ISSN (e): 2250 – 3005 || Volume, 10 || Issue, 7|| July – 2020 ||
International Journal of Computational Engineering Research (IJCER)

Extending Agile Methods with Requirements Engineering

Huayu Liu [1], Jürgen Jung [2], Quandang Sun [1]
[1]Student,
Department of Computer and Information Engineering,
Henan Normal University, 46 Jianshe East Road, Xinxiang, China
[2]Professor,
Department of Computer Science and Engineering,
Frankfurt University of Applied Sciences, Nibelungenplatz 1, 60318 Frankfurt, Germany
[1]Assistant professor,
Department of Computer and Information Engineering,
Henan Normal University, 46 Jianshe East Road, Xinxiang, China

ABSTRACT
Agile methods for tend to improve the quality of software development projects by imposing a strong customer focus and continuous risk mitigation with every iteration. It seems that they are about to substitute existing development methods in professional environment. However, agile methods are still no silver bullets as they have some limitations. They are designed for small projects that require continuous customer feedback. They also suffer from supporting internationally distributed development environments as customers and development teams need to be at the same location. Methods like for example the Scaled Agile Framework (SAFe) were arising since a couple of years in order to overcome the limitations of agile methods. They define additional coordination stages and, therefore, impose some additional complexity on applying them. The paper at hands aims at providing a pragmatic solution for the application of agile methods in a global environment. Many organizations already adopted methods and tools for requirements engineering in the past and these can be combined with agile methods. The paper summarizes requirements engineering and analysis agile methods with respect to gaps to requirements engineering methods. It will then provide an overview on how requirements engineering may complement today's agile methods.
KEYWORDS: *agile method, requirements engineering, global team, requirements repository*

Date of Submission: 10-07-2020 Date of acceptance: 26-07-2020

I. INTRODUCTION

Garbage in, Garbage out', this is the iron law of the IT industry. The quality of requirement input affects or even determines the quality of the process and delivery. Agile methods for software development have become popular over the past decades as they helped with gaining fast success in software development projects and reducing project risks. Classical methods, such as the waterfall model or the V-model, tend to be heavy-weight and document driven as they are following a strict phased approach for managing the overall project. On the contrary, agile methods (like Scrum or eXtreme Programming) focus on short release cycles for delivering tangible results in an iterative way [1]. They also reduce the management overhead significantly as they emphasize the role of the software developer as a key player in the project: the value (i.e. running software) is created by the developer.

Despite a plethora of success stories and their significant impacts, agile methods cannot be applied to any kind of software project successfully [2]. They assume a small and coherent team being at the same location with the customer. Distributed development environments or software development in a strictly regulated environment

(e.g. medical or safety critical) are, therefore, not in the scope of agile methods. A couple of agile frameworks have been developed recently in order to overcome the limitations of agile methods. Nexus and SAFe (Scaled Agile Framework) are two representative frameworks that extend agile methods with additional, coordinating activities and roles [3]. These frameworks are called "scalable" because they also allow for the development of large and complex software systems.

However, these agile frameworks bring additional complexities on the management of software projects. Furthermore, they rather focus on managing and synchronizing agile projects but lack support for aligning projects with the whole corporation [4]. The paper at hand is following a different approach by incorporating Requirements Engineering into agile methods. The discipline of Requirements Engineering (RE) is well-known and has been successfully implemented in many companies [5]. People are skilled in managing requirements and corresponding tools are already available. In the agile development mode, how to improve the efficiency of RE and ensure the quality of requirements input is a huge challenge. The approach suggested here aims at extending agile methods with respect to:

- Distributed teams: RE supports collaboration within an agile team, even if team members are at different locations.
- Project alignment: RE supports collaboration and communication among agile project teams
- Corporate requirements: RE supports following general requirements (e.g. regulations, compliance rules) across all software projects.

The paper is structured as follows: Section 2 provides an overview of the requirements engineering discipline. It presents typical roles involved as well as a life cycle for requirements. Section 3 summarises popular agile methods, analyses them concerning their gaps with respect to requirements engineering and highlights the roles of missing RE activities. Section 4 proposes the methods of filling in the RE gaps in agile methods. The objective will be in reusing RE activities and roles in order to extend agile methods. RE will then be coordinating activity for collaboration and communication. The application of such an approach is then documented in section 5. It shows how Scrum is extended by RE in a global logistics company. Then we make the conclusion in section 6.

II. REQUIREMENTS ENGINEERING

2.1 Purpose of Requirement Engineering

Requirement Engineering is the process of defining, documenting and maintaining the requirements. Requirements engineering corresponds to the first stage before system design in the life cycle of software systems development. Therefore, the purpose of requirements engineering is straightforward: determine customer requirements and define all external characteristics of the system in the vision by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication and subsequent implementation [6].

Requirements Engineering Process consists of the following main activities:

1. Requirements elicitation
2. Requirements specification
3 .Requirements verification and validation
4. Requirements management

2.1.1 Requirements Elicitation

This phase of requirements engineering focuses on collecting and gathering data and information from various stakeholders in the project domain, including stakeholders, users, constraints, environment, standards, etc. [7].

But numerous factors can conspire to hamper your requirements elicitation efforts. The following are several areas of potential elicitation risk and suggestions for how to avoid them [8]:

Product vision and project scope: Scope creep is more likely if the stakeholders lack a shared understanding of what the product is supposed to be (and not be) and do. Early in the project, write a vision and scope document that contains your business requirements, and use it to guide decisions about new or modified requirements.

