mgm A12-Project Standards

Handling Software Projects With the A12 Development Platform

January 2026

Innovation Implemented.

Legal Notice

Contents

01

Management Summary

1. Management Summary

How can robust, secure, and long-lasting enterprise software be built as quickly and cost-effectively as possible? Model-based software development provides the answer. With the help of mgm's A12 development platform, the typical efforts involved in developing business applications can be reduced — primarily by separating technical and business concerns.

However, simply using a platform does not guarantee project success. The development and operation of secure, scalable, cost-efficient, and high-performance enterprise applications requires well-thought-out and integrated production processes as well as expertise in individual development and system integration. This white paper summarizes the most important standards, procedures, and stages of software production specializing in enterprise applications, which mgm uses in projects based on A12.

What you need to know

- **Development processes – standardized instead of customized**
  The increased demands placed on enterprise software — including cyber risks, usability expectations, cloud operations, and the pursuit of digital sovereignty — require integrated development processes. Developing from scratch is neither efficient nor appropriate in today's world.

- **Partnership approach through model-based development**
  The use of a model-based development platform such as A12 enables new forms of collaborative development of customized software. Essential parts of the application are specified in models according to the low-code principle – without programming. This results in new ways of working and opportunities for participation for business departments in all phases of the software life cycle.

- **The right setup for organization and technology**
  Encapsulating domain-specific content in models and rules creates independence from technology and increases the lifespan of the application. However, the right organizational and technical setup is required to ensure that modeling and software engineering work together seamlessly.

- **Platform approach based on open source**
  The consistent use of open source is a prerequisite for a long-term successful platform approach. Openness and transparency prevent vendor lock-in, ensure connectivity to neighboring systems, and enable independent security analyses.

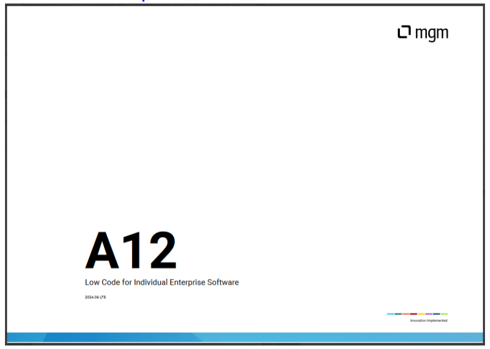
- **Automation with model-based tooling**
  The use of models opens up new possibilities for automating key steps in the software development process. In addition to a tailored project methodology, mgm has also developed new accompanying tools for quality assurance that are aligned with the model-based approach.

- **Controllable use of AI**
  AI creates new opportunities, but it must also remain controllable. The A12 platform creates a forward-looking framework for the controlled, effective, and responsible use of generative AI and agent-based systems. AI models for different use cases can be easily integrated.

- **Security by Design – Safe from the start**
  The A12 runtime platform offers robust and continuously security-tested components. To secure applications based on this platform, mgm also relies on early threat modeling, lean application security during development, and automated security analyses, among other things.

What is A12?

A12 is a platform for developing enterprise applications in complex IT landscapes. It relies on model-driven software engineering (MDSE) and brings the low-code principle to the world of enterprise software. As an open platform, A12 simplifies the integration of best-of-breed solutions and the use of AI at all levels. A12's Modeling Environment provides tools to create and maintain parts of an application over the long term as independent business logic modules without programming experience. A12's Runtime Platform provides the flexibility needed to evolve business-critical applications with professional individual software development, AI support and system integration into fully integrated enterprise applications. An introduction to A12 is provided in the white paper A12 - Low Code for Individual Enterprise Software


mgm

A12

Low Code for Individual Enterprise Software

2024.06 LTS

Innovation Implemented.

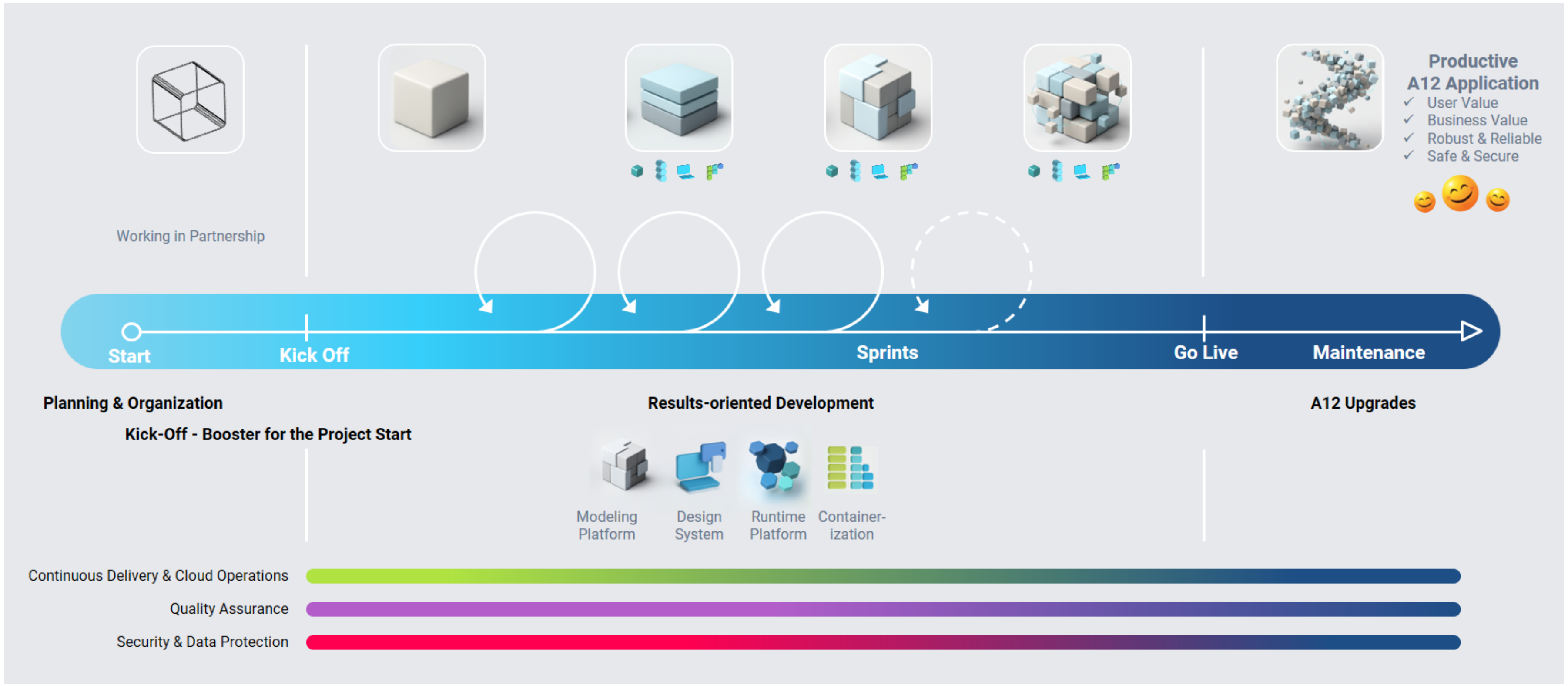


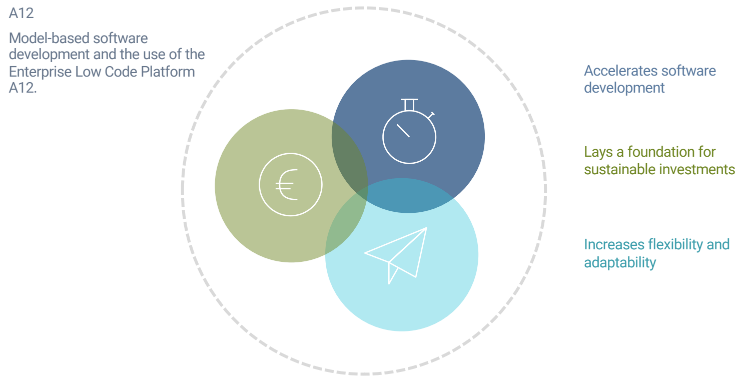
Figure 1 – *Overview of the development process with A12*

02

Industrialized Software Production

As the digital transformation progresses, software for individual business processes is becoming an increasingly important competitive factor. At the same time, the demands on the software development process are increasing - primarily due to higher requirements in terms of usability, security and the rapid provision of enterprise software. The higher degree of complexity, the use of the low-code paradigm and the capabilities of AI are changing the development process and motivating industrialized software production.

Whether a public authority, insurance company or online retailer, no major organization today can do without individual software that makes its processes more efficient and enables new services. The digital transformation has heralded a change in which the provision and use of software has a significant impact on how competitive a company is, what innovations it can implement and how well it meets customer expectations.


A12
Model-based software development and the use of the Enterprise Low Code Platform A12.
Accelerates software development
Lays a foundation for sustainable investments
Increases flexibility and adaptability

Complexity Dimensions of Enterprise Software

With the increased importance of enterprise software, the demands on software development have also risen dramatically. Applications must be developed as quickly as possible and seamlessly integrated into existing IT landscapes. At the same time, they must be performant and scalable, offer a consistent user experience and meet high security requirements. Flexible and quickly implementable adaptability is also becoming increasingly important - especially with regard to business-driven changes, but also against the backdrop of technological leaps.

Classic Cost Factors

One of the biggest challenges facing enterprise software in the long term is dealing with change. Model-based software development and the use of the A12 platform offer a solution. It

- accelerates software development through the use of ready-made components, modeling tools, and code generators, as well as the use of AI that can be validated with QA tools,
- increases the flexibility and adaptability of the developed software, as parts of the application are maintained by experts without programming, and
- lays the foundation for sustainable investments in long-lasting software, which can also be raised to new future technology levels thanks to a clear separation of technology and domain expertise.

For software development, the low-code approach represents a paradigm shift that decisively changes the way in which applications are developed and maintained. In the enterprise environment, however, the use of a low-code platform is never the only solution to all problems. There are always functional and technical as well as functional and non-functional requirements that need to be developed individually. And there is always the challenge of system integration. Because enterprise software can only develop its full potential in interaction with existing and future applications.

In project practice, low-code practices, AI support, individual development and system integration must therefore mesh seamlessly in order to bring high-quality enterprise software into production. This requires a well-thought-out organizational and technical framework.

The User Experience Needs to Be Spot On

Business applications are not traditionally known for stunning (interaction) design and a convincing user experience. Today, however, these quality features are indispensable for enterprise software. User expectations have risen dramatically. Those who are used to the convenience of modern web applications in their private lives will be reluctant to use outdated, purely functional software that does not rely on contemporary design principles in a professional environment.

In addition, different rules apply to B2B applications than in the B2C context. They are aimed at professional users who often work with the software on a daily basis and over a long period of time. Keyboard shortcuts and other productivity-enhancing features such as search and filter operations are commonplace. The areas of user interface (UI) and user experience (UX) are therefore not only essential factors for the acceptance of business applications. They also contribute to productivity by ensuring that users can complete their tasks as efficiently and purposefully as possible. To address the increased demands on interface design and user experience, A12 applications rely on the Plasma design system, which was developed specifically for the requirements of enterprise applications.

Rising Cyber Risks

Business applications are now more interconnected than ever before. They extend across more and more areas of the company and therefore offer greater productivity. At the same time, however, they are also exposed to new risks. With the increasing level of digitization of organizations, the dark business of cybercrime has also increased significantly in recent years.

The Bundesamt für Sicherheit in der Informationstechnik (German Federal Office for Information Security, BSI) describes the IT security situation in Germany as tense to critical. According to the BSI report on the state of IT security in Germany in 2024, an average of 309,000 new variants of malware were added every day between July 1, 2023, and June 30, 2024. From 2022 to 2023, the number of daily discovered vulnerabilities in software products increased by 14 percent. Cyber criminals are operating in an increasingly differentiated underground economy and have significantly expanded their extortion methods, according to the status report. A successful ransomware attack can paralyze an entire company and cause massive damage.

When developing enterprise software, security therefore has a completely different significance than it did just a few years ago. Isolated penetration tests prior to the launch of a software product are no longer sufficient. Instead, it is much more important to integrate IT security engineering expertise directly into the development process and avoid potentially insecure implementations and configurations from the outset. A12's platform approach offers major advantages for a resilient application landscape. The entire A12 code base is continuously checked using automated security analyses. Centrally maintained building blocks and templates are regularly checked and configured securely by default. Since part of the code is generated in A12 applications, the risk of implementation errors is reduced. The platform thus promotes the use of best practices and facilitates compliance with corporate guidelines and security standards.

Cloud and Containerization

The development and operation of software are becoming increasingly intertwined. One technical driver for this is the trend towards microservice architectures and container technologies. Cutting into smaller pieces and packaging into autonomous units increases the flexibility and speed at which developed IT services can be provided. At the same time, however, this also creates new challenges and demanding administrative tasks relating to the orchestration of software delivery. DevOps expertise in the team is indispensable for mastering these tasks.

A12 applications are based on a well-designed microservice architecture and can be operated both on-premise and in cloud environments. A range of templates (build & deployment pipelines, Helm charts, project templates, etc.) simplifies and standardizes the transition from development to robust, high-performance, and scalable operations. With C12, mgm also offers a cloud service that is tailored to the operation of A12 applications.

„The Era of Isolated Large Enterprise Projects Is Over“

Hamarz Mehmanesh, CEO of mgm, explains his idea of industrialized software production.

mgm has been developing business software for almost 30 years. What have been the constants during this time - despite all the technical advances?

Today, as in the past, enterprise business applications are complex, networked, long-lived and driven by the - sometimes conflicting - needs of large organizations. Traditional cost drivers such as business-related adjustments to the software throughout the entire life cycle are still highly relevant. In addition, however, the requirements for enterprise software and the complexity of development have increased massively. I find it all the more astonishing that it is still common practice in the industry to simply put a team together and start from scratch. Sure, it works, no question about it. But is it still sensible and appropriate today? I don't think so.

Why not? What stands in the way of developing enterprise software on a greenfield approach with a team of recruited specialists?

The price is simply too high, it is not efficient. There are a number of new dimensions of complexity that can only be mastered through well-established, increasingly industrialized development processes efficiently. For example, the demands on the user experience have increased. First-class usability is a must - on all devices. Then there is the issue of security: due to the increasing degree of networking and ever more professional cyber criminals, applications must be uncompromisingly secured from the outset. The triumph of cloud and container technologies has also led to a closer integration of software development and operation, which brings further challenges. In addition, there are more and more legal requirements - for example in terms of data protection and accessibility. How can a purely recruited team adequately take all of this into account and build sustainable software?

What would be the alternative? What do industrialized development processes look like?

At mgm, we combine several approaches to create a kind of agile production line. Our A12 platform forms the core. This reduces the vertical integration in individual projects and also shifts the added value from IT experts to technical experts. Specialist content is modeled with the help of special tools, not programmed. Then there is the use of AI: we have robust components and clearly structured A12 models that AI models can work with very well. Pure vibe coding does not yet work reliably in an enterprise context. However, with A12 and the specially developed tools for quality assurance, security testing, and build & deployment, we have a controllable environment in which we can use generative AI and agent-based systems very effectively and responsibly. This means we are already very close to the next stage of software development.

What are the advantages of a more industrialized development process for customers who commission individual software?

The risk is reduced. The more the software development process follows the principles of industrialization, the more likely it is that high-quality, user-friendly and secure software will be delivered in a short space of time. In my opinion, the era of large, isolated enterprise projects is over. No company can justify this any longer - especially as business is becoming increasingly fast-moving. How the current platforms and low-code approaches will fare in the long term remains to be seen. The next technical innovations will come. We are therefore pursuing the approach of modeling technical content in such a way that it is independent of specific technologies - and can really be used in the long term in terms of sustainable software.

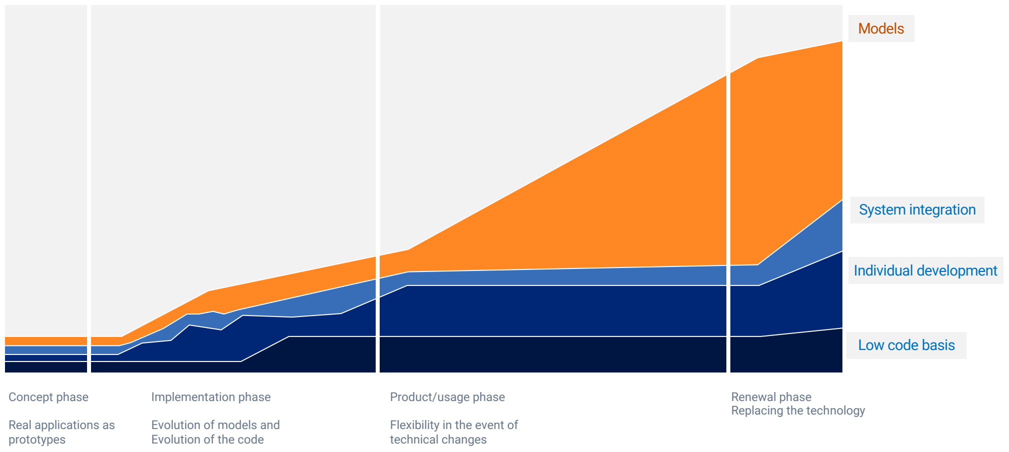
The Life Cycle of Low-Code Software

How does an individual project based on a low-code platform differ from a conventional software development project? To illustrate the key differences, let's first take a look at the most important phases in the life cycle of low-code software.

