Towards an Integrated Meta-Model for Requirements Engineering

Georgios Koutsopoulos[1], Niklas Kjellvard[2], Jonathan Magnusson[3] and Jelena Zdravkovic[1]

[1]Department of Computer and Systems Sciences, Stockholm University,
Postbox 7003, 164 07, Kista, Sweden
{georgios,jelenaz}@dsv.su.se
[2]Scania, Forskargatan 20, Södertälje, Sweden
niklas.kjellvard@scania.com
[3]NVBS, Löfströms allé 5, 172 66 Sundbyberg, Sweden
jonathan.magnusson@telia.com

Abstract. Traditional, plan-driven, requirements engineering identifies separate phases in the process with well-documented outputs associated with of each of them. The plan-driven system development is suitable for predictable projects where all properties of the end system are known or requested from the start. In many situations, however, the properties of the final system cannot be determined on beforehand requiring thus a basic part of the system to be built fast, and further enable it to evolve. For this reason, it has become more common in recent years to adopt agile development methods, which foster interactive working with customers, in short iterations, and with frequent system changes and releases. Because the plan-driven and agile approaches substantially differ in their main concepts and working steps related to requirements engineering, and the fact that larger projects often blend them, we have identified a need for establishing relationships between them through an integrated meta-model. The final artifact contains the elements of both agile and plan-driven requirements engineering, supporting thus their separate, or hybrid use, which we have illustrated and thereby discussed and concluded this research-in-progress study.

Keywords: Requirements Engineering, Plan-driven, Agile, User Story, Conceptual Modeling

1 Introduction

Traditional Requirements Engineering (RE) methods have been around for almost half a century, and their concepts are well established. The best known of these is the waterfall model introduced in 70's; a sequential flow of steps, where the outcome of each system development step serves as input to the next step [1]. System requirements are completely determined up front in the requirements analysis phase, which is followed by the subsequent phases of design, testing, implementation, and maintenance. In the next two decades, the incremental model emerged, where the requirements are specified in increments which are clearly defined - typically the core

system functionality is defined in first increments, or a functionality involving the highest risk in the development or use of the system. The requirements are therefore prioritized; those with a higher priority were earlier specified and passed to the design. A wide-known system development method relying on the incremental model is Rational Unified Process (RUP) developed in the late 90's, where, as for system requirements, the main artifact was the model of Use Cases, which evolved over the development phases inception, elaboration, construction, and transition [2]. Compared to these traditional, i.e. plan-driven development approaches, which are incorporating a defined process and working procedures, repeatability and predictability, extensive documentation, up-front architecture, validation and other, its agile counterparts are relatively young and different [3]. The basic principles of agile system development are established in the early 2000s [4]. Instead of extensive documentation and planning, emphasis is instead placed on individuals' interactions in processes and when using tools. The agile RE can be summarized as focusing on interaction between customers and development team, brief requirements, with their finalization during the development process for avoiding volatile decisions [5]. The most known prominent agile system development methods are Extreme Programming (XP) [6], and Scrum [7], while other related include Crystal, Adaptive Software Development, and even other. Agile approaches consider design and implementation to be the central activities in the development process, while documentation and analysis of requirements are less considered.

Insufficient RE can lead a project to go over its budget, and over the timeframe [8]. Correcting detected deficiencies in the requirements during the programming phase can cost up to many times more, compared to whether these deficiencies would be rectified already during the requirements analysis and based on a complete documentation. On the other side, as competition in industry has increased, the way to develop systems and software has gained a greater focus on product delivery and thus it has become more important to be able to collect and manage requirements in a more flexible way, using agile development methods [3].

Agile methods are effective when the system can be developed with a small co-located team communicating informally. This may be a constraint for development of bigger systems requiring larger development teams and lot of analysis before implementation - in such a situation a plan-driven approach may better fit; in addition, such systems may require more documentation and traceability of the development artifacts to communicate the original intentions of the developers to the support team.

The main problem in this study revolves around the fact that concepts and phases in the plan-driven RE methods are well established and well elaborated, but that it is not clear how they relate to their agile counterparts, and whether the concepts and phases overlap. It is not therefore clear whether and how the approaches could be blended, or benefit from one another. We have therefore designed an integrated meta-model, to explore the similarities and differences between the two approaches, and to unify their concepts and relationships. The solution represents an opportunity to demonstrate how project managers, requirements engineers and developers can combine concepts and methods of both traditional and agile RE approaches.