Time spent on requirements development: Tight project schedules often pressure managers and customers into glossing over the requirements because they believe that if the developers don't start coding immediately, they won't finish on time. Record how much effort you spend on requirements development for each project so that you can judge whether it was sufficient and improve your planning for future projects. Agile development approaches allow construction to begin earlier than on projects following a waterfall life cycle.

Customer engagement: Insufficient customer involvement during the project increases the chance of an expectation gap. Identify stakeholders, customers, and user classes early in the project. Determine who will serve as the literal voice of the user for each user class. Engage key stake holders as product champions. Make sure product champions fulfill their commitments so you elicit the correct needs.

Completeness and correctness of requirements specifications: Elicit user requirements that map to business requirements to ensure that the solution will deliver what the customers really need. Devise usage scenarios, write tests from the requirements, and have customers define their acceptance criteria. Create prototypes to make the requirements more meaningful for users and to elicit specific feedback from them. Enlist customer representatives to review the requirements and analysis models.

2.1.2 Requirements specification

This activity is used to produce formal software requirement models from communication. All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality.

The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc. But just because requirements are communicated on paper or in writing doesn't mean they are actually understood. [8]

Requirements understanding: Different interpretations of the requirements by developers and customers lead to expectation gaps, in which the delivered product fails to satisfy customer needs. Peer reviews of requirements by developers, testers, and customers can mitigate this risk. Trained and experienced business analysts will acquire the right information and write high-quality specifications. Creating models and prototypes that represent the requirements from multiple perspectives can reveal fuzzy, ambiguous requirements.

Time pressure to proceed despite open issues: It is a good idea to mark areas of the requirements that need further work with TBD (to be determined) or as issues, but it's risky to proceed with construction if these haven't been resolved. Record who is responsible for closing each open issue and the target date for resolution.

Ambiguous terminology: Create a glossary to define business and technical terms that might be interpreted differently by different readers. Requirements reviews can help participants reach a common understanding of terms and concepts.

Design included in requirements: Design elements that are included in the requirements place constraints on the options available to developers. Unnecessary constraints inhibit the creation of optimal designs. Review the requirements to make sure they emphasize what needs to be done to solve the business problem, rather than dictating the solution.

2.1.3 Requirements verification and validation

Verification: It refers to the set of tasks that ensure that software correctly implements a specific function. **Validation**: It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. If requirements are not validated, errors would process to the successive stages resulting in a lot of modification and rework.

The main steps for this process include:

1. The requirements should be consistent with all the other requirements i.e. no two requirements should conflict with each other.
2. The requirements should be complete in every sense.
3. The requirements should be practically achievable.

Reviewing of requirements with stakeholders, buddy checks, making test cases, etc. are some of the methods used for this. In addition, ensure that traceability is achieved during the validation and verification process to facilitate the later audit trail [7].

2.1.4 Requirements management

Requirement management is the process of analysing, documenting, tracking, prioritizing and agreeing on the requirement and controlling the communication to relevant stakeholders. Due to the tendency of changing in requirements, it should be ensured that the SRS (Software Requirements Specification) is as modifiable as possible to incorporate changes in requirements specified by the end users at later stages, too. Being able to modify the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process.

