mgm

mgm A12-Projektstandards

Abwicklung von Softwareprojekten mit der Entwicklungsplattform A12

Januar 2026

Innovation Implemented.

Impressum

mgm technology partners GmbH
Taunusstr. 23
80807 München
Tel +49 89 / 35 86 80-0

Gerichtsstand und Erfüllungsort: München

www.mgm-tp.com

Inhaltsverzeichnis

01

Management Summary

Wie lässt sich robuste, sichere und langlebige Enterprise Software bauen – und zwar möglichst schnell und möglichst wirtschaftlich? Eine Antwort darauf liefert die modellbasierte Softwareentwicklung. Mit Hilfe der Entwicklungsplattform A12 von mgm lassen sich typische Aufwände der Entwicklung von Geschäftsanwendungen reduzieren – allen voran durch eine Trennung von Fachlichkeit und Technik.

Doch allein die Verwendung einer Plattform garantiert noch keinen Projekterfolg. Die Entwicklung und der Betrieb von sicheren, skalierbaren, kosteneffizienten und performanten Unternehmensanwendungen erfordert durchdachte und integrierte Produktionsprozesse sowie Expertise in der Individualentwicklung und Systemintegration. Dieses Whitepaper fasst die wichtigsten Standards, Vorgehensweisen und Stationen einer auf Enterprise-Anwendungen spezialisierten Softwareproduktion zusammen, die mgm bei Projekten auf Basis von A12 einsetzt.

Das Wichtigste in Kürze

- **Entwicklungsprozesse - standardisiert statt individuell**
  Die gestiegenen Anforderungen an Enterprise Software - u.a. durch Cyber-Risiken, Usability-Erwartungen, Cloud-Betrieb und das Streben nach digitaler Souveränität - erfordern integrierte Entwicklungsprozesse. Eine Entwicklung auf der grünen Wiese ist weder effizient noch zeitgemäß.

- **Partnerschaftlicher Ansatz durch modellbasierte Entwicklung**
  Der Einsatz einer modellbasierten Entwicklungsplattform wie A12 ermöglicht neue Formen der partnerschaftlichen Entwicklung von Individualsoftware. Wesentliche Teile der Anwendung sind nach dem Low-Code-Prinzip in Modellen spezifiziert – ohne Programmierung. Daraus ergeben sich neue Arbeitsweisen und Partizipationsmöglichkeiten für Fachbereiche in allen Phasen des Softwarelebenszyklus.

- **Passendes Setup für Organisation und Technik**
  Die Kapselung fachlicher Inhalte in Modellen und Regeln schafft Unabhängigkeit von Technik und erhöht die Lebensdauer der Anwendung. Damit Modellierung und Software Engineering nahtlos ineinandergreifen, braucht es aber das richtige organisatorische und technische Setup.

- **Plattform-Ansatz auf Basis von Open Source**
  Der konsequente Einsatz von Open Source ist die Voraussetzung für einen langfristig erfolgreichen Plattform-Ansatz. Die Offenheit und Transparenz beugt Vendor Lock-in vor, sichert die Anschlussfähigkeit an benachbarte Systeme und ermöglicht unabhängige Sicherheitsanalysen.

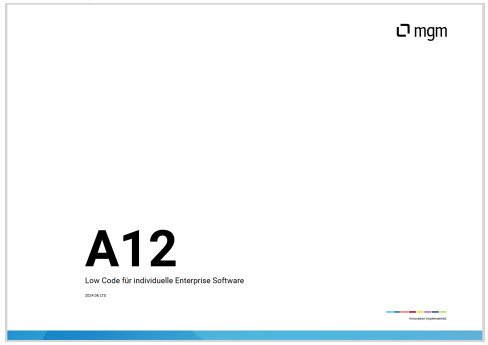
- **Automatisierung mit modellbasiertem Tooling**
  Der Einsatz von Modellen eröffnet neue Möglichkeiten, wesentliche Schritte des Softwareentwicklungsprozesses zu automatisieren. mgm hat neben einer darauf abgestimmten Projektmethodik auch neue begleitende Werkzeuge für die Qualitätssicherung entwickelt, die auf den modellbasierten Ansatz abgestimmt sind.

