ADM: An Agile Template for Requirements Documentation

Hind Kalfat[1,2], Mourad Oussalah[2] and Azeddine Chikh[1]
[1]*LRIT Laboratory, Tlemcen University, Tlemcen, Algeria*
[2]*LS2N Laboratory, Nantes University, Nantes, France*

Keywords: Agile Methods, Software Requirements Documentation, Requirements Engineering, Metamodeling, Metrics.

Abstract: Requirement documentation is one of the main activities conducted during software requirements engineering which contributes to the success of the project if done effectively. In agile, teams tend to produce minimal documentation because they are much more focused on software development. Yet, this is also due to the lack of clear guidelines on what needs to be documented and how it should be done. This paper proposes an approach based on three key axes: documentation, agile, and metrics. We use in our design a metamodel to provide various document templates that are tailored to specific user needs. These templates can be adapted to different contexts such as traditional or agile development. In order to address the issue of requirements documentation in agile context, we propose a custom document template to help agile teams while creating the software requirements documentation.

1 INTRODUCTION

Software requirements engineering (SRE) is considered as the most important phase in the software development cycle, this is why requirements must be identified, documented and managed effectively. However, SRE applied in agile context faces a significant documentation problem (Sherif et al., 2022).

First, Agile teams are mainly based on communication, they consider documentation activities as a waste of time which leads them to produce poor documentation (Theunissen et al., 2022) and this can be challenging in many situations (e.g. distributed teams, large teams, team member leaving the company, complex projects, etc.) (Heck and Zaidman, 2018)

Second, the documentation solution used in traditional development cannot work with agile approach and this is due to its heavy process and rigid structure. Therefore, this leads agile teams to consider documentation as a burden and end up neglecting it.

The main objective of this research is to propose a new software requirements documentation (SRD) process which allows to deliver a new SRD product that both fit with agile principles. Indeed, these process and product must be measurable to make sure that they fit to agile principles but also to compare them with traditional and other existing agile solutions and this would require the use of specific metrics.

In this research work we will deal with three fundamental domains of knowledge which are: **A**RE for Agile Requirements Engineering; SR**D** for Software Requirements Documentation; RD**M** for Requirements Documentation Metrics.

The solution was named ADM by combining the three terms: Agile, Documentation and Metrics.

To achieve our objective, we must first understand how agile methods work. Thus, we choose one single agile method and examine its SRE process in depth, in order to define all the documentation needs of agile teams. Therefore the documentation process is derived from the ARE process.

In order to describe how this requirements documentation can be designed, we propose a metamodel gathering all the essential concepts. Then we explain how to instantiate it in order to get different templates of documents that are suited to the needs of its creators. We also suggest a custom document template that may be useful for agile teams when designing the requirements document for their projects.

The rest of this document is structured as follows: Section 2 presents the main concepts of this research study. Then, the literature of documentation in traditional and agile is reviewed in section 3. In section 4 we describe the approach of documentation

Kalfat, H., Oussalah, M. and Chikh, A.
ADM: An Agile Template for Requirements Documentation.
DOI: 10.5220/0012122400003538
In *Proceedings of the 18th International Conference on Software Technologies (ICSOFT 2023)*, pages 494-501
ISBN: 978-989-758-665-1; ISSN: 2184-2833

in agile called ADM. In section 5 we discuss our findings. Finally, in Section 6, we conclude the paper.

2 BACKGROUND

This section provides the background of the three knowledge domains of this research.

2.1 Agile Requirements Engineering

In order to understand how requirements engineering (RE) works in agile development context, we set a single agile method on which to apply our research. We chose Scrum as it is the most popular agile development method (Sutherland & Sutherland, 2014).

In this research work, we adopt the Software Process Engineering Meta-Model (SPEM) introduced by the Object Management Group (OMG) to describe processes (OMG, 2005). SPEM and Scrum share similar foundations. SPEM focuses on Activity, Role, and Work product, as shown in Figure 1. While Scrum focuses on Ceremonies, Roles, and Artefacts. Despite different terminology, it is obvious that ceremonies correspond to activities and artefacts refer to work products. In this research we use the concepts described in the Scrum guide (Schwaber and Sutherland, 2011). However, we add two ceremonies that we consider important as they are related to SRE, which are: Product envisioning and release planning.