The rest of the paper is organized as follows. Section 2 outlines a brief background on plan-driven and agile RE and related studies. Section 3 presents the main theoretical proposal, the 2 individual meta-models, and their integration to a unified model artifact, and illustrates its use. A discussion is given in section 4 including concluding remarks and future work.

2 Background and Related Work

2.1 Plan-driven and Agile Approaches

RE is a process that involves many activities, as well as there are many different techniques and methods developed to support it. The traditional, plan-driven process is well established and thus there are clear guidelines for the activities and concepts used, which are generally divided into four phases [8]. *Elicitation* aims at initial collection of requirements from various sources, such as stakeholders, users, organizational documentation and existing systems; various techniques are used, such as interviews, workshops, observation, and other. During the *Documentation* phase, the collected requirements are formally specified using an adequate notation, typically in combination of natural language and different models such as Use Cases and domain class diagrams. In the *Negotiation* phase, the requirements are analyzed for various possible conflicts – logical, business-related, or other, and accordingly prioritized. *Validation* is performed to ensure that the documented individual requirements and the entire *requirement specification* is understandable, consistent, complete, and meets stakeholders' needs. In parallel to these phases, requirements change management records and track changes using specified dependency and traceability between the requirements. The upper part of Figure 1 briefly visualizes the overall plan-driven system development process:

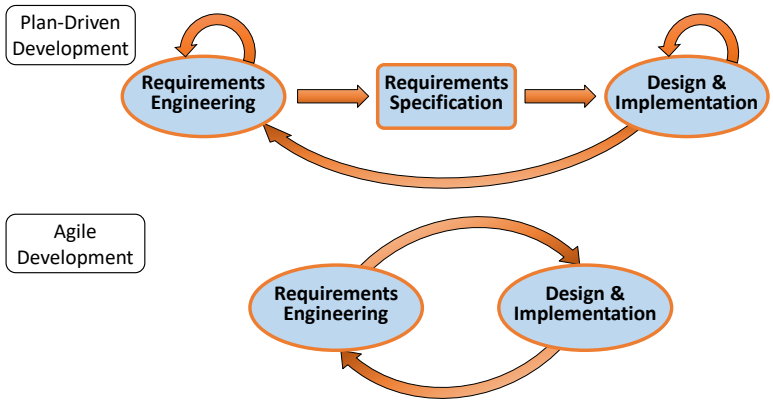

Fig. 1. Plan-driven and Agile RE.

The agile process (the bottom part of Figure 1), in contrast, has been envisioned to include *practices* such as tight and on-going communication with the customer, story-

driven modeling, short development iterations with frequent releases, limited documentation, designing and re-designing requirements as new evolve, or existing ones change, relying as well on test cases as validation [9, 10]. The methods for agile development evolved with XP and especially with Scrum [11]. These methods, as well as the other mentioned in Introduction, differ in which of the agile practices they promote the most, as well as how they consider the management of requirements. The main common concept for the RE in agile methods has become Use Story. It is used to describe a system requirement in the following way:

[Story Title] (A line naming the story)
As a [Role]
I want to [Functionality]
So I can/get [Benefit]

The User Story defines an action performed by a user in a specific role. The Functionality can be seen as a function that the system should perform in order to give a particular type of user (Role) an advantage (Benefit). A User Story is further detailed in terms of conversation/scenario and test procedures.

2.2 Related Studies

A plethora of studies concerning the duality of plan-driven and agile RE exist in the literature. Several studies concern research with a focus in the comparison between the two approaches or within an approach like [12], where strengths and weaknesses of agile approaches have been identified and discussed in a systematic comparison. A possible reason behind this delimitation lies in the fact that there is research work supporting that agile and traditional RE are juxtaposed and opposite in nature, even though the objective of all approaches remains the same [13]. Nevertheless, despite this fact, there is no lack of studies attempting to bridge the gap, by introducing or exploring approaches that are characterized as integrated, mixed or hybrid.

For example, in [14], a literature review was conducted that explored the types of hybrid approaches based on the way that the approaches are being combined. Two main categories were identified. The first one, called “Hybrid by phases” concerns approaches that combine the application of agile and plan-driven approaches per phase of the project while the second one, namely “Hybrid by methods” utilizes mixed methods, for example, a plan-driven estimation tool during agile development. What was also identified is that the hybrid approaches are efficient both in IT and non IT projects. Using parameters like project size, criticality, rate of changes, culture, and people, a theoretical model was developed for the evaluation of hybrid approaches.

