International Journal of Management, Knowledge and Learning | ISSN 2232-5697

Improving Software Quality and Transparency within an Overall IT Life Cycle Process with Quality Gates

Wei Xu
Fielmann AG, Germany
wei.xu@fielmann.com

Klaus Greve
Fielmann AG, Germany
Klaus.greve@fielmann.com

Marius Schlage
Fielmann AG, Germany
Marius.schlage@fielmann.com

Justus Holler
Fielmann AG, Germany
Justus.holler@fielmann.com

Jochen Hagen
Fielmann AG, Germany
Jochen.hagen@fielmann.com

Abstract:

This paper describes a model containing specific quality gates for software development projects. Combined with defined roles, these quality gates facilitate communication and transparency regarding the progress within the project and the software product. The motivation for the described model became apparent as the number of software system implementations grew under the auspices of distributed departments or subsidiaries.

We evaluate the quality model concerning current software development projects in classical and agile environments, one of which we describe in this paper. The project setting includes multiple business departments, business IT, internal delivery IT and external delivery IT departments. It deals with the company's central procurement platform.

Study methodology: Design Science

Findings: Specific roles combined with defined quality gates help improve the communication flow, software product transparency and quality.

Originality/value: Description of a model which can be implemented for quality assurance in software development projects as part of an overall governance process for large-scale organisations.

Introduction

The setting of this paper is one of the biggest multinational optometric and acoustic companies in central Europe, with more than 900 brick-and-mortar stores and a rapidly developing omnichannel business model. Its central headquarters supports and issues governance guidelines for the whole multinational retail group. One of these best-practice guidelines is an IT implementation process which ensures compliant, documented, and efficient system integration within the group architecture. The model described within this paper is part of this overarching IT life cycle governance process to specifically foster quality in software engineering projects, no matter if the project relies solely on in-house personnel or is (partly) supported by external personnel. The importance and motivation for the described model

became apparent as the number of software system implementations grew under the auspices of distributed departments or subsidiaries with no or little IT background and no explicit awareness of internal and external compliance guidelines. Several of these implementation projects did not meet the quality or efficiency level that the company demanded to continue its omnichannel journey at the ambitious pace that the company had set.

In the next chapter, we present a brief review of relevant literature. According to our design science approach (Hevner et al., 2004), we continue with the description of objectives for an improved solution, which is supported by a survey we conducted. The model ("artefact") is then described in more detail regarding the process, roles, and, most specifically, the quality gates. The demonstration and evaluation are done in "Example of the Implementation of the Quality Gate Model". We conclude this paper by discussing our evaluation, including known limitations and an outlook to further research.

Literature review and related work

In software development projects, one of the critical goals is to ensure the quality of the project's outcome (Project Management Institute, 2021). Software quality assurance (SQA) is a set of standards, procedures, and processes to be performed during development. Bhanushali (2023) and Galin (2004) concluded that there is a direct relationship between software testing and quality.

In general, testing models have the same test levels, but the design and focus of the process may differ depending on the product, organisational form, and project framework. Hrabovská et al. (2019) reviewed 17 models with a systematic literature review. They found that while all testing models are applicable in general contexts, not every model is suitable for all domains or every organisation, and to achieve a software quality model, the use of a meta-model that offers a well-defined structure for quality models to ensure their operationalisation and adaption is recommended (Kläs et al., 2010).

The Project Management Book of Knowledge (2021) emphasises in its 7th edition the effectiveness of tailoring to accomplish a suitable framework for the project. Tailoring is a principle to be carried out continuously to maintain the best approach for the deliverables (Rodrigues et al., 2023).

The Project Management Institute (2021) distinguishes between adaptive, predictive, and hybrid project approaches to solving complex and complicated requirements, while complexity can be characterised by using the Cynefin framework (Snowden & Boone, 2007).

Adaptive projects are characterised by requirements that make it difficult to predict the project's solution due to possible changes in the requirements over time (Project Management Institute, 2021), while predictive projects involve well-defined requirements. The solution can be implemented by a plan (Boehm & Turner, 2003). Hybrid models are characterised by a combination of adaptive and predictive approaches.

A predictive approach can be considered as a simple, straightforward project method. The limitations are its inflexibility, the difficulty in addressing changes, and the higher risk of releasing a solution that does not meet the requester's requirements (Kerzner, 2017).

Agile project management is a method that can be used for complex projects with changing requirements over time. It focuses on collaboration, flexibility, and continuous improvement. Agile methods have become a popular project method for software development and are based on a set of core values that guide the teams' work (Beck et al., 2001).

