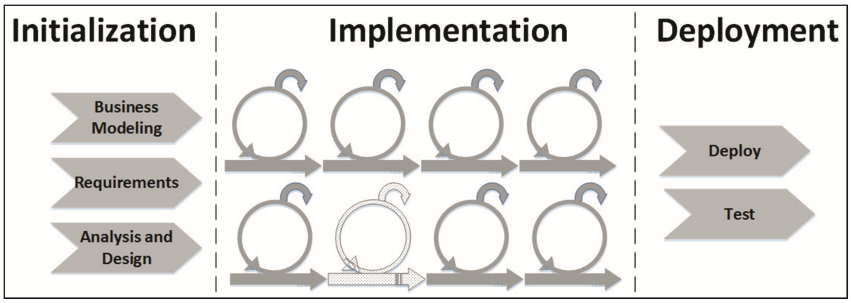

Fig. 2. Overview of the process executed in the iFloW project

and incremental releases in the form of Scrum sprints. This phase is similar to typical Scrum process. The UML models from the previous phase were just verified for changes, except for some new use cases referring newly discovered requirements. It should be noted that Fig. 2 depicts seven cycles performed like typical Scrum sprints (circles with filled border), and one cycle that rather addressed architectural refactoring (circle with dashed border). This different cycle may not be required in other R&D projects if the architecture is stable during the whole project. Finally, the Deployment phase is similar to the Transition phase of RUP. This phase included modeling a UML deployment diagram. This phase is not detailed in this paper due to size reasons.

4.1 Initialization Phase

Within the initialization phase, the objective was to develop a product backlog artifact in order to start the development phase in the form of sprints that, due to the perceived complexity of the project, was delivered together with widely accepted forms of requirements documentation (the overall process is depicted in Fig. 3).

Since iFlow is a R&D software project, some research activities were conducted throughout the project and the research outputs had to be documented. This project kicked–off as a typical waterfall process, and initial tasks were conducted to specify the project scope, to characterize the domain and the organization's (logistics–related) activities, to define terms, and to analyze flows, legacy software and data.

The organization's logistics–related processes and current gaps were documented in a report designated as '*As-Is report*'. Then, the requirements were elicited, formally specified in the form of UML use cases (see Fig. 4), a list of quality (non-functional) requirements and in a first version of the logical architecture (UML component diagram). This set of requirements was documented in a report designated as '*To–Be report*' (which was constantly updated as the implementation went along). Both use case models (especially the '*To-Be*') were used as basis to define a '*Product Backlog*'. This differs from other agile frameworks where, for instance, although complementary and able to be used together [8], in Scrum backlogs are composed of user stories. A user story is a customer-centric characterization of a requirement, containing only the information needed for the project developers to see clearly what is required to implement [29]. Use cases are used in backlogs in [30, 31].

