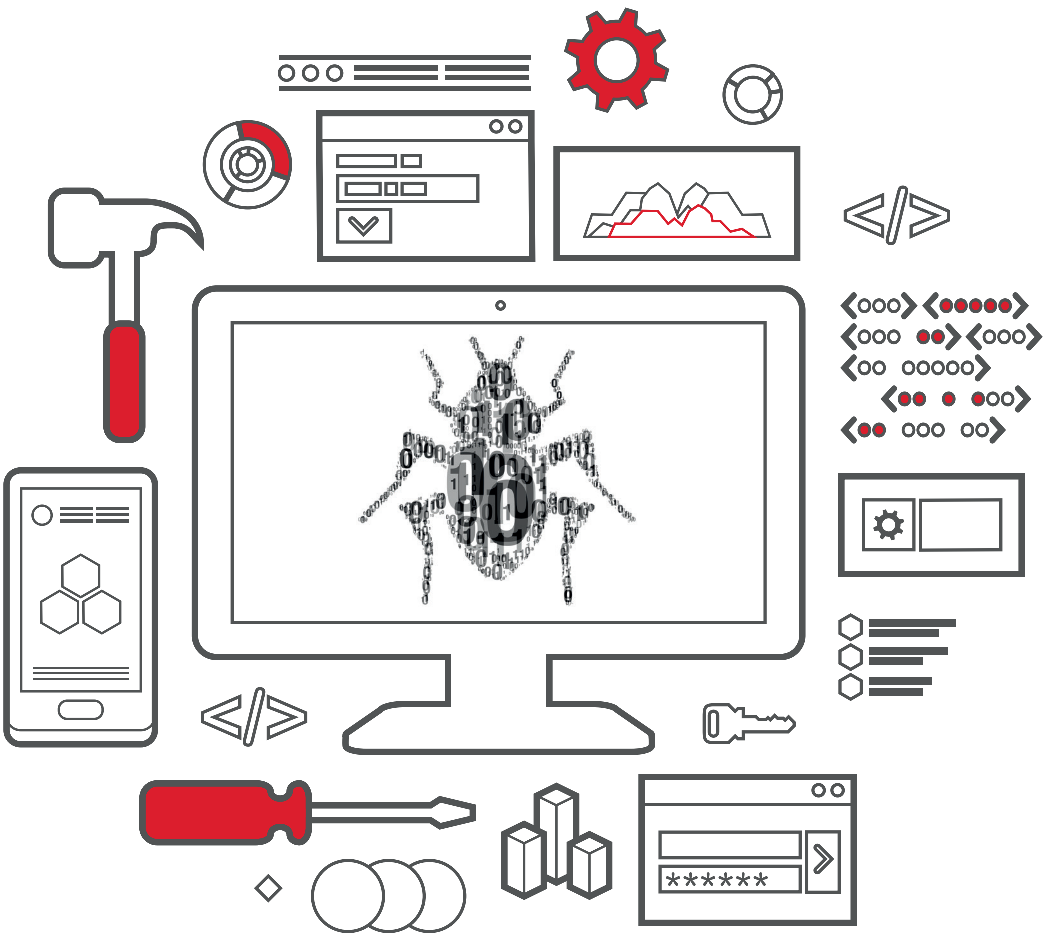
Quality



Managing Software Quality Holistically rather than Fixing the Bugs

The framework for quality managers to enable their organization to deliver high-quality software

Porsche Consulting

Strategic Vision. Smart Implementation.

Introduction

The quality management system for software is the framework that enables an organization to consistently deliver high-quality results in software development (figure 1). Quality management that supports software development and the integration of software and hardware represents significant levers of improvement, as the increase in a product system's complexity is largely driven by software.

Increasing digitalization in most industries has resulted in a high demand for tailored support using quality management for software. Porsche Consulting has developed a highly effective approach for its quality management system for software that addresses the most relevant levers of improvement in software development while specifically taking into account a software's synchronization with hardware development. This holistic approach includes three sprints. A company-specific procedure model is developed and matched to a tailored quality organization, which is in turn aligned with processes supported by appropriate IT tools.

The business case is built on the potential of minimizing debugging, currently accounting for more than 30 percent of capacity in software development. The amount currently used for management, documentation, and support can be reduced to about 50 percent by means of a tailored and coordinated approach to the quality management of software.

Quality management system for software

is a framework that enables an organization to consistently deliver high-quality results in software development.