In a project based on A12, artifacts are created on a **model level** and on a **code level**. The course of the project is characterized by an iterative approach at both levels.

The code level is divided into three areas:

- **Low Code Basis**: Technical components and artifacts that are part of the A12 platform
- **Individual Development**: Individually developed components, extensions and artifacts
- **System Integration**: Code for interaction with external systems



Concept Phase: Real Applications as Prototypes

A decisive difference to conventional software development projects can be seen in the concept phase. There are no classic prototypes, which are designed as mere mockups and dummies to convey an initial feel for the application and cannot be used any further afterwards. Instead, initial data and UI models as well as workflows are created, which are used directly in a real application.

Initial results are therefore not only visible very quickly. As a real application, they provide a much more concrete basis for deriving further requirements. Thanks to the Plasma design system, the applications follow a well thought-out UI/UX concept even in the earliest phases and have a modern look and feel.

Implementation Phase: Evolution of the Models and Evolution of the Code

The implementation phase is characterized by successive extensions at the model and code level. The data models are extended until they map all the domain-specific aspects relevant to the application. The UI models and workflows are adapted accordingly in order to bring all necessary interactions with the business objects into the user interface and to offer users the most efficient flow of the respective work processes.

At the code level, requirements are implemented in two strands. Some of the requirements are implemented using the standard components of A12. The remaining requirements are implemented individually on the project side. Here again, the extent to which they can flow back into the further development of A12 as candidates is checked.

Production and Usage Phase: Flexibility in the Event of Technical Changes

In general, software is no longer subject to change until it is no longer in use. The model and code levels also continue to evolve during the production and usage phase. At the code level, however, the changes are usually very minor and mainly characterized by technical updates. More extensive new requirements may also necessitate further implementation phases. At the model level, on the other hand, changes typically occur frequently. The business side continues to develop. This is a key advantage of model-based development: business requirements can be implemented much faster and with less effort. There is no need for individual programming. The specialist department can adapt the software independently with the help of the modeling tools.

Renewal Phase: Replacing the Technology

The final phase of the software lifecycle is usually decommissioning. Technology is developing rapidly. After just five years, enterprise software is considered technically obsolete. With a development platform like A12, however, there is an alternative to decommissioning: technical renewal. After all, why build a completely new application when the continuously maintained models map the technical core of the software accurately and up to date? The isolation of the domain knowledge in the models makes it possible in principle to replace technical components. In addition, the technical basis is continuously kept up to date through A12 updates.

03

Planning and Organization

Good planning is half the battle when developing enterprise software. In addition to choosing a suitable project management methodology, it is important to set up a competent team, establish committees for project management and establish a common understanding for cooperation. mgm relies on needs-based organizational structures that are individually balanced depending on the project.

In terms of project management, an A12 project does not differ significantly from a classic enterprise software project. In the case of complex enterprise software, the low-code approach is embedded in classic development and management structures that have proven themselves in the context of agile, individual software development. Accordingly, mgm focuses on a complete project environment and needs-based organizational structures for A12 projects, which can be individually balanced depending on the project.

Project Scope

A project is a plan with defined start and target criteria for the production of a product or service with specific resources and requirements. It can start at different points in the life cycle of an application. When we refer to a **A12 project** below, we mean - unless otherwise indicated - the scenario of a **new development**. It typically ranges from the concept phase to the application going live. As enterprise software is designed for longevity, further projects typically also take place during the usage phase. They usually focus on specific extensions or integrations with other systems.

Consulting, Change Management & Communication

The success of software depends crucially on its acceptance by users. Even the best technical solution that meets all formal requirements can fail if it is not accepted or understood by users. A12 applications counteract this from the outset by actively involving the responsible departments in the development process using the low-code approach. It makes a huge difference whether you are presented with an application that has been developed entirely externally or whether you have developed parts of it yourself and may be responsible for it independently in the long term. Change management and communication in A12 projects are consistently geared toward this new development constellation.

Laying the Foundation for Change

At the start of a project, the organizational changes that will result from the new application are analyzed. Which processes will change? Which stakeholders are affected? And how can existing structures be supported or adapted? It is particularly important to determine which technical aspects and workflows will be mapped in A12 models in the future and how the responsible departments will be involved in this process. In the public sector in particular, for example, domain-specific experts who are responsible for the digital implementation of specialist laws may be located in completely different departments or agencies. Here, it is necessary to examine how they can be involved in the software development and maintenance process in the long term in order to enable end-to-end digitization.

Empowering and Involving Employees

By using A12's low-code tools, affected employees are involved in the change at an early stage. This promotes their acceptance of new ways of working and helps to actively address resistance. mgm offers target group-specific A12 training paths to empower users and continuously train and involve them throughout the development process. Employees are not only familiarized with the new tools and systems, but also become active participants in shaping the change. This creates a culture of openness and shared progress.

Include Development Status in Communication

A clear and targeted communication strategy forms the basis for project success. It creates transparency, provides early information about changes, and gives those affected the opportunity to get involved. Different stakeholder groups require individually tailored communication formats that take their perspectives and needs into account. However, the application itself speaks the clearest language. Here, A12 offers the advantage that executable development stages are available very quickly and can be made accessible to stakeholders via test platforms. They can test the application at an early stage and influence further development by providing feedback to the development team.

A12-Enablement / Trainings

To ensure that the project team has the necessary knowledge to work with the A12 development platform, mgm offers various training courses and associated certifications. This ensures that the client's team members and partners can quickly become productive with the tools and technical components of A12.

A12 Enablement consists of training paths with basic and specialist training courses for the various roles and is supplemented by optional training courses (on accessibility).

In addition, mgm's training program includes a range of security training courses. The following offers are particularly relevant in the context of A12 project:

- Best practices for secure web applications
- Secure Coding for Java
- DevSecOps: Security in the CI/CD pipeline
- Cloud security best practices

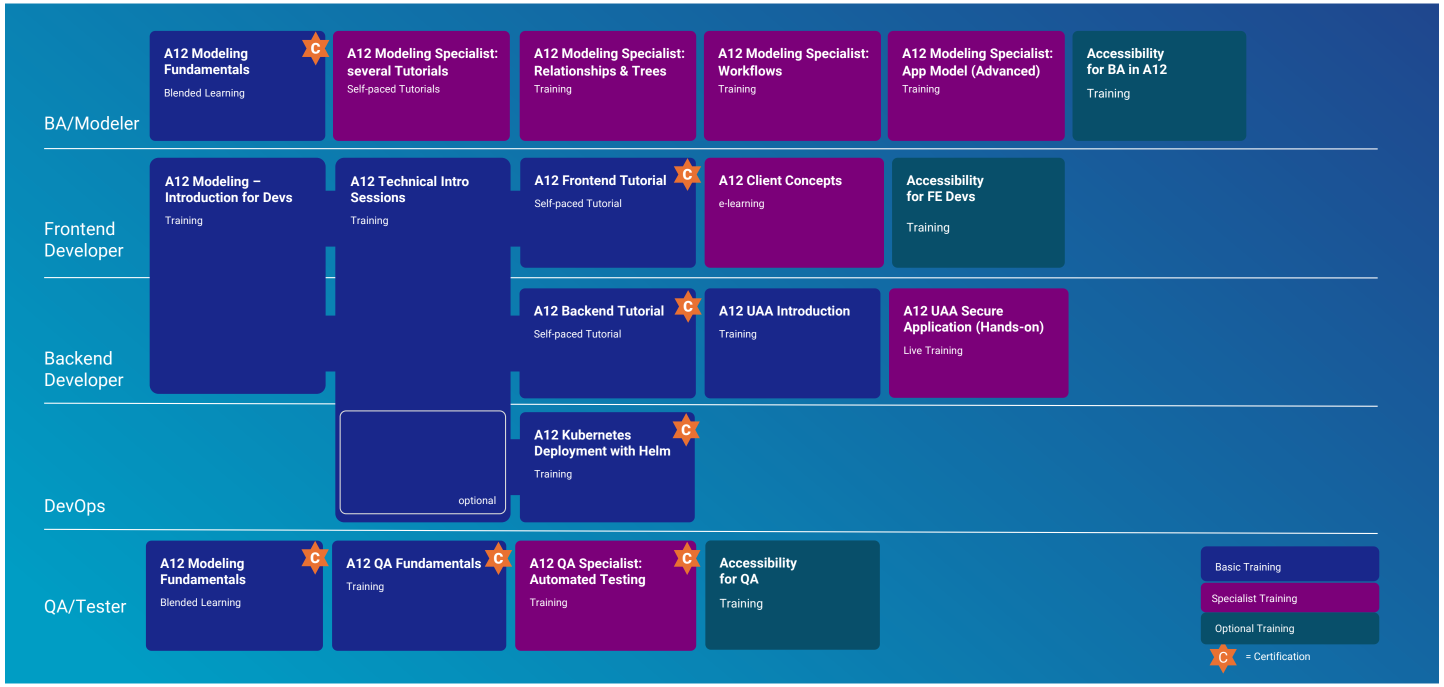


Figure 2 – *The A12 Enablement Path includes a series of training courses and certifications for various roles.*

Support of the Project Team by Cross-Sectional Teams

In addition to onboarding support from the training team and based on several large-scale projects over many years, cross-project organizational and support structures have been established within mgm. They help to incorporate the experience and best practices from successful enterprise projects into new A12 projects.

In addition to project teams, mgm relies on a number of cross-sectional teams that specialize in central disciplines of software engineering such as technical project infrastructure (TPI), security, data science & AI, quality assurance and UI/UX. They support the project teams selectively as required, are available to answer more specific technical questions and are an important guarantee for the success of the project.

In addition to the cross-sectional teams, the Professional Services (PS) team has also been established within mgm (see p. 24). It acts as an interface between the project teams and the A12 team, which further develops and maintains the platform.

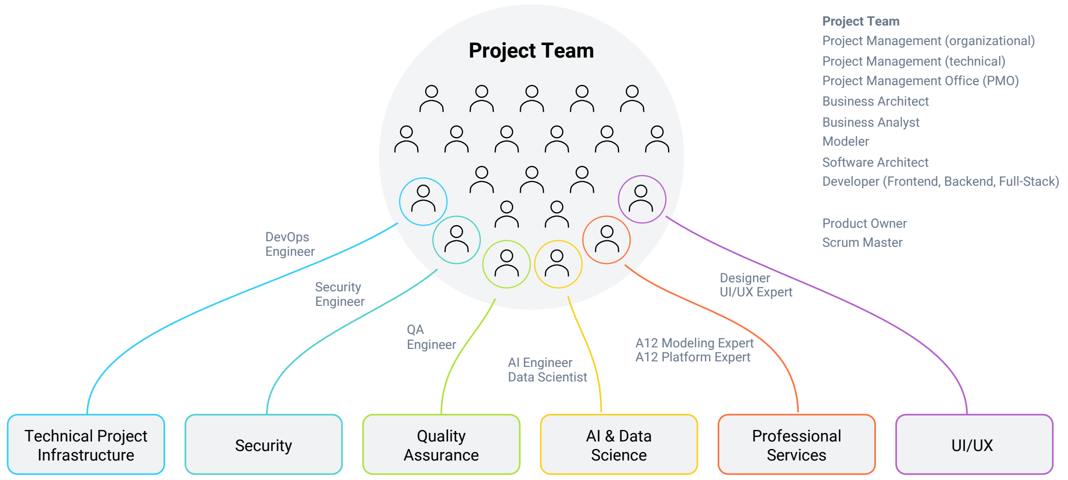


Figure 3 – *Specialists from cross-functional teams contribute their expertise as members of the project team as needed.*

Working in Partnership

The main goal of an A12 project is the effective implementation of the requirements in an efficiently usable, high-quality solution. Close cooperation with the client is an important prerequisite for achieving this goal. mgm sees itself as a flexible development partner that can be involved in various constellations as required. The spectrum ranges from projects in which the client and mgm each implement an A12 project together with their own development teams to projects in which mgm assumes complete responsibility for development.

If mgm is responsible for the A12 project, the minimum scope of the client's involvement is based on the degree of maturity of the detailed requirements at the start of the project. The more details still need to be worked out together and the stronger the focus can or should be on optimizing the business processes to be covered, the more extensive the involvement should be.

By using A12, the client can optionally participate in the modeling. The following constellations are possible and are suitable depending on the project phase and scope of modeling:

- mgm takes over the complete modeling. This is particularly recommended at the start of the project, as experienced modelers can speed up the initial development of the application.
- The client and mgm share the modeling tasks.
- The client is independently responsible for the domain-specific modeling. This constellation is particularly useful if the application is confronted with many and regular changes - for example, due to frequent legal changes.

In the case of integration, mgm trains the client's experts/business analysts in the creation of A12 models. This division of tasks makes a lot of sense because the models always reflect the contents of the business domain and the client's experts know this business domain best. mgm offers support for modeling questions and is responsible for the entire development and technical tasks. This approach has proven itself in the area of tax. Here, specialist teams independently define fields, plausibility rules and forms based on the legal requirements.

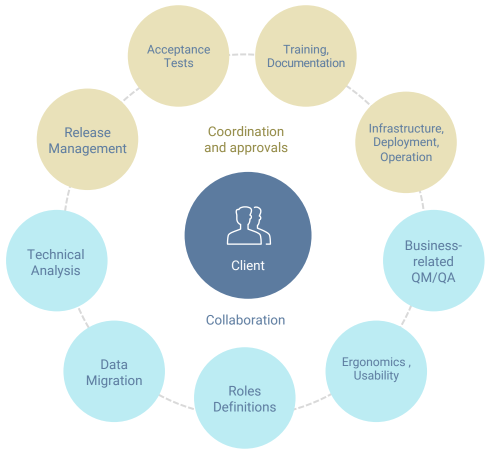


Figure 4 – *Examples of involvement and handover points for cooperation*

Time and Milestone Planning

The use of low-code enables drastic speed advantages to be achieved in software development. The first executable version of an application and the minimum viable product (MVP) of a business application are available very quickly. The amount of time saved ultimately depends on how intensively the project makes use of standard components. The more standard solutions from A12 are used, the tighter the time and milestone planning can be.
In a fully agile A12 project, new functions and criteria are implemented directly across teams to increase quality. At the beginning, the client and mgm develop a rough roadmap with prioritization, which is continuously and proactively adapted to changing requirements. Based on many years of experience, mgm considers the Plan - Do - Check - Act principle (PDCA) in combination with a structured and continuously prioritized backlog to be the most suitable iterative process. The early creation of a sprint burndown makes it possible to estimate how many sprints will be necessary to complete the respective tasks by assigning story points. This makes it possible to compensate for the reduced predictability of an agile project and create a realistic schedule and milestone plan.

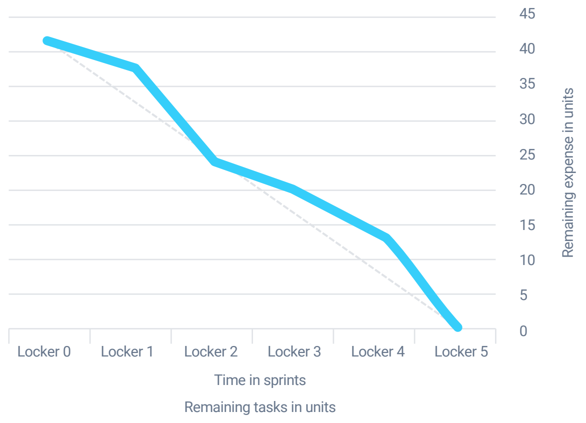


Figure 5 – *Example of an ideal sprint burndown*

04

Kick-off - Booster for the Project Start

Important decisions are made at the beginning of a project that influence the entire course of the project. mgm has developed a structured kick-off process that steers projects quickly and purposefully in the right direction. By involving teams of experts, we take key topics such as technical project infrastructure, data protection, AI, security and change & communication into account right from the start.

Setting up the Project Infrastructure With Expert Teams

To help new projects based on A12 get off to a smooth start, mgm has developed a structured kick-off process. The project management receives valuable feedback from experienced experts right from the start in order to set the right course for the project. All of mgm's specialized cross-divisional teams and the A12-PS team are involved in the process. Among other things, it should provide clarity,

- what infrastructure the project requires,
- how the software is provided to the customer, and
- how work processes are organized in the project (JIRA workflow, agile development, etc.).

With the help of a questionnaire, all key areas are systematically examined - from the ISMS risk assessment and the QA methodology to data protection issues and security-relevant aspects.
Based on this, mgm sets up the infrastructure and associated processes - from the ticket system and the provision of a wiki for documentation to the configuration of a self-developed solution for time recording and controlling of project expenses.
In addition, the project will be included in the A12 community within mgm. This gives it the opportunity, for example, to help shape the further development of the platform by participating in surveys.