- **Beherrschbare KI-Nutzung**
  KI schafft neue Möglichkeiten, muss aber auch beherrschbar bleiben. Die A12-Plattform schafft einen zukunftsweisenden Rahmen, um generative KI und agentenbasierte Systeme kontrolliert, effektiv und verantwortungsbewusst einzusetzen. KI-Modelle für unterschiedliche Anwendungszwecke können problemlos eingebunden werden.

- **Security by Design - Sicherheit von Anfang an**
  Die Laufzeitplattform von A12 bietet robuste und kontinuierlich sicherheitsüberprüfte Komponenten. Um darauf aufbauende Anwendungen abzusichern, setzt mgm auch projektseitig u.a. auf frühes Threat Modeling, entwicklungsbegleitende Lean Application Security und automatisierte Security-Analysen.

Was ist A12?

A12 ist eine Plattform für die Entwicklung von Unternehmensanwendungen in komplexen IT-Landschaften. Sie setzt auf modellbasiertes Software Engineering (MDSE) und erschließt das Low Code-Prinzip für die Welt der Enterprise Software. Als offene Plattform vereinfacht A12 die Integration von Best-of-Breed-Lösungen und den Einsatz von KI auf allen Ebenen. Die Modellierungsumgebung von A12 stellt Werkzeuge bereit, um Teile einer Anwendung ohne Programmierkenntnisse zu erstellen und als unabhängige Geschäftslogik-Module langfristig zu pflegen. Die Laufzeitplattform von A12 bietet die nötige Flexibilität, um geschäftskritische Applikationen mit professioneller Individualsoftwareentwicklung, KI-Unterstützung und Systemintegration zu voll integrierten Unternehmensanwendungen zu entwickeln. Eine Einführung in A12 bietet das Whitepaper A12 - Low Code für individuelle Enterprise Software