© Porsche Consulting

Figure 1. Definition of a management system for software quality

Software quality management requires a systematic and holistic approach

Software may not always be visible, but errors related to its use can easily have tangible and costly consequences. Irish Rail's signal failure in May 2019 or British Airways' computer breakdown at London Heathrow in 2017 are just two examples of numerous software errors costing millions of euros and directly affecting the population.

A report by the Standish Group[1] (figure 2) indicates that 71 percent of all software projects do not meet the general project conditions. Only 29 percent are finished successfully, while 52 percent exceed budgets and deadlines or do not fulfill requisite functionality. An additional 19 percent of projects are cancelled.

71%
of all software projects do not meet the general project conditions.

52%
of all software projects exceed budgets and deadlines or do not fulfill requisite functionality.

19%
of all software projects are cancelled.

© Porsche Consulting

Figure 2. Share of software projects that do not meet the conditions or requirements or are cancelled.[1]

According to a 2012 report by Capers Jones[2], a specialist in software engineering, the amount of actual coding in software development decreases as its quality increases. Coding encompasses only 18 percent of the development phase. Typically, almost 50 cents out of every Euro will go to finding and fixing bugs. However, costs associated with fixing bugs are often neither measured nor tracked systematically. The potential of a systematic approach to software quality becomes apparent which includes defect prevention, pretests, defect removal as well as formal testing based on various methods.

The customer's perception of products and services is changing rapidly. They expect any combination of hardware and software to be easily operable, highly intuitive, and bug-free from the outset. Software quality has become one of the most critical success factors leading to customer satisfaction. Porsche Consulting has developed an innovative framework for quality management to assist their clients in meeting these expectations.

A consistent, common understanding of the software system's architectural split is essential to effective quality management. Its architecture lays the foundation for applying quality methods and processes as well as defining software maturity. The latter's inclusion in software quality management requires evaluating maturity and considering the software's function, its technical structure and its procedural aspects. Maturity should not be understood as a constant state, but rather as evolving dynamically from software development and the product development process. It may therefore be suitable to review maturity and specific criteria at well-defined milestones during the project.

The relevance of a quality management system for software is based on a series of particular challenges and key characteristics:

- Expanded use of software applications causes software development to gain importance within the value chain.
- Increasing device connectivity requires higher software standards regarding interface communication architecture.
- The growing complexity of software applications requires a systematic approach to deliver high-quality results.
- The properties and requirements of novel products with regard to software cannot be accurately defined at the start of product development.
- Customers may not be able to articulate or quantify their requirements regarding software or can only do so in a subjective way.

[1] https://www.standishgroup.com/sample_research_files/DemoPRBR.pdf
[2] https://www.ifpug.org/content/documents/Jones-SoftwareDefectOriginsAndRemovalMethodsDraft5.pdf

A

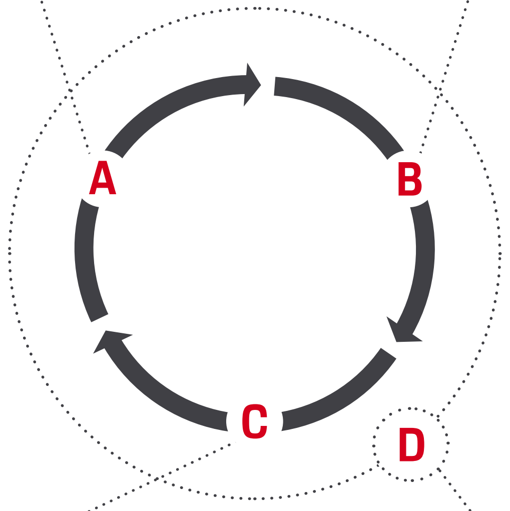
User Interface

Assurance of direct interaction between the user and the technical system

B

Sensors, logic, and actuators

Sensors detect current state, logic captures difference as compared to ideal state, actuators adjust accordingly



C

ECU (electronic control unit)

Electronic modules process input from sensor into output to actuators according to logic

D

Surrounding technical system

The system is subject to interference by user and to adjustments by actuators

© Porsche Consulting

Figure 3. Embedded software as an example for the application of quality management system for software

Software quality depends on programmers and thus on human factors to a significant extent. Consequently, both customers and programmers as well as the quality department's organization are the main aspects in the design of a quality management system for software. In addition, software development is characterized by continual updates and incremental improvements during all phases of creation, production, and actual usage, which can help eliminate errors before customers are aware of them. Quality management for software is demonstrated using the example of embedded software (figure 3), which is characterized by four aspects: the user interface; sensors, logic, and actuators; the electronic control unit (ECU); and the surrounding technical system. These aspects can be extrapolated to fields beyond the automotive industry.