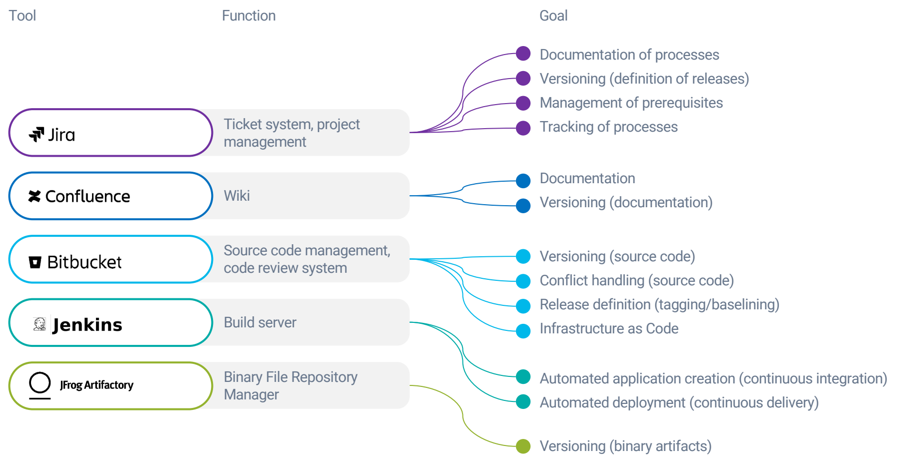
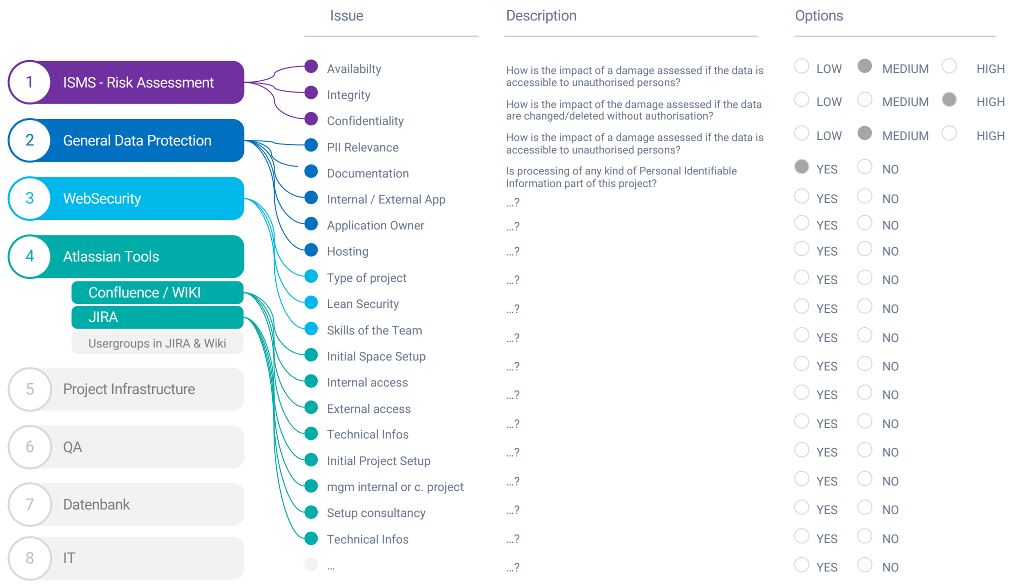

Tool
Function
Goal
Jira
Ticket system, project management
Documentation of processes
Versioning (definition of releases)
Management of prerequisites
Tracking of processes
Confluence
Wiki
Documentation
Versioning (documentation)
Bitbucket
Source code management, code review system
Versioning (source code)
Conflict handling (source code)
Release definition (tagging/baselining)
Infrastructure as Code
Jenkins
Build server
Automated application creation (continuous integration)
Automated deployment (continuous delivery)
JFrog Artifactory
Binary File Repository Manager
Versioning (binary artifacts)

Figure 6 – *Tools used as standard and their core functions*

4. Kick-off - Booster for the Project Start



Issue	Description	Options
Availabilty	How is the impact of a damage assessed if the data is accessible to unauthorised persons?	LOW [✓ MEDIUM] HIGH
Integrity	How is the impact of the damage assessed if the data are changed/deleted without authorisation?	LOW MEDIUM [✓ HIGH]
Confidentiality	How is the impact of a damage assessed if the data is accessible to unauthorised persons?	LOW [✓ MEDIUM] HIGH
PII Relevance	Is processing of any kind of Personal Identifiable Information part of this project?	[✓ YES] NO
Documentation		
Internal / External App	...?	YES NO
Application Owner	...?	YES NO
Hosting	...?	YES NO
Type of project	...?	YES NO
Lean Security	...?	YES NO
Skills of the Team	...?	YES NO
Initial Space Setup	...?	YES NO
Internal access	...?	YES NO
External access	...?	YES NO
Technical Infos	...?	YES NO
Initial Project Setup	...?	YES NO
mgm internal or c. project	...?	YES NO
Setup consultancy	...?	YES NO
Technical Infos	...?	YES NO
...	...?	YES NO

Figure 7 – *mgm offers a range of templates to systematically capture key challenges in complex A12 projects*

Technical Initialization of an A12 Project

In A12 projects, the use of the A12 platform creates a number of general constraints. The low-code approach implies that as many parts of the application as possible are modeled in addition to the required parts of individual development. In order to dovetail these development strands correctly from the outset, it is recommended that the architectural and functional concept of the application be roughly planned, even in agile projects. It initially determines how developed and modeled artefacts will flow into one another over the course of the project.

A Three-Person Team Sets the Right Course

If everyone involved in a large-scale project starts at the same time from the outset, there is a very high probability that they will initially run in different directions. To prevent this from happening, we initially deliberately chose a very small core team. It consists typically of three project-experienced people who have sound experience in the following areas:

- Architecture
- Domain-specific modeling / Business analysis / A12 modeling
- Frontend Development / Backend Development

Ideally, these three people have already familiarized themselves with the technical context of the application to be created before the initialization phase. Building on this knowledge, they analyze the requirements in detail and develop a coordinated concept and a common technical basis.

The initialization team is supported by mgm's specialized cross-sectional teams, ensuring that important aspects relating to security, data science & AI, quality assurance, UI/UX design, accessibility and technical infrastructure are taken into account right from the start.

The initialization team involves representatives from the cross-sectional teams in important decisions on a selective basis. At the end of the initialization phase, the concept and the technical basis are also reviewed and approved by the cross-sectional teams.

Coordinated Business and Technology Concept Answers Key Questions

From a conceptual point of view, the following results arise during the initialization phase:

- **Architecture concept**
  A sufficiently detailed architecture concept defines which A12 components is used and which new components are required, how they are cut and how they interact with each other.
- **Business-related model**
  The business-related model describes how the application maps business content - both with the means of A12 modeling and at the code level. Among other things, the model answers the question of which business content can be created with the A12 tools in the future and how the A12 models fit into the higher-level business modeling of the entire application.

These concepts form an important basis for development and ideally create a foundation that lasts throughout the course of the project. However, it is also a fact that not all questions can be answered conclusively in the initialization phase. It is just as important to consciously omit decisions that do not need to be made at the start of the project as it is to make decisions that point the way forward.

“ *The initialization phase of an A12 project is extremely exciting and sets an important course for the entire course of the project. It's about getting the application up and running properly with a small team - and in such a way that all sub-teams and disciplines work together in a clearly interlinked manner from the outset. This is very challenging, but a lot of fun. The integration of A12's low-code capabilities into complex individual software offers huge potential, but also requires completely new ways of thinking. Each project also brings its own technical and professional challenges*

Volker Schmitt
Senior Architect at mgm

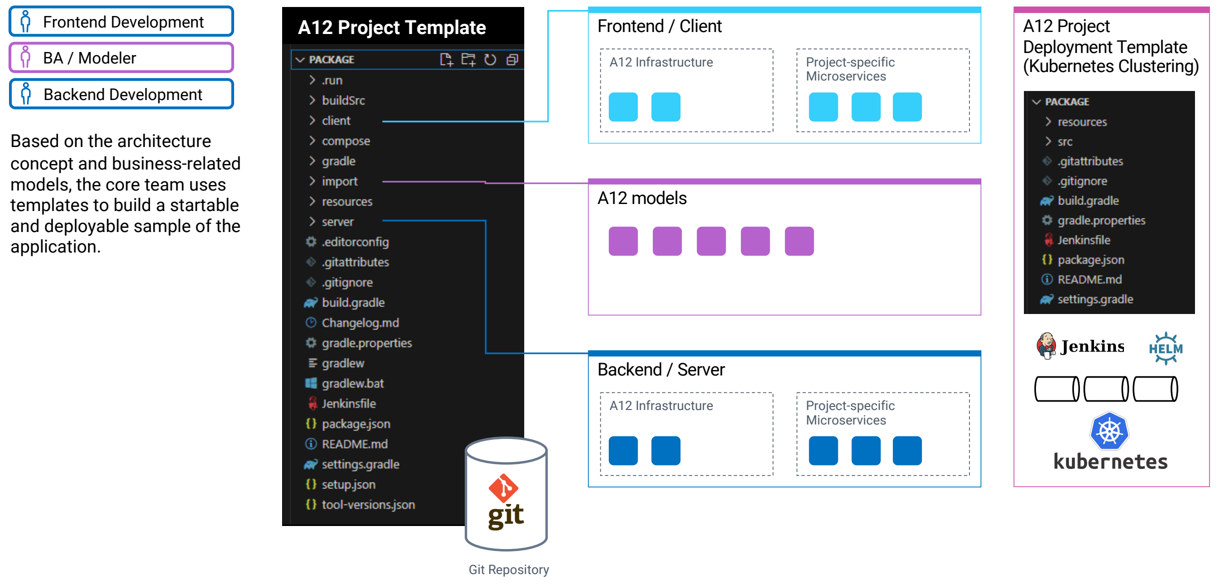
.

Deployable Sample of the Application as a Starting Point

In addition to creating the agreed concepts, the initialization team has the task of laying the foundation for a launchable and deployable application and setting up project-specific development and test environments. A key objective is to enable and promote interdisciplinary collaboration between business analysis and modeling, front-end and back-end development and technical project infrastructure.

Technically, this means creating a Git repository based on the A12 templates. Java, Node.js, Gradle and Docker are required to build and launch the project. An IDE and A12's own modeling tools are also required - above all the Simple Model Editor (SME). A12 already offers basics such as the A12 Project Template, the Helm A12 Stack and the A12 Build and Deployment Pipelines. In addition to source code for the client and server, the template also contains fully modeled sample applications and other required components such as Keycloak, the workflow engine and a PostgreSQL database.

The technical basis that needs to be created reflects the designed architecture and the functional model. It ensures that individual sub-areas can be technically integrated right from the start. This also includes the direct integration of A12 models for which the development team is not responsible. All team members work on a shared repository from day 1. This prevents silos from developing and certain parts of the application only having to be merged during the course of the project.



Based on the architecture concept and business-related models, the core team uses templates to build a startable and deployable sample of the application.

Figure 8 – *Templates and a standardized structure speed up the start of A12 projects*

Organizational Initialization of the Project

Parallel to the technical initialization, an organizational initialization of the project takes place. The project management and PMO primarily work together here. They use the project infrastructure that has already been set up to provide essential guidelines for the team. This includes the following aspects in particular:

- **Goal description**
  A shared vision is an essential prerequisite for goal-oriented development. It serves as an overarching orientation aid in the form of a product vision in an agile project. A well-formulated goal description is valid in the long term and helps the team to focus on the essentials.
- **Checklists for onboarding and offboarding**
  In no long-term enterprise software project is the team fixed from start to finish. There are always moments when new team members join or existing ones leave the team. In such cases, checklists help to introduce new team members more quickly and to clearly structure departures and handovers.
- **Organigram**
  How is the team made up? Who takes on which roles and tasks? What rights and obligations form the basis of the joint work for all team members? All these questions are answered when creating an organizational chart that is continuously kept up to date.
- **Communication matrix**
  A communication matrix makes it transparent which regular meetings are planned and who participates in them. It also explicitly shows which reporting channels and communication chains have been agreed within the team and in collaboration with the client and partners.
- **Documentation structure**
  The design of an initial structure for the documentation makes it easier for the team to record interim statuses and project results. It also makes it transparent in which areas the documentation may still have gaps.
- **Roadmap**
  The creation of a roadmap helps to schedule all foreseeable necessary work and milestones. It helps with planning and setting priorities.
- **Team calendar**
  A team calendar is set up to establish a shared view of project deadlines. Absences of team members or other stakeholders are also entered here, for example.

Get the Team on Board on the Basis of the Developed Foundation

Overall, the initialization phase aims to make fundamental decisions at the start of the project and define work packages that resolve as many dependencies as possible. All sub-teams involved are then brought on board - ideally in a joint team kick-off on site. Right from the start, they work on the basic framework of the application created during the initialization phase and benefit from the established organizational framework.

Interface to the A12 team: Professional Services

Professional Services (PS) forms the interface within mgm between the customer projects and the A12 team, which is responsible for the further development of the platform. The PS team takes special care of the needs of customer projects and is available to them as a direct contact:

Business Professional Services (BPS) specializes in business requirements. The team supports business analysts and technical experts of customer projects in modeling technical content with A12. It negotiates priorities for requirements. It also offers training and tutorials, promotes the development of a knowledge base and creates standards and templates for technical concepts, documentation and quality assurance.

Technical Professional Services (TPS) specializes in technical requirements. The team acts as a point of contact for the development team. It manages bug reports and coordinates A12 patch releases for customer projects. In addition, TPS supports quality assurance from a technical perspective and provides technical documentation and tutorials for developers.

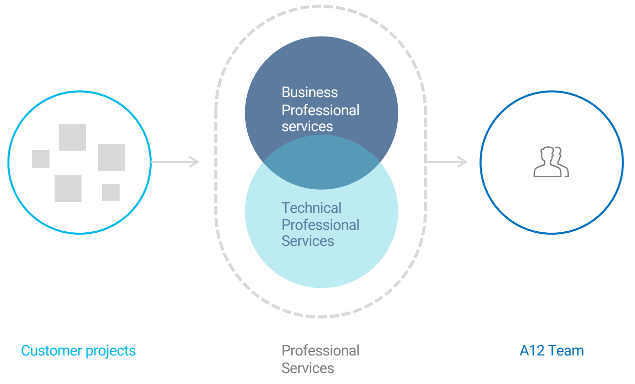

Business Professional services
Technical Professional Services
Customer projects
Professional Services
A12 Team

Figure 9 – *mgm's Professional Services teams are available to the A12 projects as contact persons for questions relating to the platform and are in close contact with the A12 platform team*

Templates for A12 Projects

Project Template

The A12 project template forms the technical starting point for A12 projects. It provides a standardized setup for applications that use all essential components of A12 - Client, Data Services and UAA. It draws on best practices from several successful A12 projects and offers a well thought-out, minimal, easily expandable configuration out-of-the-box.

Templates for Cluster Environments

Cluster environments in cloud infrastructures are becoming the de facto standard for operating business applications. A12 therefore offers a standardized and field-tested setup for the development and operation of A12 Applications in Kubernetes clusters. In combination with the project template, it enables cluster-based development and production environments to be set up very quickly. Predefined, versatile build & deployment pipelines (see chapter Continuous Delivery & Operations) automate numerous processes and ensure that the development team can concentrate on the actual value creation and does not lose any time on configuration at the start of the project.

App Factory

The A12 App Factory enables the fast and automated provision of A12 environments - i.e. the technical infrastructure for the execution of A12 Applications. It is designed for development, testing and production purposes, is based on the A12 Project Template and supports deployment to various target environments.

GetA12 - All A12 Resources in One Place

The GetA12 platform (https://geta12.com/) is the central entry point for projects that develop software based on A12. Access is restricted to registered users who are active in A12 projects.

GetA12 provides the following resources, among others:

- **Modeling environment** In GetA12, business analysts can find the latest version of the modeling tools. You can download the respective installer for Windows, MacOS and Ubuntu.
- **Documentation** From the explanation of basic concepts and the functionality of modeling techniques to the APIs of the A12 products - the extensive documentation contains a wealth of important information and is the central reference work for the A12 platform.
- **Release overview** In addition to a clear presentation of the release lines and the versions of individual A12 products used in them, GetA12 contains changelogs and the announcement of new releases.
- **A12 Discourse** The A12 Discourse exchange platform acts as both a forum and a knowledge base for the A12 community. The platform brings together developers and analysts from the A12 team and from the projects that use A12.
- **A12 Artifactory** All current technical artifacts for the use of the A12 platform are always available to the development team in the A12 Artifactory. This includes repositories for Docker, Helm, Maven and npm.
- **Training catalog** For all essential roles within an A12 project - from front and back-end development to modeling, DevOps and QA - the training catalog lists which training courses and tutorials are available.

05

Results-Oriented Development

mgm defines its success by the number of systems brought into production that contribute to the long-term business success of its customers. In the implementation phase of software projects based on A12, we therefore consistently focus on the value for the user and the value for the business.

Implementing Requirements: Modeling or Programming?

In an A12 project, requirements can basically be implemented in three ways:

- modeling with the help of the A12 editors
- implementation with standard platform components
- implementation in the form of an individual solution

Modeling domain-specific contents

The requirements that can be implemented by means of modeling can usually be identified quickly and easily. The decisive factor is the modeling scope of the platform. A12 can be used to model almost all aspects relating to domain-specific contents - from data models and the modeling of forms and workflows to the structure of the application.