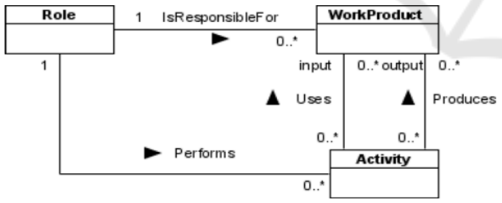

Figure 1: Conceptual model of SPEM 1.1.

The activities commonly found in ARE include: Discovery, refinement, prioritization, review, and documentation. There is a correspondence between Scrum ceremonies and ARE activities. As shown in Figure 2, each ARE activity is associated with its corresponding Scrum ceremony. Documentation, on the other hand, is needed several times in a single iteration and can be applied in parallel with the other activities. In Figure 2 we have selected the potential points where documentation will be needed, either for updating or for consultation. We notice here that the documentation is the pivot of all the ARE process.

2.2 Software Requirements Documentation

In this research work, documentation is treated through two dimensions: the process dimension and the product dimension. The first one is defined by the 3 concepts: Activity, role and product (product at micro-level) and allows to generate the second dimension which represents the final product of the documentation (product at macro-level) and which is no more than a composition of products from the micro-level.

In this research we rely on the knowledge management cycle (Girard and Girard, 2015), from which we derive the following documentation activities to define the SRD process: Create/ Update: tasks that feed SRD; Organize: tasks that structure, transform SRD content; Share: set of tasks that share and exchange parts of SRD; Access: tasks that retrieve SRD content.

2.3 Requirements Documentation Metrics

Metrics are a way to measure quantitatively the performance, quality or other characteristics of a process or system (de Oliveira, 2020).

The Goal Question Metric (GQM) is a goal oriented approach proposed by Basili et al. (Basili et al., 1994). It is divided into three levels:

- Conceptual level (Goal): At this level a measurement goal must be set;
- Operational level (Question) Here a set of questions must be asked to achieve the goal;
- Quantitative level (Metric): The metrics for answering the above questions must be defined.

3 RELATED WORKS

The SRE in general and SRD in particular are seen differently in each software development context: traditional and agile. A summary of related work to SRD in traditional RE (TRE) and ARE is presented in the subsections below.

3.1 Documentation in TRE

In TRE process, such as in Waterfall approach, documentation is created, updated and maintained by the business analyst (Robertson and Robertson, 2012). This is done at the very beginning of the

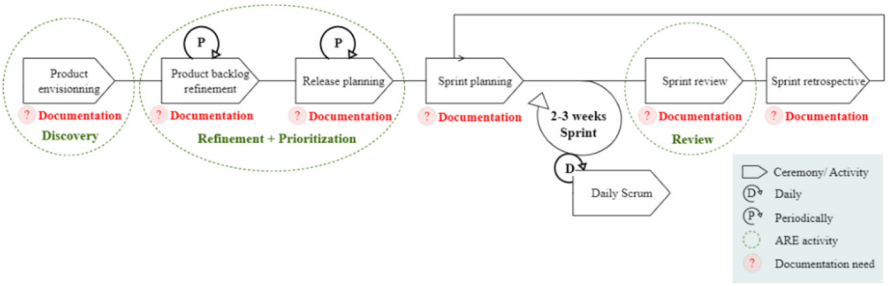

Figure 2: Agile requirements engineering activities in Scrum.

project, during the RE process. These activities are carried out sequentially, which means that all requirements are captured and documented and then carefully followed by the development team throughout the rest of the project.

The design of TRE must respect certain desirable characteristics to ensure its quality. Jalote (Jalote, 2012) identified and measured 24 quality attributes for requirements documents.

The business analyst should follow templates to guide him with what should be included in the documentation. The IEEE 830 is a widely recognised standard which offers a complete and simple structure for software requirements specification (SRS) (IEEE Computer Society et al., 1998). This template should be used in conjunction with other documents to describe other types of information like business requirements, software reviews, etc.

3.2 Documentation in ARE

In the literature, there are very few studies that address SRD activities in ARE. Instead, documentation is guided by a set of practices supported by the teams (comprising both RE-related activities and artefacts) and not by a sequential process. Here we collect the most common practices in agile.

User stories are the most used requirements notation in agile projects (Jarzębowicz and Połocka, 2017). A user story is described by the following template: “As a <Role> I want <Goal>, so that <Benefit>”. To further detail a requirement and better understand it, agile teams rely on face to face communication.

Personas help project team to gain common understanding concerning user and stakeholder. This technique is used to describe an imaginary person that will represent a certain target group of users. Hess et al. (Hess et al., 2017) observed that persona does not bring details about role-descriptions or descriptions of other relevant stakeholders who do not use the system but might have a strong influence on certain of its requirements.

