Concept of Quality Digital Twin in Agile Development

Tsuyoshi Nakajima[1], Alessandro Simonetta[2]

[1]Shibaura Institute of Technology, 3-7-5 Toyosu, Koto-ku, Tokyo, 135-848, Japan
[2]Department of Enterprise Engineering University of Rome Tor Vergata, Rome, Italy

Abstract
Agile development is essential for effective digital transformation. However, since agile development tends to focus on functional implementation, the realization of true user value (quality in a broad sense) is often neglected. To support realization of true user value in agile, this paper proposes the concept of a quality digital twin, a quality representation of the target system synchronized at each iteration. The agile development focuses on the quality of target product itself more than that of processes and intermediate products, and therefore SQuaRE quality model can be much more applicable to the agile. It uses the SQuaRE quality framework for defining quality requirements, and engineering and evaluating the system. This concept enables the realization of a support system for user-driven, high-quality agile development. Its feasibility is demonstrated by showing system / data structure and usage of the quality digital twin.

Keywords
Agile, digital twin, quality management, SQuaRE

1. Introduction

Digital Transformation is the transformation of products, services, and business models based on the needs of customers and society by responding to changes in the business environment and utilizing information technology [1]. Digital Transformation can perform high value creation and development efficiency by repeating the planning, prototyping, operation, and evaluation of ideas quickly, with the user taking the initiative. Agile development is one of the foundations required for this.

However, because agile development tends to focus on the implementation of functional requirements, the realization and evaluation of quality requirements are often neglected. To solve such a problem, we propose a concept of a support system for user-driven high-quality agile development (hereinafter called "quality digital twin") by applying the digital twin concept to quality management in agile development.

The Quality Digital Twin has the quality state model as its core, which models the quality of products in the agile development. The model is based on the quality models of the ISO/IEC 25010, including product quality models and quality-in-use model [2]. By considering user value with the quality-in-use model, refining and implementing quality requirements/functions/architecture, and iterating testing including non-functional testing, static analysis, and quality evaluation based on user reviews in each sprint, it is possible to simultaneously progress the quality estimation and evolve quality requirements.

This paper presents the issues of quality achievement in agile development and concept and design of the quality digital twin.

2. AGILE DEVELOPMENT AND ITS ISSUES

2.1. Agile development

Agile software development refers to a group of software development methodologies based on iterative development to deal with uncertainty and risk, as shown in 1 . Its important practical activities include short iterations with a certain time window, continuous release (automation, providing and evaluating working products), user participation, and test-first and refactoring. Agile development is effective when the problem to be solved is complex (i.e., its requirements are vague and/or changing, and the solution is unknown), and is adjusted by repeatedly investigating, understanding, and dealing with the problem [3]. This paper addresses Scrum, which is a lightweight process framework for agile development, and the most widely used one [4].

2.2. Issues to user-driven agile development

Digital transformation can perform high value creation and development efficiency through rapid repetition of user-driven idea planning, prototyping, operation, and evaluation [1]. Therefore, agile development is one of the necessity foundations for this. However, user involvement is often limited to input of requirements and eval-

4 th International Workshop on Experience with SQuaRE Series and its Future Direction, December 06, 2022, Tokyo, Japan
tsnaka@shibaura-it.ac.jp (T. Nakajima);
alessandro.simonetta@gmail.com (A. Simonetta)
0000-0002-9721-4763 (T. Nakajima); 0000-0003-2002-9815 (A. Simonetta)

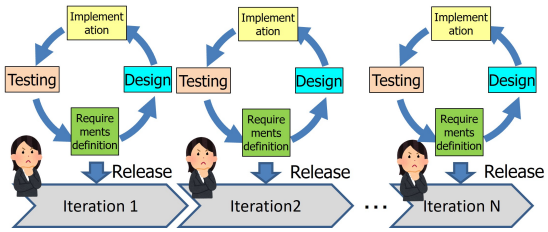

Figure 1: Agile development

uation and review of deliverables at each development cycle, and users rarely are involved in decision-making in managing the projects. As a result, agile development tends to derail from creating the value that users expect for the cost [5].

We believe that one of the reasons why users cannot be involved in development decisions is that there is no quantitative and intuitive way to determine whether the current quality of the target product is sufficient for the value they want to achieve, and what and how much is missing.