The decisions as to which requirements the team implements with the help of A12 standard components or individually are usually more complex. They require in-depth knowledge of the technical components of A12.

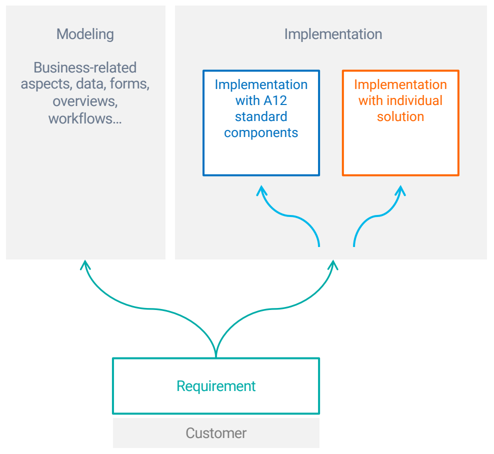

Modeling
Business-related aspects, data, forms, overviews, workflows...
Implementation
Implementation with A12 standard components
Implementation with individual solution
Requirement
Customer

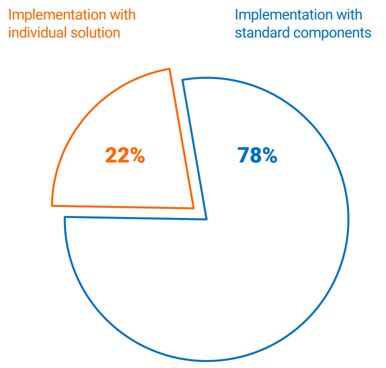
Figure 10 – *Categorization of requirements*

Implementation With Standard Components vs. Individual Implementation

Especially at the beginning of an A12 project, unless the project team already has extensive experience with the A12 platform, the following questions arise time and again: Can a specific requirement be implemented using the standard A12 components? Or does the team need to implement a custom solution to realize the requirement?

As a model-based development platform, A12 is explicitly designed for extensions and adaptations through individual software development. The APIs of the technical components provide numerous extension points that can be used to integrate your own code or implement your own solutions. In principle, however, the more a project relies on standard components from A12, the lower the implementation costs. To ensure that the development team exploits the full potential of the A12 platform and does not implement any unnecessary individual solutions, the TPS team provides support with implementation issues. It has comprehensive full-stack expertise of the A12 platform and works with the project team to determine which requirements can be covered with current and upcoming A12 features and how, and which individual solutions are necessary on the project side.

The proportion of requirements that are implemented with A12 standard components or individually varies from project to project. The project team should always compare the cost of individual solutions with their importance for the use of the software. Is a customized solution really necessary for a certain feature? Or does a standard component fulfill the essential purpose? Here, it is always important to question your own ideas and to accept compromise solutions on a case-by-case basis. This is the only way to take advantage of the productivity benefits of the low-code principle. Ideally, the proportion of individually implemented solutions for requirements should fluctuate between 5 and 25 percent.



Ideally, the proportion of individually implemented solutions for requirements should fluctuate between 5 and 25 percent.

UI/UX & Accessibility

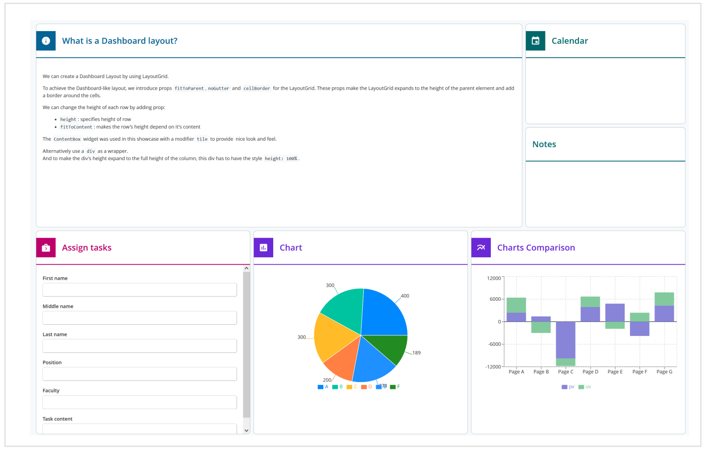
Users also expect first-class usability from enterprise software. The demand for a coherent and accessible user experience meets a high level of content complexity and the typically high information density of business applications. mgm has therefore developed its own design system: Plasma. Together with a guideline for accessibility, it ensures appealing and at the same time effectively usable software.

Plasma – UI/UX-Standard

The generated web interfaces of an A12 Application are HTML5-compliant, responsive and cross-browser compatible for all common modern web browsers.

On the client side, A12 enables the rapid creation of modularized front-ends. The client framework is based on the established React/Redux ecosystem, integrates A12 UI components such as Engines and Widgets and calls the A12 services in the backend via HTTP/REST endpoints.

We use Widgets to design the web interfaces of the application. Widgets are reusable UI components that follow Plasma design conventions and UX concepts. They support business applications that run on desktops, tablets and smartphones with keyboard, mouse and touch input. The components offer an easy-to-use, well-documented, strongly typed API and are individually extensible and customizable.



We can create a Dashboard Layout by using LayoutGrid.

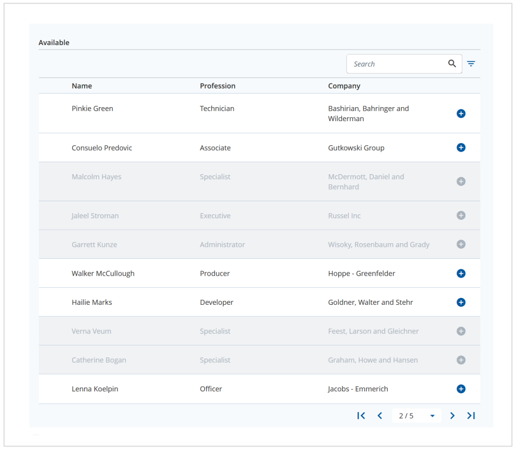
To achieve the Dashboard-like layout, we introduce props `fitToParent`, `noGutter` and `cellBorder` for the LayoutGrid. These props make the LayoutGrid expands to the height of the parent element and add a border around the cells.

We can change the height of each row by adding prop:

- `height` : specifies height of row
- `fitToContent` : makes the row's height depend on it's content

The `ContentBox` widget was used in this showcase with a modifier `tile` to provide nice look and feel.

Alternatively use a `div` as a wrapper.
And to make the div's height expand to the full height of the column, this div has to have the style `height: 100%`.



Available

Search

Name	Profession	Company
Pinkie Green	Technician	Bashirian, Bahringer and Wilderman
Consuelo Predovic	Associate	Gutkowski Group
Malcolm Hayes	Specialist	McDermott, Daniel and Bernhard
Jaleel Stroman	Executive	Russel Inc
Garrett Kunze	Administrator	Wisoky, Rosenbaum and Grady
Walker McCullough	Producer	Hoppe - Greenfelder
Hailie Marks	Developer	Goldner, Walter and Stehr
Verna Veum	Specialist	Feest, Larson and Gleichner
Catherine Bogan	Specialist	Graham, Howe and Hansen
Lenna Koelpin	Officer	Jacobs - Emmerich

2 / 5

Figure 11 – *The Plasma design system offers numerous solution patterns for the design of enterprise applications*

Accessibility - Guidelines for Projects

The importance of accessibility - also abbreviated as A11Y, a short form of accessibility - is continuously increasing in web applications. This is demonstrated not least by the stricter legal situation: for several years now, public bodies within the EU have been obliged to make websites and mobile applications accessible in accordance with Directive 2016/2102. According to the European Accessibility Act, all websites and web applications must be accessible from 2025, with a few exceptions.

We generally recommend taking accessibility into account in projects in order to ensure non-discriminatory access for users with disabilities. When developing accessible web applications with A12, mgm follows the criteria of the BITV test and the specifications of the WCAG.

Many accessibility-related functions are provided by A12 on the platform side and work out-of-the-box. The components of the A12 user interface, such as widgets and engines, are continuously checked for accessibility requirements and further developed accordingly. Nevertheless, a number of aspects must also be taken into account in project practice to ensure the accessibility of a A12 Application. There are always specific requirements that cannot be covered by the platform. Implementation on the project side is essential. This affects almost all areas - from design, modeling and development to testing.

To support A12 projects in the implementation, a detailed guide for accessible A12 Applications is available in the GetA12 documentation platform. It offers practical assistance and includes background information on accessibility and its certification, as well as checklists on design specifications and requirements for modeling and development.

Central A11Y Standards

The most important international standard for the accessibility of web applications is the **WebContent Accessibility Guidelines (WCAG)**. They are published by the World Wide Web Consortium (W3C). In Germany, the **Barrierefreie-Informationstechnik-Verordnung(BITV)** forms a central legal basis. It is essentially based on the WCAG specifications and implements EU Directive 2016/2102, according to which websites and apps of official bodies must be accessible. For official accessibility certification, applications must pass the BITV test - an extensive test procedure comprising almost 100 test steps.

06

A12 Upgrades

With the switch to new A12 versions, A12 applications benefit from new capabilities of the underlying platform. The recommended annual upgrade keeps the technical basis of the application up to date and increases its lifespan. To keep migration efforts to a minimum, the platform team provides a range of tools.

Platform Approach: Evolving Technical Foundation

Every individual software project requires a technical foundation. In conventional projects that are created from scratch, the team selects the tech stack independently. It decides on technologies, frameworks, and third-party libraries and uses them to build a technical platform that is unique in a sense. In classic individual projects, this technical foundation usually remains static. The effort required to update it can only be justified in certain cases, and obsolescence is inevitable. Even if the technical basis is state-of-the-art at the start of the project, it will gradually become outdated.

A12 applications, on the other hand, are built on a technical foundation that does not have to be created and maintained by the project team. The A12 team at mgm, which has more than 100 members, and the A12 community take on this task and continuously develop the technical basis. Continuous platform updates increase the security of the technical basis and introduce new functionalities and innovations at regular intervals. While in a conventional individual project, the technology is selected once and typically not touched again after go-live, the platform approach enables continuous technical evolution of the software.

The benefits at a glance

- A12 Updates keep the technical basis up to date. The tech stack is based on industry standards and is continuously being further developed.
- The A12 platform team manages third-party libraries on the platform and closes security gaps with A12 security updates.
- The project team is relieved of some of its usual workload and can concentrate on specific project challenges.
- The lifespan of the application is increased because the technical basis continues to develop and does not gradually become obsolete, as is the case with conventional individual projects.

A12 Release Lines

The A12 team releases a new **Release Line** with Long Term Support (LTS) once a year. It is supported for two years. Each new release line may contain **Breaking Changes**. These require adaptations on the project side, for example due to changes to the APIs.

For each release line, the A12 team continuously publishes **Product Updates** in the form of **Minor Releases** and **Patch Releases**. Minor releases bring improved features, while patch releases fix bugs and close newly identified security vulnerabilities in 3rd party libraries. Product updates do not contain breaking changes. They can be installed without any manual effort.

As an integral part of the A12 release policy, regular **Security Patches** are published every two months. If critical vulnerabilities are identified, mgm also provides corresponding patches as quickly as possible. In addition, the continuous review of all security messages of the components used in A12 is part of our build & deployment process.

Recommendations

- A12 projects should upgrade to the latest release line once a year.
- Minor and patch releases should be installed continuously and promptly to minimize security risks and benefit from more stable components.

Migration to New A12 Versions

When switching to a new release line, projects benefit from new features of the A12 platform. However, this usually involves migration efforts, which should be taken into account early on in the planning stage. All relevant steps are documented in the release notes. To minimize the effort involved, the A12 platform team provides a range of support tools:

- **Model Migration**
  Model migration is usually necessary whenever switching to a new release line. It unlocks the new model-driven features of the latest A12 version for the A12 application. Scripts automate the process. The Simple Model Editor automatically detects when a workspace contains outdated model versions and can update them automatically.
- **Data Migration**
  Data migration is only necessary if changes to serialization, persistence structure, or index structure occur in a new release line. This happens rather rarely. If this occurs, the respective release line provides a script to automatically transfer data to the new structure.
- **Code Migration**
  In the event of breaking changes (e.g., changes to APIs) in A12 components that are used in an A12 application, code adjustments are required on the application side. To keep these to a minimum, A12 relies on auto-refactoring tools such as OpenRewrite and provides codemods for new release lines. These help to automatically make changes such as renaming classes, packages, and methods.

The exact amount of remaining migration effort depends on many factors, but the following trade-off generally applies: **The further the project deviates from the A12 standards, the higher the migration effort for platform updates will be.** To ensure continuous modernization of the application and reduce follow-up costs, deviations from the provided standards should only be made for good reason in A12 applications.

Help Shape the Future of A12

A12 is continuously being further developed. Non-functional requirements in particular, such as those arising from new legal requirements, are very likely to be implemented at platform level in a timely manner. In addition, projects have numerous opportunities to contribute to the further development of the A12 platform:

- **Report Bugs**
  A Jira ticket template is available for reporting bugs. Its uniform structure and instructions for completion facilitate the systematic description of errors that will be fixed as part of regular updates.
- **Set Requirementes**
  The ability to submit requirements directly to the A12 base is a unique feature of the A12 platform that is not usually available on other low-code platforms. This gives projects the opportunity to influence the further development of the platform very directly. A detailed consolidation process has been established internally at mgm for incoming request tickets, in which the possible implementation of the request within the A12 platform is reviewed.
- **Share Model Repositories**
  By making parts of their A12 models openly available, projects are making an important contribution to the growth of the A12 ecosystem. This opens up great potential, particularly in the public sector, for avoiding duplication of effort and promoting cross-application interoperability.
- **Contribute Code**
  A12 projects may result in functionalities that can be fed back into the platform, for example as extensions based on specific extension points of the A12 components or, in the long term, as future core components of the A12 platform.

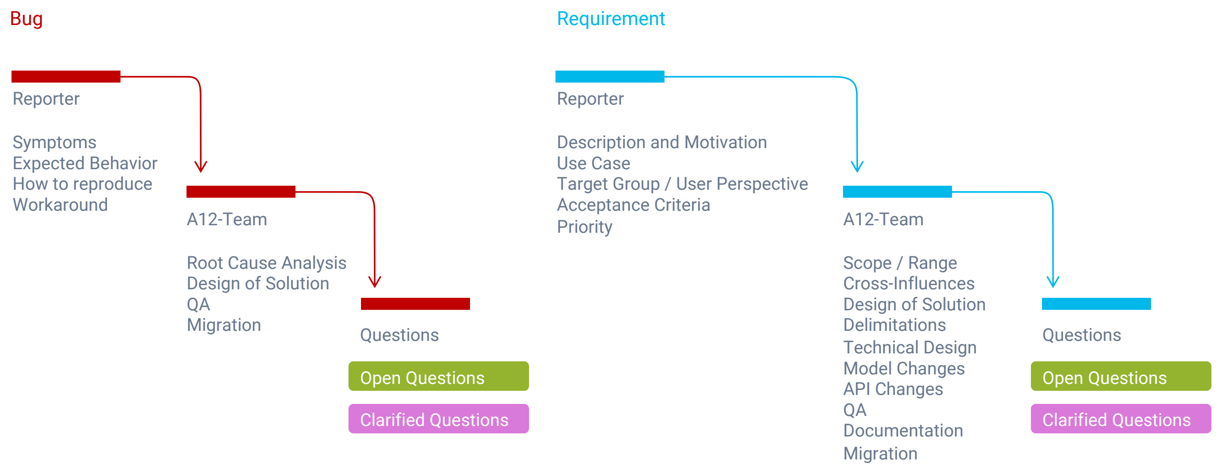


Figure 12 – *Standardized structure of bug and requirements tickets*

07

Continuous Delivery & Cloud Operations

Enterprise software is alive and constantly subject to change. Changes must be made available quickly both during development and after going live. mgm relies on standardized methods and environments along the entire development, build and deployment processes. They are specially designed for the operation of containerized applications in modern cloud infrastructures.

How can source code be reliably turned into executable software? This question is at the heart of build management, which has seen an enormous increase in complexity in recent years. Monoliths are being replaced by scalable and flexibly deployable microservices. The main drivers for this were the triumph of agile development methods and the business-driven demand to bring software innovations into production more quickly. The increasing spread of DevOps practices is also ensuring that development and operation are merging ever more seamlessly.

mgm meets this development with a series of standards relating to the required environments and the build and deployment processes.

Operation of A12 Applications on Kubernetes Clusters

Even though a variety of environments, systems and services are involved in today's enterprise software development, an essential division into two areas remains: There is a landscape in which software is developed and an environment in which the software is executed. For both levels, mgm relies on a series of standards for A12 projects that enable fast setup and automate essential build & deployment processes.