The field of RE is evolving and there is interest in identifying how the perspectives on what is considered traditional have been changing over the years. For example, 15 years ago, including Use Case modeling in RE projects was still considered a hybrid approach [15], while nowadays, Use Cases are considered part of the tradition,

There have been research studies contributing towards the integration of the two approaches as a means to incorporate the benefits of both and avoid their drawbacks. One such study is [16], where, in order to avoid testing strategies on real projects, two hybrid approaches for requirements prioritization have been introduced and empirical

simulations were conducted to compare the efficiency of agile, plan-driven and mixed approaches. Their results suggest that mixed strategies outperform agile and plan-driven approaches, even in their dominant areas, that is, large and complex projects with stable requirements or small and dynamic projects, for plan-driven and agile approaches respectively [16].

The development of RE meta-models as a means to provide practical support has also been identified as a theme in the existing literature. Meta-modeling has been encountered in several general and domain-specific studies regarding RE. In [17], a meta-model has been developed for agile development with a focus on the variety of existing process models. The meta-model aims to contribute towards a common definition in the area and the development of a unified language for the agile development processes. Another meta-model was proposed in [18], with a focus on the description, representation and relationships of the concepts that the requirements engineers should elicit and specify. The meta-model is specifically used to guide the RE process in the domain of embedded systems. Meta-modeling has also been employed in [19]. The focus of this study lies in the concept of traceability, and specifically, the traceability of non-functional requirements. A plan-driven meta-model has been enhanced with agile concepts and the specifically modified result is contributing towards the resolution of issues in this specific research area. Another meta-model for security requirements has been introduced in [20], based on existing risk-based security RE meta-models. Finally, in [21] a meta-model is introduced, optimized for modeling and managing requirements for the System Families approach.

3 An Integrated Requirements Engineering Meta-Model

The purpose of this study was to design an integrated meta-model of agile and plan-driven RE methods. The aim of the meta-model is to provide an increased understanding of the similarities, differences and relationships between the two approaches, by presenting their concepts in a single artifact.

The design approach was to first separately conceptualize the meta-models of the agile and plan-driven methods based on the literature study to obtain correct understandings about them individually. Having these two meta-models visualized, has enabled the identification of similarities and related concepts, along with their integration into one. In addition, analyzing and understanding them individually was made possible. The main requirements established in the design process were as following:

- The integrated meta-model should contain existing concepts from agile and plan-driven RE methods and be reasonably complete with regard to the conceptualization of each of the methods.
- The integrated meta-model should enable independent use of any of the existing individual RE methods, by preserving the concepts and the relationships of the individual methods.

- The integrated-meta model should give a basis for combining the methods by establishing the relationships between the concepts of the individual methods to enable using them combined.

The integration process engaged the four authors of this study, and three reviewers – a senior academic, a project manager having a long experience of work with different RE methods, and a Scrum-certificated expert. The authors made the proposals for the integrated meta-model based on the individual ones, which were in iterations analyzed and commented by the reviewers, until a commonly agreed result was obtained.

3.1 Agile and Traditional Requirements Engineering Meta-Models

Agile development approaches mainly differ from each other in the activities performed in the development phases, while the common part is that the elicitation of requirements is mainly done through User Stories, as we explained in Section 2. The figure below depicts a meta-model of the common agile concepts as they have been in the latest years harmonized among the methods mentioned in Introduction and in sections 2.

