Requirement elicitation for adaptive standards development

Marion Toussaint [*,**] **Sylvere Krima** [***]
Allison Barnard Feeney [****] **Herve Panetto** [**]

[*] *Associate, NIST, 100 Bureau Drive, Gaithersburg, MD, 20899, USA, (e-mail: marion.toussaint@nist.gov)*
[**] *Université de Lorraine, CNRS, CRAN, 54000 Nancy, France*
[***] *Engisis LLC, 10411 Motor City Dr Ste 750, Bethesda, MD, 20817-1289, USA, (email: sylvere.krima@engisis.com)*
[****] *NIST, 100 Bureau Drive, Gaithersburg, MD, 20899, USA*

Abstract: The recent digitization of manufacturing, also referred to as the fourth industrial revolution (or Industry 4.0), heavily relies on information standards for the exchange and integration of digital data across manufacturers and their partners. Standards are complex projects with a long lifecycle, often developed in a predictive environment, which no longer aligned with a fast paced industry. Adaptive environments have been proven to be an answer to this problem, but most project requirements management solutions have not evolved to support this shift. In this paper, we discuss the new challenges brought by a shift to adaptive environments, and introduce a solution to offer better requirement elicitation during standards development management with increased traceability and visibility.

Keywords: information standards, requirement elicitation, requirement model, agile, adaptive development

1. INTRODUCTION

Information standards are powerful tools for innovation and productivity in many domains (Gallaher et al., 2004; Blind, 2009; Guasch et al., 2007). When it comes to manufacturing, information standards are seen as a key enabler to the digitization of the manufacturing industry (Fischer et al., 2015; Helu et al., 2017; Hedberg et al., 2016). But standards development is relatively complex. It is a long process that includes many stakeholders, from different organizations, geographically dispersed, and on a volunteer basis, making their contribution and participation irregular. The human resources available depend on the experts' schedules and their organizations' needs, which makes it difficult to have continuity and consistency in standards development. Moreover, because their participation is irregular, it becomes difficult for every member of the standard development process to keep up with the past and current activities of the development. To summarize, the standard development process is long, irregular, and difficult to plan.

Information standards are mainly developed using the predictive or waterfall model for project management. The waterfall model is a sequential development model, in which each phase of the development must be completed in order to proceed to the next phase (Balaji and Murugaiyan, 2012). In predictive models such as the waterfall model, the project requirements and deliverables are defined at the beginning of the project (PMI, 2017), and if some requirements need to be changed or added, they will most likely not be implemented in the current development iteration (Balaji and Murugaiyan, 2012). According to ISO, standards development iterations last between 18 and 48 months (ISO, 2017), which means that: i) in the best-case scenario, new requirements will be addressed up to 18 months after being identified, ii) in the worst-case scenario, new requirements will be addressed up to 48 months after being identified. The current length and management of standards development iterations are not necessarily adapted to the needs of the industry. In the industry, strong market competition leads to a short product life cycle (Sapp et al., 2021) and requirements change often and faster than the development of standard iterations. Besides, during the long development iterations, new technologies, processes, and needs are developed, potentially making the published standards at odds with the industrial reality.

An alternative to the waterfall model for project management is to use an adaptive method (also known as Agile) (Shameem et al., 2018; Thummadi et al., 2011). Unlike waterfall, Agile follows an iterative and incremental approach. This model consists of short iterative release cycles, which requires more transparency and visibility, stakeholders to be more involved and notified more regularly of the progress of the project (Edeki, 2015). Agile is mainly used to achieve high quality projects in short periods, better collaboration between all stakeholders, and reduced time to market (Kumar and Bhatia, 2012). Agile can be implemented through many frameworks such as Scrum, XP, or Lean Kanban (Stellman and Greene, 2013). Most Agile frameworks are designed for small development teams. However, standards development often requires several (large) teams to work together. The SAFe Framework

by Scaled Agile (Agile, 2019) is a viable alternative for standards development due to its support of large, multi-disciplinary, and distributed teams. SAFe methods can bring benefits to the development teams of model-based standards, as demonstrated in (Sapp et al., 2021). The report (Sapp et al., 2021) demonstrates the benefits of adopting an agile framework and tool-chain in support of the standards development processes, illustrated with ISO 10303 (Pratt, 2005), a large and complex standard widely used in the manufacturing industry.