Ramesh et al. (Ramesh et al., 2010) identify six agile practices for RE which include prototyping for documentation. This practice is used to communicate between the development team and the customer, reducing the margin of error and allowing the customer to provide feedback.

Based on this state of the art, we believe that it is necessary to find a new approach describing the process for creating SRD in agile context.

4 ADM APPROACH

The ADM approach proposed in this paper is a combination of three concepts: (i) Documentation, (ii) Agility and (iii) Metrics. In order to describe ADM we use metamodeling based on the following three levels of abstraction defined by the OMG (ISO/IEC, 2007):

- M2: The metamodel that defines all the ADM concepts and the relations between them;
- M1: Document templates that are instantiated from the metamodel;
- M0: Documents created for real projects through the application of ADM by agile teams.

4.1 ADM Metamodel (M2 Level)

First, we analysed the application of the RE process in the agile Scrum framework. This helped to understand how each activity is executed and what it needs as information for its proper functioning. We then identified the documentation needs in this process to be able to improve it through ADM solution. From here we can conclude that ADM is derived from ARE.

Figure 3 represents the metamodel that can be instantiated to create different templates of documents. The project team has the flexibility to

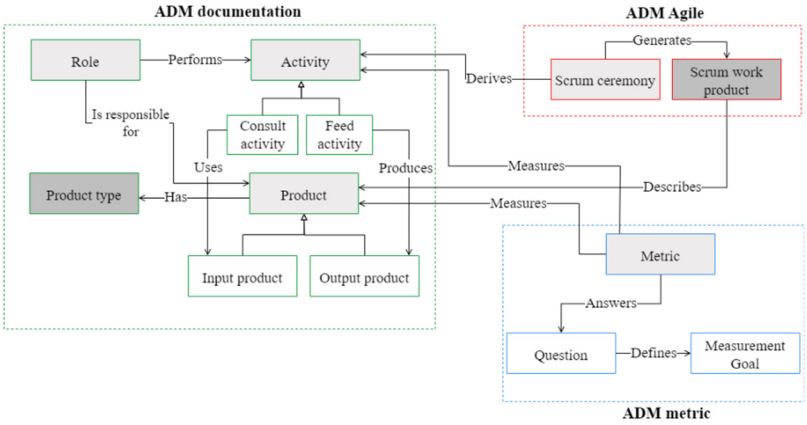

Figure 3: ADM Metamodel.

choose the most suitable template based on their specific needs. The sub-sections below present the concepts of the metamodel.

4.1.1 ADM Agile

The A of agile in ADM is defined by the two following concepts:

- Scrum ceremony: represents any activity held during a Scrum process. In this research work we take into account all the ARE activities that can influence the SRD process;
- Scrum work product: this concept is used to show agile teams the link between the products they use to describe the requirements information and the ADM products.

4.1.2 ADM Documentation

The D of documentation in ADM is defined by the following four basic concepts:

- Activity: documentation activities in ADM are derived from Scrum ceremonies. They can be of two kinds: activities to update the documentation by adding or updating information and others to consult it by accessing existing information;
- Role: represents the roles involved in each activity taking into account that there is not a unique annotation to represent the requirements from the different points of view of project team members (Pires et al., 2011);
- Product: represents the documentation products at micro level. Input product are used in consult activity and output product are produced during update activity;
- Product type: this concept is introduced for better organisation of the documentation, it describes the proper writing style for each ADM product. When instantiated, it has defined values to be used and does not depend on the template of the document.

4.1.3 ADM Metrics

As we use the GQM approach for the measurement, the M of metrics in ADM is defined by the three concepts: Goal, Question and Metric which have already been explained in **section 2.3**.

4.2 ADM Document Template (M1 Level)

After creating the metamodel, we instantiate it to create a document template. A document template can vary according to the needs of its creator. Using the ADM metamodel we can obtain the following types of templates:

Existing Agile Templates: As mentioned before, in agile there are many techniques, each allowing to create a fragment of documentation. It would therefore be possible to instantiate the ADM metamodel to obtain for each of these techniques a different document template.

Custom Template: Table 3 in the appendix presents a custom template developed for agile methods, which is obtained by deriving from the Scrum activities the essential concepts defining the documentation process. It's worth noting that there may be other customised templates depending on the preferences of the team members. In order to define these templates, it is first necessary to establish the invariant products which represent the important elements to be included in the documentation, whatever the project to be developed.