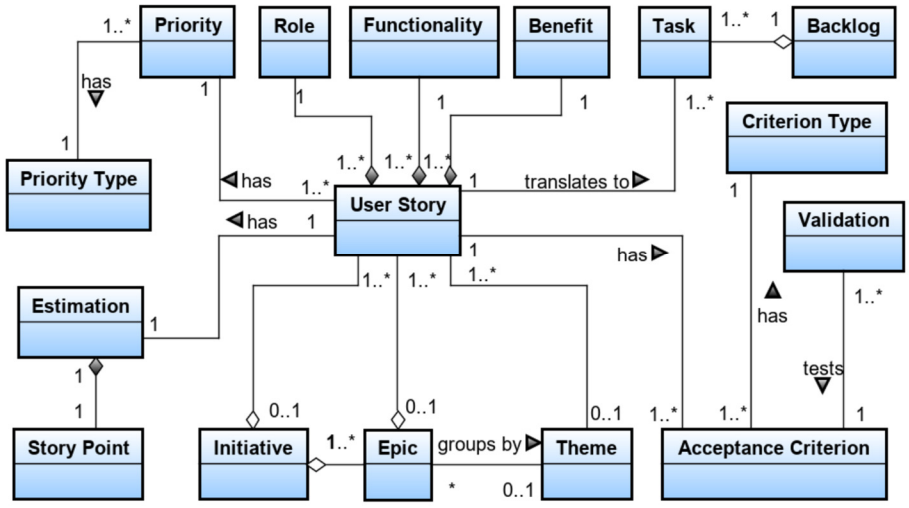

Fig. 2. Agile RE meta-model.

The central element in the meta-model is User Story, describing an informal description of a system function, written in a natural language from a user's perspective consisting of three parts: Role, Functionality and Benefit. Role is a type of user, from whose perspective the User Story is written. Functionality represents a desired system behavior. Benefit is the advantage provided by the system function to the user. Task is a broken-down part of User Story, which contributes to the completion of the story. Acceptance Criterion is a checklist designed to determine if one User Story is logically correct and complete which should pass Validation. A criterion belongs to its powertype Criterion Type. Estimation is a measurement of the time and the resources

User Story's development requires. It is usually accompanied by Story Point(s), which is a measure of the relative size and complexity of the User Story. Epic is a large story containing multiple User Stories related through common Theme(s). Priority - User Stories are prioritized by the customer in order to determine which stories should be first developed. Several Epics or User Stories are grouped through common Themes. Several Epics or User Stories grouped hierarchically make Initiative. Backlog is a priority list, including Tasks, of the stories to be developed in an iteration. A popular concept used in the agile literature is Feature, however, the lack of consensus on its definition may result in possible confusion, therefore, it has been omitted from the meta-model. The Functionality element refers to a low-level feature, while the Epic element refers to a high-level feature.

Plan-driven requirements specification, unlike agile, has a focus on artifacts and phases (see Figure 3). The concepts and the relations have been constructed based on [8]:

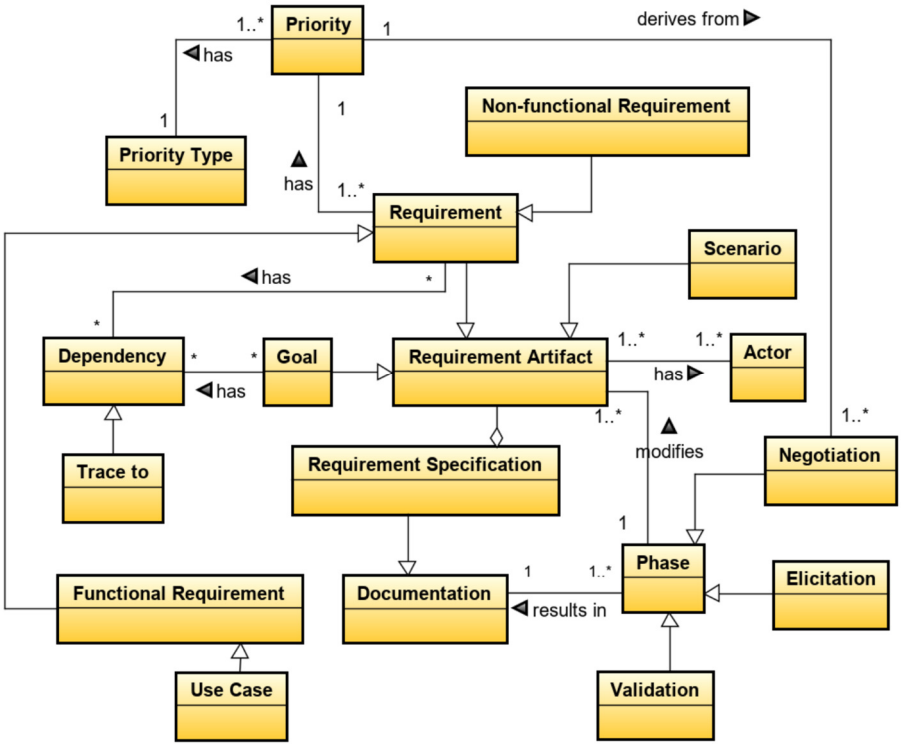

Fig. 3. Plan-driven RE meta-model.