One popular method among the agile methodologies is SCRUM, which was developed by Ken Schwaber and Jeff Sutherland (Schwaber & Sutherland, 2011). SCRUM is designed for small, cross-functional teams and focuses on iterative and incremental releases of the project. It is adaptable and encourages collaboration, communication and continuous improvement (Schwaber & Sutherland, 2011). However, Scrum does not have explicit workflow management or visualisation of internal workflows (Bhavsar et al., 2020); therefore, KANBAN (Anderson, 2010), a visual management system from lean manufacturing is used, which transforms SCRUM to SCRUMBAN in order to address the limitation (Reddy, 2015). To address the challenge of scaling agile, SAFe (*SAFe 5.0 Framework*, 2022) was developed. SAFe is designed to manage agile at scale and provides a structure for multiple teams and projects to meet the required standards (Ebert & Paasivaara, 2017).

An essential component in predictive and adaptive approaches is the concept of quality assurance. Software testing underlines one aspect of necessary quality assurance throughout development (Mahfuz, 2016). Continuous testing refers to testing throughout the entire project to ensure that the project meets the requirements and identifies bugs early. Pair programming is an agile practice emphasising collaboration and continuous testing (Beck, 2000). Quality control underlines the aspect of monitoring a deliverable through its entire life cycle, including the phase transition (Project Management Institute, 2021).

Cooper (1990) introduces a stage-gate process as a framework for managing the development of new products. He outlines stages, each with its own gate, which ensure that the development is moving in the right direction and that the resource allocation is used efficiently. Although his work does not reference software projects, this method is also utilised in IT projects (Karlstrom & Runeson, 2005). The concept of quality gates as a method for quality control is based on the stage-gate process (Ambartsoumian et al., 2011). Quality gates are set up to ensure that quality is maintained throughout the entire project, and a system of gatekeepers is responsible for reviewing the gate criteria (Crispin & Gregory, 2009).

Necessity of a quality assurance model within the overall IT lifecycle process

Due to the increasing internationalisation of the company, with a corresponding number of software-related projects distributed within this group setting, it was necessary to implement a more centrally controlled governance process supporting the whole lifecycle of software products. To address the relevant pain points, we conducted a survey intending to reach all people involved in software products within the company. Based on our lead question, "Are there any indications of obstacles in the phase transitions of (software) products?" we focused on the following areas of interest:

1. How would you describe the quality assurance process of a solution from an end2end perspective?
2. How would you describe the status tracking and alignment of the requirement to deploy the process?
3. In which way are the subprocesses of the requirement to deploy value stream dependent on each other?
4. How do you know the deployed solution’s quality is sufficient?

Of the 55 employees contacted, around 30% participated in the survey. Due to the number of responses and the distribution across queried organisational characteristics, it was neither possible to group the responses according to these characteristics nor to examine the statistical

significance. Regardless of this limitation, which prevents a deeper analysis, clear tendencies can be identified. These were presumed to be the indicators sought for the lead question.

The participants were asked to respond on a scale from 1 (low) to 5 (high). Based on the result of 2.63 (standard deviation: 0.81) for the question "Are you aware of the acceptance criteria that are used to gradually improve the maturity level of dependent software projects?", there is an indication of at least one obstacle according to the defined assumption. The opposing statement confirms this: "How often would changes in requirements have been avoidable if there had been better coordination between dependent software projects?" with an average value of 4.00. More than 75% of the answers were 4 or 5.

Furthermore, there is a need for transparent tracking of the maturity level of all software products (to be developed), with an average rating of 3.87. In addition to an operational view, the management level could also be a component for further consideration, as a "superordinate quality assurance model across all software projects with minimum agreed acceptance criteria" with an average rating of 3.50 was identified as a potentially useful tool.

In summary, the following statements were identified against the background of software product development:

- Changes in requirements could have been avoided if the interdependencies of projects were better aligned.
- A holistic quality assurance model that includes all software products and defined acceptance criteria is a potentially useful tool.

The Software Quality Gate Model

This chapter introduces our Software Quality Gate Model as a solution for quality assurance in software development projects. This model has been derived using various adaptive and predictive project management methods to define a hybrid management framework. Its primary objective is to enable our organisation to discern quality assurance requirements across the entire development lifecycle, fostering a culture of quality among individuals. The model, characterised by its holistic nature, strategically incorporates the capabilities of individuals to facilitate the evolution of organisational quality maturity.