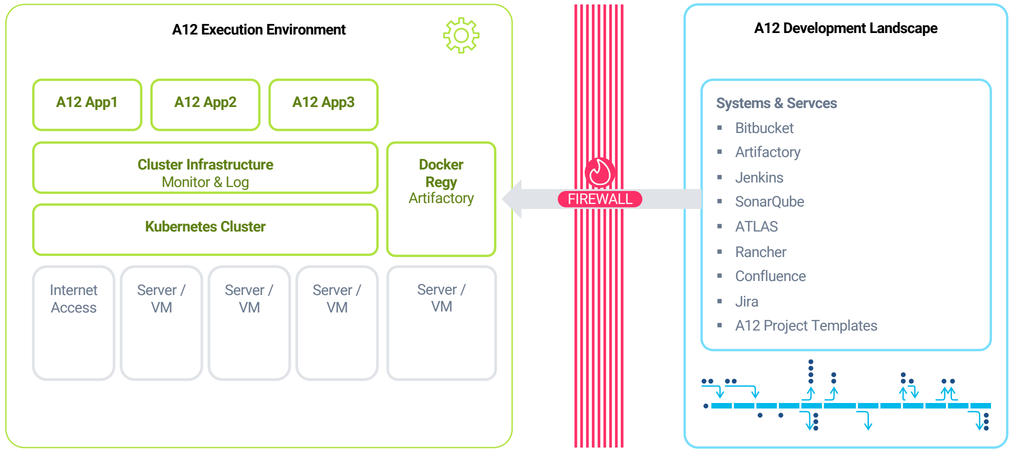


Figure 13 – *Schematic overview of the operating and development environments*

The most important infrastructural pillars for the development environment (Bitbucket, Jira, etc.) are described on p. **??**. For the environment for the execution of A12 Applications, mgm relies on Kubernetes clusters as standard. Based on the experience from several large software projects, we have made a selection of tools from the Kubernetes ecosystem that we recommend as the standard stack. In principle, however, A12 Applications can be operated with different technology stacks - depending on the specifications of the respective hoster.

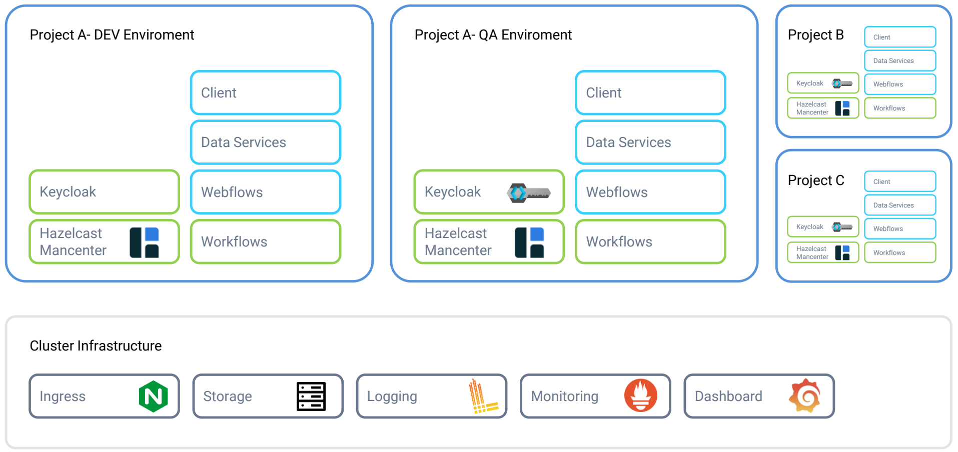


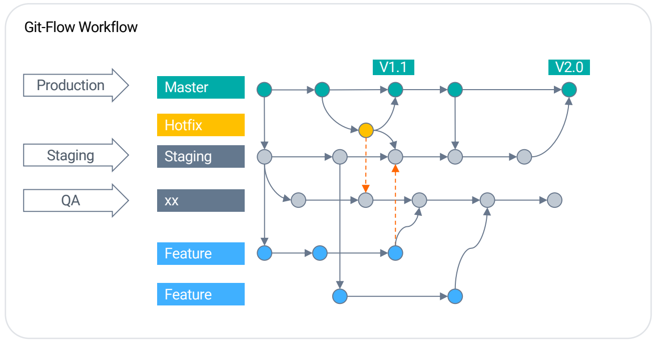
Figure 14 – *Standardized project environments speed up the setup in the cluster*

Code Management and Versioning

A system for versioning source code and models is an essential prerequisite for implementing the CI/CD pipeline. mgm relies on **Git-Repositories**. As a graphical interface for code reviews, **Bitbucket** is the tool of choice. For the versioning scheme of the software developed as part of the project, we recommend **Semantic Versioning** (https://semver.org).

The specific branching model to be used is determined in coordination with the release procedure and the project plan. In general, the Git-flow workflow is very suitable for enterprise software projects, as it supports the feature branches and release branches typically required.

The structure of the repositories follows a defined standard. In addition to a repository for the program code, there is always an adjacent repository for deployment-related content. This contains configurations that describe how the software is automatically built and deployed.



Code Management

Figure 16 – *Example of a Git workflow*

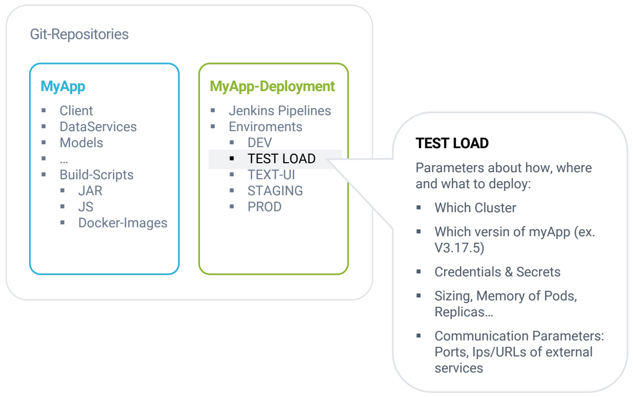


Figure 15 – *Standardized repository structure*

CI/CD Pipelines

In order to transform source code into robust and quality-assured software, mgm uses build and deployment pipelines that have been developed in-house and proven in several projects. The following figure outlines the basic structure. From left to right, the pipeline outlines the path from code to production-ready software. The pipe segments indicate specific operations with the development statuses, which are largely automated. The leading system is the Jenkins Build Server - a widely used open source automation tool. It starts and orchestrates the build and deployment jobs.

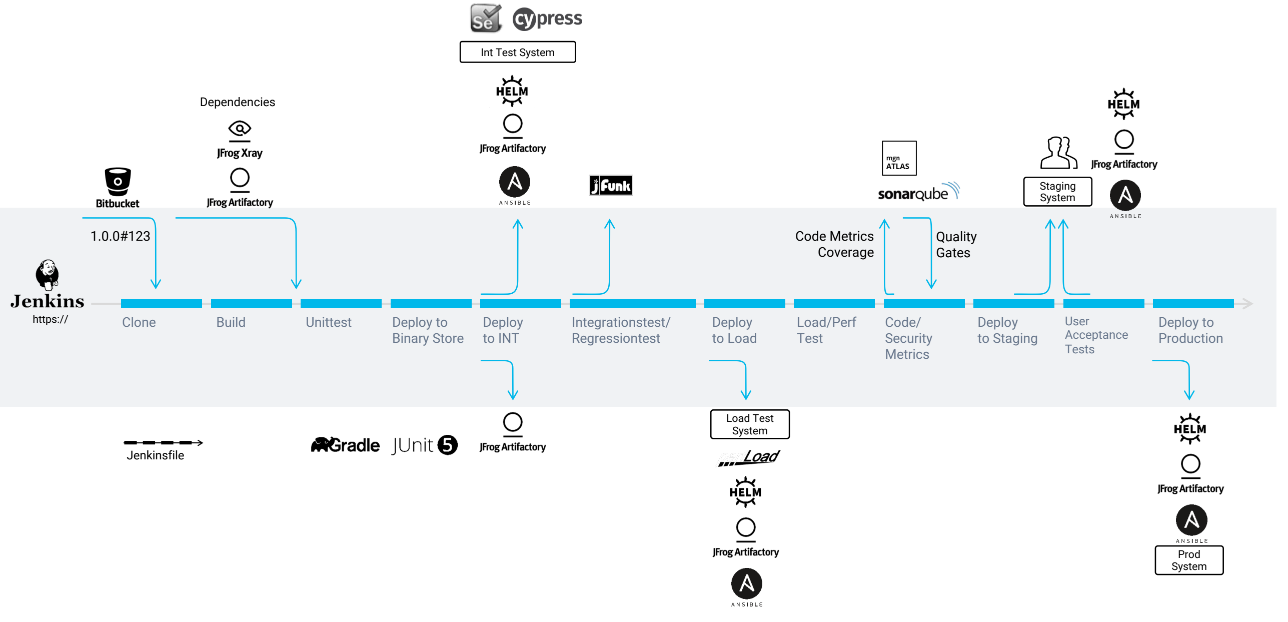


Figure 17 – *Jenkins orchestrates the individual steps of the CI/CD pipeline*

In the first two tube segments, A12 models and source code are pulled from the source code management (Git) and built. After building, the generated artifacts are stored in a container registry. From here, the images can be installed on test systems in further steps and tested with appropriate test tools (Selenium, JFunk, Perfload, Cypress). If required, load tests (e.g. with Perfload) can be carried out in a further step. After a successful build, the artifacts are subjected to a static code analysis in order to automatically detect potential errors and bugs. Finally, in the last tube segment, the software is deployed to a production system.

During the development process, there are several occasions for building the software. The build and deployment process is clearly defined for each of these occasions.

	Executer	Tests	Result / Artifact	Published	Deployed
REVIEW Builds (Snapshot Builds)	Developer Jenkins on PR Review	Unit Tests	Name-version-SNAPSHOT	NOT Public To ~/.m2/repository To executor/.m2/repository	On dev notebook Manually on private DEV environment
NIGHTLY Builds	Jenkins daily (or other fixed timeframe)	Unit Tests QF / Integration Tests	Name-version-BUILDNR	Public To Artifactory	Automatically on INT environment
FEATURE Builds	Jenkins non demand	Unit Tests QF / Integration Tests	Name-version-SNAPSHOT	NOT Public To executor/.m2/repository	On dev notebook Manually on DEV environment
RELEASE Builds	Jenkins non demand	Unit Tests QF / Integration Tests	Name-version	Public To Artifactory	Manually on PROD

Figure 18 – *Overview of the different types of builds*

Dependency Management

One efficiency driver in the development of modern business applications is the reuse of program code. There is no need to constantly reinvent the wheel: In practice, development teams therefore repeatedly use existing libraries and program packages. They can come from in-house development or from external sources - typically from open source initiatives. If your own code calls such libraries, this is referred to as dependencies.

Careful management of these dependencies as well as the version status and licensing conditions of the libraries used is important in order to minimize risks and avoid ending up in “dependency hell”. After all, if a vulnerability becomes known in a third-party library, your own software is also potentially at risk. And if license conditions are not adhered to, there is a risk of serious copyright infringements.

As part of the build and deployment pipeline, mgm therefore also uses **Software Composition Analysis (SCA)** - an automated process for analyzing the open source components used. This prevents software versions with serious security problems from going through the entire build process in the first place. We use our product **ATLAS** (https://www.mgm-sp.com/mgm-atlas) with **JFrog Xray**.

Legislators have also recognized the importance of making software dependencies on third-party components more transparent. As part of the Cyber Resilience Act, manufacturers across Europe will be obliged to document which software components are included in their software products in the future with **Software Bills of Material (SBOM)**. For the A12 platform, this requirement is already met today with the help of ATLAS. This option is also available to customer projects for A12 Applications.

From Repositories to Environments

As the development statuses flow through the pipeline, they are uploaded to specific environments at four stations. If we compare the pipeline with a production line, these stations are important points for processing the workpiece. The INT, TEST-LOAD and STAGING environments are closely related to quality assurance measures, which are presented in more detail in the following chapter.

ENVIRONMENT	PURPOSE
INT TEST	Integration tests, regression tests
LOAD TEST	Further quality assurance measures such as load tests
STAGING	Final quality assurance; user acceptance tests; technical acceptance tests of release candidates
PROD	Productive operation in Kubernetes Cluster

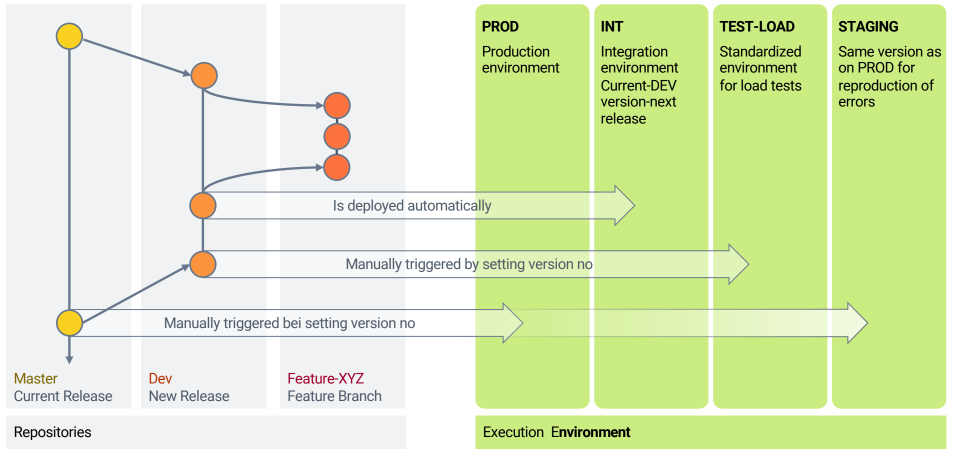

PROD
Production environment
INT
Integration environment Current-DEV version-next release
TEST-LOAD
Standardized environment for load tests
STAGING
Same version as on PROD for reproduction of errors
Is deployed automatically
Manually triggered by setting version no
Manually triggered bei setting version no
Master
Current Release
Dev
New Release
Feature-XYZ
Feature Branch
Repositories
Execution Environment

Figure 19 – *Overview of the individual deployments*

08

Quality Assurance

A12 projects are based on quality-assured components of the A12 platform on the one hand. On the other hand, it is also important to set up QA measures for solutions implemented on the project side. mgm uses a range of tools from its own research and development - from a test data generator and a test management tool to advanced automation tools. In combination with a proven methodology, the result is uncompromisingly high-quality software

.

Methodology and Approach

Placing Quality Assurance in the Software Life Cycle

Quality assurance accompanies the entire software life cycle — from the initial idea to productive operation. It is an integral part of the development process, which sets different priorities in each phase, thus ensuring sustainable quality gains.

- **Requirements and concept phase**
  Quality assurance begins as early as the requirements gathering stage. Precise documentation, reviews involving the customer, and early validation of models ensure that misinterpretations are avoided and a robust basis for development is created.
- **Design and development phase**
  During implementation, quality assurance is closely intertwined with development. Code reviews, static analyses, and continuous integration ensure quality at the technical level. Technical implementations are tested promptly and specifically through verifications and regression tests. For applications with high load requirements, initial tests are performed on performance-relevant components. Quality is not checked downstream, but is developed from the outset.
- **Testing and integration phase**
  In this phase, system and integration tests – both manual and automated – ensure that the functions work together. In addition, non-functional tests are carried out to cover all relevant quality aspects.
- **Handover and acceptance**
  Once development is complete, the software and all documentation are handed over to the customer. Acceptance tests are carried out on this basis. Only after formal acceptance is the solution released and can be transferred to productive operation.
- **Operation and maintenance**
  Quality assurance does not end after the go-live. Feedback from support and operations enables tests and measures for future versions to be further developed in a targeted manner.

Very Early Testing

mgm generally relies on the “Very Early Testing” methodology developed in-house and successfully used in many projects. It aims to find problems in the project as early as possible in the process, when it is still easy and inexpensive to eliminate them. Tests are carried out more frequently than with the classic methodology, where testing is usually only carried out at the end of development. It is therefore necessary to strive for a high degree of test automation and to work closely with development.

This paradigm of “very early testing” is already being used in the development of the A12 platform and is also strongly recommended on the project side.

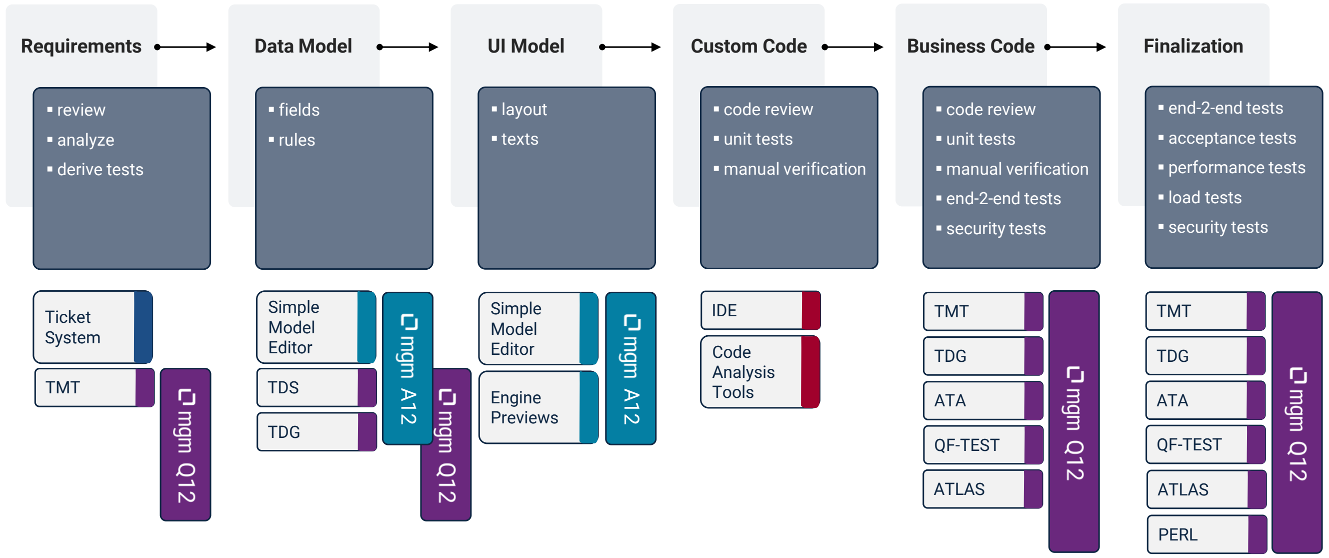


Figure 20 – *Overview of which tools are used in which phases of the model-based development process*

Software Quality Assurance in the Context of A12 Projects

Quality assurance in projects based on the mgm A12 platform differs in key aspects from classic software development projects. The model-based low-code architecture shifts the focus of testing, changes cost structures, and opens up new possibilities for automation. Quality assurance must specifically address these special features in order to make optimal use of the platform's added value.