While becoming agile can bring many benefits to standards development processes, it also creates some challenges. Because requirements are defined, analyzed, and processed on a more frequent basis, properly capturing, tracking, and managing them becomes crucial, more time consuming, and more complex. While the different phases of requirements engineering present many challenges (Besrour et al., 2016), most of the literature focuses on conflict management (Shah and Patel, 2014; Davis et al., 2006; In and Roy, 2001; Gambo et al., 2015; Decker et al., 2007) and predictive environment, leaving out requirements elicitation, a key step to their management (PMI, 2017). In this paper, we present a solution for requirements elicitation in an adaptive information standards development environment that enables maximal transparency, traceability, and visibility of the project activities.

2. RELATED WORK

Requirements elicitation is a critical step in project development and is sometimes considered a complex process (Sharma and Pandey, 2013; Fernandes et al., 2012). Many methods have been developed to capture and represent requirements to facilitate this process. In this section, we present some of the most representative methods of the state of the art.

The Requirements Interchange Format, also known as ReqIF, is a standardized XML-based format aimed to support the exchange of requirements, as well as their associated metadata, between different management tools (Adedjouma et al., 2011). This nonproprietary format satisfies the industry needs for exchanging requirements data between different companies without having to share the same management tools (OMG, 2016). ReqIF is supported by almost all existing requirements management tools and is widely used. A ReqIF model is composed of specification objects, called SpecObject, to represent the requirements. These objects contain multiple user-defined attributes and the relationships between different objects are also represented.

The Systems Modeling Language (SysML) is a modeling language for the specification, analysis, design, and verification of a broad range of complex systems (SysML, 2019). SysML includes different types of diagrams, one of which is used to represent text-based requirements. This diagram can effectively capture functional as well as non-functional requirements. This SysML diagram captures the relationship between requirements, system models, and test cases (Hause, 2006). A SysML requirement is mainly defined by a unique identifier and a text-based definition, but other properties can be included such as priority, status, or verification status (Roques, 2015). An example of a SysML

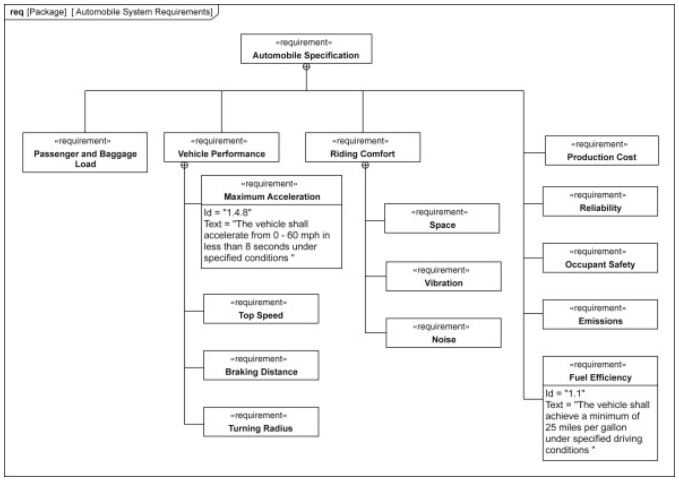

Fig. 1. Example of a SysML requirements diagram for the Automobile System Specification. (S. Friedenthal and Steiner, 2015)