We utilise quality gates to partition the development cycle into evenly timed phases. This strategy provides flexibility in implementing diverse project management methods to attain the predetermined level of quality maturity for the underlying deliverable.

This chapter comprises two parts: a definition of the model's components and an initial application of the model in a project.

Definition of the Software Quality Gate Model Components

In the following definition, the model is outlined in its default form, meeting the minimal requirements for its application. A pivotal feature of the model lies in its ability to enhance itself by assimilating lessons learned within the organisation, specifically emphasising transparent communication among participants. While adherence to guidelines for each phase and quality gate remains imperative, those employing the model are entrusted with defining lower-level processes and selecting tools. Consequently, even during the inaugural use of the model, a maturity level exceeding the default version can be realised within the confines of the project or organisation. The definition encompasses components, the meeting structure, and a risk-managed fast lane for the model.

Definition of a Phase:

A phase is defined as a distinct interval during which specific events and activities occur to build a business request. The solution development begins by extracting the requirements from the business request. Deliverables represent requirements in varying maturity statuses, progressing through phases subject to acceptance criteria. Phases, including Request, Conceptual Design, Development, System-Integration, Business-Integration, and Go-Live/Hypercare, may be time-limited depending on the chosen project framework. Each phase has a unique mission statement that outlines its primary focus and objectives. The acceptance criteria are used to identify the specific activities necessary for producing deliverables for the consecutive phase. The default phases follow a sequential order, each building on the previous one, as stated in Table 1 below.

Table 1: Default phases and mission statements

#	Phase name	Mission statement
1	Request	Business requirements are evaluated
2	Conceptual Design	Teams can start work
3	Development	Teams are done with their implementation
4	System-Integration	New implementations work with/within its system/product
5	Business Integration	System with demanded new implementations works within business IT landscape
6	Go-Live/Hypercare	System is released, and hypercare is done

Definition of a Quality Gate

A quality gate (QG) is a decision point for a deliverable at the end of a phase that monitors the maturity of a solution against the defined acceptance criteria specific to that gate. It marks a point where the deliverable is assessed against the agreed criteria, and a decision is made upon them to enter the next phase. The quality gates are enumerated to ease the monitoring of progress.

Our defined default model has the following acceptance criteria for each gate: organisation, software test, and documentation for clarity and ease of reading (cf. Table 2).

Table 2: Quality Gates

Quality Gate	1	2	3	4	5	6
Acceptance criteria organisation	The cost benefit is assessed.	IT-system selected non-functional system requirements assessed functional-system requirements assessed	requirements are developed	all QG3 features are integrated into the system	system with all QG3 features is integrated into IT landscape Go-Live recommendation (test report and risk evaluation) cutover list release preparations	cut overdone software released planned hypercare executed monitoring extended to new features lessons learned
Acceptance criteria test	not available	necessary test levels are decided based on the requirements	QG2 defined test levels exercised run a regression test suite no bugs of category A (showstopper) and B (no workaround) testcases and test infrastructure, including test data, are defined, and prepared for system test	QG2 defined test levels exercised run a regression test suite no bugs of category A (showstopper) and B (no workaround) necessary training provided test infrastructure ready, necessary test data provided	User acceptance Test of demanded features with user's authorisation Run the regression test suite e2e Tests (if necessary) with peripheral systems within the business IT landscape no nugs of category A (showstopper) and B (no workaround)	test completion test report update the regression test suite
Acceptance criteria documents	vision/strategy request-tickets requirements	business vision documents decision documents requirement documents	decision documents functional documentation technical documentation	decision documents test report of QG3/QG4 training documents	decision documents Go-Live documentation cutover list release documentation	hypercare/production bugs documentation

Definition of Roles in the Quality Gate Model

In addition to the project roles, such as program manager, project leader, project manager, and project team, typically found in organisations or project frameworks, the quality gate is not dependent on specific organisational roles. It can be implemented in any participant setting. It is essential to define the role of the Quality Gate Owner in order to manage and lead the model effectively.

The Quality Gate Owner, whether an individual or a group, manages the entire phase within their Quality Gate and ensures that the defined acceptance criteria are met. They are accountable for scheduling alignment meetings with all project members, sharing status updates, and managing impediments as needed. As the gatekeeper of the quality gate at the end of the phase, the Quality Gate Owner is responsible for checking that the deliverables meet the agreed acceptance criteria. Their decision on whether to proceed to the next phase and hand over to the next Quality Gate Owner is based on the acceptance criteria, their alignment with other team members and Quality Gate Owners, and a traffic light system: green for fully approved, yellow for partially approved, and red for not approved. The handover between Quality Gate Owners is viewed as transitioning towards the next maturity state. The responsibility for deliverables is shared among all Quality Gate Owners throughout the entire Quality Gate Model. Each Quality Gate Owner is considered the leader of their respective phase, and all Quality Gate Owners together are leaders of the Quality Gate Model. Additionally, the Quality Gate Owner monitors and initiates continuous improvement processes that have been addressed by lessons learned, focusing on end-to-end improvements in capabilities, tooling, and processes.