Focus of QA

- **Quality-assured A12 components**
  Both the modeling components and the runtime components of the A12 platform are already quality-assured by mgm – functionally and non-functionally, manually and automatically. For projects, this means that repeated checks of basic platform functions are no longer necessary, and testing efforts can focus on technical modeling and individual development.

- **Shift in testing focus**
  Model editors and preview engines allow requirements to be checked and validated on the data and UI side early on in the development phase. Test cases are thus more closely aligned with model artifacts and their functional correctness. This enables immediate feedback on feasibility and consistency, avoiding long loops until implementation is complete.

- **Quality assurance for custom code**
  The proportion of individually developed functions is greatly reduced in A12 projects. Established QA measures apply to these areas. This ensures that the necessary care is taken for customized extensions without compromising the efficiency gains of the model-based approach.

- **Automation potential & A12-specialized tooling**
  Thanks to its architecture and A12-integrated tools, A12 is particularly well suited for the automation of QA activities. Test data generation or UI test automation can be set up immediately on the basis of standardized A12 components. This results in a significant reduction in QA preparation efforts.

QA in A12 projects benefits from a quality-assured platform basis, early technical validation through models, and a high degree of automation. These advantages must be exploited consistently. The remaining individual code is secured with established QA procedures.

Responsibilities and Roles in QA

In the software QA process, various roles work closely together to map the quality of software across all phases of the development cycle. Key roles in the software QA process are:

- **Test experts**
  Test experts are responsible for the design, specification, execution, and documentation of tests — both functional and non-functional — throughout the QA process. They have detailed knowledge of the customer's or project's subject matter. Performing non-functional tests (e.g., load, performance, security, or accessibility tests) requires specialized knowledge of the respective test procedures and tools, which is usually covered by specialists.

  – **Manual/Automated**
    Depending on the project requirements, test experts work in the discipline of manual testing or test automation. Both areas are part of the role profile, but are not necessarily covered by the same person.

  – **In mgm projects, manual test experts and test experts with automation skills often work closely together;** depending on the task at hand, test design, including manual test execution on the one hand, and test automation on the other, are covered either by division of labor or by a single person. Careful test analysis, test design, and initial manual test execution form the basis for subsequent test automation. While stable test cases are identified and technically reviewed in the first step, test automation engineers contribute their expertise to automate these tests efficiently and maintainably – especially for recurring regression tests. This results in an integrated QA process in which both roles build on each other and jointly contribute to sustainable software quality.

- **Test manager**
  In larger or particularly complex projects, the use of a test manager can be useful and necessary. The test manager is responsible for planning, controlling, and monitoring all QA activities in the project. This primarily includes developing the QA strategy, resource management, selecting suitable methods and tools, and reporting to stakeholders.

- **Additional QA roles**
  Depending on the size and complexity of the project, additional roles such as TestOps engineers or external testers and auditors may be involved. They take on specific tasks that go beyond the core profile of test experts and test managers.

Software Quality Assurance as a Separate Service

In addition to development-accompanying quality assurance in development projects, mgm offers testing-as-a-service and QA expert consulting. This allows customers to flexibly access experienced test experts, test managers, or automation specialists — whether for individual phases such as acceptance tests or as continuous support in their own development projects.

- Testing-as-a-Service
- QA expert consulting

In addition, mgm's own UI/UX competence team is available for questions and support in areas such as accessibility testing:

- Accessibility/A11Y

Q12 Quality Assurance Landscape

What is Q12?

Efficient quality assurance of enterprise software is hardly conceivable without specialized tools. They structure processes, reduce manual effort, and create the basis for reproducible results. At mgm, the Q12 quality assurance landscape forms the core of our QA tools. The company's own toolchain was developed based on the requirements of real projects and covers all central areas of modern QA – from test planning and automation to security, performance, and accessibility.

How is Q12 structured?

Q12 comprises several specialized tools that can be used individually or combined flexibly. Some of them (TDS, TDG, and ATA) are specifically designed for A12-based applications, while the others can be used in all enterprise applications (A12 and non-A12).

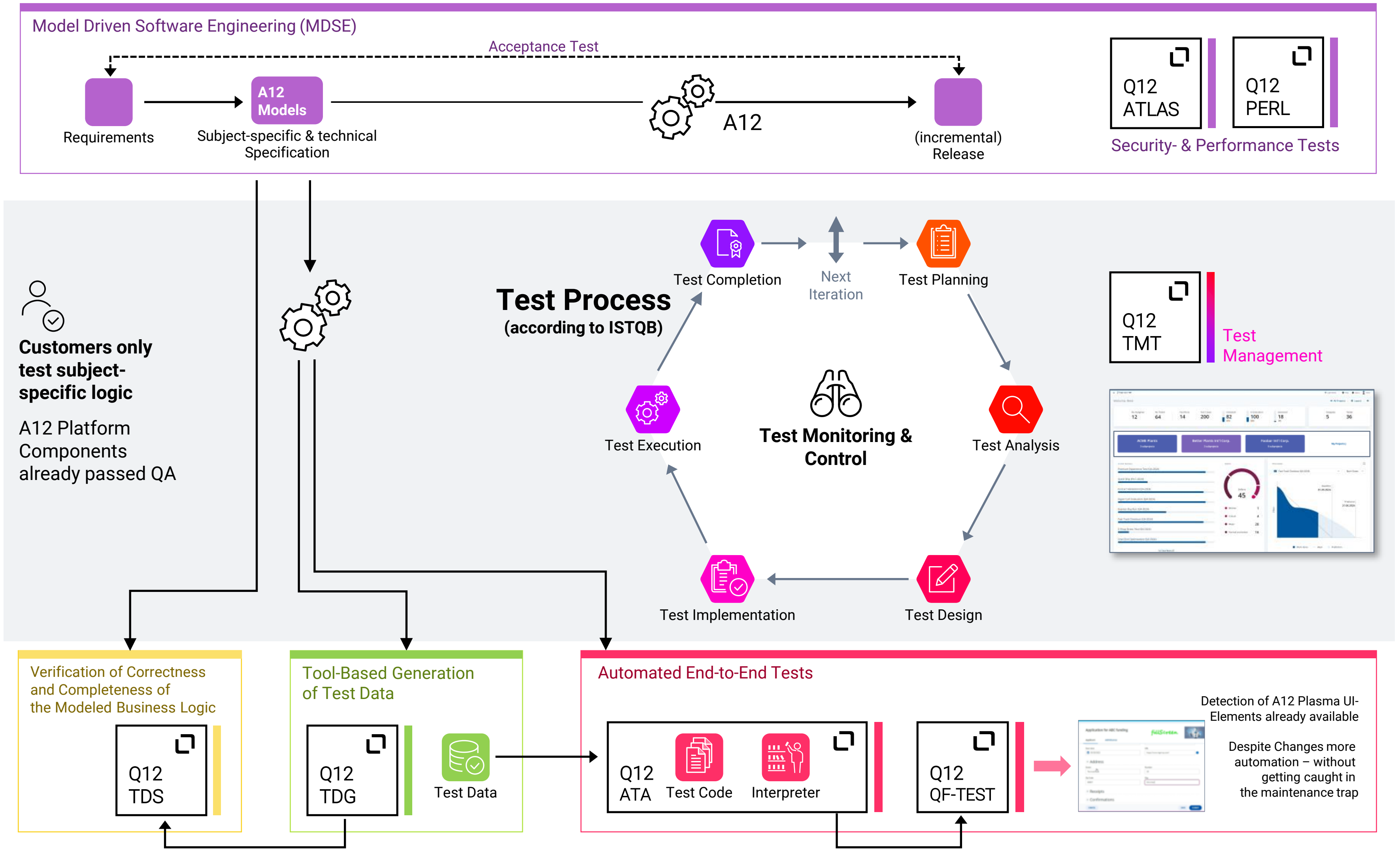
- **Q12-TDS (Test Data Suite)**
  Functional testing of data models with a focus on rules, field definitions, mappings, and calculations in model-based A12 applications
- **Q12-TDG (Test Data Generator)**
  Automatic generation of valid test data for complex forms in model-based A12 applications
- **Q12-ATA (Automated Test Automation)**
  Automated generation of code for use in UI test automation in model-based A12 applications
- **Q12-TMT (Test Management Tool)**
  Central platform for documenting test cases and reporting on test execution
- **Q12-QF-TEST (UI Test Automation)**
  Creation, management, and execution of tests for automated testing of graphical user interfaces
- **Q12-ATLAS (Automated Toolset for Lean Application Security)**
  Orchestration of analysis tools for automated application security testing
- **Q12-PERL (Performance & Load Tests)**
  Analysis, execution, and monitoring of performance and load tests

What is special about Q12?

- **Model-driven automation**
  Model-based low-code development is changing the QA process and requires specialized tools. Q12 offers tailor-made solutions that are specifically designed for A12 applications. They embed quality assurance early on in the development process in an automated and highly efficient manner.
- **Customizable tooling**
  Q12 is expandable. The tools can be integrated with third-party tools. If necessary, mgm can adapt the tools to the organizational or technical specifics of the customer's organization or add specific features.
- **Control framework for AI use**
  AI and low code combined bring enormous productivity benefits. Q12 establishes an effective control and governance framework for AI-based A12 applications and minimizes the risks of AI-related errors and hallucinations.
- **Tried and tested in enterprise projects**
  All tools in Q12 have been developed based on the needs and requirements of large enterprise software projects. They address all the key challenges in quality assurance for mature business applications.

What use cases is Q12 designed for?

- **A12 applications**
  Thanks to specialized tools, Q12 unlocks the principles of model-based software development for quality assurance. In conjunction with universally applicable tools, this creates a holistic QA solution in the A12 environment. The clever use of models shifts QA activities to early project phases, reduces manual work, and creates a level of automation that is difficult to achieve in classic projects.
- **Enterprise software without A12 reference**
  For enterprise applications outside the A12 environment, the universal Q12 tools provide a powerful foundation for modern quality assurance across all development phases. They support the quality assurance of complex business and web applications through integrated test management, automated test execution, and intelligent analysis functions in the areas of performance, load, and security.


Model Driven Software Engineering (MDSE)
Acceptance Test
Requirements
A12 Models
Subject-specific & technical Specification
A12
(incremental) Release
Q12 ATLAS
Q12 PERL
Security- & Performance Tests
Customers only test subject-specific logic
A12 Platform Components already passed QA
Test Process (according to ISTQB)
Test Completion
Next Iteration
Test Planning
Test Execution
Test Monitoring & Control
Test Analysis
Test Implementation
Test Design
Q12 TMT
Test Management
Verification of Correctness and Completeness of the Modeled Business Logic
Q12 TDS
Tool-Based Generation of Test Data
Q12 TDG
Test Data
Automated End-to-End Tests
Q12 ATA
Test Code
Interpreter
Q12 QF-TEST
Detection of A12 Plasma UI-Elements already available
Despite Changes more automation – without getting caught in the maintenance trap

Test Levels & Types

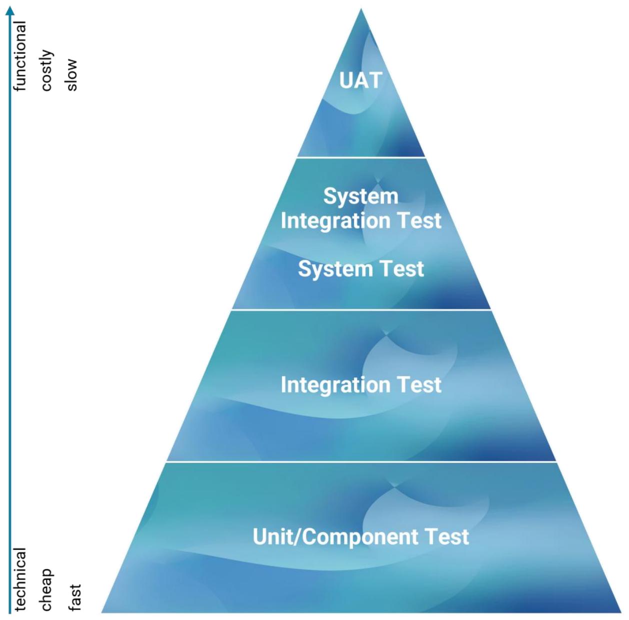
Test Levels

Test levels organize test activities along the development and integration steps of a system. They determine the level at which testing is performed, the primary objectives, and who is typically responsible. The test pyramid is used here to visualize the test levels. The model originally comes from test automation, but can also be applied to the structure by levels – regardless of whether the tests are performed manually or automatically. It illustrates that the focus is on the lower levels, while the higher levels contain fewer but more comprehensive tests. A deviation from this principle, as occurs in the so-called ice cream cone anti-pattern — too many GUI tests and too few unit and integration tests — is considered an indication of an unbalanced testing strategy.

Key testing levels are:

- **Component or unit tests**
  Testing of individual program units or modules in isolation. The aim is to detect implementation errors at an early stage. These tests are usually automated and are mainly carried out by developers.
- **Integration tests**
  Testing of the interfaces and interactions between individual components. The focus is on data transfers, communication channels, and the interaction of technical components.
- **System tests**
  Checking the entire system as a fully integrated solution. Tests are carried out to ensure that all functional and non-functional requirements are met. This also includes performance, security, and usability tests.
- **System integration tests**
  Validating the interaction of the system with other or external systems. The focus is on ensuring correct interface functionality and stability across system boundaries.
- **Acceptance tests (user acceptance tests (UAT))** End-to-end tests from the perspective of users or business stakeholders. The aim is to verify that the system meets the functional requirements and business needs. Acceptance tests form the basis for the final release decision.

The clear differentiation according to test levels ensures that quality is checked both at a technical level and from a functional perspective: From the broad base of quick, technical tests to the narrow tip of complex, business-oriented tests. The higher the level, the more functional, costly, and slower the tests become – at the same time, their importance for functional validation increases.

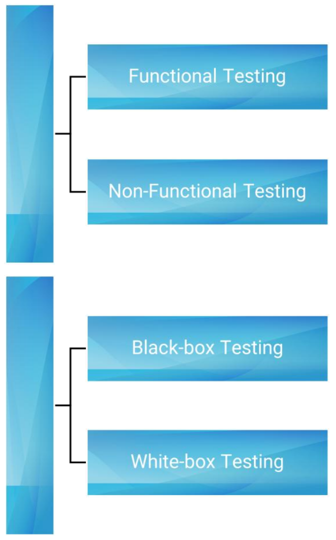


Test Types

Test types describe different perspectives from which a system can be tested. They can be applied independently of the test levels and supplement them with specific questions. While the test levels determine when testing takes place in the development process, test types answer the question of how and with what focus testing is carried out. However, there is no uniform definition of test types in practice; terms are often used differently depending on the industry or company. For initial guidance, we therefore rely on the ISTQB standard. Here, a distinction is made between two classification approaches:

- **Functional vs. non-functional**
  This distinction refers to the content focus of the test. Functional tests answer the question of whether the system does what has been specified. Non-functional tests, on the other hand, evaluate characteristics such as performance, security, user-friendliness, or accessibility. The delimitation of non-functional characteristics is based on the quality characteristics of ISO/IEC 25010.
- **Black box vs. white box**
  This perspective considers the derivation of the tests. Black box tests are based on specifications or user stories and test exclusively via interfaces and observable behavior. White box tests, on the other hand, access the internal structure of code or architecture and derive their tests from this.

These two classification approaches create a versatile grid that combines different perspectives on quality. It allows tests to be classified both according to their content focus (functional or non-functional) and their derivation method (black-box or white-box). This ensures that test activities can be planned and carried out systematically – regardless of level, process model, or technical implementation.