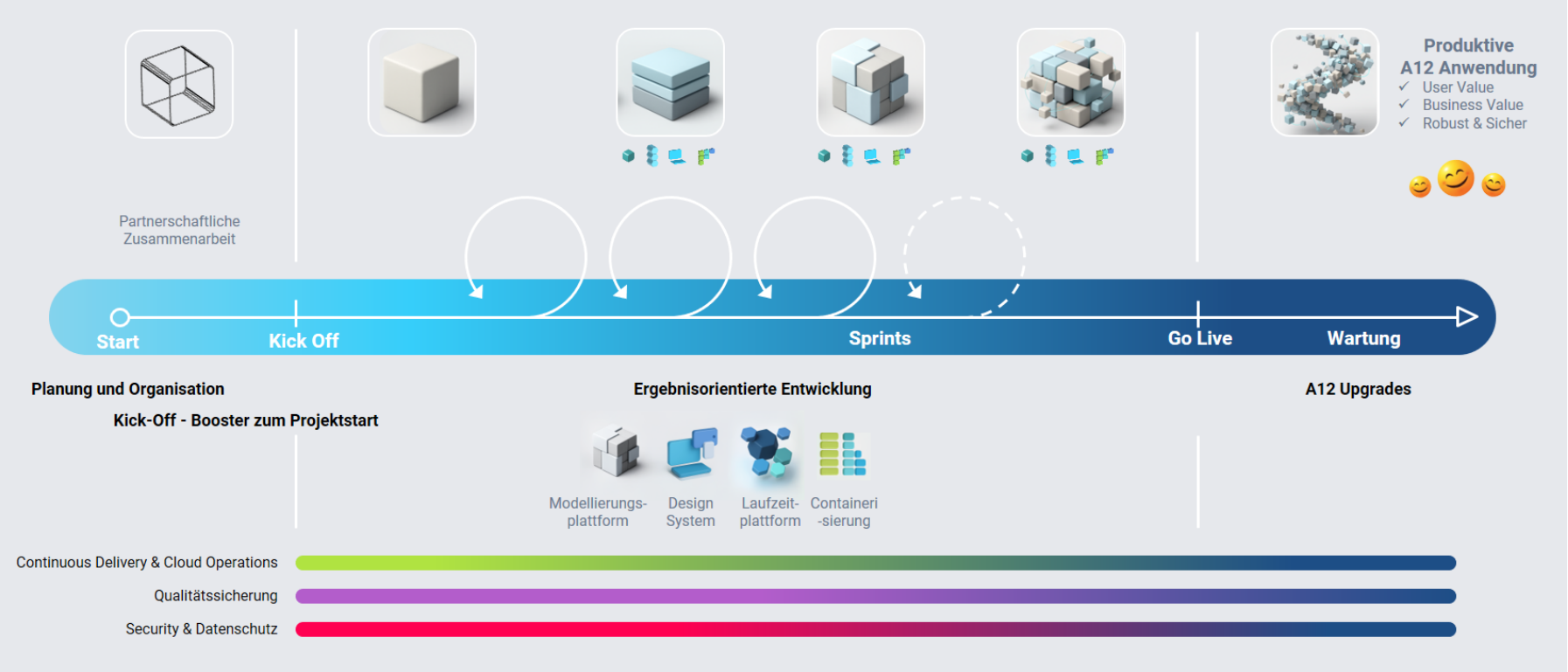


Abbildung 1 – *Der Entwicklungsprozess mit A12 im Überblick*

02

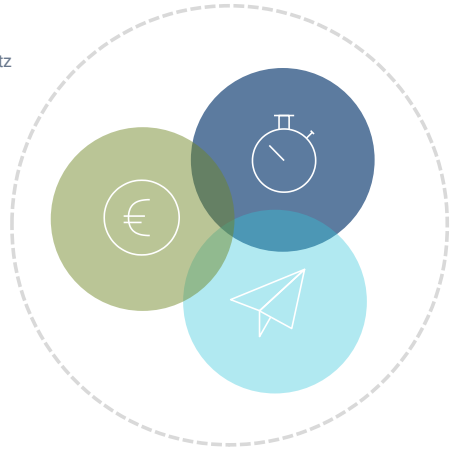
Industrialisierte Softwareproduktion

Mit der fortschreitenden digitalen Transformation wird Software für individuelle Geschäftsprozesse ein immer wichtigerer Wettbewerbsfaktor. Gleichzeitig steigen die Anforderungen an den Softwareentwicklungsprozess – vor allem durch höhere Ansprüche an die Bedienbarkeit, Sicherheit und die schnelle Bereitstellung von Enterprise Software. Der höhere Grad an Komplexität, der Einsatz des Low Code Paradigmas und die Fähigkeiten von KI verändern den Entwicklungsprozess und motivieren eine industrialisierte Softwareproduktion.

Ob Behörde, Versicherung oder Versandhändler, keine größere Organisation kommt heute ohne individuelle Software aus, die ihre Abläufe effizienter gestaltet und neue Services ermöglicht. Die digitale Transformation hat einen Wandel eingeläutet, in dem die Bereitstellung und Nutzung von Software wesentlich beeinflusst, wie wettbewerbsfähig ein Unternehmen ist, welche Innovationen es umsetzen kann und wie gut es die Erwartungen der Kundschaft erfüllt.

A12

Modellbasierte Softwareentwicklung und der Einsatz der Enterprise Low Code Plattform A12.



Beschleunigt die Softwareentwicklung

Legt einen Grundstein für Nachhaltige Investitionen

Steigert die Flexibilität und Anpassbarkeit

Komplexitäts-Dimensionen von Enterprise Software

Mit der gestiegenen Bedeutung von Enterprise Software sind auch die Anforderungen an die Softwareentwicklung drastisch gestiegen. Anwendungen müssen möglichst schnell entwickelt und nahtlos in bestehende IT-Landschaften integriert werden. Gleichzeitig müssen sie performant und skalierbar sein, eine stimmige Nutzererfahrung bieten, und hohen Sicherheitsanforderungen genügen. Immer wichtiger wird auch eine flexible und schnell umsetzbare Anpassbarkeit – vor allem in Hinblick auf fachlich getriebene Änderungen, aber auch vor dem Hintergrund von Technologiesprüngen.