Requirement is an expression for a desired system behavior. A Functional Requirement is describing a task to be performed by a system; it can be described using different formalism, one common is by Use Case, a collection of actions that defines the interaction between an actor and the system. Non-Functional Requirement describes a

quality (sometimes referred as “constraint”) of the system, such as performance, security, or other. Goal describes an intended result to be achieved by the system. A Requirement, or a Goal, or a Scenario, is identified through Elicitation and when specified during Documentation, it becomes a Requirements Artifact included in a Requirements Specification. A Requirement Artifact has an Actor responsible for it, such as a stakeholder, or a Use Case actor. Negotiation phase aims at identifying and resolving any conflicts, as well as determining Priority of requirements by following a Priority Type rule. Validation phase takes the focus on quality control of Requirement Artifact(s).

3.2 The Integrated Meta-Model

The integration process has resulted in the meta-model shown in Figure 4. The blue color is used to depict the concepts of the agile meta-model, the orange for the plan-driven, while the green is depicting the concepts common for the both methods.

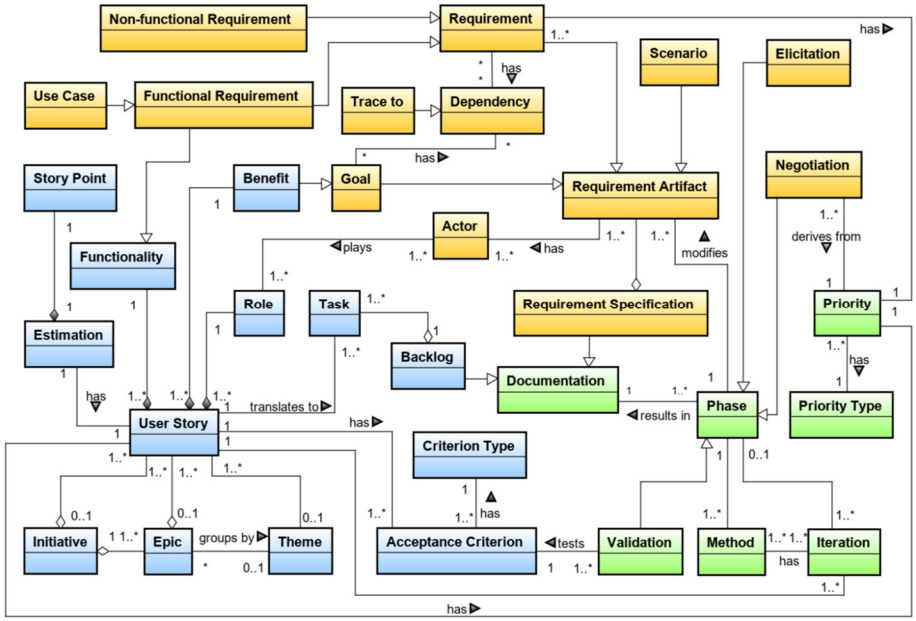

Fig. 4. Integrated RE meta-model.

In brief, the integrated meta-model depicts several concepts which bear the same name and are used in the same way in the both methods; it also depicts new relationships and classes, which were established to link similar concepts of the methods. Details are explored in Table 1 below:

Table 1. Unified, linked and new elements of the integrated meta-model.

Class/Relationship	Description
Priority	Priority is found in both agile and traditional requirements management, and has the same meaning.
Phase	Collection class for the phases of plan-driven RE, including Validation, which is also found in agile requirements management.
Validation	Found in both agile and plan-driven RE, but with different meanings. Within agile, Validation is aimed at testing the Acceptance Criterion, while in the plan-driven, it aims at reviewing the requirement specification.
Method	A class that contains the various methods of requirements management, such as agile and plan-driven.
Iteration	It is added to create a bridge between Method, Phase and User Story. There is at least one iteration, both in the plan-driven phases and in the agile collection of User Stories.
Documentation	Does not refer to the documentation phase taking place during the course of the project within plan-driven RE (Requirements Specification), as well as within agile requirements engineering (Backlog), but to the set of documents derived from the phase.
Backlog - Documentation	Backlog is an outcome of documenting in the agile method.
Functionality – Functional requirement	A Functionality of a User Story in the agile method corresponds to a Functional Requirement in the plan-driven method.
Benefit - Goal	In the agile method, the Benefit of a User Story may coincide with a Goal in the plan driven method.
Role - Actor	The Role of a User Story in the agile method corresponds to the Actor in the plan-driven method, such as to stakeholder or a Use Case actor.