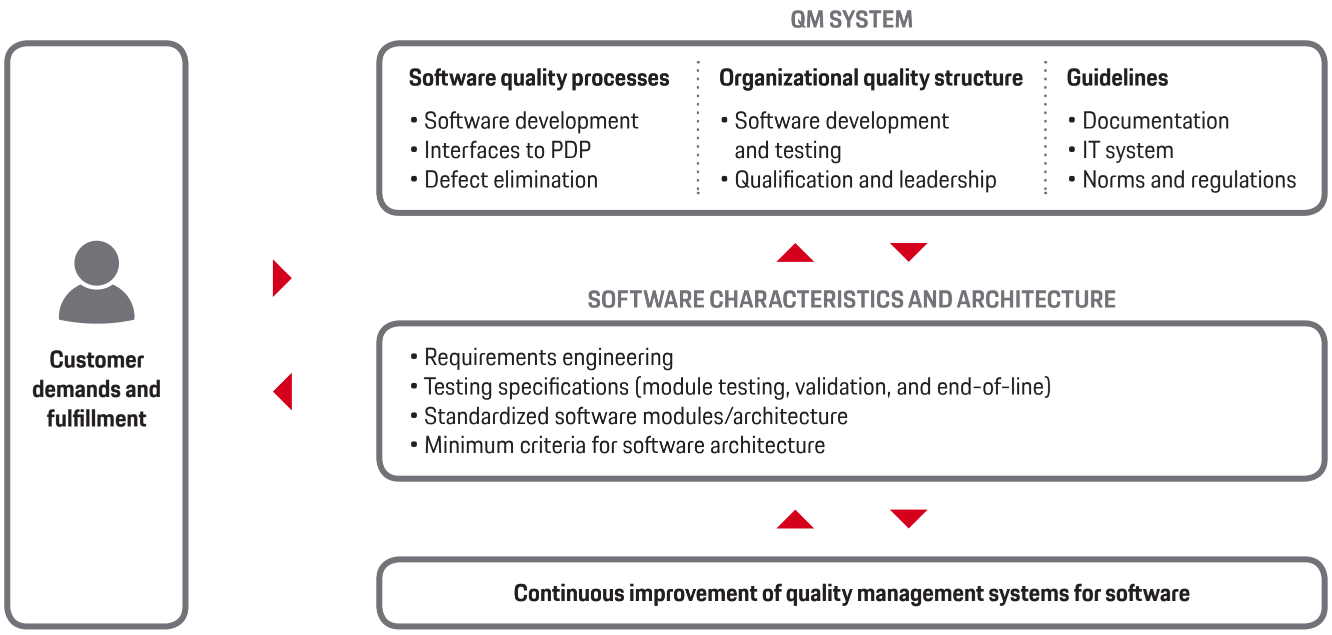
The six pillars of quality management systems for software

The framework, as depicted in figure 4, has been derived from several standards (e.g., ISO9000 family, ASPICE, and CMMI) and includes the following relevant aspects to ensure high-quality software development:

- Enabling an organization to deal with the exponential increase in complexity of software functions
- Dealing with the increasing necessity to integrate software functions into the development process
- Improving delivery time, customer acceptance, and flexibility to change
- Providing a holistic approach for software development throughout the life cycle

Understanding customer demands is the key starting point. Software quality processes and frameworks offer a systematic approach to ensuring high-quality software. The appropriate, relatively agile development process takes hardware development into account. Existing product development processes (PDP), and defect elimination processes are also considered.

An organizational structure capable of addressing the challenges inherent in software quality processes needs to be individualized to an organization's background, focus, strategic outlook, and the like. Guidelines like ISO90003, ISO15504, ISO25000, and other IEEE-regulations play a decisive role in customer acceptance but also in compliance to regulations. A successful quality management system for software needs to address specific software characteristics—such as approaches to software testing; modular, platform-based architecture; and a product's ability to receive updates over-the-air. A quality management system should undergo continuous improvement throughout its application in development projects.