Klassische Aufwandstreiber

Zu den größten Herausforderungen von Enterprise Software gehört langfristig der Umgang mit Änderungen. Einen Lösungsansatz bietet die modellbasierte Softwareentwicklung und der Einsatz der Plattform A12. Sie

- beschleunigt die Softwareentwicklung durch die Verwendung vorgefertigter Komponenten, Modellierungswerkzeuge und Code-Generatoren sowie den mit QS-Werkzeugen validierbaren Einsatz von KI,
- steigert die Flexibilität und Anpassbarkeit der entwickelten Software, da Teile der Anwendung ohne Programmierung von Fachexperten gepflegt werden, und
- legt einen Grundstein für nachhaltige Investitionen in langlebige Software, die durch eine klare Trennung von Technik und Fachlichkeit auch auf neue künftige Technologiestufen gehoben werden kann.

Für die Softwareentwicklung bedeutet der Low Code-Ansatz einen Paradigmenwechsel, der die Art und Weise der Entwicklung sowie der Pflege von Anwendungen an einigen Punkten entscheidend verändert. Im Enterprise-Umfeld ist die Verwendung einer Low Code Plattform jedoch nie allein die Lösung aller Probleme. Es gibt immer fachliche und technische sowie funktionale und nicht-funktionale Anforderungen, die einer individuellen Entwicklung bedürfen. Und es gibt immer die Herausforderung der Systemintegration. Denn nur im Zusammenspiel mit bestehenden und künftigen Anwendungen entfaltet Enterprise Software ihr volles Potenzial.

In der Projektpraxis müssen deshalb Low Code-Praktiken, KI-Unterstützung, Individualentwicklung und Systemintegration nahtlos eineinandergreifen, um hochqualitative Enterprise Software in Produktion zu bringen. Dafür bedarf es eines gut durchdachten organisatorischen und technischen Rahmens.

Das Nutzererlebnis muss stimmen

Geschäftsanwendungen sind klassischerweise nicht für ein umwerfendes (Interaktions-)Design und ein überzeugendes Nutzererlebnis bekannt. Heute sind diese Qualitätsmerkmale für Enterprise Software jedoch unabdingbar. Die Ansprüche der Nutzer:innen sind drastisch gestiegen. Wer den Komfort moderner Web-Anwendungen im Privaten gewohnt ist, wird im beruflichen Umfeld nur mit Widerwillen eine veraltet wirkende, rein funktionale Software benutzen, die nicht auf zeitgemäße Gestaltungsprinzipien setzt.

Darüber hinaus gelten für B2B-Anwendungen andere Regeln als im B2C-Kontext. Sie richten sich an professionelle Nutzer:innen, die oft täglich und über einen langen Zeitraum mit der Software arbeiten. Keyboard-Shortcuts und weitere produktivitätssteigernde Funktionen wie Such- und Filteroperationen sind an der Tagesordnung. Die Bereiche User Interface (UI) und User Experience (UX) sind deshalb nicht nur wesentliche Faktoren für die Akzeptanz von Geschäftsanwendungen. Sie zahlen auch auf die Produktivität ein, indem sie dafür sorgen, dass Nutzer:innen möglichst zielgerichtet ihre Aufgaben effizient erledigen können. Um die gestiegenen Ansprüche an das Interfacedesign und die Nutzererfahrung zu adressieren, setzen A12-Anwendungen auf das Designsystem Plasma, das speziell für die Anforderungen von Enterprise-Anwendungen entwickelt wurde.

Steigende Cyber-Risiken

Geschäftsanwendungen sind heute vernetzter als je zuvor. Sie erstrecken sich über immer mehr Unternehmensbereiche und bieten dadurch eine höhere Produktivität. Gleichzeitig sind sie aber auch neuen Risiken ausgesetzt. Mit dem steigenden Digitalisierungsgrad von Organisationen nahm auch das dunkle Geschäft der Cyberkriminalität in den vergangenen Jahren deutlich zu.

So bezeichnet das Bundesamt für Sicherheit in der Informationstechnik die IT-Sicherheitslage in Deutschland als angespannt bis kritisch. Laut dem Bericht zur Lage der IT-Sicherheit in Deutschland 2024 traten im Zeitraum vom 1. Juli 2023 bis zum 30. Juni 2024 im Schnitt täglich 309.000 neue Varianten von Schadsoftware hinzu. Von 2022 bis 2023 stieg die Anzahl der täglich bekannt gewordenen Schwachstellen in Softwareprodukten um 14 Prozent. Cyber-Kriminelle agieren in einer zunehmend ausdifferenzierten Untergrundökonomie und haben laut dem Lagebericht ihre Erpressungsmethoden deutlich ausgeweitet.