2.2 Requirement Life Cycle

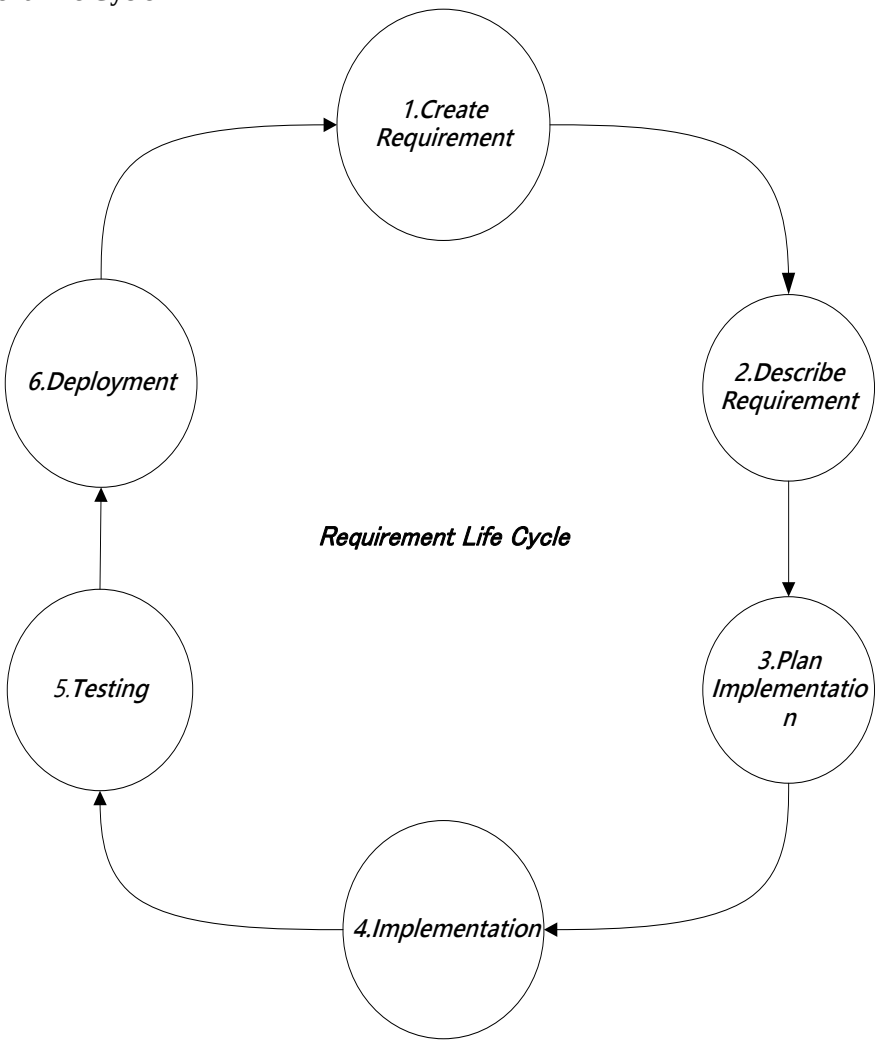

Figure 1: Requirement Life Cycle

The above picture is our definition of the requirements life cycle, which is similar to the software development life cycle, but the main difference is that the requirements lifecycle focuses on a single requirement developing only. In software development, the focus is on a complete system. However, in the entire process of the software development life cycle, there are still missing some stages which related to requirements, such as requirements analysis and requirements design. Because the requirement life cycle only covers the planning and implementation of one requirement.

The requirement life cycle consists of following activities [9]:

Create Requirement: The business requirement is created in this phase and stored in a requirements repository. It focuses on the main stakeholder—which is the customer. The main stakeholder has a new idea about the software system (i.e. product) and it will be recorded as a new requirement. The description is rather high level and needs to be refined later together with a requirements manager. This description will be the common reference for the software developer in order to implement the new requirement.

Describe Requirement: Each requirement needs to be described in detail so that it can be used for system implementation. The business stakeholder will detail out the description together with a business analyst or a requirement engineer. Meetings with Requirement engineer, stakeholders and Business analyst are held in order

to determine the requirements with respect to the following aspects: Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during the requirements specification phase. Requirements are analysed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.
Finally, an entry in the requirement backlog is created which serves the purpose of guideline for the next phase of the model. The testing team follows the Requirement Testing Life Cycle and starts the Test Planning phase after the requirements planning is completed.
Plan Implementation: In this phase, the implementation of a requirement will be planned. The development team needs to determine which modules are affected by the new requirement. They will also estimate the effort required for implementing the requirement in the system. There are usually plenty of requirements and not all of them can be implemented at once. Hence, the team needs to schedule the implementation of each requirement based on effort and availability of resources.
Implementation: On receiving system design documents, the work is divided into modules / units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer.
Testing: After the code is developed it is tested against the requirement specification to make sure that the product is solving the needs addressed and gathered during the requirements phase. During this phase all types of functional testing as unit testing, integration testing, system testing, and acceptance testing are done as well as non-functional testing are also done.
Deployment: After successful testing the product is delivered / deployed to the customer for their use.
As soon as the product is given to the customers, they will first do the beta testing. If any changes are required or if any bugs are caught, then they will report it to the engineering team. Once those changes are made or the bugs are fixed then the final deployment will happen.

2.3 Stakeholders in Requirements Engineering

Stakeholder analysis is a process of identifying a person, group or a company which can affect or get affected by a decision, activity, or the outcome of the software project. It is important in order to identify the exact requirements of the project and what various stakeholders are expecting from the project outcome [10].

2.3.1 Internal Stakeholder

An internal stakeholder can be a person, group or a company that is directly involved in the project. For example :

Project Manager: Responsible for managing the whole project. Project Manager is generally never involved in producing the end product, but he / she controls, monitors and manages the activities involved in the production.
1. Responses for project
2. Represents the customer and other stakeholders
3. Sets the directions for whole project
4. Acquires / distributes resources
5. Coordinates friction for all of stakeholders

Team Leader (Manager):
1. Takes input from PO (Purchase Order)
2. Shields team from external interferences
3. Coordinates the team / communication
4. Removes obstacles
5. Promotes Agile / coaching
6. Coordinates friction for the whole team

Requirement Engineer: According to product planning or project requirements, Requirement Engineer cooperates with marketing, sales planning and other departments to analyze, clarify and optimize user needs, and writes a demand analysis report based on original requirements, and collaborates with the technical department to design prototypes
1. Identifies the need for needs, and can propose reasonable improvements to provide business units with better cost-effective solutions
2. Refines demand coordination between users and R & D
3. Assesses the costs and benefits of demand
4. Collectes user satisfaction and varies feedbacks on the system, and regularly summarizes and suggestes improvement plans

Project Team
1. Plans the individual projects / estimates

2. Works on all aspects of product development
3. Transforms a requirement into a product
4. Communicates
5. Reports obstacles

Company: The Organisation who has taken up the project and whose employees are directly involved in the development of the project. Provides or distributes resources for the project

Funders: Provide funds and resources for the successful completion of the project

Business Analyst: Communicates with customers and let them understand data-driven processes and how they can make products, services, software, and hardware more efficient and add value. They must articulate these ideas, but they also must balance them with technical feasibility and financial and functional rationality.

Software Architect: This job responsibility is to transform the customer's needs into a standardized development plan and text during the development of a software project, and to formulate the overall structure of the project, and guide the entire development team to complete this plan.

2.3.2 External Stakeholder

An external stakeholder is the one who is linked indirectly to the project but has significant contribution in the successful completion of the project. For example:

Customer: Specifies the requirements of the project and helps in the elicitation process of the requirement gathering phase. Customer is the one for whom the project is being developed.

Supplier: Supplies essential services and equipment for the project.

Government: Makes policies which help in better working of the organisation.

III. AGILE METHODS AND REQUIREMENTS ENGINEERING

3.1 Overview on Agile Methods

What is an agile method?

Agile methods are approaches to develop software. They aim to give an organization the ability of delivering fast, with high quality products and the highest priority to satisfy the customer. It can be done by delivering early and continuous valuable software. And now that Agile is also applied outside software and needed to what does it mean to be agile e.g. in Human Resources (HR), in marketing and sales [11].