Traditional Template: The traditional template is generally represented using IEEE 830 standard.

This section explains how the metamodel is instantiated. The instantiation is done according to the concepts represented in grey in Figure 3. Dark grey concepts are instantiated the same way regardless of the document template, while light grey concepts depend on the document template chosen.

4.2.1 Instantiation of Metrics

We have instantiated five metrics for the template we have chosen to describe using the GQM method as shown below.

- Goal: To define the goal we want to achieve we use the structure proposed by Basili (Basili and Rombach, 1988): Analyse documentation needs in Scrum for the purpose of proposing a new document template with respect to project team needs from the view point of agile team in the context of ARE.
- Question: the questions related to the defined goal are:
  a. How well is the documentation maintained?
  b. Who is involved in the documentation activities?
  c. Does the documentation product contain the necessary information to support ARE process?
  d. Is the documentation accessible to all the project team members?
  e. Does the documentation product match all the profiles of the project team?
- Metric: The metrics used to answer the previous questions correspond to a specific template that we have chosen to instantiate. These metrics are suited to the agile context (Kupiainen, 2015) and are divided into two as follows (see Table 1).

Table 1: ADM metrics.

Metric nature	Associated questions	Metric
Process Metrics	a	Maintenance
	b	Collaboration
Product metrics	c	Coverage
	d	Accessibility
	e	Adaptability to profiles

4.2.2 Instantiation of Scrum Work Product

Table 3 in the appendix lists Scrum products that are commonly used by agile teams and associates them with the corresponding Scrum ceremonies (Rubin, 2012).

4.2.3 Instantiation of Product Types

In order to better structure and organize ADM products we propose to classify them into families and sub-families such as:

- Deliverables: All information that is directly related to software product increments;
- Communication: information that flows between team members and assists in the development of software product increments;
- Best practices: The decisions made by team members to promote continuous improvement of work.

4.2.4 Document Template Example

Using an extract from the template provided in the appendix, we explain the instantiation of the remaining concepts, focusing in particular on the sprint review ceremony (see Table 3). During this Scrum ceremony, the development team accesses to the existing prototypes in order to present them to the customer. The customer gives his feedback on the product increment presented and include it directly to the documentation. The documentation activity here is to organize or, more precisely, to annotate, as the information created is used to annotate existing content. Finally, the customer may have new requirements to add for the next iterations, with the help of the product owner.

4.3 ADM Document (M0 Level)

In this section we list a set of values that should be followed by the team when creating documentation with ADM. A practical example is then presented, showing how ADM is used to create an agile and efficient documentation.

4.3.1 ADM Values

Based on the principle mentioned previously that agile is seen as a baseline and that SRD is derived from ARE, in this section we also use the agile values as a baseline and derive new documentation related values from them (Fowler and Highsmith, 2001).

From ‘People and their interactions more than processes and tools’ we can derive ‘Documentation is the team's memory’. The agile manifesto encourages the involvement of team members and their ability to communicate effectively with each other. However, it should be noted that verbal communication is subject

to amnesia. Documentation can thus play the role of memory to record efficiently any information belonging to this discussion between the team members. This memory can be either individual or collective. Thus, each team member can access both his own documentation space and the shared space to interact with others.

From 'Operational software more than exhaustive documentation' we can derive 'The documentation is adapted to its user's profile'. Documentation cannot be considered exhaustive if it is adapted to the needs of the users. This means that the content must continuously correspond to the context of its use but also to the profile of its user. The client, for example, will not have access to the same content as the development team.

From 'Collaboration with customers more than contractual negotiation' we can derive 'The customer is involved in the SRD process'. Documentation should contain a space for the customer where he can exchange with the team and give feedback on the project and software product.

From 'Adapting to change more than following a plan' we can derive 'The documentation process is continuous'. One of the main challenges of changing requirements is to keep the documentation up to date over time. Therefore, the documentation process must be continuous and conducted in parallel with the SRE process.

4.3.2 ADM Document Example

This section uses an example of a project mentioned in (Rubin, 2012) that explains the development stages of a product called Smart-Review4You (or Simply SR4U). Thus we instantiate the custom template to show how documentation is designed for SR4U, as shown in Table 2. However we do not use all the elements of the template as some of them are very specific to the project.

Table 2: Project documentation using ADM custom template.