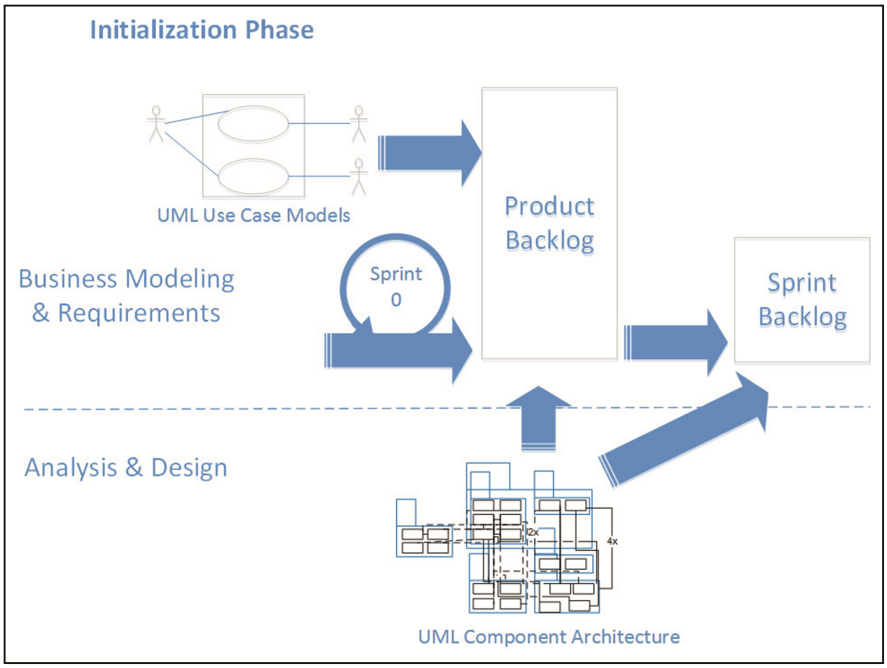

Fig. 3. Initialization phase

The use case diagram illustrated in Fig. 4 shows the overall use case model of the iFloW project. For the purpose of this paper, the use cases are not described, as the diagram is used only for demonstration purposes. Each of the use cases were functionally decomposed, which resulted in a total of 90 lower level use cases.

The Initialization phase ends with a Sprint 0 ceremony. Most of the technological research was performed during this ceremony, prior to the implementation in the following sprints. Like in a typical Sprint 0, each item (use case) was prioritized by its perceived value from stakeholders, in this case by using MoSCoW prioritization technique [32]. For this task, the PO provided input on the perceived business value of the requirement. On the other hand, R&D Coordinators identified critical implementation issues.

Also, each use case was estimated related to a quantitative effort for its implementation, where it was defined that the effort for each sprint corresponds to a total of 20 points (which resulted in approximately five points per week) as basis for distribution of these points per use case. This was a decision made by PO and R&D Coordinators. A commonly used technique is use case points [33], however in this project this technique was not used. Rather, and following a comparative technique similar to a planning poker [34]. Finally, based in efforts and prioritizations, the remaining step to define the '*Sprint Backlogs*' (which use cases from the '*Product Backlog*' to implement during the sprint) and to plan them.

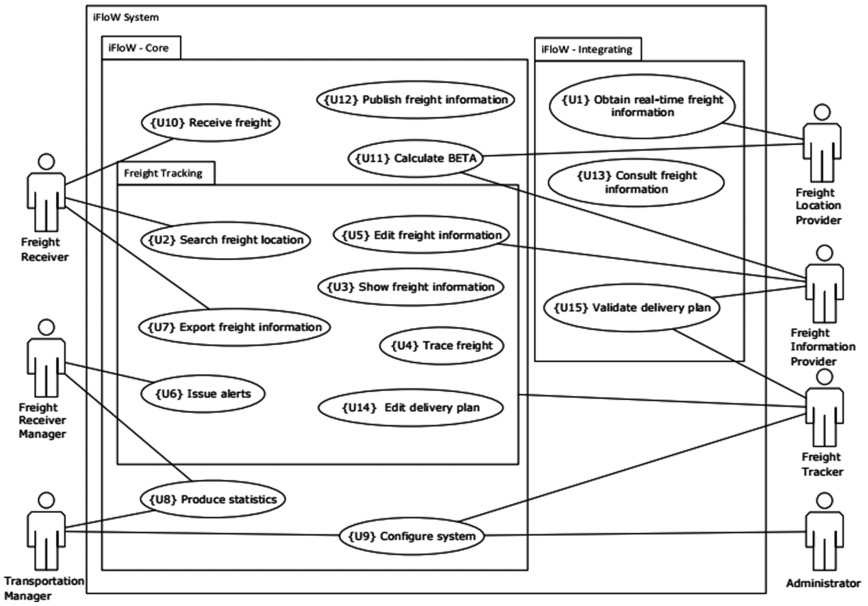

Fig. 4. Use case diagram

4.2 Implementation Phase

Within the implementation phase, the use cases from the '*Product Backlog*' were implemented iteratively and incrementally during eight four-week Scrum sprints. In this phase, typical Scrum iterations were performed, where each '*Sprint Backlog'* is a selected subset from the '*Product Backlog*' (see Fig. 5).