Bei der Entwicklung von Enterprise Software hat deshalb das Thema Sicherheit einen ganz anderen Stellenwert als noch vor wenigen Jahren. Vereinzelte Penetrationstests vor der Inbetriebnahme eines Softwareprodukts sind längst nicht mehr ausreichend. Es geht vielmehr darum, die Expertise rund um das IT Security Engineering direkt in den Entwicklungsprozess zu integrieren und potenziell unsichere Implementierungen und Konfigurationen von Anfang an zu vermeiden. Der Plattform-Ansatz von A12 bietet hier große Vorteile für eine resiliente Anwendungslandschaft. Die gesamte Code-Basis von A12 wird durchgängig mit automatisierten Security-Analysen überprüft. Zentral gepflegte Bausteine und Templates werden regelmäßig geprüft und per Default sicher konfiguriert. Da bei A12-Anwendungen ein Teil des Codes generiert wird, sinkt das Risiko von Implementierungsfehlern. Die Plattform fördert dadurch den Einsatz von Best Practices und erleichtert die Einhaltung von Compliance-Vorgaben und Security-Standards.

Cloud und Containerisierung

Die Entwicklung und der Betrieb von Software greifen immer stärker ineinander. Ein technischer Treiber dafür ist der Trend zu Microservice-Architekturen und Containertechnologien. Der Schnitt in kleinere Versatzstücke und die Paketierung in autonome Einheiten erhöht die Flexibilität und das Tempo, in dem entwickelte IT-Services bereitgestellt werden können. Gleichzeitig entstehen daraus aber auch neue Herausforderungen und anspruchsvolle administrative Aufgaben rund um die Orchestrierung der Auslieferung von Software. Für die Bewältigung dieser Aufgaben ist DevOps-Expertise im Team unabdingbar.

A12-Anwendungen basieren auf einer durchdachten Microservice-Architektur und lassen sich sowohl On-Premise als auch in Cloud-Umgebungen betreiben. Eine Reihe von Templates (Build & Deployment Pipelines, Helm Charts, Project Template...) vereinfacht und standardisiert den Übergang von der Entwicklung zu einem robusten, performanten und skalierbaren Betrieb. Mit C12 bietet mgm zudem ein Cloud-Angebot, das maßgeschneidert auf den Betrieb von A12-Anwendungen abgestimmt ist.

„Die Ära isolierter großer Enterprise-Projekte ist vorbei“

Hamarz Mehmanesh, CEO von mgm, erläutert im Gespräch seine Vorstellung einer industrialisierten Softwareproduktion.

Seit fast 30 Jahren entwickelt mgm Geschäftssoftware. Welche Konstanten gab es in dieser Zeit – trotz aller technischen Umbrüche?

Heute wie damals sind Enterprise Geschäftsanwendungen komplex, vernetzt, langlebig und von den – teils auch gegenläufigen – Bedürfnissen großer Organisationen getrieben. Klassische Aufwandstreiber wie fachlich bedingte Anpassungen der Software entlang des gesamten Lebenszyklus sind weiterhin hochrelevant. Darüber hinaus sind die Anforderungen an Enterprise Software und die Komplexität der Entwicklung aber massiv gestiegen. Umso erstaunlicher finde ich die in der Branche immer noch weit verbreitete Praxis, dass man einfach ein Team zusammenstellt und auf der grünen Wiese loslegt. Das funktioniert zwar, keine Frage. Aber ist es heute noch sinnvoll und zeitgemäß? Ich denke nicht.

Warum nicht? Was steht dem Ansatz entgegen, Enterprise Software auf der grünen Wiese mit einem Team aus rekrutierten Spezialist:innen zu entwickeln?