09

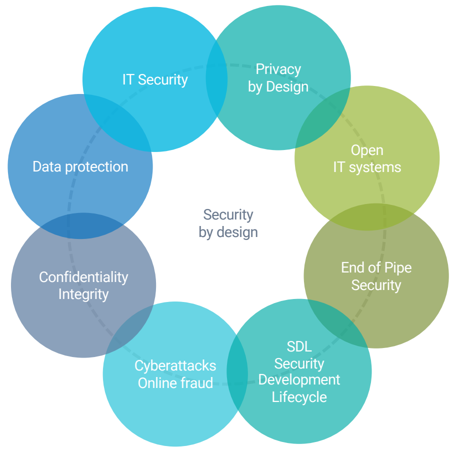
Security & Data Protection

Due to a higher degree of networking, enterprise software is much more exposed today than it was a few years ago. The risks of cyber attacks are increasing. In order to counter them in a targeted manner, mgm experts from a company unit specializing in security are involved in the project right from the start. With the security toolset ATLAS, automated security analyses also ensure that vulnerabilities and configuration problems can be quickly identified and eliminated.

Security by Design and Lean Application Security

A12 follows the principle of security by design. Security requirements are taken into account right from the start in order to prevent potential vulnerabilities. Security experts accompany all phases of development - from early requirements and architectural decisions through to acceptance tests.

As part of A12 projects, we rely on our self-developed approach **Lean Application Security**. This is based on the security awareness of all stakeholders and the expertise of the development team. All software developers and project managers at mgm receive appropriate training, which is regularly refreshed. Security experts accompany the projects. They monitor compliance with the self-imposed quality standards and create an iterative threat analysis ("threat modeling"), which provides information about the threats to the application at hand at all times. The transparent presentation of these threats in turn sharpens the team's awareness of the necessary steps to avoid corresponding vulnerabilities. Lean security means that an application is submitted to the final penetration test that is already secure from the ground up instead of trying to make the application secure by means of the penetration test alone, as is often the case.



mgm ATLAS

To ensure the highest possible level of security, mgm relies on automated security analyses - both for the A12 platform and for A12 projects. The specially developed security toolset ATLAS is used for this. As a flexible platform for security scanners, ATLAS integrates a range of tools such as OWASP ZAP and Nikto. It enables automated security tests and provides consolidated reporting - also through the integration of reports in Sonarqube. Among other things, ATLAS checks for known vulnerabilities in third-party components, detects configuration problems such as missing HTTP security headers and tests how robust APIs are against attacks such as injection attacks.

Overall, ATLAS combines analyses from the following areas:

- Software Composition Analysis (SCA)
- Static Application Security Testing (SAST)
- Dynamic Application Security Testing (DAST)
- Interactive Application Security Testing (IAST)

ATLAS is integrated directly into the build pipeline. Which scans are to be executed in detail can be easily configured in a YAML file. A vulnerability management dashboard clearly displays the results of the scans.

The use of ATLAS promotes a shift-left development approach for security-relevant aspects in A12 projects and minimizes the risks of vulnerabilities that are only detected at a late stage. As the security toolset can be integrated into the build process right from the start, it provides valuable feedback on potential security vulnerabilities throughout the entire development process.

The Vulnerability Management Dashboard also provides A12 projects with an assessment and evaluation of findings that are known to the A12 team. This favors a uniform approach to closing security gaps.

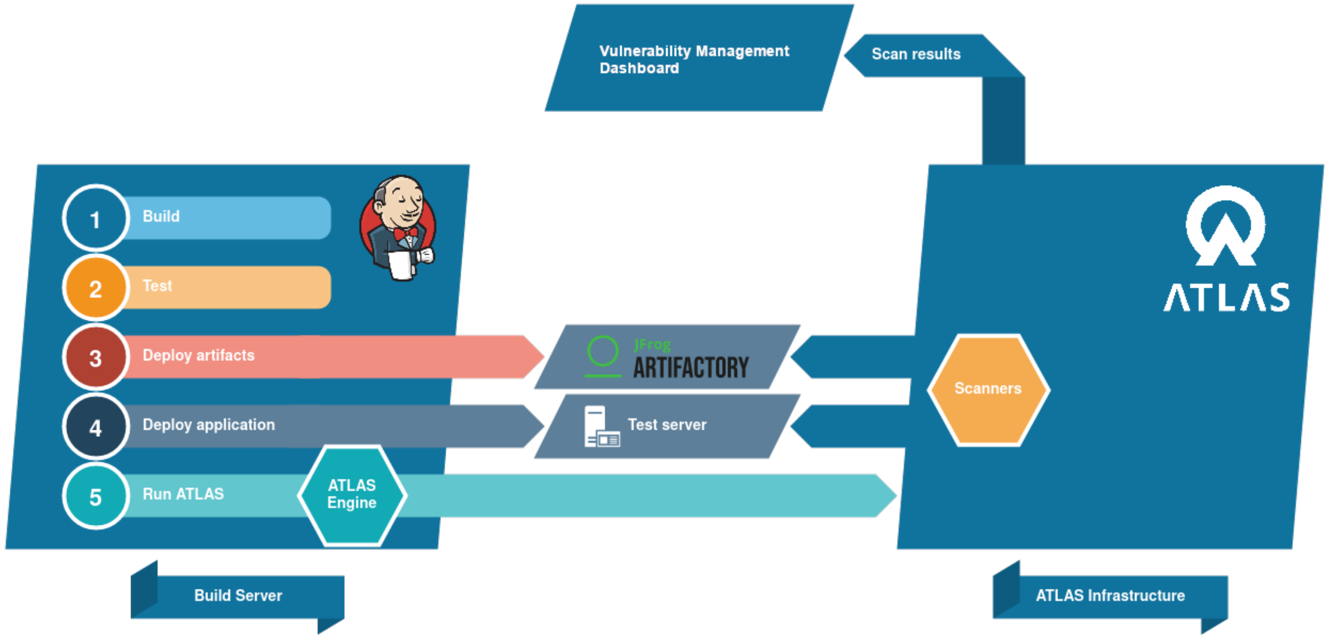


Figure 21 – *Basic architecture of ATLAS*

A12 Security Guidelines

The A12 Security Guidelines summarize a series of guidelines, best practices and recommendations for the secure use of A12. They are available on the GetA12 documentation platform and outline what a secure standard configuration looks like based on the A12 Project Template. In addition to securing service endpoints and the approach for a logging strategy taking into account the requirements of the General Data Protection Regulation, the documentation contains tips for a secure configuration of Keycloak as an identity provider and recommendations for the use of security headers.

Threat Modeling

mgm ensures the security of A12 as a platform through the various measures described above. However, as soon as a new product is created on the basis of A12, this product is exposed to individual threats that depend, among other things, on the processed data, the exposure, e.g. on the Internet, and the peripheral systems. These threats must be identified at an early stage and dealt with appropriately, i.e. counteracted (prevention), monitored (reaction) or accepted because, for example, the risk is low and the costs of measures would be disproportionately high. This process of identifying individual threats and evaluating possible countermeasures is called threat modeling and is offered by mgm as a service as part of the development of an A12 Application so that this application receives as much security as necessary - without incurring unnecessary costs for superfluous measures.

Data Protection: Responsibility in the Project Context

In most cases, an A12 application also processes personal data, such as names, email addresses, login information, or similar details. A12 takes data protection requirements into account from the outset and supports legally compliant implementation in all project phases. The platform follows the principle of “privacy by design & default” and offers data protection-friendly default settings in the A12 Project Template.

The **A12 Security Guidelines**, in conjunction with mgm’s internal **Secure Software Development Guidelines**, contain practical recommendations for the secure configuration of service endpoints, the implementation of a logging strategy that takes applicable data protection requirements into account, and the integration of identity providers such as Keycloak.

Aspects such as role and rights management, the minimization of personal data, storage duration and data deletion, as well as appropriate technical and organizational measures during processing should be taken into account as early as the design stage of an application – especially when recording functional requirements.

Responsibility for compliance with data protection requirements lies with the respective customer or project organization. In close coordination with the responsible data protection officers, issues such as the lawfulness of processing, the protection of data subjects’ rights, the performance of risk analyses and data protection impact assessments, and the maintenance of the record of processing activities must also be taken into account.

The close integration of security and data protection creates a holistic protective approach that meets regulatory requirements while strengthening technical security. A12 projects benefit from clear guidelines that help to process personal data responsibly – efficiently, transparently, and in line with high European standards.

Appendix A

Project Team - Roles and Tasks

ROLE	TASKS
Project Management (organizational)	⊕ Planning and control of tasks ⊕ Tracking of status and degree of completion ⊕ Reporting ⊕ Contact person for the client’s project management ⊕ Resource & risk management ⊕ Change management & escalation ⊕ Team building & unity
Project Management (technical)	⊕ Technical coordination ⊕ Technical responsibility ⊕ Delivery management ⊕ Release Management ⊕ Management of the development team ⊕ Reporting ⊕ Risk & problem management

ROLE	TASKS
Project Management Office (PMO)	⊕ Administrative project support ⊕ On-/Offboarding ⊕ Invoice preparation
Business Architect	⊕ Conception of a consistent technical logic ⊕ Superordinate functional application design
Business Analyst	⊕ Technical contact for the client ⊕ Requirements analysis ⊕ Concept consolidation and technical documentation
Modeler	⊕ Implementation of functional and business requirements in A12 models ⊕ Comparison of requirements with the A12 feature scope at modeling level ⊕ Fit-gap analysis development vs. modeling
Software Architect	⊕ Technical contact for the client ⊕ Overall architecture and systems involved ⊕ Control of the development tasks ⊕ Monitoring of non-functional requirements
Developer (Frontend, Backend, Full-Stack)	⊕ Detailed design, programming and documentation of the implementation of the technical requirements based on the architectural design ⊕ Creation and execution of unit tests ⊕ Execution of code reviews ⊕ Presentation of results in formats such as sprint reviews
QA Engineer	⊕ Conception of the QA activities accompanying development ⊕ Definition of test cases and test data ⊕ Execution and documentation of the tests ⊕ Support for preparation and implementation for the partial releases and the overall solution ⊕ Customer support during acceptance tests

A. Project Team - Roles and Tasks

ROLE	TASKS
Tech QA Engineer	⊕ Conception of the required QA automation ⊕ Design and implementation of load and performance tests ⊕ Ensuring the availability of the test environment per release
DevOps Engineer	⊕ Build and deployment process ⊕ Provision of necessary infrastructure (preparation of test system, maintenance and shared repository) ⊕ Configuration management ⊕ Release Management
Security Engineer	⊕ Ensuring compliance with security guidelines during ongoing development ⊕ Application Security ⊕ Security tests ⊕ Penetration tests
Designer / UI/UX Expert	⊕ Conception of style guides ⊕ Design of the user interfaces ⊕ Implementation of accessibility ⊕ Application of basic operating paradigms ⊕ Development and prioritization of ergonomic requirements
Product Owner* (by proxy, usually by the client)	⊕ Requirements analysis and creation of the product vision and roadmap ⊕ Clarification of business requirements ⊕ Prioritization of requirements ⊕ Regular feedback with all project participants ⊕ Acceptance and verification ⊕ Responsible for the success of the product
Scrum Master*	⊕ Project structure (Jira/Confluence) ⊕ Responsibility for processes and workflows incl. continuous improvement ⊕ Metrics & Key figures ⊕ Responsibility for collaboration and moderation ⊕ Team coach for increasing productivity

Appendix B

Project Management Methodology

mgm ensures that the cooperation of all parties involved in the project runs smoothly and that all goals are achieved efficiently and in a resource-saving manner. Which project management methodology is used depends in detail on the structures and requirements of the client. In the following, we outline a number of important factors and approaches that have proven themselves in our project practice.

- **Stakeholder Analysis and Management**

  For successful project implementation and acceptance of the application, it is crucial to consider the interests of key internal and external stakeholders - and to involve them accordingly, depending on the relevance of the project. Typical stakeholders are representatives of system interfaces, user groups, administrators and department heads.

- **Definition of Roles, Responsibilities and Accountabilities**

  Both the roles relating to technical development services and organizational management must be clearly defined and transparent. To this end, criteria are first defined that describe the responsibilities within a task in more detail - for example, according to the RACI technique. Then the responsibilities for each task area are defined according to the specified grid for each role. mgm uses a responsibility assignment diagram here. This avoids misunderstandings that would otherwise lead to conflict situations and a risk that is difficult to calculate.

- **Risk Analysis and Preparation of Risk Management**

  We recommend a detailed risk analysis to identify project risks. It identifies the project risks in the respective implementation phases and estimates their probability of occurrence, assesses the extent of damage and describes countermeasures. This includes risks in the technical implementation and in the area of personnel as well as external risks. The analysis should lead to continuous risk management.

- **Development of a Communication and Reporting Channel**

  A communication plan describes which stakeholders receive which type of information when and with which method via which medium. A communication matrix is used to agree with the stakeholders (e.g. project manager, steering committee, works council) which media should be used and how. The selection of an optimal communication and reporting channel depends on the defined project roles as well as individual and cultural aspects.

- **Creating a Project Glossary**

  Every software project involves a variety of technical and specialist aspects that can be interpreted differently. In order to promote a common understanding as early as possible, a project-related glossary should be created immediately after the start of the project. It contains the most important definitions of terms and helps to avoid misunderstandings. As the project progresses, further definitions can be added successively or the initial definitions can be sharpened.

Decision Guidance for the Right Methodology

Some situations are clearly defined and predictable. For example, if we look at a garden maze from a bird's eye view, we can use simple means to find a path from the target location in the middle through the branching paths to the exit. We can design a plan that reliably leads to the goal. In enterprise software development, however, we repeatedly encounter situations that are dynamic and unpredictable in parts. Here, the situation is more like walking through a wild jungle to reach an underlying goal. The path is not completely predictable. Weather conditions such as heavy rain can make areas impassable. Here, we make our way step by step towards the goal, constantly observing our surroundings and adapting the route accordingly.

In the terminology of software projects, we also speak of a waterfall model in situations such as the garden maze and the clearly definable plan. In the case of the jungle, however, we are operating in the world of agile methods, which provide for an iterative, step-by-step approach in a dynamic environment. But when is it a labyrinth and when is it a jungle? Spoiler alert: In most software projects, both cases exist in different shades. There are aspects that are known and can be planned very well. And there are aspects that are largely uncertain or subject to great dynamics.

For successful project management, it is crucial to know which situation we are currently in - and how we can best navigate this situation in order to achieve a desired goal. mgm relies on the Cynefin framework when choosing a suitable methodology. It distinguishes between four situations, each of which entails different recommendations for action:

- **Simple:** The situation is known and there are clear causal relationships. The application of best practices reliably leads to the goal.
- **Complicated:** The situation is not fully known and requires in-depth analysis. However, it can ultimately be countered with a clearly structured approach and a plan. Good forecasts and valid estimates are possible. However, decisions must be made early on and changes can only be addressed to a limited extent.
- **Complex:** The situation is confusing and can only be tackled through an experimental approach. Solutions are found step by step. While forecasts are difficult, the procedure allows for quick reactions. Decisions can be made late.
- **Chaotic:** The situation is uncontrollable, but requires a direct response. A more detailed assessment is carried out downstream.

The classification of situations in this scheme is not fixed. The assessment of a situation can change over the course of the project. For example, if the team penetrates a complex situation deeply enough, it can initially be classified as complicated - and later perhaps as simple. The more situations that fall into the "simple" category over the course of the project, the better. This is where processes can typically be automated and the digital dividend of the project increased.

Agile Methodologies

During implementation, mgm relies on common agile methods such as **SCRUM** or **Kanban**.

Both methods describe an iterative process in which tasks, functions and criteria are organized in a product backlog. Within this backlog, they are prioritized and then processed by the team based on the prioritization. mgm ensures that the full potential of an iterative process can be used. Regular feedback meetings with all stakeholders allow processes and functions to be re-evaluated and discussed changes to be implemented immediately. By using these methods efficiently, mgm therefore actively contributes to improving the workflow and productivity of the entire team. Another advantage of an agile A12 project is the avoidance of a long planning phase. The procedure provides for the fastest possible creation of an executable first product version, which is expanded iteratively - always with a clear view of the requirements of the client and future users. The use of the A12 platform and the low-code approach accelerate development and ensure that the first version already has a comparatively high level of maturity.

Customer-Specific Customization and Hybrid Models

As there is no one-size-fits-all strategy in terms of software development, all A12 projects are customized. This can mean combining approaches from agile and classic methodologies. This approach is usually chosen in order to avoid the disadvantages of a single methodology and to do justice to the circumstances of the client's organization. The individually structured weighting allows only part of the team to operate according to agile methodologies (e.g. product development), while other parts of the team (higher-level management, controlling or marketing) follow the approaches of classic project management. This may be due to classic hierarchical company structures in which agile working is difficult to implement. The final weighting of the classic and agile parts of the hybrid solution is worked out jointly by the client and mgm, whereby individual areas are checked for their complexity and associated plannability using the Cynefin framework.