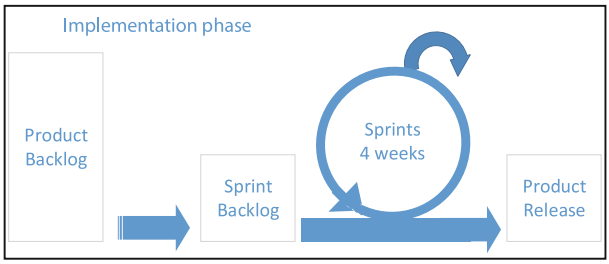

Fig. 5. The implementation phase

In Fig. 6 is depicted an example of a '*Sprint Backlog'* tracking sheet, which allowed monitoring through a breakdown chart. In this sheet, use cases were included until the total effort reached 20 points. Then, the use case implementation was monitored throughout each of the four weeks that compose the Sprint, by registering the remaining units (from the initial estimation) that were implemented during that week. This way,

the tracking sheet allowed monitoring the implemented functionalities (and thus, the value delivered to the organization), but also if the Sprint's initial planned target (column SPx Target) was obtained.

Sprint #4Tracking Sheet

Project	iFloW
Sprint #	4
Start date	05/01/14

				week			
				1	2	3	4
Task ID	**Description**	**SP4 Target (%)**	**Initial estimate (units)**	**Units Left**	**Units Left**	**Units Left**	**Units Left**
{U9.5}	Configure delivery plan	100%	6	5	3	1	1(b)
{U14}	Edit delivery plan	100%	3	2	1	1	1(b)
{U15}	Validate delivery plan	100%	2	2	2	1	1(b)
{U10}	Receive Freight (TS_SP3_U10_04)	100%	1	1	1	1	1
{U1.1.1}	Obtain freight information from *Forwarder A* from European port	25%	2	2	2	1	0
{U1.1.2}	Obtain freight information from *Forwarder A* from Asian port	25%	2	2	2	1	0
{U12}	Publish freight information	25%	2	2	2	1	0
{U13}	Consult freight information	25%	2	2	2	1	0
	Total estimate units		20				
	Remaining units (actual)			18	15	8	1
	Remaining units (ideal)			15,0	10,0	5,0	0,0

Fig. 6. Example of a sprint backlog based in use cases

Each sprint has a standard planning and structure consisting of several milestones, previously negotiated by the project members:

- **Sprint development**: lasts four weeks, and is allocated to the development of the items from the '*Sprint Backlog*';
- **Sprint Monitoring meeting**: takes place in second week to show sprint progress and monitor sprint tasks. The attendees are the Product Owner, R&D coordination and development team;
- **Sprint Verification and Validation (V + V) meeting**: takes place in the fourth (*i.e.*, last) week and the goal is to test and validate the requirements implemented by the development team. The attendees are the Product Owner, the development team, a member of the Bosch IT department, and an assigned Product User from Bosch. In each Sprint V + V meeting, the Product User was assigned a different user from

Logistics department so the performed tests could encompass different insights from the organization. During the sprint, if any requirement (use case) is moved to a next sprint due to a given constraint and will not be presented in this meeting, the team is notified;