Der Preis ist einfach zu hoch, es ist nicht effizient. Es gibt nun mal eine Reihe neuer Komplexitätsdimensionen, die nur durch eingespielte, zunehmend industrialisierte Entwicklungsprozesse effizient bewältigt werden können. So ist der Anspruch an das Nutzererlebnis gestiegen. Erstklassige Bedienbarkeit ist ein Muss – und zwar auf allen Geräten. Dann das Thema Security: Aufgrund des zunehmenden Grads an Vernetzung und immer professioneller agierender Cyber-Krimineller müssen Anwendungen von Anfang an kompromisslos abgesichert werden. Der Siegeszug der Cloud und Container-Technologien hat außerdem eine engere Verzahnung von Entwicklung und Betrieb der Software bedingt, die weitere Herausforderungen mitbringt. Darüber hinaus gibt es immer mehr gesetzliche Anforderungen – zum Beispiel im Datenschutz und bei der Barrierefreiheit. Wie soll ein rein rekrutiertes Team all das angemessen berücksichtigen und nachhaltige Software bauen?

Was wäre denn die Alternative? Wie sehen industrialisierte Entwicklungsprozesse aus?

Wir kombinieren bei mgm mehrere Ansätze, um eine Art agile Fertigungsstraße aufzubauen. Den Kern bildet unsere Plattform A12. Damit reduzieren wir die Fertigungstiefe in den Einzelprojekten und verlagern zusätzlich die Wertschöpfung von IT-Experten auf Fachexperten. Fachliche Inhalte werden mit Hilfe spezieller Werkzeuge modelliert, nicht programmiert. Dann der Einsatz von KI: Wir haben robuste Komponenten und klar strukturierte A12-Modelle, mit denen KI-Modelle sehr gut arbeiten können. Reines Vibe Coding funktioniert im Enterprise-Kontext noch nicht verlässlich. Mit A12 und den speziell dafür entwickelten Tools für Qualitätssicherung, Security-Prüfungen und Build & Deployment haben wir aber einen kontrollierbaren Rahmen, in dem wir generative KI und agentenbasierte Systeme sehr effektiv und verantwortungsbewusst einsetzen können. Damit sind wir der nächsten Evolutionsstufe der Softwareentwicklung schon heute sehr nah.

Welche Vorteile ergeben sich durch einen stärker industrialisierten Entwicklungsprozess für Kunden, die Individualsoftware in Auftrag geben?

Das Risiko sinkt. Je stärker der Softwareentwicklungsprozess den Prinzipien der Industrialisierung folgt, desto wahrscheinlicher ist die Bereitstellung hochqualitativer, nutzerfreundlicher und sicherer Software in kurzer Zeit. Die Ära großer isolierter Enterprise-Projekte ist meiner Ansicht nach vorbei. Das kann kein Unternehmen mehr verantworten – zumal das Geschäft immer schnelllebiger wird. Wie sich die aktuellen Plattformen und Low Code Ansätze langfristig schlagen, wird sich zeigen. Die nächsten technischen Innovationen werden kommen. Wir verfolgen deshalb den Ansatz, fachliche Inhalte derart zu modellieren, dass sie unabhängig von bestimmten Technologien sind – und wirklich langfristig weitergenutzt werden können im Sinne nachhaltiger Software.