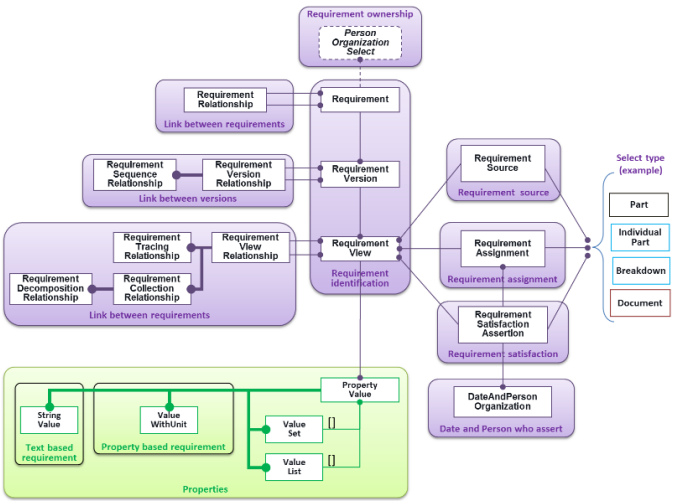

Fig. 2. STEP AP 242 Requirements model. (AP242, 2013)

requirement diagram for an Automobile System is shown in Figure 1. This example illustrates the capture of text-based requirements and the relations between them.

ISO 10303, also known as the Standard for the Exchange of Product data (STEP), is a widely adopted standard that aims to provide a complete and unambiguous description of manufacturing products, usable throughout their life cycle, regardless of the IT support used. STEP enables manufacturing applications to exchange and share data regardless of the specificities of the different systems that exchange and share information. STEP AP242 (ISO, 2020) includes a model for requirements management, as illustrated in Figure 2. This model focuses principally on capturing requirements definitions, as well as their relationships, validation criteria, and source (AP242, 2013).

These models focus mainly on a limited and textual definition of the requirements, and the relationships between them. These models were designed to support the representation and exchange of requirements and not meant to support advanced requirements management and planning activities (PMI, 2017) in an adaptive environment. As discussed previously, in transitioning from a waterfall to

an adaptive project management approach, requirements management takes a different and more important role, and requires more information and more visibility, to be executed properly (Institute, 2017). Besides, working with distributed teams and volunteer human resources requires better traceability and visibility of both decisions and contributions. An ideal solution should focus on defining the information to capture in order to overcome the following challenges (Sapp et al., 2021):

- (N1) It should ensure the traceability (and visibility) of resources: 1) to associate people to work items and tasks to which they contribute or supervise (i.e., their level of engagement), 2) to link people to the meetings they attend, and 3) to link work items to meetings during which they are discussed or to follow the progress of the work items;
- (N2) It should ensure the traceability (and visibility) of decisions: 1) by linking people to the decision they make, 2) by linking decisions to the meetings during which they are taken, and 3) by following the evolution of decisions regarding work items and requirements as the meetings progress;
- (N3) It should ensure requirements definition management by linking requirements to their source and a context in order to be able to validate them with their owner.

3. PRESENTATION OF THE MODEL

None of the models presented in the previous section captures enough information to overcome the challenges we presented because they lack a (precise) definition of 1) the requirements' context and source, 2) the requirement ownership information, 3) the associated resource(s) management, 4) the associated work breakdown, 5) key decisions, and 6) tracking information between requirements and associated deliverables. Consequently, we developed our own requirements elicitation model. This model can be used on its own or as an extension to the previously introduced solutions (i.e., ReqIF, SysML, and AP242). Figure 3 illustrates this elicitation model we propose. Our model is a UML Class diagram and is composed of six classes (Meeting, Requirement, Work Item, Task, Member, and Decision) in response to our three main needs: requirements definition management and traceability, decisions traceability, and commitment and resources traceability. This model supports our needs as shown in Table 1.

The objective of the proposed model is to keep track of decisions and requirements throughout the development of a project, which is crucial for long projects involving different distributed stakeholders. This model can be used to understand the level of engagement of the different stakeholders and thus be able to more easily manage the available resources. Moreover, this model allows us to record all decisions taken during meetings, making that information available to people who could not attend. It also allows us to ensure the traceability of requirements from their creation to their implementation in order to validate them directly with the people who had expressed these needs in the first place.