Illustration 1.

With this illustration, the intention is to demonstrate how the proposed integrated meta-model (Figure 4) can contribute to better coordination in larger projects, where the management of requirements is done by following both agile and plan-driven approaches.

The example is based on a project where a new CRM system, in a 3-layer architecture is to be developed. The project consists of a group that works agile (Agile-g) and the other that works according to a plan-driven approach (PlanD-g). The Agile-g manages and develops the requirements for the presentation layer, including user interface, while PlanD-g focuses on the Database and Business Logic layers. This is for the reason that within the agile requirements management the focus is on what the customer wants to obtain, in short iterations, and with the releases of limited functional software. Therefore, Agile-g starts by collecting User Stories, such as:

– As a [Seller] I want to be able to [Get call lists for prospects] So that I can get [Increased efficiency in my sales]

- As a [Seller] I want to be able to [See sales statistics per item for specific time period] So that I can get [Optimized Sales Opportunities]
  - As a [Seller] I want to be able to [Get detailed sales statistics for each customer] So that I can get [Optimized Sales Opportunities]

Through the shared access to Documentation (Figure 4), PlanD-g gets the access to the written User Stories, and accordingly can initiate elicitation of requirements for its system parts. Based on the relationship between Role and Actor, Plan-D gets access to Actor, which allows identification of responsible Actors (stakeholders, and even Use Case actors) for Requirements, in this case - Seller. The main high-level Requirements elicited, negotiated and validated by PlanD-g are:

- FR1: the system shall provide the customer database,
- FR2: the system shall provide periodic sales statistics for individual customers,
- FR3: the system shall provide customer statistics.

Like for the given example, Agile-g can obtain, through a common relationship with Documentation, the access to any elicited requirement, including related Non-functional Requirements to the existing ones, through Dependency. In this way, Agile-g can, in an early iteration, conduct acceptance tests for validation of some important Requirements such as non-functional (performance, reliability, or other) which would otherwise not have been considered or prioritized in User Stories knowing that their focus lies on functionality (see also Section 2).

Illustration 2.
The following simple example demonstrates a utility of the meta-model by illustrating how it can facilitate an asynchronous application of both approaches on a system. The example concerns upgrading a payment system, in particular, an automated self-checkout functionality is being introduced. Originally, the requirements for the system had been elicited using a plan-based approach, while the requirements for an upgrade are elicited following an agile approach. The existing specification would include a plethora of detailed requirements, however, only a small fragment concerning the payment is illustrated and depicted in Table 1, to serve the example purpose.

Table 2. A fragment of the fictitious requirements specification.

ID	Description	Trace from
G1	Process customer payments more efficiently.	
FR1	The system should process payments.	G1
USE1	Process payments	FR1
NFR1	The system should calculate payments in 2 seconds.	G1

A functional requirement and a non-functional requirement, namely FR1 and NFR1 respectively, have been derived and are traced from goal G1. A Use Case (USE1) has

been documented in the original specification along with its associated traceability, as shown in Table 1, its priority as high, and Cashier as its actor. The level of detail in the description of USE1 also allowed the inclusion of a main and an alternative flow along with pre-conditions and post-conditions. In particular, the main flow concerns payment by cash, the alternative flow card payments, a pre-condition that the system is operating and is connected to a database and a bank, and post-condition that the payment has been processed and registered and a receipt has been printed.

Instantiating the model would result in an instance of the Use Case class, namely USE1 and G1 would be an instance of the Goal class, which are both associated with an instance of the Trace To class to depict their traceability link. An association with the Priority class assigns Priority values of a specific Priority Type during the Negotiation phase. This can also be reflected in the model using associations. The association with the Requirements Specification class would reflect the documented state of the requirements.

The agile approach for upgrading the system would suggest starting from the beginning by eliciting User Stories. However, the integrated meta-model enables the potential to reverse engineer User Stories from the existing Use Case information in the Requirements specification. The shared access to the Documentation class would provide enough information to identify required tasks, and in return, User Stories. Any alternative flow can possibly generate a User Story, the roles are existing as actors, the benefits exist as goals and pre-conditions and post-conditions can serve as acceptance criteria along with any possible dependence with non-functional requirements. Traceability and priority can also be extracted.