When was it born?

In 2000, Agile Manifesto and the Twelve Principles were published. The Manifesto has four main values:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan
- From these four main values, the following principles are formulated. Those principles represent what does it mean to be agile in detail:
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Customer satisfaction by rapid, continuous delivery of useful software
- Even late changes in requirements are welcomed
- Close daily cooperation between businesspeople and developers
- Face-to-face conversation is the best form of communication
- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity
- Self-organizing teams
- Regular adaptation to changing circumstances

There are multiple interpretations of agile methodology. These interpretations are different but in general they focus on delivering products that matches budget, time with high quality and customer satisfaction. Here is the requirement life cycle of the most popular agile methods:

3.2 eXtreme Programming (XP)

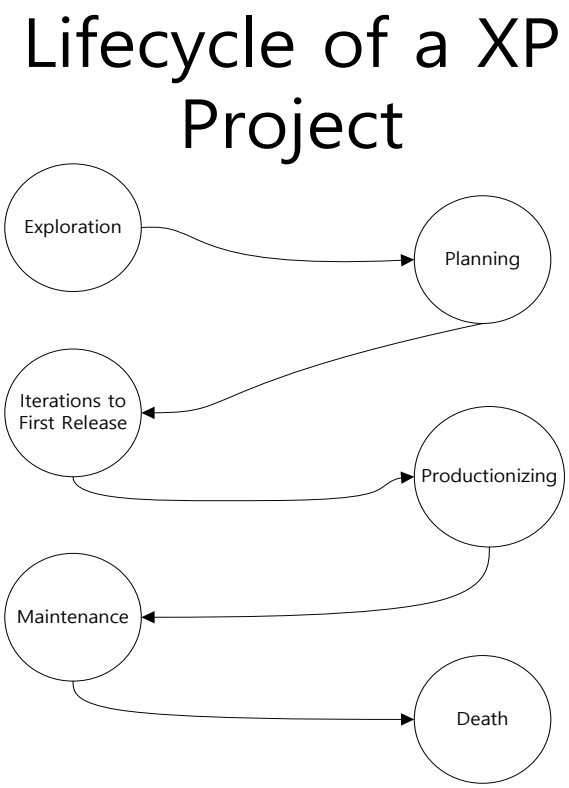

Figure 2. Life Cycle of a XP Project

What is eXtreme Programming (XP)?
Extreme programming is a lightweight and smart software development method. It depends on values of correspondence, simplicity, courage [12, 13], and criticism. It breaks down complex development processes into relatively simple small cycles. Through active communication, feedback and other series of methods, developers and customers can be very clear about the development progress, changes, unresolved issues and potential difficulties, and adjust the development process in a timely manner according to the actual situation.

When was it born?
Extreme Programming (XP) [14] was proposed by KentBeck in 1996. It is a software engineering methodology and is one of the most productive methodologies in agile software development. As with other agile methodologies, the fundamental difference between Extreme Programming and traditional methodologies is that it places more emphasis on adaptability and difficulties. In March 1996, Kent finally introduced a new software development concept, XP, into a project for DaimlerChrysler. XP is suitable for small team development.

What are its advantages
Extreme Programming is successful because it stresses customer satisfaction. Instead of delivering everything you could possibly want on some date far in the future, this process delivers the software you need as you need it. Extreme Programming empowers your developers to confidently respond to changing customer requirements, even late in the life cycle.

3.3 Scrum

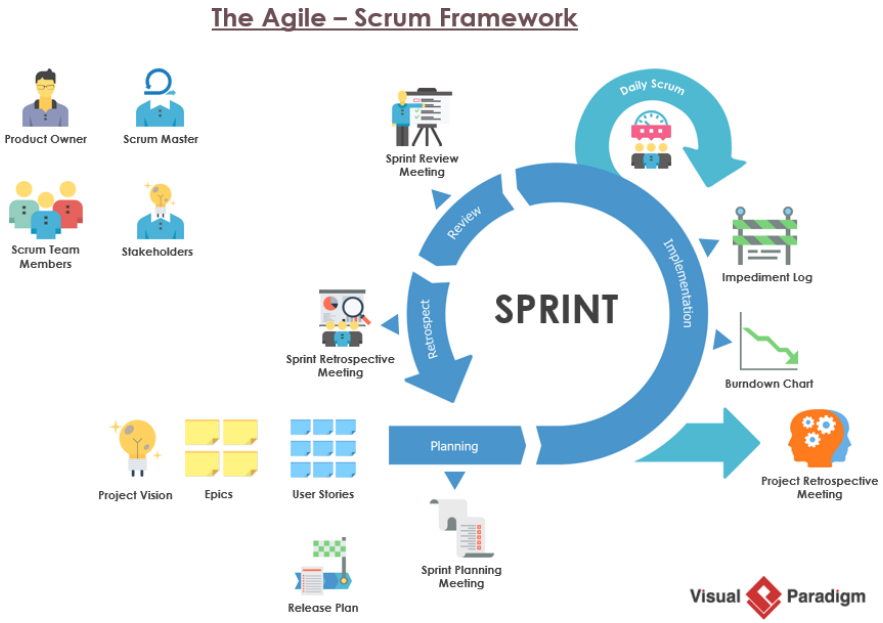

Figure 3. The Agile-Scrum Framework [15]

What is Scrum?