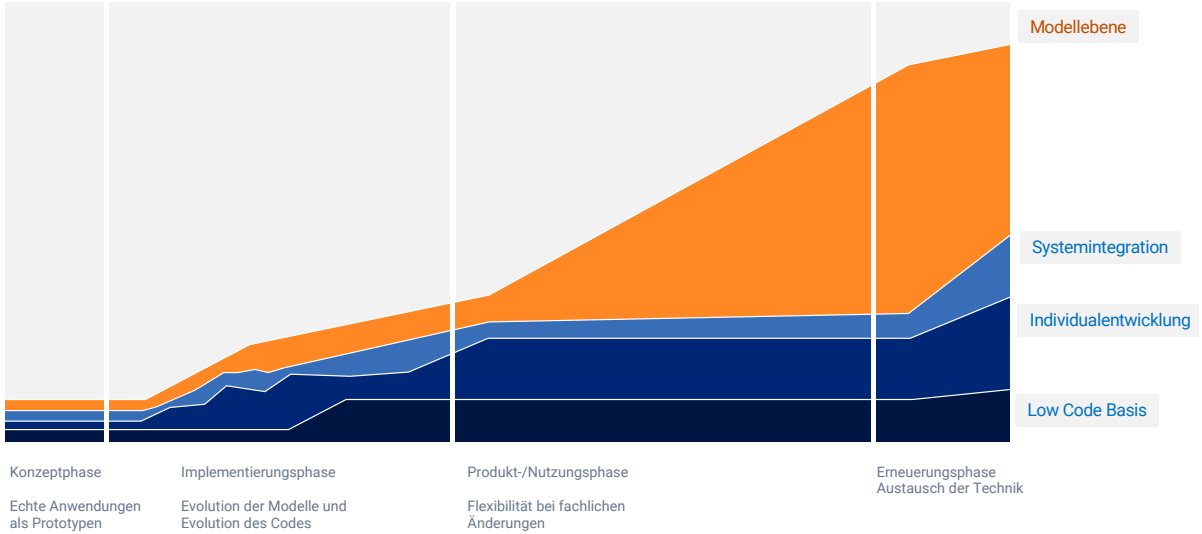
Der Lebenszyklus von Low Code Software

Wie unterscheidet sich ein Individualprojekt auf Basis einer Low Code Plattform von einem herkömmlichen Softwareentwicklungsprojekt? Um die wesentlichen Unterschiede zu verdeutlichen, werfen wir zunächst einen Blick auf die wichtigsten Phasen im Lebenszyklus von Low Code Software.

Bei einem Projekt auf Basis von A12 entstehen Artefakte auf einer **Modell-Ebene** und einer **Code-Ebene**. Der Projektverlauf ist auf beiden Ebenen durch ein iteratives Vorgehen charakterisiert.

Die Code-Ebene ist in drei Bereiche unterteilt:

- **Low Code Basis**: Technische Komponenten und Artefakte, die Teil der A12 Plattform sind
- **Individualentwicklung**: Individuell entwickelte Komponenten, Erweiterungen und Artefakte
- **Systemintegration**: Code für die Interaktion mit externen Systemen



Konzeptphase: Echte Anwendungen als Prototypen

Ein entscheidender Unterschied zu konventionellen Softwareentwicklungsprojekten zeigt sich in der Konzeptphase. Es gibt keine klassischen Prototypen, die als Mockups und Attrappen ein erstes Gefühl für die Anwendung vermitteln sollen und danach nicht weiter nutzbar sind. Statt dessen werden erste Daten- und UI-Modelle sowie Workflows erstellt, die direkt in eine echte Anwendung kommen.

Erste Ergebnisse sind dadurch nicht nur sehr schnell sichtbar. Sie stellen als echte Anwendung eine viel konkretere Grundlage dar, um weitere Anforderungen abzuleiten. Durch das Plasma Designsystem folgen die Anwendungen auch in den frühesten Phasen einem durchdachten UI/UX-Konzept und haben ein modernes Look & Feel.

Implementierungsphase: Evolution der Modelle und Evolution des Codes

Die Implementierungsphase ist durch sukzessive Erweiterungen auf der Modell- und der Code-Ebene gekennzeichnet. Die Datenmodelle werden so lange erweitert, bis sie alle für die Anwendung relevanten fachlichen Aspekte abbilden. Die UI-Modelle und Workflows werden entsprechend angepasst, um alle nötigen Interaktionen mit den Geschäftsobjekten in die Benutzerschnittstelle zu bringen und den Anwendern einen möglichst effizienten Fluss der jeweiligen Arbeitsvorgänge zu bieten.

Auf der Code-Ebene werden Anforderungen in zwei Strängen umgesetzt. Ein Teil der Anforderungen wird mit Hilfe der Standardkomponenten von A12 umgesetzt. Die restlichen Anforderungen werden individuell projektseitig implementiert. Hier wird wiederum geprüft, inwieweit sie als Kandidaten in die Weiterentwicklung von A12 zurückfließen können.

Produktions- und Nutzungsphase: Flexibilität bei fachlichen Änderungen

Software unterliegt in der Regel erst dann keinen Änderungen mehr, wenn sie nicht mehr genutzt wird. Auch in der Produktions- und Nutzungsphase entwickeln sich die Modell- und die Code-Ebene weiter. Auf der Code-Ebene sind die Änderungen eher gering und maßgeblich durch technische Updates charakterisiert. Umfangreichere neue Anforderungen können auch weitere Implementierungsphasen bedingen. Auf der Modell-Ebene hingegen treten Änderungen typischerweise häufig auf. Die Fachlichkeit entwickelt sich weiter. Hier zeigt sich ein wesentlicher Vorteil der modellbasierten Entwicklung: Fachliche Anpassungen sind wesentlich schneller und mit geringerem Aufwand umsetzbar. Eine individuelle Ausprogrammierung entfällt. Der Fachbereich kann mit Hilfe der Modellierungswerkzeuge die Software eigenständig anpassen.