2.3. Quality management of agile development

In the waterfall development, the quality of the target product is defined, decomposing it into quality activities and their goals for intermediate products in each development phase. The project tries to achieve them to assure the quality of the target product indirectly. In contrast, in the agile development, the quality to be achieved is mainly the value demanded by the user, and its resources are concentrated on creating an executable product to realize and confirm the value. The quality is achieved through repeated product evaluations with user participation at each iteration [6]. From a quality management perspective, agile development, compared to the waterfall, has the advantage that its quality goals are business-oriented and the way to check their achievement is direct.

Modeling languages such as use cases in UML may mitigate the above weakness of waterfall development to some extent because they are easy to understand even for non-experts. However, the quality of these models does not directly imply the quality of the system, since they only can define the functionality of the system, not its quality.

In the agile development, on the other hand, user values are defined by the functionality the product should have. Therefore, the development team tends to focus on realizing the functionality, not user values themselves. To prevent this, the user review is conducted to evaluate the product at the end of each iteration. However, because the user review tends to be subjective, some quality aspects that the users are not directly benefited from is often neglected to confirm. Agile development itself does not have the support to overcome these problems and achieve quality in a comprehensive and balanced manner.

3. QUALITY DIGITAL TWIN CONCEPT

3.1. Digital twin

A digital twin (DT) is a living, intelligent and evolving model, being the virtual counterpart of a physical entity or process for practical purpose, such as monitoring and control, future prediction and planning, and conceptual design and development [7].

From the results of the literature review [7] [8] [9], we found that DT consists of all or a partial combination of the following five functions as shown in 2 (Generic form of DT).

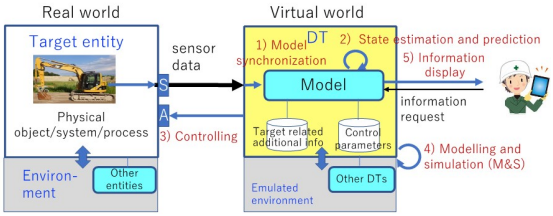

Figure 2: Generic form of Digital Twin

- Model synchronization: Synchronization of the real-world target and the virtual world model from data sensed from the real world one
- State estimation and prediction: Estimation and prediction of target internal states from the model (for monitoring)
- Controlling: Controlling the real-world target based on 2)
- Modelling and simulation: Future forecasting and planning through simulation and optimization by changing inputs and control parameters
- Information display: Display of information about the monitoring of the estimated internal state of the target, the results of future predictions, or the provision of additional information tied to the model.

3.2. Application of DT concept to quality management in agile development and three hypotheses

In waterfall development, quality is estimated using the results of quality assurance activities (often processes)

such as testing and reviews of intermediate deliverables. In contrast, agile development does not produce intermediate deliverables, but instead produces a workable product in a short cycle, making it possible to measure quality data directly on the product. Based on this, we hypothesized that the ISO/IEC 25000 (SQuaRE) series of product-focused quality models and quality estimation using these models would be easily applicable to agile development (Hypothesis 1).

The quality of the software in the real-world is not directly visible, but is estimated through quality measurement and evaluation. The SQuaRE quality models and the results of quality measurement and evaluation using the models (their data repository is called quality state model) can be regarded as a virtual world model for the invisible quality of real-world software products. The iteration cycles of agile enables model synchronization between them. From these considerations, we have another hypothesis (Hypothesis 2) that the quality state model can be applied to quality management in agile development. Based on these hypotheses, we propose Quality Digital Twin (Quality DT), a quality management support system using the quality state model.

Comparing to the generic form of DT, the following concept of Quality DT was drawn:

1. Model synchronization: Definition a of quality-in-use requirements (at the start of the development), story development for quality requirements (at the start of each iteration), and then quality evaluation based on testing, static analysis and user reviews (at the end of the iteration).
2. State estimation and prediction: Monitoring of current quality and its deviation from the goal
3. Controlling: no counterpart
4. Modelling and simulation: What-of analysis for Quality Management, including development (iterations and each sprint) planning
5. Information display: Visualization of estimation and prediction of quality and risks

Furthermore, we believe that the quality digital twin would allow users to not only provide input and feedback to the agile development but also take the initiative in decision making on it. This is because the quality digital twin has the potential to provide users for intuitive and quantitative quality estimation in a way that only quality assurance professionals have been able to do (Hypothesis 3).

4. Considerations on application to agile development

4.1. Plan it; Do + Evaluation

We take the approach of only rough support for decomposition of quality requirements in sprint planning, and evaluating the adequacy of the decomposition and correcting it during sprint evaluation. This is because the agile development does not fit with time-consuming long-term and complete planning and requirements definition.