Scrum is an agile process framework for managing complex knowledge work, with an initial emphasis on software development, although it has been used in other fields and is slowly starting to be explored for other complex work, research and advanced technologies. It is designed for teams of ten or fewer members, who break their work into goals that can be completed within timeboxed iterations, called sprints, no longer than one month and most commonly two weeks, then track progress and re-plan in 15-minute time-boxed stand-up meetings, called daily scrums.

When was it born?

Hirotaka Takeuchi and Ikujiro Nonaka introduced the term scrum in the context of product development in their 1986 Harvard Business Review article.

In the early 1990s, Ken Schwaber used what would become Scrum at his company, Advanced Development Methods, while Jeff Sutherland, John Scumniotales and Jeff McKenna developed a similar approach at Easel Corporation, referring to it as scrum.

Ken and Jeff worked together to integrate their ideas into a single framework, Scrum. They tested Scrum and continually improved it, leading to their 1995 paper, contributions to the Agile Manifesto in 2001, and the worldwide spread and use of Scrum since 2002.

What are its advantages?

Surm is a lightweight, iterative and incremental framework for managing complex work. The framework challenges assumptions of the traditional, sequential approach to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved.

3.4 Kan Ban

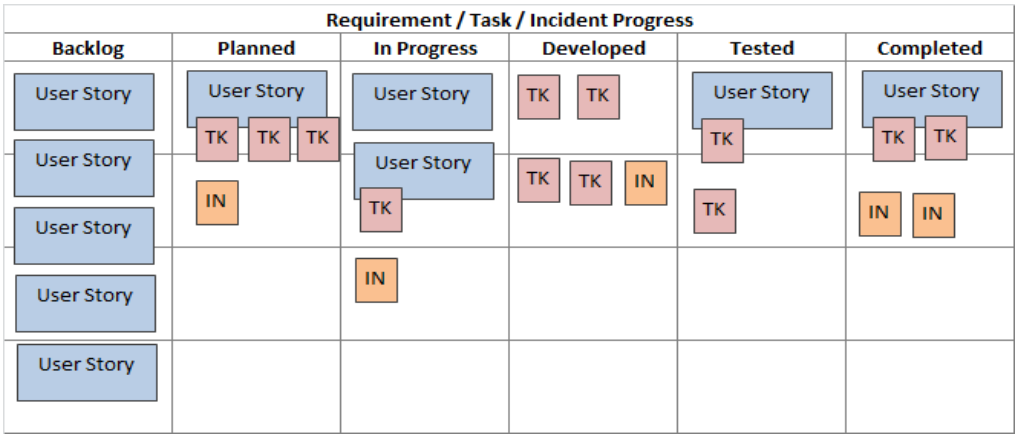

Figure 4. Kan Ban workflow

What is Kan Ban?

Kanban – literally translated from Japanese means a signboard or billboard – started life as a scheduling system in the motor industry and is now one of the fastest growth areas in the agile market. It′s easy to grasp, simple to implement and costs pretty much nothing. A big advantage to Kanban is that it can be used by individuals or teams to manage their workload or on full-scale projects. It′s a brilliant introduction to working in an agile way and there′s more to it than meets the eye [16]. Agile development draws on the visual management concepts behind it. Kan Ban makes project management the biggest visualization, but Kan ban can manage the process, record the details and history of the user story.

When was it born?

Kanban uses the rate of demand to control the rate of production, passing demand from the end customer up through the chain of customer-store processes. In 1953, Toyota applied this logic in their main plant machine shop.

What are its advantages

- Flexibility
- Maximum visualization
- Troubleshoot team communication
- Focus on continuous delivery
- Reduction of wasted work / wasted time
- Increased productivity
- Increased efficiency
- Team members' ability to focus

3.5 Gap Analysis

The agile methods presented in the previous section aim at managing requirements for agile projects. There are some differences to classical requirements engineering which will be described in this section.

3.5.1 Requirement

(a) eXtreme Programming (XP)

In XP, they use plans to expend requirements at the beginning of the development. The team will do an activity called Software Release Planning: Customers articulate requirements, and developers estimate development

costs and risks. The client develops a rough project plan based on development costs, risks, and the importance of each requirement. The initial project plan is not necessary (or impossible) to be very accurate, because the development costs, risks, and importance of each requirement are not static. Moreover, the plan will be continuously adjusted to be precise during the implementation process [17].
In the development process, like scrum, XP needs to create an iteration plan, but the iteration length of a Sprint of XP is roughly 1 ~ 2 weeks, and the iteration length of Scrum is generally 2 ~ 4 weeks: customers can always see the developer′s work done, and can change requirement in time. In an iteration of XP, if a User Story has not been implemented, you can consider replacing it with another requirement. The principle of replacement is that the amount of time required to implement the requirements is equal.

(b) Scrum
Backlog is a special term in Scrum, which means a collection of to-do items. The Product Backlog is a quantified user requirement that expresses itemized requirements that need to be developed. The Sprint Backlog (task list) is a task that needs to be completed in one iteration. It is also the most used Backlog in the development process. It is very detailed.
In Scrum they don′t use Demand analysis reports, as we know, Agile development follows a principle that everything that is finally done is meaningful to customers. In Scrum, The Product owner create Backlogs, every Backblog that is completed by Sprint is a function that allows users to use it completely. For the developer, he will use the Demo method to show the results, and he will attach great importance to quality. In every sprint, the develop team will create a Sprint Backlog to show the tasks.
Moreover, before every sprint is finished, no requirements can be added, and the Scrum Master is strictly controlled, and the development team is not allowed to receive interference. But in XP, you can consider replacing it.