3.1 Meeting Class

The *Meeting* class represents the different properties that characterize a meeting and contains several properties to keep track of all the details of the meeting:

- The *id* property uniquely identifies the meeting;
- The *event* property defines the event during which the meeting took place, but this property is not mandatory because meetings are not necessarily included in a particular event;
- The *place*, *date* and *time* properties correspond respectively to the place, the date, and the time at which the meeting was held;
- The *type* property defines if the meeting is in person or virtual. If the meeting is virtual, the *place* property is left empty;
- The *subject* property defines the main subject of the meeting from a high-level point of view, while the *topic* property represents the agenda of the meeting which is the list of items that need to be discussed during the meeting;
- The *goal* property represents the objective of the meeting;
- The *future points* property represents topics that will have to be discussed during the next meeting.

The *Meeting* class has relationships with some of the other classes. Firstly, the Meeting class is linked to the *Member* class: several people can attend a meeting. Secondly, there are two relationships between the *Meeting* and *Requirement* classes: requirements are created during a meeting, and requirements can be discussed and updated during one or more meetings. Thirdly, the *Meeting* class is linked to the *Work item* class: work items can be planned during one or more meetings.

3.2 Member Class

The *Member* class represents the characteristics of a stakeholder. A member can be anyone involved in the project such as a manager, a developer, or a client. A member contains two properties, the name of the person and the organization to which that person belongs. The *Member* class is linked to the following classes:

- To the *Meeting* class: several persons can present one or more topics during a meeting, but the presenter(s) must be one of the meeting's attendees;
- To the *Decision* class: every member can express an opinion about requirements or a work item;
- To the *Requirement* class: a requirement has one or more authors;
- To the *Work item* class across two different relationships: one relation represents the work item's supervisor while the other one represents the team members working on it. We considered that a supervisor contributes to the work item that he supervises and that it is not necessary to create multiple relation links between a member and the same work item. Several work items can have the same supervisor and one or more team members working on it;
- To the *Task* class: one or more persons can contribute to some tasks.

Table 1. List of the UML classes from our elicitation model and the needs they meet

Needs	Model Classes
Requirements definition management and traceability (N3)	Member, Meeting and Requirements classes
Decisions traceability (N2)	Decision, Member and Meeting classes
Commitment and resources traceability (N1)	Requirement, Work Item, Task, Meeting and Member classes

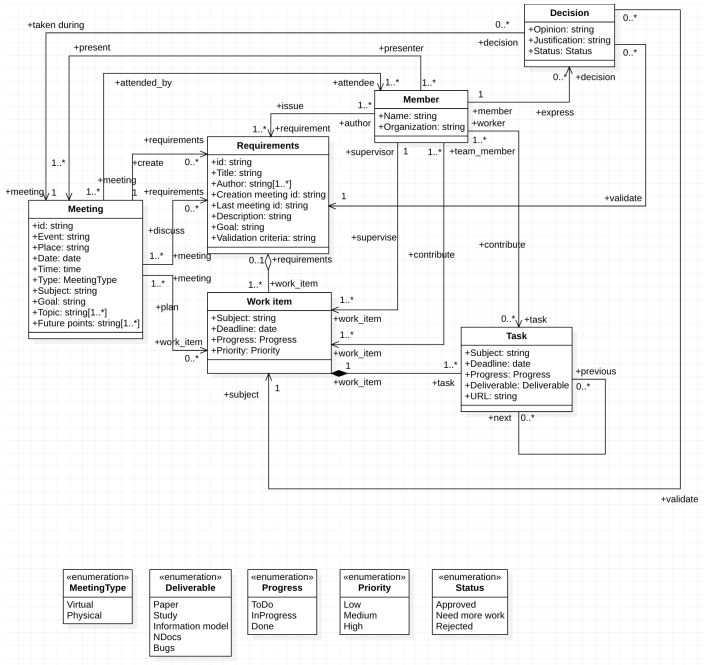

Fig. 3. Requirements elicitation model (UML Class Diagram)

3.3 Decision Class

During a meeting, each attendee is free to express an opinion on the different topics discussed. The objective of the *Decision* class is to keep track of the decisions expressed during a meeting. This class contains three properties to characterize these decisions. An attendee's opinion on a particular requirement or work item and its justification are respectively represented by the *opinion* and *justification* properties. The last property defines the status of the decision and this property can currently only take one of the following three values: *Approved*, *Need more word*, or *Rejected*.