Definition of Quality Gate Meetings

The Quality Gate Meeting is a coordination meeting that involves at least all Quality Gate Owners and can be extended to all project members and interested people. Its main objective is to obtain a clear and comprehensive understanding of the quality maturity of the requirement, including quality gate checks, rechecks of requirements with the previous status “with limited quality maturity”, and checks against the Gantt chart. The meeting also evaluates whether the solution is still on track concerning time, scope, and budget or whether further measures must be undertaken.

Definition of the risk-managed Fastlane

The Fastlane is designed for requirements that cannot be implemented within the regular Quality Gate model, such as those related to business or regulatory issues. These requirements are time-limited rather than quality-dependent and therefore prioritised to meet the time goal rather than the agreed quality. This may result in less documentation, fewer tests, or an implementation "hack" that does not necessarily prioritise sustainability. The primary objective is to achieve a productive and functioning Proof of Concept. The traffic light at the agreed quality gates often shows yellow and red. To meet requirements that contradict the quality agreement of the Quality Gate model, specific activities must be planned and executed, such as risk analysis, risk-based test levels, documentation types, and a “firefighter” hypercare. Once the time goal is met, the requirement must be subjected to the regular Quality Gate process to achieve the agreed quality maturity and transition from a PoC-maturity to a Quality Gate releasable maturity. The Fastlane can also be utilised in design thinking processes to obtain a releasable prototype but should only utilised with care.

The overview of the Software Gate Model, depicted in Figure 1, encompasses the entire lifecycle from initialisation to operation. For illustration purposes, we designate the conductor

as the Quality Gate Owner, responsible for verifying the agreed acceptance criteria after each phase. The train is the vehicle transporting artefacts through each phase and into production.