A minimal and abstract description of USE1 would include these steps:

Main Flow:
1. Product is scanned
2. System registers price and quantity
3. Sum is requested
4. Systems calculates sum
5. Cash amount is registered
6. The system Change calculates and displays change, receipt is printed.

Alternative Flow:
5a. Card is scanned

In this particular example, USE1 provides information adequate information for the following User Stories:

- As a [Cashier] I want to be able to [Have a receipt printed] so that I can [Process customer payments more efficiently]
- As a [Cashier] I want to be able to [Calculate payments by cash] so that I can [Process customer payments more efficiently]
- As a [Cashier] I want to be able to [Calculate payments by card] so that I can [Process customer payments more efficiently]
- As a [Cashier] I want to be able to [Register payments by cash] so that I can [Process customer payments more efficiently]

- As a [Cashier] I want to be able to [Register payments by card] so that I can [Process customer payments more efficiently]

The role is still the cashier since the self-checkout machines need to be supervised by a cashier. The functionalities that are still missing before completing the initial agile iteration are the automatic delivery of receipt and the automatic return of change as long as cash payment is concerned. Collecting the two User Stories below is completing the set.

- As a [Cashier] I want to be able to [Return change to customers automatically] so that I can [Process customer payments more efficiently].
- As a [Cashier] I want to be able to [Deliver an automated receipt] so that I can [Process customer payments more efficiently]

So, the majority of the User Stories that would comprise the payment epic, have been extracted from an existing plan-driven specification. Taking also into consideration that the User Story mapping can be facilitated by the existence of traceability and priority associations, this can enable the potential to save valuable time and effort from the requirements engineers that are working on the project.

4 Discussion, Conclusions and Future Work

In the traditional view, RE is considered as critical to avoid wrong, incomplete, or ambiguous requirements, which will be as such delivered to system design. Hence, in the plan-driven approach, it is a common practice to use a process to plan successful management of requirements and creation of the system requirements specification, by doing extensive elicitation, documentation, negotiation and validation. The agile methods have, in contrast, focus to, by highly interactive practices reduce the amount of documentation, do smaller chunks of requirements specified and managed less formally. This, in turn, leads to a faster implementation, which means that user feedback comes at an earlier stage. Yet, some challenges may emerge when scaling an agile system development method across a large project and teams because of the lack of overall planning, coordination and specification. Both approaches may be seen have advantages and disadvantages depending on the development environment - agile is well suited for the projects with high levels of uncertainty in requirements, and plan-driven is more suited for low level of uncertainty projects, i.e. where requirements are rather stable.

The goal of this study has been therefore to, by individually conceptualizing the main concepts of the two approaches, design an integrated conceptualization to understand how they relate to each other. The main propose behind that is gaining insight about the differences of the two approaches, and moreover opportunity to (i) consolidate in a single place (the integrated meta-model) different methods being used in a project; and (ii) blend the approaches by, for instance, practicing the use of different or additional concepts in current agile methods in the way that they benefit from plan-based when the size and complexity of a project motivates that.

The presented integrated meta-model has been reviewed and discussed by few experts as we reported in Section 3 and its use was demonstrated by two illustrative examples. The main subjects of the future work are therefore to test the integrated meta-model in a real project, and thus also to investigate in which directions the artifact should be improved; enlarge it to include even more detailed concepts, or enrich it with the methods of use; or even extend it to support emerging approaches relying on new requirements-related notions such as for capability-driven system development [22]; or, for example, by considering digital requirements sources in addition to human customers / stakeholders.

Acknowledgment. We would like to express our gratitude to the experts mentioned in Section 3 who dedicated their time for reviewing the meta-model artifacts throughout their development.