The *Decision* class has relationships with three other classes: the *Meeting* class, the *Requirement* class, and the *Work Item* class. A decision is linked to the meeting during which this decision was taken, and to the requirement or work item to which this decision relates. Several decisions can be taken during the same meeting and in the same way, several decisions can be associated with one requirement or work item.

3.4 Requirement Class

The *Requirement* class represents the different properties that characterize a requirement. Thus, a requirement is composed of several properties. Firstly, a requirement is identified by a unique id. A requirement is also characterized by a title and an author. Besides, the *Requirement* class contains the id of the meeting during which the requirement was created and the id of the last meeting during which the requirement was updated. When a requirement is created, the *creation meeting id* and *last meeting id* properties are identical. Finally, a requirement is also characterized by its description, its goal, and its validation criteria.

3.5 Work Item Class

A project is made up of several work items that represent the different steps to reach the goal of the project. Work items help to plan and manage projects. In our case, one or more work items can be associated with a requirement. The work item is characterized by a high-level subject, a deadline, its progress, and a priority status. A work item's progress can either be *To Do*, *In Progress*, or *Done*, while its priority can either be *Low*, *Medium*, or *High*. The progress of a work item depends on the progress of the tasks that compose it: indeed, a work item cannot be marked as *Done* if all its tasks are not also marked as *Done*.

3.6 Task Class

A work item is composed of one or more tasks. The *Task* class can represent user stories, tasks, bugs, or issues. A task is characterized by five properties: a subject which defines the goal of the task, a deadline, its progress, whose value is included among *To Do*, *In Progress* or *Done*; the type of deliverable returned, and the url associated with the commit corresponding to the task's deliverable. Currently, the deliverable of a task can be a *Paper*, an *Information model*, some *NDocs*, or a *Bug*. When the task

has not yet started, the url property is left empty. In some cases, the different tasks included in a work item need to respect a certain order, represented by the relationship between the tasks.

4. DEMONSTRATION OF THE MODEL

As presented in the previous section, this model allows us to keep track of and share information regarding project meetings, decisions, requirements, and work items. This model also keeps all the stakeholders informed about the progress of the project and the different topics discussed during the different project meetings. Moreover, this model supports tracking traceability of the requirements by keeping track of their creation and their authors, as well as any changes that may have been made and the progress of the associated work items. This information is key to better understand and follow the evolution of the requirements over time and to validate them directly with the people who expressed them in the first place.

The Figure 4 illustrates a virtual technical meeting attended by five stakeholders. The main objective of this project meeting is the project management methodology with the transition to Agile. During this meeting, the 45e65d requirement on Agile as a project management method was updated and two work items were discussed. First, the "Agile" work item, associated with the 45e65d requirement that is composed of four tasks. The first task has been completed, the second is still in progress, while the other two tasks were just added during this meeting. As new tasks, they do not have contributors yet, they don't have an associated repository, and their deliverables are not yet defined. Second, the work item concerning the review of the issues log is still in progress. During the meeting, the bugs were reviewed and their resolution was planned.

Due to its design, our solution can be used on its own or integrated to an existing model. For instance, to integrate this solution to ReqIF, one would replace the *SpecObject* element of the ReqIF model with our *Requirement* class and/or replace the *System Component* element of ReqIF with our *Work Item* class

5. CONCLUSION

In this paper, we discussed the role of information standards, a key enabler to the digitization of manufacturing, and the inefficiencies in their current development process that often relies on a predictive management approach. As an alternative, we highlighted how an adaptive approach that can overcome some of these inefficiencies also comes with its own challenges, notably an increased need for traceability and visibility of requirements, their elicitation, their management, and their implementation. As this need had not been addressed, we developed and proposed a new requirement elicitation model that serves as a foundation for providing requirements and decision traceability and visibility, by recording and leveraging project meetings in a formal way.