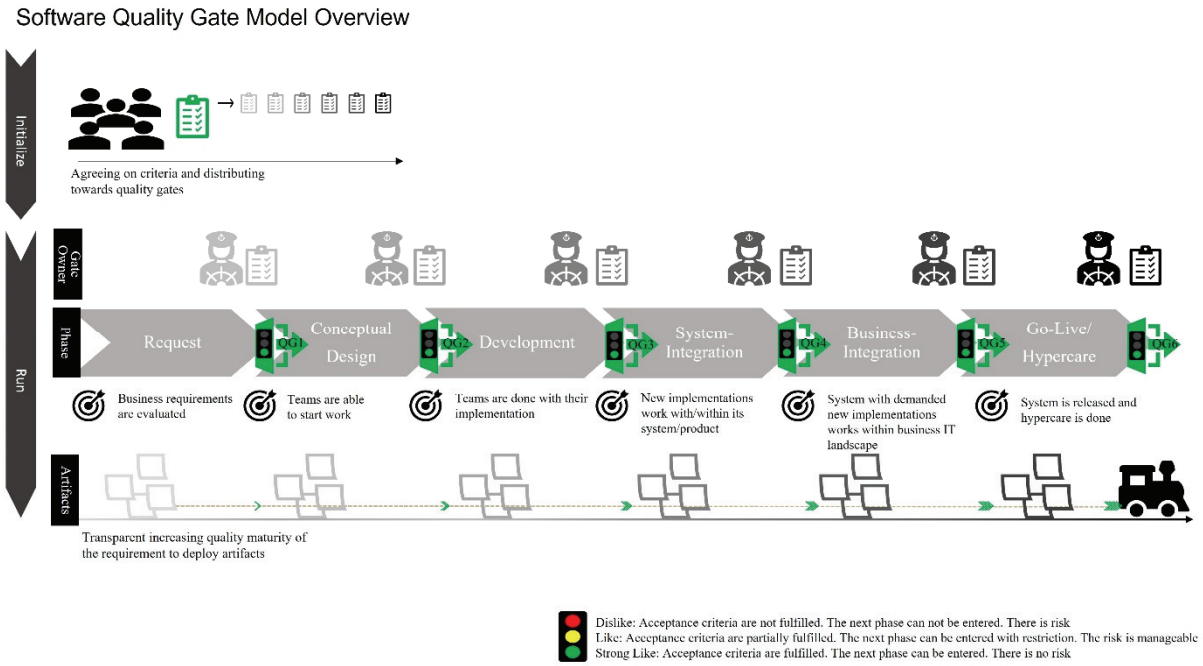

Figure 1: Software Quality Gate Model

Workflow in the model

The model's workflow is centred on the participants' attitudes towards communication, transparency, and collaboration. The model is abstract in nature and consists of different phases and acceptance criteria that must be met. These acceptance criteria set boundaries, and the participants' knowledge and problem-solving skills are used to generate processes to achieve the required documentation, implementation, testing, and test defect management. The Quality Gate Model (Figure 1: Software Quality Gate Model) outlines the processes agreed upon by the participants, which occur within the quality gate boundaries and culminate in delivering the deliverables. The quality gate owners are responsible for managing the model's risks, and they do this by ensuring the defined maturity is maintained during each quality gate alignment meeting.

Recommendation for Tools

The model itself does not have any restrictions on tools. A requirement to manage the quality gate model is that the requirements and their status can be visualised. A recommendation for visualisation is KANBAN and, therefore, a tool that supports this. Tooling for the implementation is based, by definition, on the capabilities of the project members and can vary from each organisation for documentation, testing, test automation and deployments.

Applying the model for software refactoring

The model can address not only business requests and requirements but also IT requests such as system upgrades, patches, and hotfixes for existing functionalities. It is important to note that changes to the platform can affect the functionality, resulting in the need for new implementations or fixes. As a result, all phases and quality gates must be met. Even if no new implementation is required, the proposed solution must be evaluated against the agreed-upon acceptance criteria, and additional measures such as documentation, software code, and testing must be considered.

Lessons Learned and Improvements to the Software Quality Gate Model

The Quality Gate Owner is responsible for initiating Lessons Learned meetings. These meetings may be initiated based on feedback generated during Quality Gate alignment meetings or after the process, depending on the feedback received. The focus of Lessons Learned is to enhance the entire Quality Gate Process. This can result in additional or fewer acceptance criteria in the Quality Gates and the adoption of alternative tools and methods to improve the model. Quality Gate Owners coordinate with all participants to identify efficient and effective improvements for the entire process, focusing on enhancing the process as a whole rather than improving individual phases and Quality Gates.

Example of the Implementation of the Quality Gate Model

The procurement of technical and IT resources is the responsibility of the central procurement department, which has requested a modern system to collaborate with service partners. This request has resulted in five use cases addressing purchasing, inventory management, order management, and invoicing processes. The project is part of a larger digitisation program for the department, which oversees external consulting services, technical services, IT equipment, and branch equipment. The current ordering and procurement system is based on Microsoft SharePoint, email via Excel Spreadsheets, SAP R/3 procurement and invoicing in SAP R/3.

As other departments migrate their processes from SAP R/3 to SAP S/4, the finance and accounting department, in particular, needs the central procurement department to modernise their systems and be SAP S/4 ready. Due to the dependencies between projects, the procurement project required frequent alignments and requests for other departments to advance their implementation of the procurement system and SAP S/4 migration plans.

As a result, the procurement project's scope has increased to include several processes needed in SAP S/4 and SAP S/4 Vendor Invoice management. To tackle this complexity, the procurement team's project organisation was set up with an external implementation partner responsible for SAP Ariba and an internal implementation partner consisting of five different business departments and the internal IT department. The estimated project duration was six months, and the project team included approximately 35 members with different backgrounds and knowledge of project work, using agile and waterfall concepts, depending on their previous projects.

The execution of the procurement project was viewed as a project with a generally low level of risk, making it an ideal choice for testing the specified model. The goal was to assess whether the model aligns with the criteria and aids the organisation in carrying out projects with a specific level of quality. Table 3 details the algorithm illustrating how the underlying project was aligned and integrated with the Quality Gate Model.