(c) Kan Ban
Both Scrum and Kanban are process tools, but Kanban places more emphasis on visualizing workflows, limiting ongoing work (WIP), measuring and managing processes, and clarifying process policies. Using (theoretical) models to identify opportunities for improvement, Kanban does not use demand analysis reports, but uses user cases to make backlogs. Compared with scrum, Scrum limits WIP by iteration and Kanban limits WIP by process status [18] .
The core of the Kanban board are the to-do items. These individual items are all delivery focused and must deliver business value directly or indirectly. For example, setting up a Kanban board is a legitimate item but a meeting to discuss the to-do items is merely part of the main job. The items are business delivery focused and not centred on activities. If an item on the backlog does not contribute to business goals, it should be removed.

3.5.2 Life cycle
(a) eXtreme Programming (XP)
We can describe Extreme Programming from the perspective of life cycle, first, start off by describing the desired results of the project by having customers define a set of stories. As these stories are being created, the team estimates the size of each story. This size estimate, along with relative benefit as estimated by the customer can provide an indication of relative value which the customer can use to determine priority of the stories.
If the team identifies some stories that they are unable to estimate because they don′t understand all of the technical considerations involved, they can introduce a spike to do some focused research on that particular story or a common aspect of multiple stories. The spike-time can happen before the beginning of normal iterations,this time used to do some deeply research on project.
Next, the entire team gets together to create a release plan that everyone feels reasonable. This release plan is a first pass at what stories will be delivered in a particular quarter, or release. The stories delivered should be based on what value they provide and considerations about how various stories support each other. Then the team launches into a series of weekly cycles. At the beginning of each weekly cycle, the team (including the customer) gets together to decide which stories will be realized during that week. The team then breaks those stories into tasks to be completed within that week.
At the end of the week, the team and customer review progress to date and the customer can decide whether the project should continue, or if sufficient value has been delivered.

(b) Scrum
The Scrum Life cycle consists of a series of Sprints, where the end result is a potentially shippable product increment. Inside of these sprints, all the activities necessary for the development of the product occur on a small subset of the overall product.

Below is a description of the key steps in the Scrum Lifecycle:

- Establish the Product Backlog.
- Planning. The product owner and development team conduct Sprint Planning. Determine the scope of the Sprint in the first part of Sprint Planning and the plan for delivering that scope in the second half of Sprint Planning.
- Implementation. As the Sprint progresses, development team perform the work necessary to deliver the selected product backlog items.
- Daily scrum. On a daily basis, the development team coordinate their work in a Daily Scrum.
- Review. At the end of the Sprint the development team delivers the Product Backlog Items selected during Sprint Planning. The development team holds a Sprint Review to show the customer the increment and get feedback. The development team and product owner also reflect on how the Sprint has proceeded so far and adapting their processes accordingly during a retrospective.
- Retrospect. The Team repeats steps 2–5 until the desired outcome of the product have been met.

The first and second (Generate ideas, Description) items of the traditional requirement life cycle that are not clearly present in the scrum lifecycle, and the sixth step(Retrospect) is an extra step than the traditional requirement life cycle, it means Scrum can work in Sprint progresses, and this is one of advantages for Scrum.
This does not explain the nature of Scrum or agile product development in depth. However, the ability to manage changing priorities, increased productivity, improved project visibility, improved team morale, higher effectiveness, quality, and business stakeholder satisfaction with this approach speak for themselves [19].

(c)Kan Ban

The first and second (generate ideas, description) items of the traditional requirement life cycle that are not clearly present in the Kan Ban life cycle. Planning is same like the third items of the traditional, In Progress and Developed are consistent with Step 4 (Implementation).
Kanban does not specify a time limit iteration, it can be planned, released, and process improvement separately. It can be event driven instead of a time limit, and it uses the development cycle as the measurement data for plan and process improvement.

3.5.3 Roles

(a)eXtreme Programming (XP)

In XP, they don′t have a clear need for engineers and business analysts, their work has been split into other roles on the team, Customers directly submit user stories and requirements to the team.
The customer in XP is the other half of the essential duality of extreme programming. The programmer knows how to program. The customer knows what to program. But the customer is willing to learn just as much as the programmer is. Being an XP customer is not easy. There are skills you have to learn, like writing good stories, and an attitude that will make you successful.
The best customers are those who will actually use the system being developed, but who also have a certain perspective on the problem to be solved.
Since a lot of testing responsibility lies on the shoulders of the programmers, the role of tester in an XP team is really focused on the customer. You are responsible for helping the customer choose and write functional tests. If the functional tests aren't part of the integration suite, you are responsible for running the functional tests regularly and posting the results in a prominent place. They have done some parts of BA and requirements engineer in normal Requirement Development. Because they can selectively prioritize testing based on feasibility.

(b)Scrum

Different from the traditional requirements analysis process, Scrum don′t have requirement engineers and business analysts, but the Product Owner will finish their duty.
Product Owner, the one who can determine "What the team will do", is responsible for docking with customers and create Backlog.
Scrum Master, Scrum Master works closely as a Team Leader and Product owner, and he can help team members in a timely manner. The ScrumMaster needs to know what tasks have been completed, which tasks have begun, which new tasks have been discovered, and which estimates may have changed. The ScrumMaster needs to update according to the above situation to reflect the daily workload and how many outstanding burndown charts (Burndown Chart) [20].

(c)Kan Ban

In fact, there is not too much clear role division in the Kanban process. Kanban is shared by the entire team and is responsible for all tasks that need to be delivered. Although some teams have hired agile coaches, unlike Scrum, there is no "Kanban Master" that makes everything run smoothly.

IV. ROLE IN AGILE REQUIREMENTS ENGINEERING

4.1 Roles and responsibilities

4.1.1 Agile requirements engineer

The agile requirements engineer is the designated role for specific activities to ensure that requirements engineering is implemented in agile development.