This information model, unlike existing traditional solutions, has been designed for adaptive management and captures all the information essential to properly implement an agile approach. Due to its simplicity, the model can easily be used on its own or as an extension to an existing requirements management solution. As an extension, it can extend and enrich traditional solutions that were not initially designed for adaptive management.

Once this model has been properly implemented and instantiated (i.e., data has been collected), the next challenge is to display the captured information in a meaningful way to the different project stakeholders. Because visibility is key to agile management, project information is often shared graphically in an information radiator that is easily accessible and understandable by all. Our future work will focus on identifying the most appropriate visualization techniques to display the information we are now able to collect.

REFERENCES

Adedjouma, M., Dubois, H., and Terrier, F. (2011). Requirements exchange: From specification documents to models. In *2011 16th IEEE International Conference on Engineering of Complex Computer Systems*. IEEE.

Agile, S. (2019). Safe 5.0 framework. URL `https://www.scaledagileframework.com/`. Accessed: 2020-10-10 [Online].

AP242, S. (2013). Requirement management interoperability. URL `http://www.ap242.org/requirement-interoperability`. Accessed: 2020-10-13 [Online].

Balaji, S. and Murugaiyan, D.M.S. (2012). Waterfall vs v-model vs agile: A comparative study on sdlc. *International Journal of Information Technology and Business Management*, 2(1).

Besrour, S., Rahim, L.B.A., and Dominic, P.D.D. (2016). A quantitative study to identify critical requirement engineering challenges in the context of small and medium software enterprise. In *2016 3rd International Conference On Computer And Information Sciences (ICCOINS)*. IEEE.

Blind, K. (2009). Standardisation as a catalyst for innovation. *ERIM Report Series*.

Davis, C.J., Fuller, R.M., Tremblay, M.C., and Berndt, D.J. (2006). Communication challenges in requirements elicitation and the use of the repertory grid technique. *Journal of Computer Information Systems*, 46(5).

Decker, B., E. Ras, J.R., Jaubert, P., and Rieth, M. (2007). Wiki-based stakeholder participation in requirements engineering. *IEEE Software*, 24(2).

Edeki, C. (2015). Agile software development methodology. *European Journal of Mathematics and Computer Science*, 2(1).

Fernandes, J., Duarte, D., Ribeiro, C., Farinha, C., Pereira, J.M., and da Silva, M.M. (2012). ithink : A game-based approach towards improving collaboration and participation in requirement elicitation. *Procedia Computer Science*, 15.

Fischer, K., Rosche, P., Trainer, A., Feeney, A.B., and Hedberg, T.D. (2015). Investigating the impact of standards-based interoperability for design to manufacturing and quality in the supply chain.

Gallaher, M.P., O'Connor, A.C., John L. Dettbarn, J., and Gilday, L.T. (2004). Cost analysis of inadequate