Table 3: Setup Algorithm

#	Algorithm	Implementation in the project
1	Align the phases	The process involved aligning the phases between the third-party vendor and the project team. Specifically, we needed to align the Activate Phase from the third-party vendor with our quality gate model.
2	Align the acceptance criteria	We have reviewed all the criteria outlined by the quality gate model, Activate methodology, and project team and aligned them towards the mandatory criteria.
3	Acceptance criteria distribution across quality gates	All aligned criteria from the previous step were evenly distributed across the quality gates to create equidistant phases
4	Define Quality Gate Owners	For practical reasons, a designated group, including the project lead, technical project lead, and test manager acted as the Quality Gate Owner for all gates, rather than assigning separate owners for each gate.
5	Define Quality Gate alignment meetings	The Quality Gate Owners reached a consensus that a single alignment meeting for all quality gates was adequate, which took place once a week.
6	Explain to all participants the aligned setup of phases, acceptance criteria and Quality Gate Owner	Once all phases were synchronised and the acceptance criteria for each quality gate were defined with the gate owners, a comprehensive one-page slide containing all the information was created and presented to all the quality gate participants. This included the project team of the SAP Ariba subproject, the steering wheel, and other dependent projects that were aware of the process.
7	Setup a tool for the model	The project's Tool setup utilises a single Jira project with three KANBAN boards of the "Software" project type. The Quality Gate Owner board displays use cases as user stories and has lanes aligned to the project phases, filtered by project = ARIBA AND issuetype = Story ORDER BY Rank ASC. Each user story includes acceptance criteria for the relevant quality gate and subtasks outlining the activities required to meet those criteria. The Quality Gate Owner reviews the user story and subtasks sequentially, passing them to the next phase upon completing the quality gate and creating new subtasks. The subtasks are organised on a separate board known as the working board, which is set up with three lanes (to do, do, and done), and the swim lanes are configured based on stories. This allows project members to organise themselves and address their subtasks under the relevant user story. The third board in the project displays all test bugs, which are distributed among project members. Test bugs are organised as subtasks within a test bug user story, linked to the relevant use case user story.
8	Setup implementation tools	We decided to use document templates that are familiar within the organisation in Office 365 and Atlassian Confluence. Additionally, we mandated the use of SAP ChaRM for deployment purposes. However, we have yet to choose a test automation tool.
9	Run lessons learned	A lessons-learned session has not been held since the project is ongoing.

Discussion

In this chapter, we explore how the procurement project derived advantages from employing the Quality Gate Model defined in the paper rather than another project framework. Additionally, we examine how it effectively addressed the identified pain points from the survey.

The procurement project started with five use cases that needed to be migrated from a SharePoint solution to SAP ARIBA. However, due to dependencies to other projects, there were also SAP R/3 processes that needed to be migrated to SAP S/4. The procurement project used the Quality Gate Model as an implementation method. However, this method was not selected using quality gates since the acceptance criteria for system selection were not implemented in the Quality Gate Model at that time. The project duration was estimated to be six months. Although the Quality Gate Model focuses on the quality maturity for this project, time checks have been integrated at each quality gate to check the latest possible time by which the requirement might reach the defined maturity. In the procurement project, the project staff consisted of thirty-five persons.

One of the main challenges in the project was integrating the project members' varying knowledge backgrounds and experiences in project management and tooling, ranging from classical project management to agile project management with DevOps and test automation. Minor challenges were encountered in aligning the project management method and tooling with other projects, as the ERP system S/4 is a shared system for all projects. Therefore, discussions on sharing release management with a four-system landscape of dev, test, pre-prod, and prod-system were unproductive.

The Quality Gate model focused not only on the procurement project but also dependent projects on the things that mattered – it concentrates on the question "WHAT" needs to be done in order to get a solution released before "HOW" and by whom it gets done, simplifying the complexity of project management by a three-dimensional approach. The model provided a guideline for necessary activities and the sequence of events. When additional project activities that should improve the quality came up, the first step was to check in which phase and, therefore, in which quality gate it was necessary. The second step was to check how the overall Quality Gate Model benefits from the change. In the project, this approach benefits especially from different knowledge backgrounds; it takes the focus away from individual experience and tool setups and empowers people to work together. This results in an aligned team understanding, increasing motivation and commitment to completing all necessary activities.

The setup of the Quality Gate Model framework as the leading project management framework was quickly established since it sets up checklists as a guideline, involves all project team members, empowers them, and has no tool restrictions. The role overseeing the quality gates is that of the Quality Gate Owner. As the Quality Gate Owner in this project, a group of four persons and a project lead from an external partner were selected. The main task of the Quality Gate Owner was to help the project members reach the quality gate by providing coordination and support. The Quality Gate Owner was also identified as the authority to whom any issues should be escalated. During the project, no issues regarding the quality were escalated, which we believe is due to the transparency of the progress of requirements. Since we have the phases with Quality Gates and the Quality Gate Owner frequently updated the maturity status, interested people could track the requirement's progress with a certainty of about 17% per quality gate. The quality checks occurred every time the assignee reported that they were ready for the next phase. Based on the checklist and a traffic light system, the requirement passed or did not pass to the next phase. The formal process of quality checks took place in a weekly Quality Gate alignment appointment, the frequency of which was agreed upon by the project staff. In this

meeting, the quality gate owner aligned the progress, checked the requirements against the quality gates and discussed risks and how to manage them.