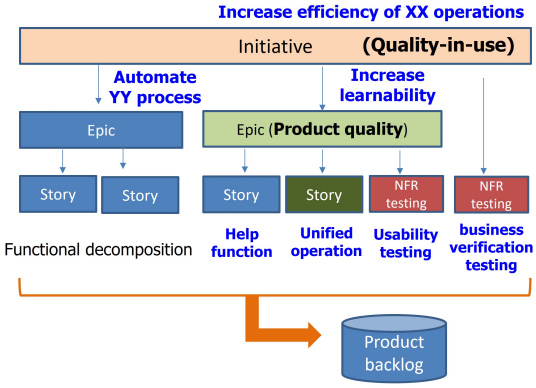

Figure 3: Quality decomposition using agile framework

In agile development, requirements continuously evolve. In other words, the quality digital twin is evaluated in every iteration to accumulate necessary quality requirements for its completion in the product backlog.

Quality decomposition is performed using the framework of Scrum: initiative, epic, and story, depending on the scale of the project. 3 shows an example of quality decomposition. First, user values are defined at the top level as quality-in-use (QiU) requirements, and then decomposed into product quality requirements, functions, architectural design, or non-functional testing. Problems in product quality discovered during development are stored in the backlog as technical liabilities.

4.2. Quality requirements in agile development

At the sprint planning, the decomposition of quality requirements to be implemented into stories is performed to put in the sprint backlog. During the execution of a sprint, the results of testing and static analysis are collected. Technical debt identified by the developer is also collected into the product backlog.

The sprint review is based on the collected quality-related data to determine the current quality. In this case, based on the decomposition at the time of planning, the

results of testing and static analysis are automatically calculated in terms of their "contribution" to the quality characteristics/sub-characteristics.

During a sprint review, the quality dashboard displays the quality evaluation results in a visual and easy-to-understand manner for the users to intuitively monitor the quality. The qualitydashboard shows the relationship between quality requirements, quality realization, and quality activities, provides the ability to edit it (8), and monitors the status of product quality. 4 shows an image of the quality status monitoring function. The achievement percentage of all the QiU requirements is displayed in the left-hand side of 4, in which QiU requirements are listed in order of their importance (three sizes of length: most important, important, and normal). In the right pie chart in 4, the angle of the arc represents the relative importance of each quality characteristic to the product, and the length of the radius represents the achievement of product quality. For example, the figure shows usability is the most important quality of the target product, and its achievement level is about 70

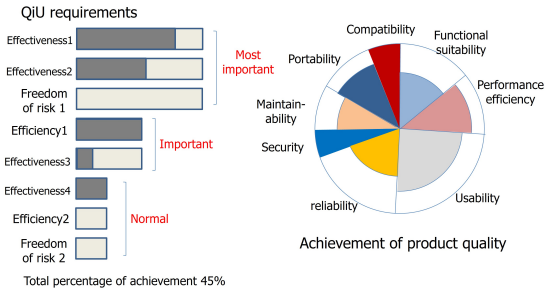

Figure 4: Image of quality status monitoring in quality dashboard

During the sprint review, user reviews the correspondence between the quality requirements and the quality-related data collected and their contribution to the requirements. For QiU requirements, the following points will be checked:

- Do the functions and PQ requirements decomposed from a QiU requirement truly contribute? Is the defined contribution level appropriate for it ?
- Is non-functional testing necessary to support the QiU requirement?
- Is this QiU requirement really necessary? Should it be changed to express what we need more clearly?

Almost the same comparative review can be performed for product quality requirements. By conducting the above sprint review for each QiU requirement, it is possible to evolve the quality requirements, making sure the degree to which the targeted quality have been achieved. Concerning product quality, we can see the balance of its achievement.

5. QUALITY STATE MODEL AND ITS APPLICATION

This section presents ideas for the organization and use of the quality state model, which is a central part of Quality DT.

5.1. SQuaRE quality models

The ISO/IEC 25000 (SQuaRE) series provides a framework for quality related to systems and software products. ISO/IEC 25010 [10] defines both the quality-in-use and product quality model for systems and software products, while ISO/IEC 25012 [11] defines the data quality model.

Quality-in-use represents the influence on stakeholders when the target entity is used under a certain context of use, while product quality and data quality are the capability of the target product and data themselves, respectively to satisfy both stated and implies needs.

5.2. Interpretation of the SQuaRE quality models in the quality state model

In-use quality

The quality state model is built on the above three quality models of SQuaRE, but to capture the decomposition relationships between quality requirements, the following interpretations of the models and their properties are added.