- **Sprint Closure and Planning meeting**: takes place at most two days after the Sprint V + V meeting, and the attendees are the Product Owner, the R&D coordination, a member of the Bosch IT department and the development team. It is similar to a Sprint Retrospective and a Sprint Planning meeting from typical Scrum, performed within the same meeting. The main goal is to analyze the progress of the implementation phase, by assessing the percentage and completion of the use case implementation and thus updating the burndown chart. If applicable, short rework actions (depicted from the Sprint V + V) are approved to perform until the end of the sprint. Additionally, the next Sprint is planned, resulting in the construction of the 'Sprint Backlog' artifact;
- **Sprint Rework meeting**: takes place the day after the Sprint Closure meeting. After Sprint V + V, some rework actions can arise due to a suggestion by the verification and validation team. If applicable, the development team has to implement these rework actions until the end of the sprint. The Sprint Rework meetings are used to validate the rework actions performed. The attendees are the assigned Product Users, Product Owner, a member of the Bosch IT department and the development team.

For implementing each use case, the team performed tasks involving several software engineering disciplines. In this paper, we use the terminology from RUP's disciplines (only for demonstration purposes) to depicts the type of effort involved within the Sprints.

Occasionally, the team performed spikes (originally defined within XP), a technique used for activities such as research, innovation, design, investigation and prototyping. With spikes, one can properly estimate the development effort associated with a requirement or even to better understand a requirement. The use of spikes in the iFloW project justifies the inclusion of the Requirements discipline in the each Sprint, as shown in Fig. 7.

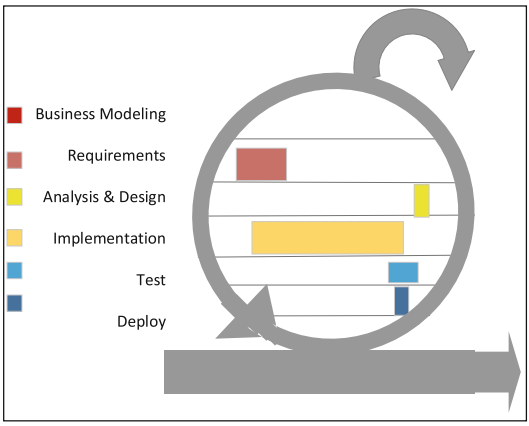

Fig. 7. The performed disciplines within the sprints (Color figure online)

In all sprints, the need for updates to the logical architecture was assessed (within the Analysis & Design). Afterwards, the typical disciplines were carried out within the sprints: Implementation, Testing and Deploy. These spikes were, in their majority, originated from middleware-based use cases (for instance, related to integration with third-party service providers, GPS, EPCIS or SAP-OER). Within the remaining use cases, the Requirements discipline was not required. Thus, in comparison with the disciplines included in Fig. 7, the Sprint performed the remaining disciplines like illustrated with exception of Requirements. In fact, it is what indeed occurs in typical Scrum process (where almost every requirements-related effort is performed before Sprint cycles, like Sprint 0 or similar).

Additionally, the implementation of some middleware-related use cases involved third-party collaboration. The implementation of those use cases must be managed from a distributed team perspective. It is required that the implementation effort by third–parties is properly aligned with the team's development process. In the case of the iFloW project, the integration with third-party service provider entities required that both teams worked together and their work aligned.

At a given point in time, both Bosch and UMinho identified the need for refactoring the code and the architecture of the system, namely to cope with security and standardization issues. Such refactoring led to a pause in the implementation tasks. The software logical architecture was revisited and the impacts were analyzed. Some design-oriented spikes (similar to architectural spikes from XP) were conducted, which then followed the re-design of the architecture. In this case, there was a focus in Analysis & Design instead of Implementation (see Fig. 8, where it is detailed the sixth Sprint included in Fig. 2). Similarly to the other sprints, this effort also lasted four weeks. This effort was required in this case but it may occur, or not, in any project.