References

1. Royce, W.W.: Managing the Development of Large Software Systems-Concepts and Techniques. 1970. In: Proceedings of IEEE WESCON. pp. 1–9. TRW (1970).
2. Booch, G., Rumbaugh, J., Jacobson, I.: The unified modeling language user guide. Addison-Wesley, Reading Mass (1998).
3. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Computers in Human Behavior. 51, 915–929 (2015). https://doi.org/10.1016/j.chb.2014.10.046.
4. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., others: Manifesto for agile software development. (2001).
5. Zowghi, D., Coulin, C.: Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In: Aurum, A. and Wohlin, C. (eds.) Engineering and Managing Software Requirements. pp. 19–46. Springer-Verlag, Berlin/Heidelberg (2005). https://doi.org/10.1007/3-540-28244-0_2.
6. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, Reading, MA (2000).
7. Schwaber, K.: SCRUM Development Process. In: Sutherland, J., Casanave, C., Miller, J., Patel, P., and Hollowell, G. (eds.) Business Object Design and Implementation. pp. 117–134. Springer London, London (1997). https://doi.org/10.1007/978-1-4471-0947-1_11.
8. Pohl, K.: Requirements engineering: fundamentals, principles, and techniques. Springer, Heidelberg ; New York (2010).
9. Leffingwell, D.: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley, Upper Saddle River, NJ (2011).
10. Ramesh, B., Cao, L., Baskerville, R.: Agile requirements engineering practices and challenges: an empirical study: Agile RE practices and challenges. Information Systems Journal. 20, 449–480 (2007). https://doi.org/10.1111/j.1365-2575.2007.00259.x.
11. Rubin, K.S.: Essential Scrum: a practical guide to the most popular agile process. Addison-Wesley, Upper Saddle River, NJ (2012).
12. Muneer, S.U., Nadeem, M., Kasi, B.: Comparison of modern techniques for analyzing NFRs in Agile: A systematic literature review. Journal of Software Engineering Practice. 3, 1–12 (2019).

13. Abbas, J.: Quintessence of Traditional and Agile Requirement Engineering. JSEA. 09, 63–70 (2016). https://doi.org/10.4236/jsea.2016.93005.
14. Imani, T., Nakano, M.: A Model for Effective Area of Hybrid Approach Combining Agile and Plan-Driven Methods in IT Project. Journal of International Association of P2M. 13, 52–70 (2018).
15. Daniels, J., Bahill, T.: The hybrid process that combines traditional requirements and use cases. Syst. Engin. 7, 303–319 (2004). https://doi.org/10.1002/sys.20013.
16. Port, D., Bui, T.: Simulating mixed agile and plan-based requirements prioritization strategies: proof-of-concept and practical implications. European Journal of Information Systems. 18, 317–331 (2009). https://doi.org/10.1057/ejis.2009.19.
17. Schön, E.-M., Sedeño, J., Mejías, M., Thomaschewski, J., Escalona, M.J.: A Metamodel for Agile Requirements Engineering. JCC. 07, 1–22 (2019). https://doi.org/10.4236/jcc.2019.72001.
18. Pereira, T., Sousa, A., Oliveira, R., Albuquerque, D., Alencar, F., Castro, J.: A Metamodel to Guide a Requirements Elicitation Process for Embedded Systems. In: 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC). pp. 101–109. IEEE, Coimbra (2018). https://doi.org/10.1109/QUATIC.2018.00023.
19. Binti Arbain, A.F., Ghani, I., Wan Kadir, W.M.N.: Agile non functional requiremnents (NFR) traceability metamodel. In: 2014 8th. Malaysian Software Engineering Conference (MySEC). pp. 228–233. IEEE, Langkawi, Malaysia (2014). https://doi.org/10.1109/MySec.2014.6986019.
20. Faily, S., Fléchais, I.: A meta-model for usable secure requirements engineering. In: Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems - SESS '10. pp. 29–35. ACM Press, Cape Town, South Africa (2010). https://doi.org/10.1145/1809100.1809105.
21. Cerón, R., Dueñas, J.C., Serrano, E., Capilla, R.: A Meta-model for Requirements Engineering in System Family Context for Software Process Improvement Using CMMI. In: Bomarius, F. and Komi-Sirviö, S. (eds.) Product Focused Software Process Improvement. pp. 173–188. Springer Berlin Heidelberg, Berlin, Heidelberg (2005). https://doi.org/10.1007/11497455_15.
22. Zdravkovic, J., Stirna, J., Kuhr, J.-C., Koç, H.: Requirements Engineering for Capability Driven Development. In: Frank, U., Loucopoulos, P., Pastor, Ó., and Petrounias, I. (eds.) The Practice of Enterprise Modeling. pp. 193–207. Springer, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45501-2_14.