The Quality-in-Use (QiU) model targets use during operation. In contrast, the ease of work during development and maintenance is partly in the Product Quality (PQ) model. In this paper, these are positioned as quality models PQQiU that considers their influence on work under a specific context of use.

- Quality in Use [QiU]
  - Effectiveness (magnitude of value created for the user)
  - Efficiency (efficiency of user tasks)
  - Satisfaction (positive impact on user's mind & attitude)
  - Freedom of risk (avoidance of negative impacts)
  - Usage Coverage (ability to perform appropriately to anticipated/future usage situations)
- Product quality characteristics for non-user use (ease of development and maintenance work) [PQ QiU]

  - Testability (ease of testing work)
  - Analyzability (ease of failure analysis)
  - Modifiability (ease of modification work)
  - Installability (ease of installation work)
  - Reusability (ease of reuse)

5.3. True product quality

The product quality characteristics other than the above represent product-specific qualities and are classified as below.

- Quality of the functions that the system has [PQ(Function), PQ(Set of Functions)]
  - Functional accuracy
  - Function appropriateness
  - Time behavior
- Quality of a set of functions
  - Functional completeness
  - Usability (cognitive ease of use)
  - Security
- Quality of the source code of the system [PQ(Code)]
  - Modularity
- Characteristics of the system with respect to possible failures [PQ(Failure)]
  - Maturity (no failures ⟵ few defects)
  - Fault tolerance (service continuity against failures)
  - Availability (service uptime against failures)
  - Recoverability (low loss against possible failures)
- Relationship with outside of system [PQ(Outside)]
  - Capacity satisfiability
  - Interoperability
  - Coexistence
  - Adaptability
  - Accessibility
  - Resource efficiency

5.4. Deployment of quality requirements in the SQuaRE quality requirements framework

ISO/IEC 25030 Quality Requirements Framework [12], [13] guides the flow of deployment of quality requirements through decomposition of the system.

Quality Digital Twin supports various deployment methods of quality requirements by using patterns. In addition, there are three patterns for the realization of quality requirements: assignment to components, functions, and architecture (structure and design guideline) [14]. When A, B, and C are quality nodes (in 6), the rule A⟶B | C means that A can be deployed into either B or C.

QiUFunction [with PQ(Function)]* | PQ(Set of Functions)* | PQ(Outside)* | PQ(Failure)* ☒ PQ QiU Function [with PQ(Function)]* | PQ(Code)* | Architecture ☒ PQ(Function) DQ* ☒ PQ(Set of Functions), PQ(outside), PQ(failure) Function [with PQ(Function)]* | DQ* |

Architecture

QiU requirements can be deployed into Function and Product quality requirements. The product quality characteristics for non-user use are developed into functions, code quality, and architecture. Architecture is embodied in the components, their relationships and principles of behavior, and design guidelines.

When a system is decomposed into itscomponents, the deployment of product qualityto them also occurs. In this case, product quality can be inherited, not inherited, or assigned among the components. In case of the assignment, a product quality requirement such as time efficiency and reliability, is decomposed into a set of new requirements with different goals, each of which assigned to each component. In case of including data components, data quality (DQ) requirements are defined for them.

5.5. Quality evaluation of the SQuaRE quality model

Quality evaluation in SQuaRE [15] selects important quality characteristics and sub-characteristics, and defines information needs for the target to measure them using quality measures, as shown in 5 . The quality measure quantifies the results of testing, user reviews, or static analysis of the target entity into one single value. The value of the quality measure is mapped to a predefined quality rating level to obtain a quality rating value.

Quality DT considers non-functional testing, static analysis, and user reviews conducted in each iteration of agile development as quality activities, and integrates the results of these activities to evaluate the achievement of the corresponding quality requirements.

5.6. Design of the quality state model

6 shows the meta-model of the quality state model. The quality state model is realized as a directed acyclic graph in which higher-level requirements are related to lower-level realizations by the link ”supports”. A support link has an attribute of “contribution,” which indicates the degree of support. The following node types are considered.