One issue that was determined in the alignment meeting is that the Quality Gate 3 ready requirement met the agreed quality standards but that the perception of the requirement was poor, which resulted in numerous requests to change the requirement. The result of the Quality Gate Owners was to start showing the requirement with dependent QG3 requirement in test sessions, i.e., performing a review with the tester and interested parties. The quality gates' status helped distinguish the testable package quickly, and the test sessions helped minimise the misunderstanding resulting in fewer new requirements and changes. We have noticed that by design, the model has only a high dimensional workflow, which we found is another strength of the model because it led to a flexible working mode for the project team without restrictions on how to reach the quality gate.

However, this strength sometimes led to confusion by people unfamiliar with the quality gate model. Because of the high degree of freedom, the quality gates can be seen as very abstract, which seems chaotic. Moreover, document types and document templates have to be realigned and refactored for the model, which seems to be a repetition for most project workers. Another weakness is that the quality gate model concentrates on the overall solution quality. This encourages project members to do further activities not listed in the quality gate to obtain an even higher maturity of their work. We experienced that the number of additional activities that are not mentioned explicitly in the gates was very limited. We saw activities in the tests, such as conducting peer reviews or performing explorative tests. However, discussions about adding activities in the quality gates rose, meaning that not only are “must haves” listed but also “nice to haves”, which would bureaucratise and slow down the release process by passing all gates. An explanation for this weakness is that we have only worked explicitly with this model in this project and have not conducted lessons learned; therefore, the change from a fixed mindset to a growth mindset was not completed, resulting in an understanding that the responsibility of the quality comes from the model, not the people.

With respect to the pain points revealed in the survey, we observed that the high degree of freedom and the maturity level in the model helped to deal with the problem of changes in requirements. The expectation was not to avoid necessary changes but to challenge unnecessary changes. This model helped us to reduce changes in two ways. The quality gates ensured that only mature requirements were planned for implementation, which limited changes due to unclear tasks, and the holistic approach for the solution helped to break requirements into complex and complicated features. The second dimension seems to be very straightforward. However, we experienced difficulties in previous projects. Once a project method is selected, issues are encountered by implementing complicated requirements using an agile methodology or complex requirements with a waterfall approach. As mentioned previously, we have noticed these kinds of requirements, and to give an insight, we explain the warehouse process for internal hardware devices. This warehousing process is a subprocess of two main processes customized in SAP R/3. The requirement was to migrate this process to SAP S/4, and it was known that it has dependencies towards the greenfield finance project, the invoicing project, and the new authorisation concept for the ERP. At first, it seemed to be a waterfall requirement, which we addressed that way, but after several days of implementation and challenging the quality gate, it seemed to be more complex. Several increments of the requirement were identified to be too complex to deal with in a waterfall model, so without hesitation, we divided the requirement into complicated and complex requirements and dealt with them accordingly. Differentiating requirements in agile and waterfall models usually evolve a hybrid model. In our model, the degree of freedom allows them to be handled accordingly and therefore did not

lead to method confusion. Moreover, the quality gates helped to align and manage the divided requirements to govern the progress of dependent requirements and release the solution.

A limitation of the current model is that it concentrates on IT solutions and only works on minor business readiness effects such as training non-IT-system related sub-processes necessary in the integration. The model also needs to be aligned with other projects in terms of transparency and to avoid potential conflicts, e.g., in mapping of the quality maturity for integration scenarios. Also, all project members must agree on working with the Quality Gate model, which can sometimes be difficult to achieve. Another limitation is that although the model is designed without a tool restriction, this is only clear for the management tools. Within the implementation and system landscape, the tools are determined by the selected IT System.

We found no empirical evidence that a holistic quality assurance model would improve or maintain the overall software quality. The procurement project, which is dependent on two other projects, constitutes a fraction of the entire system landscape in the company, i.e., less than 5% of overall system integration. In the procurement project, it was clear at which moment integration testing with peripheral systems should be planned and executed. In contrast, due to a lack of transparency, the dependent projects had to check their integration test readiness regularly. This indicated the possibility that overall agreed acceptance criteria could have resulted in less friction in aligning with dependent products, an outcome which we have flagged for attention in subsequent projects using the quality gate model.