QM SYSTEM
Software quality processes
• Software development
• Interfaces to PDP
• Defect elimination
Organizational quality structure
• Software development and testing
• Qualification and leadership
Guidelines
• Documentation
• IT system
• Norms and regulations
SOFTWARE CHARACTERISTICS AND ARCHITECTURE
• Requirements engineering
• Testing specifications (module testing, validation, and end-of-line)
• Standardized software modules/architecture
• Minimum criteria for software architecture
Continuous improvement of quality management systems for software
Customer demands and fulfillment

© Porsche Consulting

Figure 4. The main pillars of quality management system for software

Customer demands and fulfillment

Another important step is determining the product's advantages with a particular focus on the combination of software and hardware, which should then be compared to customer needs and expectations. This requires knowledge about the product's suitability to meet the required functions. Developing a suitable communication concept, creating prototypes, and learning from case studies all contribute to making the product a tangible experience for the customer. Such activities aim to:

- Increase the accuracy of cost and time estimates
- Eliminate costly changes in later phases of development
- Reduce development efforts and task duplication
- Improve communication with stakeholders
- Document details for future reference

Software quality process

Quality in software development is ensured when guiding structures are defined and systematically followed for the following three levers.

01 Development process: a tailored approach supports quality in the software development process.

02 Interfaces: quality is supported by defining interfaces in the development of software and hardware.

03 Debugging: a systematic defect elimination process ensures the required and desired quality of the final software.

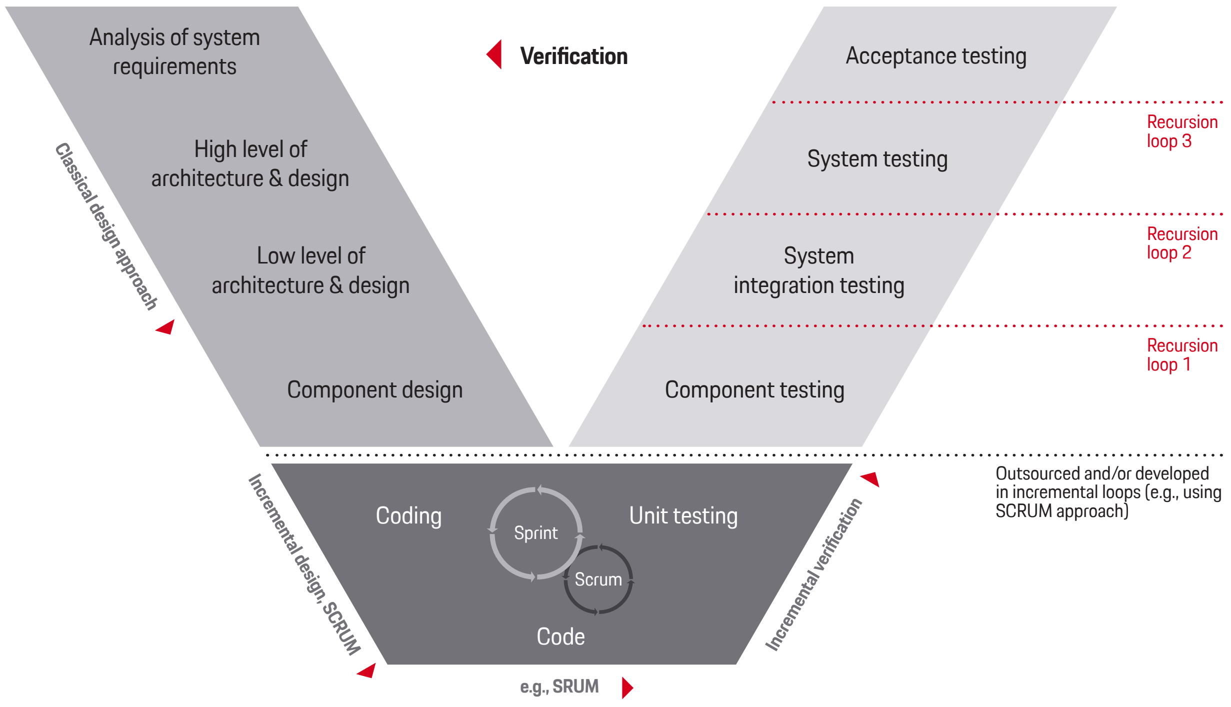
There are several generic software development models with different characteristics and ways of ensuring quality; each is particularly suited to a certain type of project (figure 5).