Product Envisioning	
Role	• **Product Owner (PO)** : Roger • **Stakeholders** : Customer and SMEs
Activity	Create
Product	**Project goal**: differentiate Review Everything, Inc., in the marketplace. **Product goal** : identify, filter, and display online reviews that includes a trainable search agent **Epics:** • As a typical user I want to teach SR4U what types of reviews to discard so that SR4U will know what characteristics to use when discarding reviews on my behalf. **Stakeholder list:** typical user, sophisticated user **Estimation:** 3 months for release 1.0
Type	Deliverables
Product Backlog Refinement	
Role	• **PO**: Roger • **Stakeholders** : Customer and SMEs
Activity	Organize/Create
Product	**Detailed US:** • As a typical user I want to tell SR4U to ignore reviews that contain specific keywords that I feel show bias in a review so that I don't see any reviews containing those keywords.
Sprint Planning	
Role	Development team
Activity	Create
Product	**Tasks:** • Code the UI Hours = 7 • Automate testsHours = 8
Type	Communication
Product	focus on improving the user experience
Type	Communication
Sprint Retrospective	
Role	Scrum Master + Development team
Activity	Create and Share
Product	Allocate more time for testing in future sprints
Type	Best practice

5 DISCUSSION

We can conclude from the state of the art that in traditional documentation, there is a well-defined model to follow but the document produced is rigid and obviously the process of its creation is not suitable for agile. On the other hand, agile is informal with no structure imposed and based on the skills and knowledge of individuals (Dingsøyr et al., 2012). For each part of the documentation there is a certain number of practices to follow and it is up to the project team to choose the practices suitable for them, the order of documentation activities and the roles involved. If this may be easy to an experienced agile team it might not be the case for a young team (Hoda et al., 2010).

If we measure the instantiated process, which represents a document template, we find that: it involves all the roles for its execution; it's accessible to all agile team members; it's adapted to profiles of project team since it specifies for each role which documentation activities and products correspond to it; it includes the information we believe is relevant to ARE process; it considers documentation as important since it sees it as part of the ARE process.

6 CONCLUSION

This paper proposes a metamodel intended to support SRD activities which are conducted in agile context. During ARE process, teams produce knowledge that needs to be documented in order to be used later in the project. Therefore, by detecting the documentation needs present in ARE process we were able to instantiate the metamodel to describe in detail how SRD should be created and used and this by answering the following questions: what should be documented? What parts of SRD products should be used? Who is involved in SRD process? When and how requirements are documented? Then, just like the values in the agile manifesto, the proposed ADM approach is accompanied by a set of values that serve the team in documenting requirements.

Taking into account the flexibility offered by agile, we integrate the notion of metrics into the solution, allowing teams to adapt the new documentation process to their needs.

In conclusion, we plan to conduct a case study with an actual Scrum team to validate and demonstrate the practical application of ADM approach. Additionally, it would be interesting to extend this solution to the other phases of software development life cycle (SDLC) and not only SRE.

REFERENCES

Basili, V. R., & Rombach, H. D. (1988). The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on software engineering*, *14*(6), 758-773.

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The goal question metric approach. Encyclopedia of software engineering, 2, 528-532.

de Oliveira, K. M. (2020, June). Practices to Define Software Measurements. In *INFORSID* (pp. 77-92).

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of systems and software*, *85*(6), 1213-1221.

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software development*, *9*(8), 28-35.

Girard, J., & Girard, J. (2015). Defining knowledge management: Toward an applied compendium. *Online Journal of Applied Knowledge Management*.

Heck, P., & Zaidman, A. (2018). A systematic literature review on quality criteria for agile requirements specifications. *Software Quality Journal*, *26*, 127-160.

Hess, A., Diebold, P., & Seyff, N. (2017, September). Towards requirements communication and documentation guidelines for agile teams. In 2017 ieee 25th international requirements engineering conference workshops (rew) (pp. 415-418). IEEE.

Hoda, R., Noble, J., & Marshall, S. (2010, July). How much is just enough? Some documentation patterns on agile projects. In *Proceedings of the 15th European Conference on Pattern Languages of Programs* (pp. 1-13).

IEEE Computer Society. Software Engineering Standards Committee, & IEEE-SA Standards Board. (1998). *IEEE recommended practice for software requirements specifications* (Vol. 830, No. 1998). IEEE.

ISO/IEC/JTC 1/SC 32. ISO/IEC 19502:2005, Information Technology - Meta Object Facility. Multiple. Distributed through American National Standards Institute, 2007

Jalote, P. (2012). *An integrated approach to software engineering*. Springer Science & Business Media.