Conclusion

In summary, initiating our quality gate model was prompted by the need to establish a foundational standard for quality assurance activities in software testing and to create a transparent model that aligns the interdependencies of projects.

Our objective was rooted in addressing the lead question: "Are there any indications of obstacles in the phase transitions of (software) products?" Through this comprehensive approach, we aimed to guide the organisation toward essential practices for software implementations and governance. As a result, we formulated a model tailored for IT software development projects, designed to be predictable and adaptable across both classical and agile project management methodologies, independent of specific tools, and dedicated to enhancing the quality maturity of the delivered software. This model, transparent and equipped with control mechanisms, was successfully developed, and implemented in a project.

Despite its notable strengths, including flexibility and a focus on quality, the model has limitations. These include the essential commitment of all participants and the need for increased transparency during concurrent software development projects. Quantifying improvements in software product quality posed challenges, but qualitatively, the model significantly enhanced transparency and communication.

Looking ahead, our future plans involve refining cross-project communication and expanding the model's usage to foster a shared understanding and better alignment between related projects so that frequent changes in requirements can be reduced. While acknowledging its limitations, we find satisfaction in the current status and value of the model within our pilot project procurement platform. To unlock additional potential, we aim to encourage the further model's utilisation in our recently designed IT life cycle process, anticipating continued positive impacts on overall project outcomes.

References

Ambartsoumian, V., Dhaliwal, J., Lee, E., Meservy, T., & Zhang, C. (2011). Implementing quality gates throughout the enterprise it production process. *Journal of Information Technology Management*, *22*(1), 28–38.

Anderson, D. J. (2010). *Kanban: Successful evolutionary change for your technology business*. Blue Hole Press.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., & Jeffries, R. (2001). *The Agile Manifesto*.

Bhanushali, A. (2023). Ensuring Software Quality Through Effective Quality Assurance Testing: Best Practices and Case Studies. *International Journal of Advances in Scientific Research and Engineering*, *26*, 1.

Bhavsar, K., Shah, V., & Gopalan, S. (2020). Scrumban: An agile integration of scrum and kanban in software engineering. *International Journal of Innovative Technology and Exploring Engineering*, *9*(4), 1626–1634.

Boehm, B., & Turner, R. (2003). Using risk to balance agile and plan-driven methods. *Computer*, *36*(6), 57–66. https://doi.org/10.1109/MC.2003.1204376

Crispin, L., & Gregory, J. (2009). *Agile testing: A practical guide for testers and agile teams*. Pearson Education.

Ebert, C., & Paasivaara, M. (2017). Scaling agile. *Ieee Software*, *34*(6), 98–103.

Galin, D. (2004). *Software Quality Assurance: From Theory to Implementation*. Pearson Education.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–105. https://doi.org/10.2307/25148625

Hrabovská, K., Rossi, B., & Pitner, T. (2019). Software testing process models benefits & drawbacks: A systematic literature review. *arXiv preprint arXiv:1901.01450*.

Karlstrom, D., & Runeson, P. (2005). Combining agile methods with stage-gate project management. *IEEE software*, *22*(3), 43–49.

Kerzner, H. (2017). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. John Wiley & Sons.

Kläs, M., Lampasona, C., Nunnenmacher, S., Wagner, S., Herrmannsdörfer, M., & Lochmann, K. (2010). How to evaluate meta-models for software quality. *Proceedings of the 20th International Workshop on Software Measurement (IWSM2010)*.

Mahfuz, A. S. (2016). *Software Quality Assurance: Integrating Testing, Security, and Audit*. CRC Press.

Project Management Institute. (2021). *The Project Management and A Guide to the Project Management Body of Knowledge* (7. Aufl.). Project Management Institute.

Reddy, A. (2015). *The Scrumban [r] evolution: Getting the most out of Agile, Scrum, and lean Kanban*. Addison-Wesley Professional.

Rodrigues, M. C., Domingues, L., & Oliveira, J. P. (2023). Tailoring: A case study on the application of the seventh principle of PMBOK 7 in a public institution. *Procedia Computer Science*, *219*, 1735–1743. https://doi.org/10.1016/j.procs.2023.01.468

SAFe 5.0 Framework. (2022). Scaled Agile Framework. https://www.scaledagileframework.com/

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

Snowden, D. J., & Boone, M. E. (2007, November 1). A Leader's Framework for Decision Making. *Harvard Business Review*. https://hbr.org/2007/11/a-leaders-framework-for-decision-making