	Waterfall model	Spiral model	V-model	Rational unified process	Agile
Key to quality assurance	Returning to previous step(s) if realized quality level does not meet requirements	Cyclical reviews of incrementally improved prototypes	Adjusting requirements and design aspects based on test results	Recurring tests, the use of guidelines, templates, reports, and checkpoints	Frequent checks, close team interaction, flexible organization
Particularly suited to	Projects in which most requirements are known and few changes are expected	Projects that focus on risk and have incomplete requirements	Projects with special focus on software and hardware integration	Projects that are driven by use cases and focus on complete software systems	Projects that require flexibility and transparency with a minimum amount of time

© Porsche Consulting

Figure 5. Alternative software development models with different characteristics

The development models that most frequently combine these three levers are the V-model and the agile approach. The V-model is characterized by a sequential path of designing software followed by an equally sequential route to verification by testing the design. Quality is ensured by adjusting requirements and design characteristics based on test results. In the agile approach, quality is ensured by continual checks and interaction with the voice of the customer. A software backlog documents all the requirements that may evolve over time. Several sprints are realized by self-organized scrum teams who maximize the development project's flexibility to react to changes.


Analysis of system requirements
High level of architecture & design
Low level of architecture & design
Component design
Classical design approach
Verification
Acceptance testing
System testing
System integration testing
Component testing
Recursion loop 3
Recursion loop 2
Recursion loop 1
Outsourced and/or developed in incremental loops (e.g., using SCRUM approach)
Coding
Sprint
Scrum
Unit testing
Code
Incremental design, SCRUM
Incremental verification
e.g., SRUM

Quick facts

- Sequential path to software design with increasing detail
- Agile approach to designing and verifying code incrementally
- Sequential path to design verification by testing with increasingly broad scope
- Recursion loops at various integration levels may require updating software backlog
- Can be extended to integrate hardware by parallel approach

© Porsche Consulting

Figure 6. The individualized hybrid between the common V-model and the agile approach

These approaches can be combined with each other and, when tailored to a development project, may result in an appropriate hybrid model. The tailored approach is characterized by a hybridized methodology one could call the "V-scrum fall" (figure 6). It allows teams to combine agile practices like scrum with elements from the V- or waterfall model and tailor them to a specific development project.

Organizational quality structure

In the context of a quality management system for software, a company-specific organizational structure is defined by five structural dimensions (figure 7). A six-step approach helps determine the most suitable organizational structure.

Step 01 | Requirements and target states
The requirements of and interfaces for customers must be known or defined, and once established, the company goals can be set and incorporated into the organizational design. It is also important to consider the requirements of the company as well as its employees. The requirements and target states provide information about how formalized the quality structure should be.

Step 02 | Process archeology
When (re)designing an organizational structure, it is necessary to analyze existing processes and work out the structures implicitly contained therein. Analysis criteria should include the complexity and repetition frequency of the processes themselves. They provide information about the degree of specialization and the possibility of making decisions in a centralized or decentralized manner.

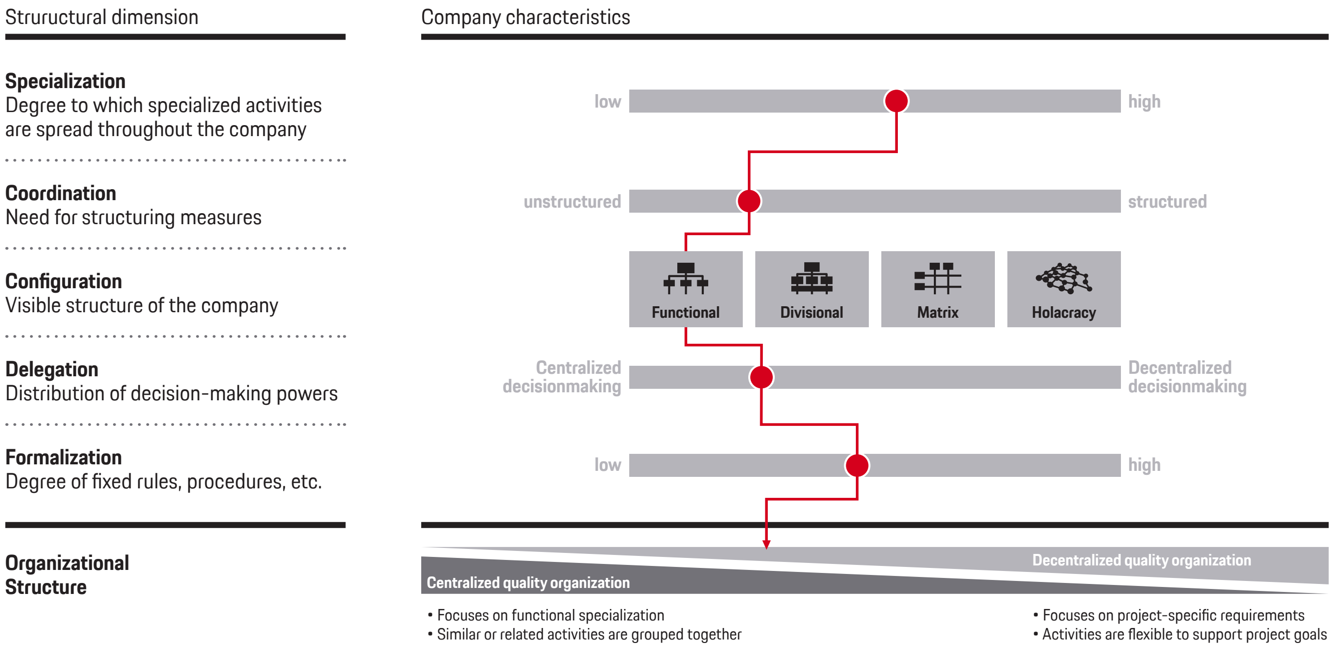
Step 03 | Task synthesis
The external appearance of the organization (configuration) is defined by the logical content and factual combination of activities resulting from process archeology and the activities' assignment to groups or teams. The organizational structure manifests itself in a company's organizational chart.