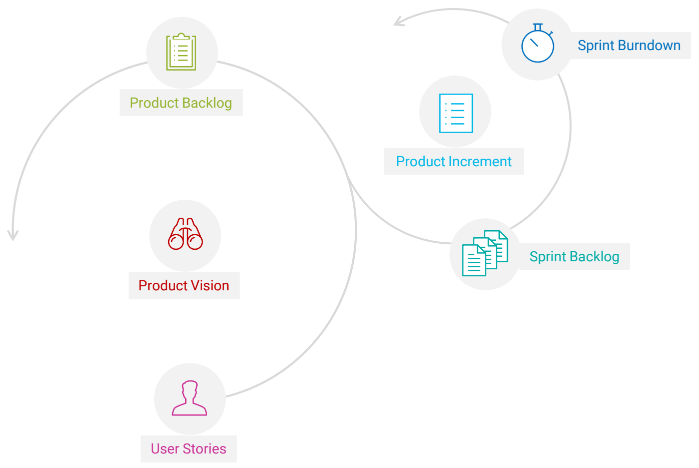


Figure B.1 – *Relationships between the most important artifacts in the agile approach*

ARTEFACT	EXPLANATION
Product Backlog	The product backlog is a structured and prioritized list of all remaining functions and tasks. The product owner is responsible for the correct maintenance of the backlog, using the parameters of business value, risk and delivery date.
Sprint Backlog	The sprint backlog contains all the specifications of the tasks that the development team plans to implement in a sprint.
Product Increment	The product increment corresponds to a list that contains all the tasks and functions completed in the current sprint. The hurdle of this artifact is the definition of the status "Done" and the heterogeneous understanding of all team members. The aim is to assemble the completed tasks into a minimum viable product (MVP) as soon as possible. As soon as a product with core functions has been developed, it is presented to the user and further developed together with them. This minimizes the risk of a product being developed without meeting the needs of the target group.
Sprint Burndown	The sprint burndown explains how quickly the user stories were processed compared to the planning.
User Stories	User stories correspond to subordinate individual tasks that are developed from the customer's perspective. Functions or options are often added to a product even though it is not clear whether the user needs or even wants them. SCRUM teams try to get around this with user stories.
Product Vision	A shared product vision enables a team to develop intrinsic motivation and to continuously compare the respective work status with the goal.

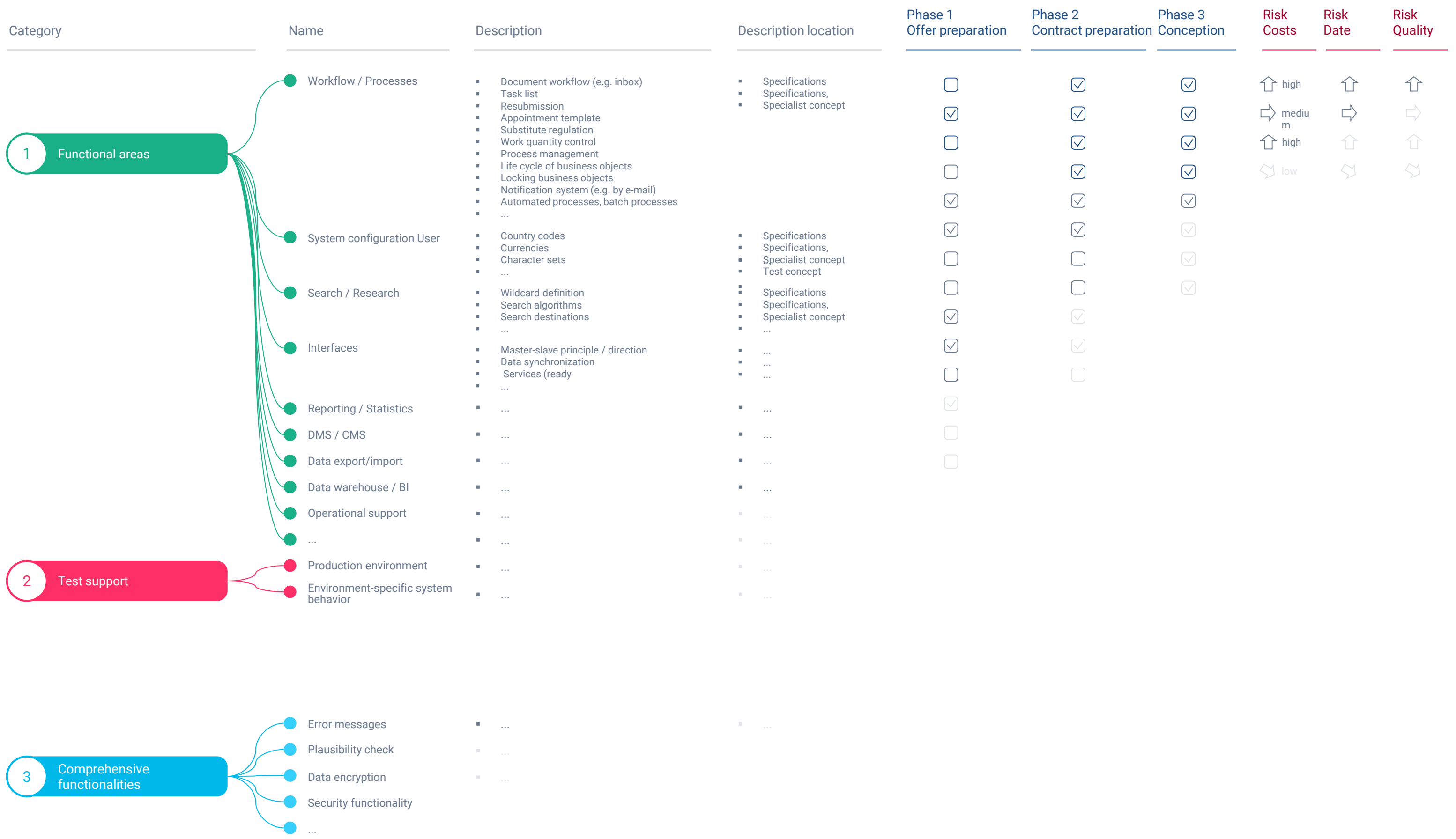
Appendix C

Systematically Recording Requirements

Based on many years of experience in large enterprise software projects, mgm has developed a structured process for analyzing and recording requirements. A series of templates and checklists help to ask the right questions at the right time and to record requirements in an appropriate level of detail.

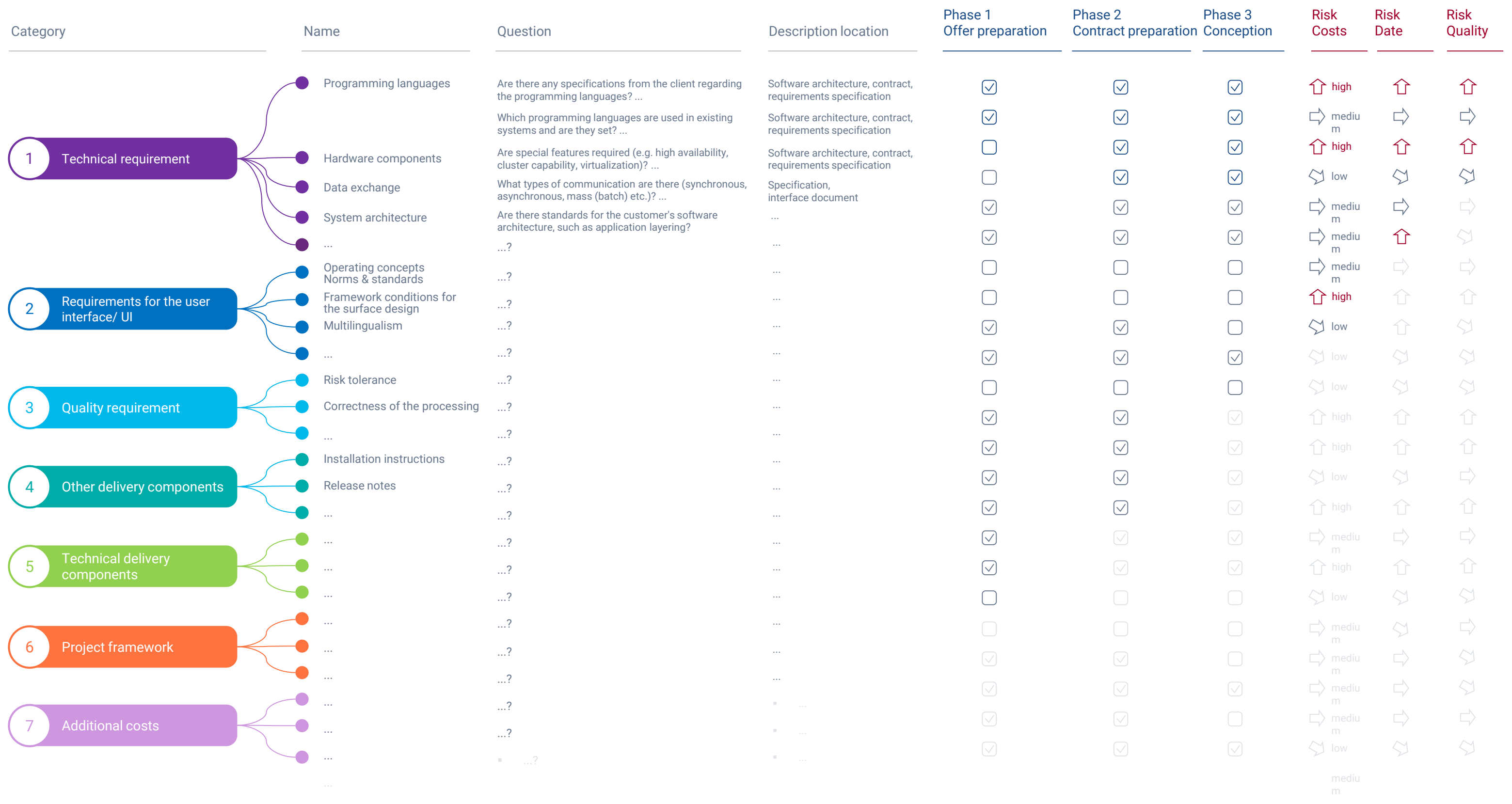
The most important source of requirements is the people involved in the project. With the help of a stakeholder checklist, we ensure that all key contacts for various roles are identified. Other, non-personal sources of requirements can be, for example, legacy systems and their documentation as well as legal requirements and standards.

When eliciting requirements, we rely on questionnaires, interviews, requirements workshops, observation/collaboration in user group processes and prototyping, among other things. The description of requirements is always target group-specific and results-oriented. Particularly in the case of domain-specific, functional requirements, joint modeling sessions can also contribute to a better understanding. In order to concretize requirements, we also define fulfillment and acceptance criteria - in addition to any acceptance criteria of the client - as required. They contain important information for implementation and also form the basis for test cases.



Category	Name	Description	Description location
1 Functional areas	Workflow / Processes	▪ Document workflow (e.g. inbox) ▪ Task list ▪ Resubmission ▪ Appointment template ▪ Substitute regulation ▪ Work quantity control ▪ Process management ▪ Life cycle of business objects ▪ Locking business objects ▪ Notification system (e.g. by e-mail) ▪ Automated processes, batch processes ▪ ...	▪ Specifications ▪ Specifications, ▪ Specialist concept
	System configuration User	▪ Country codes ▪ Currencies ▪ Character sets ▪ ...	▪ Specifications ▪ Specifications, ▪ Specialist concept ▪ Test concept
	Search / Research	▪ Wildcard definition ▪ Search algorithms ▪ Search destinations ▪ ...	▪ Specifications ▪ Specifications, ▪ Specialist concept ▪ ...
	Interfaces	▪ Master-slave principle / direction ▪ Data synchronization ▪ Services (ready ▪ ...	▪ ... ▪ ... ▪ ...
	Reporting / Statistics	▪ ...	▪ ...
	DMS / CMS	▪ ...	▪ ...
	Data export/import	▪ ...	▪ ...
	Data warehouse / BI	▪ ...	▪ ...
	Operational support	▪ ...	▪ ...
	...	▪ ...	▪ ...
2 Test support	Production environment	▪ ...	▪ ...
	Environment-specific system behavior	▪ ...	▪ ...
3 Comprehensive functionalities	Error messages	▪ ...	▪ ...
	Plausibility check	▪ ...	
	Data encryption	▪ ...	
	Security functionality		
	...		

Figure C.1 – *Collection of functional requirements*



Question	Description location
Are there any specifications from the client regarding the programming languages? ...	Software architecture, contract, requirements specification
Which programming languages are used in existing systems and are they set? ...	Software architecture, contract, requirements specification
Are special features required (e.g. high availability, cluster capability, virtualization)? ...	Software architecture, contract, requirements specification
What types of communication are there (synchronous, asynchronous, mass (batch) etc.)? ...	Specification, interface document
Are there standards for the customer's software architecture, such as application layering?	...
...?	...

Figure C.2 – *Collection of non-functional requirements*

Requirements Analysis in an Agile Project

In agile A12 projects, mgm applies cross-departmental requirements management. All requirements are viewed from the user's perspective and prioritized for each department. Prioritization is carried out in close coordination with the client. This is based on the continuous evaluation of the respective contribution to the achievement of objectives, the cost-benefit ratio and any implementation risks. In order to institutionalize the prioritization process, regular - e.g. 14-day - backlog grooming has proven successful in more complex projects.

All requirements are initially recorded as user stories from the user's perspective. Further categorization takes place based on the relevance of the respective requirement. If this is above average or the topic is overly complex, the user story becomes an epic. An epic usually consists of several stories that must be completed to fulfill the epic. The advantage of this scheme is that it offers a second level of prioritization, namely the prioritization of the epics, as well as the prioritization of the individual stories within the epic.

In certain cases, an agile project or sub-aspects may require a detailed concept in the classic sense. mgm prefers to plan only the first milestone in detail in this case. The subsequent milestones are only estimated at this point in time and further elaborated at the beginning of the respective project phase based on current results and findings. With this approach, mgm incorporates the planning aspect of classic project management into an agile approach. This means a significant reduction in the initial planning effort and enables continuous, precise planning of the ongoing A12 project with a high level of client involvement.

All requirements are documented centrally in tickets in a digital requirements management tool (Jira). This means that they - as well as the implementation status - can be viewed transparently by everyone involved and can be commented on. The requirements are always described in a target group-specific and results-oriented manner. Supplementary concepts are stored in a project wiki and are linked bidirectionally with the tickets from the requirements management tool.

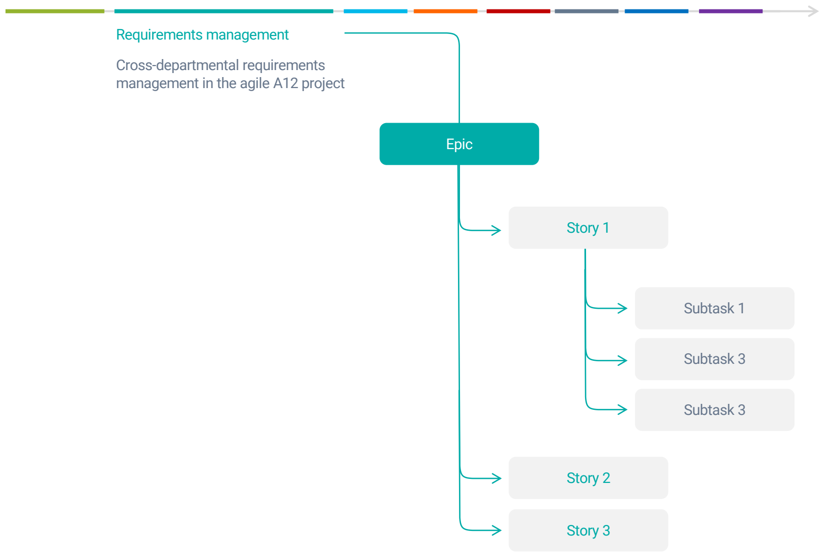


Figure C.3 – *Prioritization levels of requirements*

Glossary

A11Y Abbreviation for accessibility. 30

A12 Application Application developed on the basis of the A12 platform. 25, 29, 30, 37, 41, 53

A12 model Model that has emerged from the model-based development approach of A12 - e.g. *Document Model* in the area of data modeling and *Form Model* in the area of UI modeling. 21, 22, 40

A12 component Technical component that is part of the A12 runtime platform - e.g. "Data Services" and "Workflows". 21

A12 project Individual software project in which an A12 application is created. 13, 14, 15, 25, 27, 30, 52

A12 team Team within mgm responsible for the development of the A12 platform. 2, 24, 25, 32, 52

ATLAS security toolset developed and offered by mgm, see also https://www.mgm-sp.com/mgm-atlas. 2, 41, 50, 52

BPS Business Professional Services - a team at mgm acting as an interface between project teams and the A12 team, which provides support with business and modeling related questions. 24

project team Team responsible for the development of an A12 application as part of an A12 project. 14, 28

PS Professional Services - Umbrella term for BPS and TPS. 15, 19, 24

QA Short for Quality Assurance. 7, 42

SME Simple Model Editor - the central modeling tool of the A12 platform. 22

TPS Technical Professional Services - a team at mgm acting as an interface between the project teams and the A12 team, providing support with technical issues. 24

Have we sparked your interest? Talk to us about how A12 and clear project standards can help to drive your projects forward in a sustainable way.

www.mgm-tp.com/a12.html

mgm technology partners GmbH

Taunusstr. 23
80807 München

Germany

Tel +49 89 / 35 86 80-0

www.mgm-tp.com
info@mgm-tp.com

Innovation Implemented.