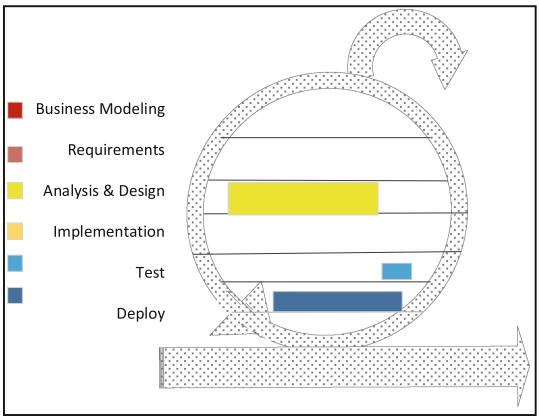

Fig. 8. The performed disciplines within the architectural spike sprint (Color figure online)

5 Lessons Learned

Defining a hybrid approach (waterfall-based during initialization and Scrum-based during implementation) in a collaborative Industry-University context arose many challenges. We believe that the inclusion of artifacts modeling and documentation strengthened the adoption of a Scrum process in these contexts. However, the entire adoption was a learning process, with advantages and disadvantages, which are detailed in this section.

5.1 Advantages

Industrial and scientific interests – typically, these kinds of collaborative Industry-University software projects aim timeliness, addressing the needs of stakeholders, rigor and access. In terms of university interests, the initial approach on documenting requirements allowed the team to gain domain knowledge, to assure the academic rigor of R&D projects. In terms of industry interests, the use of an iterative development process facilitated negotiation concerning schedules and their expectations.

Requirements documentation waterfall-based – the fact that the Product Backlog was composed of 90 use cases led to a shared perception of the system complexity that originated the need to perform proper efforts in documenting the requirements. Thus, consuming efforts in almost exclusively for requirements engineering typically performed in waterfall approaches, in the initialization phase, allowed the project team to gain the required knowledge to properly implement a system of such complexity.

Implementation Scrum-based – within a customer perspective, Bosch was always aware of the system's current state of development. The iterative development, in form of Scrum sprints, was crucial to manage Bosch's expectations, due to the periodical meetings and the incremental delivery of working software.

Use of a logical architecture – to enforce a proper organization on the set of components. The relationships among components suggest dependencies that may impact the implementation of functionalities and their inclusion in the Sprint Backlog.

Assess the logical architecture – the software logical architecture was revisited and the impacts were analyzed at the end of each iteration, in order to predict refactoring efforts. Additionally, when a change was identified, the logical architecture representation allowed to analyze which components are targeted with impacts from those changes.

5.2 Disadvantages

Effort estimation for use cases – the fact that it was a completely new development team (thus team velocity was unknown) and the need to frequently perform research spikes in order to overcome technological issues (for instance, related to GPS, EPCIS or SAP-OER) were the main obstacles for the estimation. In Scrum, estimation is performed using techniques such as planning poker, where user stories are estimated based in comparing efforts between other user stories. Due to the inexperience of the team, estimating the required effort for implementing use cases by comparing with other was itself a learning process. Such approach resulted in sprint backlogs where use cases

had not been implemented due to error in estimating and required conclusion in further sprints, and where the effort estimating of the remaining use cases (as well as rework, whenever was required, and the spikes that were performed within almost every sprints) required constant updates on every Sprint Closure and Planning meeting.

Dependence on negotiation for middleware use cases – collaborative coding among iFloW team members and service provider team members was required to implement middleware-related use cases. Most of the times the implementation required previous negotiation and agreements and the implementation did not progress at the desired velocity. The team's work reached a point where they had to pause and wait for those agreements, which resulted in the extension of use cases (and use cases with dependencies with them) through several sprints.

6 Conclusions and Outlook

The iFloW project is a collaborative R&D software project, where University researchers were contracted for developing an industrial software system. Since the project aimed at delivering a software product, the R&D coordination elements decided to use many practices available in agile process, namely in Scrum (and, to a lesser extent, in XP). During this project, there was always the concern to fulfill both industry and university needs. Thus, this project had to face key challenges as timeliness, addressing the needs of stakeholders, rigor and access. It should be noticed that the nature and context of the project created the need for firstly proposing an initialization phase (similar to what happens in waterfall model) and afterwards performing sprints throughout the implementation phase. The project ended with a deployment phase.