An agile requirements engineer is the bridge between customers and developers in a distributed environment. A good analysis of demand is the key to whether the product can meet the needs of users. Requirements engineers control the vane of project development based on understanding users and technology.

4.1.2 Responsibilities

Collect requirements: The purpose of collecting requirements is to determine the needs of various aspects of the target system. The main tasks involved are to establish a method framework for obtaining user needs, and to support and monitor the process of obtaining requirements.

Specify requirement: In this stage, as a requirement engineer, you have known the requirements of your customer; you should use some models and diagrams (use case diagrams, activity diagrams, sequence diagrams etc.) to create Requirements Specification. The requirement specification is used as the production basis for customer reference, future development, and testing. Therefore, your work results are basically mapped on this requirements specification. This document is the top priority and is the basis for judging your work.

Update requirement: Requirements update is a review of the work in the requirements specify phase to verify the consistency, feasibility, completeness, and validity of the requirements documents.

Conduct customer requirement surveys and organize customer needs, responsible for demonstrating the completed project module to customers and collecting opinions on the completed module.

Manage requirement status: In the process of requirements management, agile requirements engineers can plan priorities based on the content of the requirements, and prioritize important requirements, and make timely adjustments based on feedback during the development process

4.2 RE in Scrum

Each traditional Scrum Team includes three key roles: Product Owner, Scrum Master and Scrum Team. We try to substitute the four responsibilities of agile requirements engineer and agile requirements engineer′s role into traditional scrum development.

"Collecting requirements" can belong to Product Owner (PO) and Agile Requirement Engineer. In a traditional Scrum team, the project manager has the opportunity to communicate with the business user. We decide to add agile requirements engineers on this basis, a role that can be both customer-facing and development team-facing to complete the key "requirement in" link.

"Specify requirement" belongs to Product Owner (PO), supported by the agile requirements Engineer. The team uses the backlog to specify requirements, the product owner maintains the Product Backlog and ensures that everyone knows what is included and its priority. The product owner may need the support of others, such as an agile requirement engineer.

"Updating requirement" belongs to Agile Requirement Engineer and ScrumMaster (SM). When a business user creates a new requirement, he will submit a requirement description to Agile Requirement Engineer, and then Agile Requirement Engineer will discuss the details of the new requirement with the ScrumMaster and add the new requirement to the next sprint.

4.3 RE in eXtreme Programming (XP)

Every role in scrum can "Manage requirement status". Development team and ScrumMaster can manage requirements through Scrum meetings which all roles are present. The product owner and requirement engineer can change the priority status of requirements according to the actual situation.

The Roles that have been found effective in Extreme Programming are Developer (also called Programmer by some teams), Customer, Manager (also called tracker), and Coach. When we add the four responsibilities of agile requirements engineer and agile requirements engineers in XP, we can redefine their mission:

Customer and agile requirements engineers are responsible for "collecting requirements" and "Specify requirements". The customer will write the necessary and sufficient details in the required story. Customers can

become potential agile requirements engineers. After collecting user stories, agile requirement engineer sorts the requirements by priority and estimates the iteration time required to realize these user stories.

"Updating requirement" belongs to Manager. Managers will schedule and conduct release planning and iteration planning meetings, and participate in assessments with the team to provide feedback at the team and individual levels on how reality fits previous estimates, which ultimately helps them make better estimates next time.

Every role in XP can "Manage requirement status". Face-to-face communication is the first choice. Through the management of the status of the meeting and the realization of pairing programming, not only team members will participate, but customer representatives will always be on site.

4.4 RE in Kanban

In KanBan, roles are not explicitly specified, and they are always developed as a team. So we can substitute the agile requirements engineer into the Kanban development team member.

Agile requirements engineers are responsible for "collecting requirements", "Specify requirement" and " Updating requirement": Agile requirements engineers can obtain the requirements from the company's product department, and then enter the requirements into the backlog. During this process, the tasks should be reasonably released according to the team's situation and the appropriate priority should be set. After the project goes online, new requirements may be generated. Agile requirements engineers are always responsible for updating new requirements into the backlog.

Every role in Kanban can "Manage requirement status". Not only Agile requirements engineers can manage requirement status, but also the programmer can manage that. Actual Cycle Days is the time it takes to complete a task. If this time differs greatly from Estimated Cycle Days, the entire team will have to review and summarize what went wrong and change the details of requirement.

V. CASE STUDY

The extended version of Scrum as described in section 4.1 has been applied in a global logistics company. There were three agile teams, each of which was developing an individual system:

- Station management system: planning and controlling logistics flows in the warehouses and hubs
- Customer order system: processing customer orders and billing
- Webservices API: Global API (Application Programming Interface) for large customers

Each of these systems is used by individual departments but need to be integrated. The API offers webservices for large customers so that they can submit shipping orders to the customer order system and track the shipments' status (station management system). Product owners and development teams are distributed across several countries.

Due to the limitations of agile methods, a fan-in requirements management process has been implemented:

- A global requirements manager and a central repository for requirements have been implemented
- All requirements need to be documented in the central repository. There are workspaces for each of the three teams, but each team member can read all requirements (including the ones of other teams).
- The requirements manager supports business stakeholders with writing requirements and sore them in the repository.
- The requirements manager coordinates the assignment and consolidation of requirements together with the product owners.
- The requirements manager collaborates with the scrum master and the agile teams.