Step 04 | Responsibility definition
The responsibilities and competences of each organizational unit must be determined, which helps specify the organizational structure.

Step 05 | Interface definition
The interfaces between the organizational units are defined according to the process archeology and the preliminary organizational structure. The interface definition also helps increase the organizational structure's degree of detail.

Step 06 | Organization structure
The final step is rearranging the organization based on the target state and steering the processes to implement the change. Employee qualification, leadership, and corporate culture in particular play an important role in reorganization. These aspects need to be taken into account comprehensively.

The complex interplay between all of these steps and structural dimensions calls for an organizational design that is tailored to the individual organization. Benchmarks may help to validate a certain set-up.


Struructural dimension
Company characteristics
Specialization
Degree to which specialized activities are spread throughout the company
low
high
Coordination
Need for structuring measures
unstructured
structured
Configuration
Visible structure of the company
Functional
Divisional
Matrix
Holacracy
Delegation
Distribution of decision-making powers
Centralized decisionmaking
Decentralized decisionmaking
Formalization
Degree of fixed rules, procedures, etc.
low
high
Organizational Structure
Centralized quality organization
Decentralized quality organization
• Focuses on functional specialization
• Similar or related activities are grouped together
• Focuses on project-specific requirements
• Activities are flexible to support project goals

© Porsche Consulting

Figure 7. Five structural dimensions to consider when developing an organizational quality structure

Guidelines

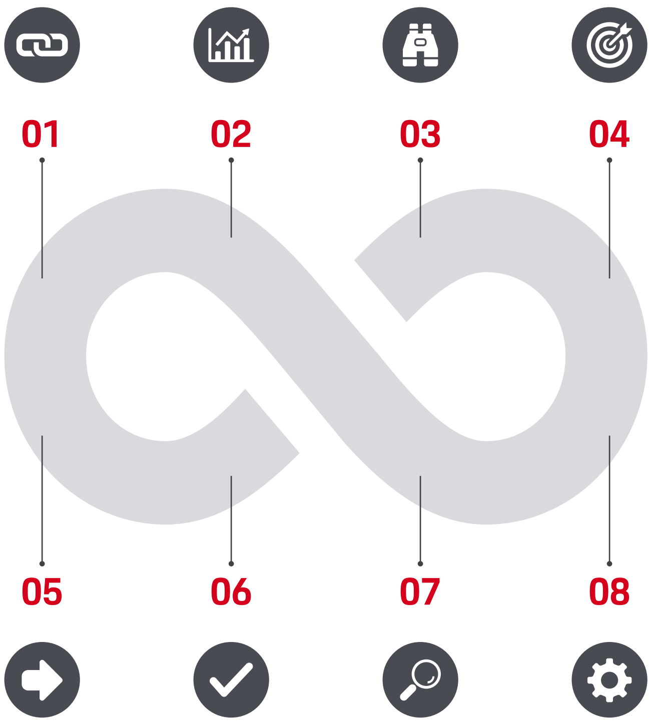
Within the framework of a quality management system for software, guidelines are understood as all measures that serve to document the development process in addition to mandatory guidelines for the software development process itself. The documentation is divided into the four dimensions of process, project, system, and quality. The right mix of methods should be tailored to business needs and should include as few system interfaces as possible. An IT-supported document management system is especially advisable for securing documentation of project results. Change and release management can be supported equally well by a suitable IT tool, however, as can software testing and life-cycle management. Software testing can be accelerated by using intelligent automation and smart analytics.