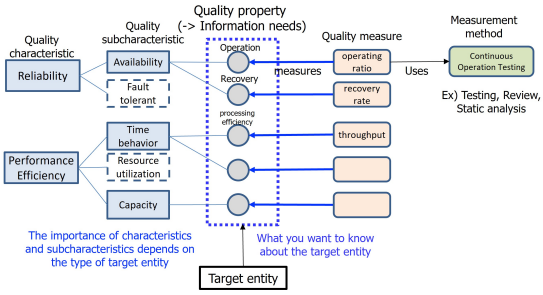

Figure 5: Quality evaluation and measurement methods

- Quality requirement: QiU, PQQiU, PQ(Set of Functions), PQ(Outside), PQ(Failure),DQ
- Quality implementation: Function [with PQ(Function)], Architecture
- Quality activity: Functional testing, Non-functional testing, Static analysis, User review

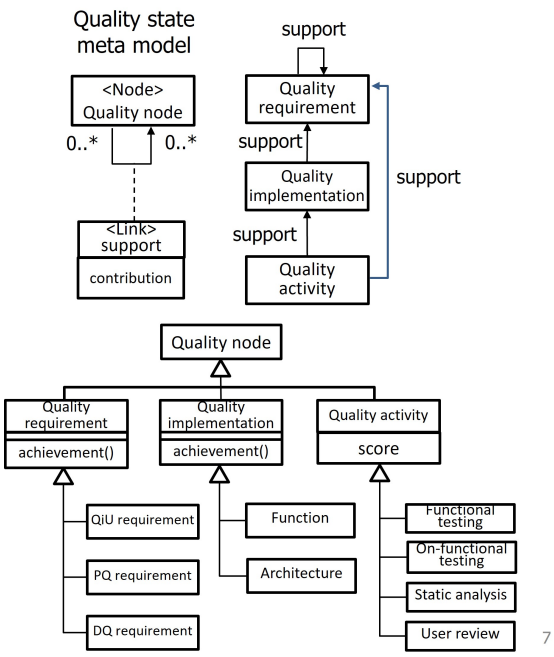

Figure 6: Metamodel of quality state model

Quality requirements can be supported by not only quality requirements themselves, but quality implementations and quality activities. Quality implementation can be supported by Functional testing for Function, Non-functional testing for Architecture, and Static analysis for Architecture.

5.7. Quality evaluation of the SQuaRE quality model

In the Quality DT, quality management of agile development is considered as a process of correctly evolving the quality state model. In other words, while iteratively developing the product, quality requirements (and their relationships) are gradually established, and at the same time, the degree of achievement of quality requirements is measured from the results of quality activities in the agile development, and corrective activities are recommended for achieving the quality goals. This prevents the agile development from deviating from the quality goals.

7 shows how the quality state model is used in a sprint. At the sprint planning, the results of requirements decomposition are input into it, the results of quality activities are collected during executing the sprint, and at the sprint review, quality evaluation are conducted on the model and then the model is reviewed and, if necessary, reorganized. The detailed usage flow is shown below.

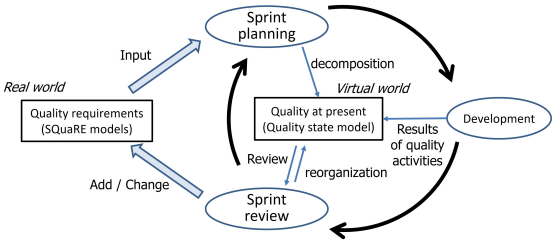

Figure 7: Relationship between sprint and quality state model

1. At sprint planning Supports relationships are tentatively established at the time of quality requirement decomposition. In sprint planning, users, product owners and developers cooperate to decompose a quality requirement into other quality requirements, quality implementations and quality activities, guided by the rules in 4.2. In the case of explicit decomposition, a new node is added to the quality state model and automatically generates a supports link to the decomposed element. The decomposition should be rough at this point because supports links can be always reviewed to reconnect later.
2. During sprint implementation The results of the quality activities are recorded.
3. At sprint review The supports relationship is reviewed at the sprint review. In the sprint review, users, product owners and developers cooperate to evaluate the results of quality activities conducted during the sprint. Prior to the review, the

results of User review, Non-functional testing and Static analysis of QiU are evaluated.
4. Functional testing: success 1/ failure 0
5. Non-functional testing, Static analysis: achievement score [0,1]
6. User review: achievement/non-achievement of specified quality requirements [0,1]

For all quality activities conducted up to this stage, the quality evaluation value of quality node n i , achievement(n i), is calculated using the following formula.

where support(n i) = n i1 , n i2 , ... , n im is the total set of nodes supporting node n i , and contribution ij is the contribution of node n ij to n i . In the evaluation review of n i , the contents and quality evaluation values of n i , all nodes supporting n i support(n i) (including the quality implementations and quality activities conducted in this sprint), and the contribution ij of each node are shown on the links, as shown in 8.