This paper presents how a R&D project for developing a software system was conducted by combining some Scrum practices with UML models dedicated to document requirements and architecture. Agile processes are commonly used among practitioners but not much in R&D projects. The main advantages that result from using Scrum practices within the implementation phase are related with the facilitated negotiation with the stakeholders concerning deadlines and their expectations regarding the system. Thus, stakeholders have the opportunity to frequently interact with the development team in short iterations, allowing them to adjust their ideas about the system.

Some issues stated as lessons learned are seen as opportunities for improvement. In order to prevent the stated errors within the sprint backlog definition in future projects, effort estimation techniques as use case points should be considered. Additionally, architecture design (and re-design) can also be improved. By using an architecture derivation method (like the 4SRS method [35]), through traceability mechanisms, requirements change during the implementation can be supported.

Acknowledgements. This research is sponsored by the Portugal Incentive System for Research and Technological Development PEst-UID/CEC/00319/2013 and by project in co–promotion nº 36265/2013 (Project HMIExcel - 2013-2015).

References

1. Royce, W.W.: Managing the development of large software systems. In: IEEE WESCON. Los Angeles (1970)
2. Boehm, B.W.: A spiral model of software development and enhancement. Computer (Long. Beach. Calif) **21**, 61–72 (1988)
3. Kruchten, P.: The rational unified process: an introduction. Addison-Wesley Professional, Boston (2004)
4. Barroca, L., Sharp, H., Salah, D., Taylor, K., Gregory, P.: Bridging the gap between research and agile practice: An evolutionary model. Int. J. Syst. Assur. Eng, Manag. 1–12 (2015)
5. Cho, J.: A hybrid software development method for large-scale projects: rational unified process with scrum. Issues Inf. Syst. **10** (2009)
6. Boehm, B.: Get ready for agile methods, with care. Computer (Long. Beach. Calif) **35**, 64–69 (2002)
7. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, Boston (2004)
8. Schwaber, K.: Scrum development process. In: Sutherland, J., Casanave, C., Miller, J., Patel, P., Hollowell, G. (eds.) Business Object Design and Implementation, pp. 117–134. Springer, Heidelberg (1997)
9. VersionOne Inc: 8th Annual State of Agile Survey (2013) http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf
10. Santos, N., Barbosa, D., Maia, P., Fernandes, F., Rebelo, M., Silva, P.V., Carvalho, S.M., Fernandes, J.M., Machado, R.J.: iFloW: an integrated logistics software system for inbound supply chain traceability. In: Mendonça, J.P., Fensterbank, S.-A., Barthet, E. (eds.) Enterprise Interoperability, Proceedings of 8th International Conference on Interoperability for Enterprise Systems and Applications (I-ESA). (in-press). Springer, Guimarães, Portugal (2016)
11. Choy, K.L., Ng, S.W.K., So, S.C.K., Liu, J.J., Lau, H.: Improving supply chain traceability with the integration of logistics information system and RFID technology. Materials Science Forum, pp. 135–155. Trans Tech Publ (2006)
12. Choy, K.L., So, S.C.K., Liu, J.J., Lau, H.: Improving logistics visibility in a supply chain: an integrated approach with radio frequency identification technology. Int. J. Integr. Supply Manag. **3**, 135–155 (2007)
13. Kandel, C., Klumpp, M., Keusgen, T.: GPS based track and trace for transparent and sustainable global supply chains. In: 17th International Conference on Concurrent Enterprising (ICE), pp. 1–8. IEEE (2011)
14. Doukidis, G.I., Chow, H.K.H., Choy, K.L., Lee, W.B., Chan, F.T.S.: Integration of web-based and RFID technology in visualizing logistics operations-a case study. Supply Chain Manag. Int. J. **12**, 221–234 (2007)
15. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. **50**, 833–859 (2008)
16. Ramos, H., Vasconcelos, A.: eXtreme enterprise architecture planning. In: 29th Annual ACM Symposium on Applied Computing (SAC), pp. 1417–1419. ACM (2014)
17. Abrahamsson, P., Conboy, K., Wang, X.: Lots done, more to do: the current state of agile systems development research. Eur. J. Inf. Syst. **18**, 281–284 (2009)
18. Niemelä, E., Vaskivuo, T.: Agile middleware of pervasive computing environments. In: Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 192–197. IEEE (2004)