Software characteristics and software architecture

The complete definition of all relevant software properties forms the basis for ensuring that all desired functionalities of the software are available and can be experienced. The requirement-based description of the software's characteristics is particularly relevant, since the early determination is crucial for a functional design of the software architecture. The software architecture has a significant impact on the functionality of the software (e.g., access or startup speed). The most suitable software architecture can only be selected if the most prevalent requirements for the software are defined. Software is developed in modules which build on each other much like building blocks. These blocks may need to be complemented to address the desired functions. In this way, software continuously evolves. Using existing software modules with proven quality enhances efficiency and in turn the quality of software development.

Continuous improvement in a systemic context

Continuous improvement of the software quality is based on a systematic and recurring approach (figure 8). These improvements aid in enhancing value for the customer. The tracking of software changes and the application of new software functionalities are as important as efficient error detection. Preventive measures should be deduced in order to avoid mistakes from the outset. A comprehensive view achieved by evaluation and prioritization is advisable.



01 Monitor the integration, deployment, release, and delivery

02 Improve the efficiency of detection

03 Identify opportunities to prevent defects

04 Focus on a comprehensive view of business risk

05 Provide a workflow for prioritizing fixes and quality improvements by preventing recurrence

06 Refine the process continuously

07 Fix bugs/defects

08 Manage the release into the next stage of release cycle

© Porsche Consulting

Figure 8. Aspects for continuously improving software quality

Measures balancing effort and benefit should be anchored in the improvement process. As a result, corrective measures and quality improvements are initiated preventively. The tracking of the implementation of measures, their effectiveness, and the inclusion of lessons learned form the basis of the framework for quality management of software. Eight success factors have been identified that particularly contribute to continuous improvement (figure 9).

Implementing learned lessons and anchoring them in the organzation is a transparent pursuit of potential and the realization of measures and goals derived from such a pursuit. Successfully implementing measures enables the realization of strategies.

01
Mind-set
Holistic mind-set about quality

02
Reactivity
Learning, implementing, learning

03
Flexible organization
Implementing changes

04
Information network
Using available data

05
Modularity
Building blocks

06
Learning from customer response
Customer use

07
Cybersecurity & data security
Protecting data

08
Training, development
Taking challenges

© Porsche Consulting

Figure 9. Success factors fostering continuous improvement

Business case

The return on investment in quality management for software is not always quantifiable, but this is also true for other quality measures. While costs linked to certain quality improvement measures can be quantified, their benefits often cannot. For example, averted damage can hardly be monetized and the costs of nonconformity to regulations only anticipated. The business case of a quality management system for software is built on the observation that about 70 percent of software development does not add value (Figure 10). Only 30 percent of activities add value through coding or creative work that focuses on realizing the desired functionality.