Figure 8: Review of a quality node and its supporting nodes

In addition, the user can also check the degree to which quality requirements have been met and the progress made since the last time. By checking the correspondence and contribution of the supporting nodes, it is possible to review whether the implementation and activities truly support the goal quality requirements and whether their contribution is appropriate.

5.8. Challenges in realizing Quality DT

The system that realizes the quality digital twin consists of three functions: a quality requirement decomposition support function, a quality evaluation function, and a quality dashboard function, as shown in 9.

The followings need to be addressed for each function.

- Quality requirement decomposition function: The description and decomposition of quality requirements must be user-driven. It is necessary to support appropriate decomposition without difficulty for users.
- Quality evaluation function: It is necessary to establish a method to link the results of non-functional testing and static analysis to quality evaluation. For static analysis, we plan to refer to ISO/IEC 5055 [16].
- Quality dashboard function: what information and how needed to displayed to help users take the lead in quality management.

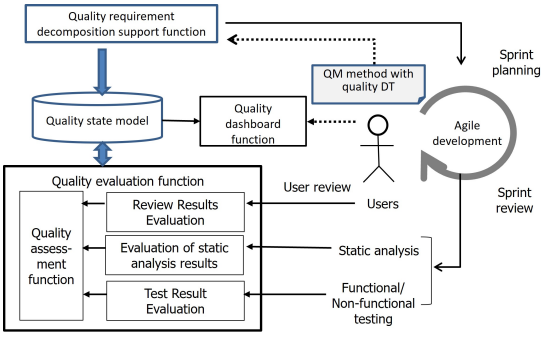

Figure 9: Relationship between sprint and quality state model

6. Conclusion

We proposed the concept of Quality Digital Twin, which visualizes the quality of products in the agile development, using the quality models of the ISO/IEC 25010. This concept enables the realization of a system that supports user-driven, high-quality agile development. The feasibility of the concept is demonstrated by showing an example of the structure and usage of the quality state model, which is the core of the concept, as well as three issues for the realization of the concept. We plan to conduct prototyping and experiments to demonstrate the effectiveness of the system.

References

[1] G. Vial, Understanding digital transformation: A review and a research agenda, The Journal of Strategic Information Systems 28 (2019) 118–144. URL: https://www.sciencedirect.com/science/article/pii/S0963868717302196. doi:https://doi.org/10.1016/j.jsis.2019.01.003, sI: Review issue.

[2] T. Komiyama, A. Motoei, Establishing international standards for systems and software quality requirements and evaluation, IWESQ@APSEC (2020).

[3] S. McConnell, More effective Agile: A roadmap for software leaders., Construx Press, 2019.

[4] B. Meyer, The Ugly, the Hype and the Good: an assessment of the agile approach; Agile, Springer, Cham, 2014.

[5] C. Ebert, H. C. Duarte, Digital transformation, IEEE Software 35 (2018) 16 21.

[6] K. Kautz, Participatory design activities and agile software development, IFIP WG 8.2/8.6 International Working Conference (2010) 303–316.

[7] B. R. Barricelli, E. Casiraghi, D. Fogli, A survey on digital twin: definitions, characteristics, applica-

tions, and design implications, IEEE access (????) 167653–167671.

[8] K. Y. H. Lim, P. Zheng, C.-H. Chen, A state-of-the-art survey of Digital Twin: techniques, engineering product lifecycle management and business innovation perspectives, Journal of Intelligent Manufacturing 31 (2020) 1313–1337.

[9] F. Tao, et al. , Digital twin in industry: State-of-the-art, IEEE Transactions on Industrial Informatics 15.4 (2018) 2405–2415.

[10] ISO/IEC 25010:2011 "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.", 2011.

[11] ISO/IEC 25012:2008 Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model, 2008.

[12] ISO/IEC 25030:2019 Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Quality requirements framework, 2019.

[13] T. Nakajima, T. Komiyama, Applying Quality Requirements Framework to an IoT System and its Evaluation, International Journal on Advances in Internet Technology 12 (2019).

[14] P. Clements, et al., Documenting software architectures: views and beyond (2nd edition), Addison-Wesley Professional, 2010.

[15] T. Nakajima, About the Framework of Quality Evaluation Using SQuaRE, APSEC (2020).

[16] ISO/IEC 5055 Information technology—Software measurement—Software quality measurement—Automated source code quality measures, 2021.