Jarzębowicz, A., & Połocka, K. (2017, September). Selecting requirements documentation techniques for software projects: a survey study. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)* (pp. 1189-1198). IEEE.

Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). Using metrics in Agile and Lean Software Development–A systematic literature review of industrial studies. *Information and software technology*.

OMG. Software Process Engineering Metamodel Specification, Version 1.1. Formal/2005-01-06, Object Management Group, 2005.

Pires, P. F., Delicato, F. C., Cóbe, R., Batista, T., Davis, J. G., & Song, J. H. (2011). Integrating ontologies, model driven, and CNL in a multi-viewed approach for requirements engineering. *Requirements Engineering*, *16*, 133-160.

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, *20*(5).

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley.

Sherif, E., Helmy, W., & Hassan, G. (2022). Agile Requirements Engineering's Challenges. *International Conference on Software and Data Technologies*.

Schwaber, K., & Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, *21*(1), 1-38.

Sutherland, J., & Sutherland, J. J. (2014). *Scrum: the art of doing twice the work in half the time*. Currency.

Theunissen, T., van Heesch, U., & Avgeriou, P. (2022). A mapping study on documentation in Continuous Software Development. *Information and software technology*, *142*, 106733.

APPENDIX

This section shows the instantiation of ADM at M1 level in Table 3 using as main sources (Rubin, 2012) (Sutherland & Sutherland, 2014) (Schwaber and Sutherland, 2011). The invariant products of the documentation that we deemed necessary for the use of this custom template are indexed by the asterisk symbol (*), while the other are considered optional.

Table 3: ADM instantiation – Custom template.

Agile		Documentation						Metric
Scrum ceremony	Scrum work product	Activity		Role	Product		Product type	Metric
		Update	Consult		Input	Output		
Product envisioning	Product vision + Roadmap + High-level features	Create	-	PO + Stakeholders	-	Project goal + Product goal* + Product vision + Project scope + Product scope + High-level user + stories (Epics)* + Stakeholders list* + User personas + Product roadmap + Estimation(Budget and time)*	Deliverables	▪ Maintenance ▪ Collaboration ▪ Coverage ▪ Accessibility ▪ Adaptability toprofiles
Product backlog refinement	Product backlog	Organize	-	PO + Stakeholders	Epics	User stories*	Deliverables	
		Create	-	PO + Stakeholders	-	US (for more details)	Deliverables	
		Create	-	PO + DT	-	Acceptance criteria	Deliverables	
		-	Access	PO + DT	PB items	-	Deliverables	
		Create	-	PO + stakeholders	-	Story points*	Deliverables	
		Organize	-	PO + DT + Stakeholders + SM	User stories +Story points	User stories	Deliverables	
		Share	-	PO + DT	-	Prioritized PB	Deliverables	
Release planning	Release plan	-	Access	PO + DT	Product vision + Prioritized PB	-	Deliverables	
		Create	-	PO + DT + Stakeholders	-	(Release goal* + Time estimation + Team velocity)	Deliverables + Communication	
			Access	PO + DT + Stakeholders	Story points		Deliverables	
Sprint planning	Sprint backlog + Sprint goal	-	Access	DT	PB items	-	Deliverables	
		Create + Share	-	PO + DT + SM + Customer	-	Definition of done + Sprint goal	Deliverables	
		Organize	-	DT	SB + Release goal + Sprintgoal	SB items	Deliverables	
		Create	-	DT	-	Tasks*	Communication	
		-	Share	DT		SB	Deliverables	
Daily scrum	Burndown chart	-	Access	DT	SB + US	-	Deliverables	
		Create + Share	-	DT + SM	-	Burndown chart + Difficulties + Solutions	Communication + Best practice	
Sprint review	Sprint plan	-	Access	DT + Customer	Prototypes	-	Deliverables	
		Organize (Annotate)	-	Customer	-	Feedback*	Deliverables + Communication	
		Create + Share	-	Customer + PO	-	New requirements + Feedback	Deliverables + Communication	
Retrospective	List of Improvement actions	-	Access	SM + DT + PO (optional)	Burndown chart	-	Deliverables	
		Create + Share	-	SM + DT	-	Sprint review report*	Best practice	

Product owner (PO); Scrum master (SM); Development team (DT); User story (US); Product backlog (PB); Sprint backlog (SB)