A business user who wants a new requirement to be implemented will first align with the requirements manager. The business user needs to write the requirements description and the requirements manager supports him/her in case of any question or concern. The requirements manager then discusses the new requirement with all product owners. They can check whether a similar requirement already exist and to which team it should be assigned. This will foster re-use as several requirements can be bundled and the corresponding software feature be reused by others. Prioritization is done by all product owners, facilitated by the requirements manager.

Any kind of information concerning a requirement needs to be provided or updated in the central repository. Hence, the lifecycle of a requirement is maintained by the whole team in the tool. The development team needs to change its status to "in development" after starting to implement implementation. The development team is also responsible for updating the status for each phase according to the life cycle. Any kind of change in a requirement needs to be done by the person who is currently in charge of it. If the developer needs more details, he or she consults the requirements manager and the product owner so that the change can be done.

This approach proved to be quite successful in the company as it only required minimal changes in the organization and easily allowed for collaboration among distributed teams. The repository, furthermore, provided transparency on the development status of all requirements.

VI. SUMMARY

The present paper describes an approach to combine agile software development methods with classical requirements engineering. Agile methods are very popular today as they help with gaining early successes in software development by focusing on customer needs and reducing risk. However, they still have some restrictions in distributed environments and large projects. Instead of adopting one of the appearing frameworks for scaling agile methods, the approach in this paper starts with applying concepts and roles that are already common in today′s organisations. There are certified requirements engineers available that can support collecting and specifying requirements for agile teams. Such a requirements engineer can bridge geographical distances and distribute requirements over related projects.

The approach presented here is based on a literature review and practical experience by the authors. It has been adopted in an international company with the headquarter in Germany. Customers (i.e. the users of the system) are working globally and development teams are located in various countries. The agile requirements engineer need to collect requirements globally and assign them to the corresponding team. He/she also helps with describing requirements in more detail as business users are not experienced with formal descriptions. Managing requirements is a shared responsibility between all stakeholders (product owner, agile requirements engineer, development team).

Research is only at an early stage as the approach will have to be evaluated in further organisations. This will support improving the approach and make it more robust for a professional environment. We especially aim at introducing the approach to international companies (e.g. China). This will also show whether further requirements engineering concepts can be adopted so that agile methods can be extended, but without adding administrative overhead.

REFERENCES

[1]. F.Anwer,S.Aftab,S.S.Muhammad Shah and U.Waheed, ”Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum”, Int J. Computer Science and Telecommunications , Vol. 8, no. 2, 2017

[2]. TURK,D. Limitations of agile software processes. International Conference on Extreme Programming & Flexible Processes in Software Engineering. 2002

[3]. Uludag, M. , Kleehaus, M. , Dreymann, N. , Kabelin, C. , & Matthes, F. Investigating the Adoption and Application of Large-Scale Scrum at a German Automobile Manufacturer. 14th ACM/IEEE International Conference on Global Software Engineering. ACM, 2019

[4]. Uludag, M. , Kleehaus, M. , Caprano, C. , & Matthes, F. Identifying and Structuring Challenges in Large-Scale Agile Development Based on a Structured Literature Review. 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference. IEEE, 2018.

[5]. Mahmood Niazi. Improving the requirements engineering process: a process oriented approach, 2002.

[6]. Nuseibeh, Bashar, Easterbrook, & Steve. Requirements engineering: a roadmap. proc icse the future of software engineering may, 35--46, p. 1, 2000.

[7]. Isonkobong Udousoro. Effective Requirement Engineering Process Model in Software Engineering. 8(1), p. 3, 2020.

[8]. Karl Wiegers, Joy Beatty, ”Software Requirements (Developer Best Practices 3rd Edition” Microsoft Press; 3 edition, p. 543-545, 2013.

[9]. Jilvan Pinheiro, ”Software Development Life Cycle (SDLC) phases” ,https://medium.com/@jilvanpinheiro/

[10]. Ralph Young, & Ralph Young. The requirements engineering handbook.. DBLP, 2003.

[11]. Saad Abdullatif Alhuzami. “Engineering in Agile Software Development”. King Saud University.

[12]. Zainab Sultan, Rabiya Abbas, Shahid Nazir, & S. Asim, Analytical review on test cases prioritization techniques: an empirical study. international journal of advanced computer science & applications, 8(2), 2017.

[13]. Sehrish Alam, Shahid Nazir Bhatti & Dr. Amr Mohsen Jadi. Impact and Challenges of Requirement Engineering in Agile Methodologies: A Systematic Review. 8(4), 2017.

[14]. Dominik Maximini. The Scrum Culture. Springer International Publishing, 2015.

[15]. Scrum vs Waterfall vs Agile vs Lean vs Kanban , https://www.visual-paradigm.com/scrum/scrum-vs-waterfall-vs-agile-vs-lean-vs-kanban/

[16]. Jutta Eckstein ,John Buck, ”Company-wide Agility with Beyond Budgeting, Open Space &Sociocracy: Survive & Thrive on Disruption”, CreateSpace Independent Publishing Platform, 2018.

[17]. Erich Gamma,”Extreme Programming Explained”, p. 6, http://index-of.co.uk/.

[18]. Ahmad, M. O. , Dennehy, D. , Conboy, K. , & Oivo, M. Kanban in software engineering: a systematic mapping study. Journal of Systems & Software, S0164121217302820, p. 5. 2017.

[19]. Boehm, Barry, Turner, Richard. “Balancing Agility and Discipline: A Guide for the Perplexed” lecture notes in computer science, 108(7):316 - 325, 2004.

[20]. Moreira, Mario E, ”The Agile Enterprise : Building and Running Agile Organizations / by Mario E. Moreira”, Pearson Education Limited, p. 82-83, 2015.