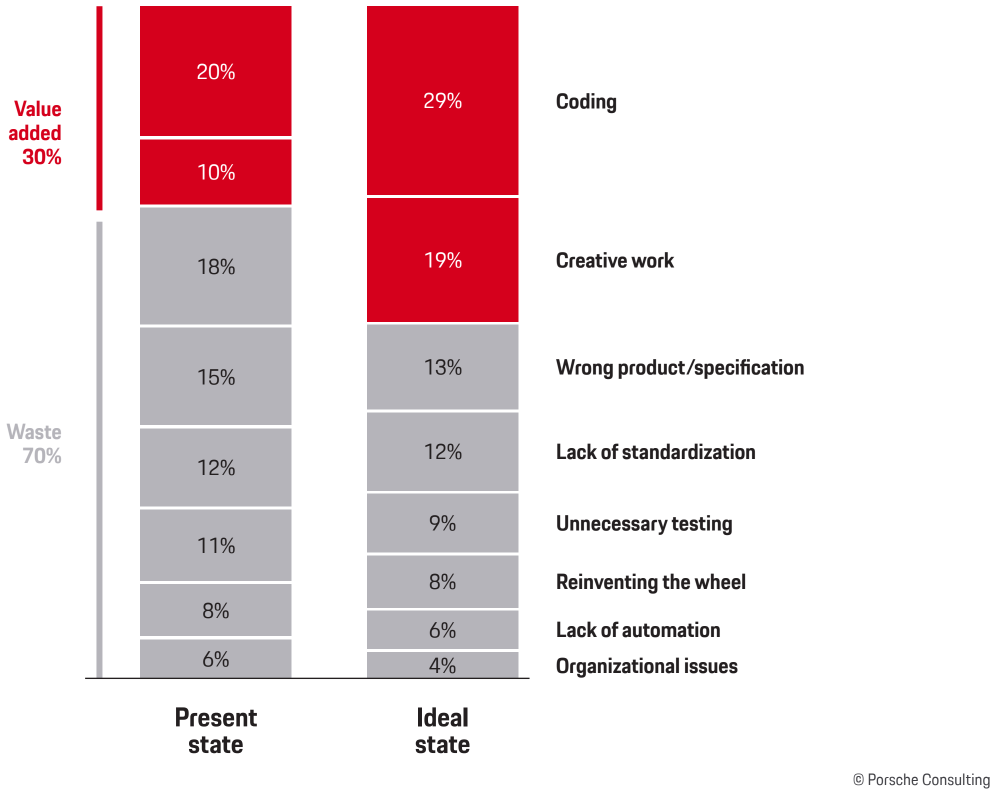


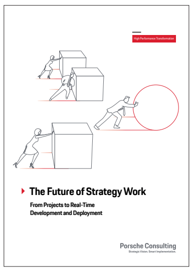
Figure 10. The share of activities during software development that add value can be increased significantly (estimation by Porsche Consulting based on survey conducted in 2017)

Identifying the root cause of software errors becomes more difficult as networking increases among systems and their software. The combination of software and hardware amplifies this difficulty and emphasizes the necessity and potential of a systematic approach to software quality. The quality management system for software as introduced is able to address the specific needs and challenges in software quality management that an organization faces.

Further reading

Auswirkungen der Elektrifizierung des Automobils auf den deutschen Maschinenbau
Porsche Consulting

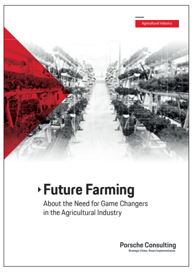
Auswirkungen der Elektrifizierung des Automobils auf den deutschen Maschinenbau


The Future of Strategy Work
From Projects to Real-Time Development and Deployment
Porsche Consulting

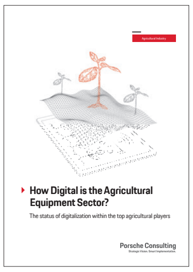
The Future of Strategy Work

Business Process Management Reloaded
Eight levers to turn process management into a competitive advantage of high-performance enterprises
Porsche Consulting

Business Process Management Reloaded


Future Farming
About the Need for Game Changers in the Agricultural Industry
Porsche Consulting

Future Farming


How Digital is the Agricultural Equipment Sector?
The status of digitalization within the top agricultural players
Porsche Consulting

How Digital is the Agricultural Equipment Sector

Authors

Oliver Stahl
Associate Partner

Contact
+49 170 911 4330

Michael Bartholdt
Senior Berater

Sebastian Roth
Senior Berater

Dr. Dominik Rößle
Senior Berater

Porsche Consulting

Headquartered in Bietigheim-Bissingen, Porsche Consulting GmbH is a subsidiary of the Stuttgart-based sports car manufacturer Dr. Ing. h.c. F. Porsche AG. Founded in 1994, the company currently employs 600 people and is among the top 10 management consultancies in Germany (Lünendonk analysis). Active around the globe, it has offices in Stuttgart, Hamburg, Munich and Berlin as well as in Milan, São Paulo, Atlanta, Belmont (Silicon Valley) and Shanghai. Following the principle of "Strategic Vision, Smart Implementation," its experts support companies worldwide primarily with their major transformations, the improvement of their performance, and enhancement of their innovative capacity. Their clients are large corporations and medium-sized companies in the automotive, aviation and aerospace industries, as well as industrial goods. Other clients originate from the financial services, consumer goods, retail, and construction sectors.

Strategic Vision. Smart Implementation.

As a leading consultancy for putting strategies into practice, we have a clear mission: we generate competitive advantage on the basis of measurable results. We think strategically and act pragmatically. We always focus on people—out of principle. This is because success comes from working together with our clients and their employees. We can only reach our aim if we trigger enthusiasm for necessary changes in everyone involved.

Porsche Consulting

Stuttgart | Hamburg | Munich | Berlin | Milan | São Paulo | Atlanta | Belmont | Shanghai

www.porsche-consulting.com

© Porsche Consulting 2019