19. Välimäki, A., Kääriäinen, J.: Patterns for distributed scrum—a case study. In: Mertins, K., Ruggaber, R., Popplewell, K., Xiaofei, X. (eds.) Enterprise interoperability III, pp. 85–97. Springer, Heidelberg (2008)
20. Dingsøyr, T., Moe, N.B.: Towards Principles of Large-Scale Agile Development: A Summary of the Workshop at XP2014 and a revised research agenda (2014)
21. Eckstein, J.: Agile Software Development With Distributed Teams: Staying Agile in a Global World. Addison-Wesley, Boston (2013)
22. Sutherland, J., Viktorov, A., Blount, J.: Adaptive engineering of large software projects with distributed/outsourced teams. In: Proceedings of the International Conference on Complex Systems, Boston, MA, USA, pp. 25–30 (2006)
23. Cristal, M., Wildt, D., Prikladnicki, R.: Usage of scrum practices within a global company. In: IEEE International Conference on Global Software Engineering (ICGSE), pp. 222–226. IEEE (2008)
24. Costa, N., Santos, N., Ferreira, N., Machado, R.J.: Delivering user stories for implementing logical software architectures by multiple scrum teams. In: Murgante, B., Misra, S., Rocha, A.M.A., Torre, C., Rocha, J.G., Falcão, M.I., Taniar, D., Apduhan, B.O., Gervasi, O. (eds.) ICCSA 2014, Part III. LNCS, vol. 8581, pp. 747–762. Springer, Heidelberg (2014)
25. Kerievsky, J.: Industrial XP: Making XP work in large organizations. Exec. Report. Cut. Consort. **6** (2005)
26. Fernandes, G., Pinto, E.B., Machado, R.J., Araújo, M., Pontes, A.: A program and project management approach for collaborative university-industry R&D funded contracts. Procedia Comput. Sci. **64**, 1065–1074 (2015)
27. Pellegrinelli, S.: What's in a name: Project or programme? Int. J. Proj. Manag. **29**(2), 232–240 (2011)
28. Schwaber, K., Beedle, M.: Agile Software Development with Scrum, 1st edn. Prentice Hall PTR, Upper Saddle River (2001). ISBN: 0130676349
29. Ambler, S., Lines, M.: Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press, Boston (2012)
30. Kroll, P., MacIsaac, B.: Agility and Discipline Made Easy: Practices from OpenUP and RUP. Pearson Education, Boston (2006)
31. Jacobson, I., Spence, I., Bittner, K.: Use case 2.0: The Definite Guide. Ivar Jacobson International (2011)
32. Waters, K.: Prioritization using moscow. Agil. Plan. (2009)
33. Anda, B., Dreiem, H., Jørgensen, M.: Estimating software development effort based on use cases-experiences from industry. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 487–502. Springer, Heidelberg (2001)
34. Grenning, J.: Planning poker or how to avoid analysis paralysis while release planning. Hawthorn Woods Renaiss. Softw. Consult. **3**, 1–3 (2002)
35. Ferreira, N., Santos, N., Machado, R., Fernandes, J.E., Gasević, D.: A V-model approach for business process requirements elicitation in cloud design. In: Bouguettaya, A., Sheng, Q.Z., Daniel, F. (eds.) Advanced Web Services, pp. 551–578. Springer, New York (2014)