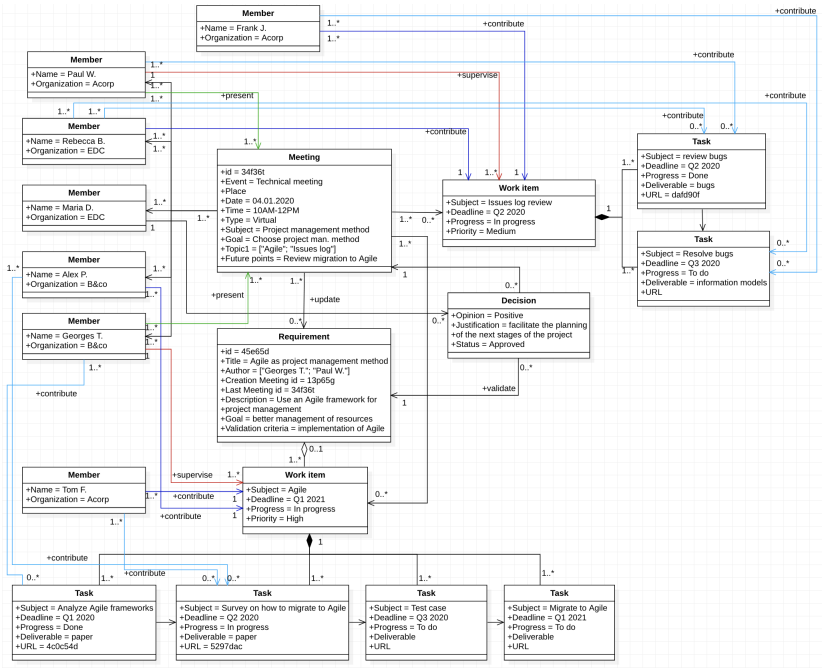

Fig. 4. Simple example of our Requirement Elicitation model (UML Object Diagram)

interoperability in the u.s. capital facilities industry. *National Institute of Standards and Technology (NIST)*.

Gambo, I.P., Soriyan, A.H., and Ikono, R.N. (2015). A proposed process model for requirements engineering using delphi techniques for prioritization. *International Journal of Information Technology and Computer Science*.

Guasch, J., Racine, J.L., Sánchez, I., and Diop, M. (2007). *Quality Systems and Standards for a Competitive Edge*. The World Bank.

Hause, M. (2006). The sysml modelling language. In *Fifth European Systems Engineering Conference*. INCOSE.

Hedberg, T.D., Feeney, A.B., Helu, M.M., and Camelio, J.A. (2016). Towards a lifecycle information framework and technology in manufacturing. *ASME Journal of Computing and Information Science in Engineering*.

Helu, M.M., Hedberg, T.D., and Feeney, A.B. (2017). Reference architecture to integrate heterogeneous manufacturing systems for the digital thread. *CIRP Journal of Manufacturing Science and Technology*.

In, H. and Roy, S. (2001). Visualization issues for software requirements negotiation. In *25th Annual International Computer Software and Applications Conference*. IEEE.

Institute, P.M. (2017). *Agile Practice Guide*. Project Management Institute.

ISO (2017). Target date planner. Accessed: 2020-10-10 [Online].

ISO (2020). Industrial automation systems and integration — product data representation and exchange — part 242: Application protocol: Managed model-based 3d engineering.

Kumar, G. and Bhatia, P.K. (2012). Impact of agile methodology on software development process. *International Journal of Computer Technology and Electronics Engineering*, 2(4).

OMG (2016). Requirements interchange format (reqif). URL `https://www.omg.org/reqif/`. Accessed: 2020-10-11 [Online].

PMI (2017). A guide to the project management body of knowledge (pmbok guide). Project Management Institute, 6 edition.

Pratt, M.J. (2005). Iso 10303, the step standard for product data exchange, and its plm capabilities. *International Journal Of Product Lifecycle Management*, 1(1).

Roques, P. (2015). How modeling can be useful to better define and trace requirements. *Requirements Engineering Magazine*. Accessed: 2020-10-13 [Online].

S. Friedenthal, A.M. and Steiner, R. (2015). *A Practical Guide to SysML, Third Edition*. Morgan Kaufmann.

Sapp, B., Harvey, M., Toussaint, M., Krima, S., Feeney, A.B., and Panetto, H. (2021). Agile for model-based-standards development. *NIST Advanced Manufacturing Series 100-40*.

Shah, T. and Patel, S.V. (2014). A review of requirement engineering issues and challenges in various software development methods. *International Journal of Computer Applications*, 99(15).

Shameem, M., Kumar, C., Chandra, B., and Khan, A. (2018). Impact of requirements volatility and flexible management on gsd project success: A study based on the dimensions of requirements volatility. *International Journal of Agile Systems and Management*.

Sharma, S. and Pandey, S.K. (2013). Revisiting requirements elicitation techniques. *International Journal of Computer Applications*, 75(12).

Stellman, A. and Greene, J. (2013). *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*. O'Reilly Media.

SysML (2019). Sysml open source project. URL `https://sysml.org`. Accessed: 2020-10-11 [Online].

Thummadi, B., Shiv, O., and Lyytinen, K. (2011). Enacted routines in agile and waterfall processes. In *2011 Agile Conference*. IEEE.