Erneuerungsphase: Austausch der Technik

Die letzte Phase des Software-Lebenszyklus ist üblicherweise die Stilllegung. Die Technik entwickelt sich rasant weiter. Bereits nach fünf Jahren gilt Enterprise Software technisch als veraltet. Mit einer Entwicklungsplattform wie A12 gibt es jedoch eine Alternative zur Stilllegung: die technische Erneuerung bzw. Evolution. Denn warum sollte man eine komplett neue Anwendung bauen, wenn die kontinuierlich gepflegten Modelle den fachlichen Kern der Software aktuell und akkurat abbilden? Die Isolierung der Fachlichkeit in den Modellen ermöglicht prinzipiell einen Austausch technischer Komponenten. Darüber hinaus wird die technische Basis durch A12-Updates kontinuierlich aktuell gehalten.

03

Planung und Organisation

Gute Planung ist die halbe Miete für die Entwicklung von Enterprise Software. Neben der Wahl einer geeigneten Projektmanagement-Methodik gilt es, ein kompetentes Team aufzustellen, Gremien zur Projektsteuerung einzurichten, und ein gemeinsames Verständnis für die Zusammenarbeit zu etablieren. mgm setzt auf bedarfsgerechte Organisationsstrukturen, die je nach Projekt individuell austariert werden.

Hinsichtlich des Projektmanagements unterscheidet sich ein A12 Projekt nicht wesentlich von einem klassischen Enterprise Software-Projekt. Bei komplexer Enterprise Software ist der Low Code-Ansatz eingebettet in klassische Entwicklungs- und Managementstrukturen, die sich im Rahmen der agilen, individuellen Softwareentwicklung bewährt haben. Dementsprechend legt mgm bei A12-Projekten einen großen Wert auf eine vollständige Projektumgebung und bedarfsgerechte Organisationsstrukturen, die je nach Projekt individuell austariert werden können.

Projekt-Scope

Ein Projekt ist ein Vorhaben mit definierten Start- und Zielkriterien zur Herstellung eines Produkts oder einer Dienstleistung mit bestimmten Ressourcen und Anforderungen. Es kann an unterschiedlichen Punkten des Lebenszyklus einer Anwendung ansetzen.
Wenn wir im Folgenden von einem **A12-Projekt** sprechen, ist damit – sofern nicht anderweitig gekennzeichnet – das Szenario einer **Neuentwicklung** gemeint. Es reicht typischerweise von der Konzeptphase bis zur Inbetriebnahme der Anwendung.
Da Enterprise Software für Langlebigkeit ausgelegt ist, finden typischerweise auch in der Nutzungsphase weitere Projekte statt. Sie fokussieren sich in der Regel auf bestimmte Erweiterungen oder Integrationen mit anderen Systemen.

Consulting, Change Management & Kommunikation

Der Erfolg einer Software hängt entscheidend von ihrer Akzeptanz bei den Nutzenden ab. Selbst die technisch beste Lösung, die sämtliche formalen Anforderungen erfüllt, kann scheitern, wenn sie von den Anwendern nicht akzeptiert oder verstanden wird. A12-Anwendungen wirken dem von Anfang an entgegen, indem sie die verantwortlichen Fachbereiche mit dem Low Code-Ansatz aktiv in die Entwicklung einbeziehen. Es macht einen gewaltigen Unterschied, eine komplett extern entwickelte Anwendung vorgesetzt zu bekommen, oder Teile davon selbst entwickelt zu haben und möglicherweise langfristig eigenständig zu verantworten. Change Management und Kommunikation sind in A12-Projekten konsequent auf diese neuartige Entwicklungskonstellation ausgerichtet.

Grundlagen für den Wandel schaffen

Zu Beginn eines Projekts werden die organisatorischen Veränderungen analysiert, die durch die neue Anwendung entstehen. Welche Prozesse ändern sich? Welche Stakeholder sind betroffen? Und wie können bestehende Strukturen unterstützt oder angepasst werden? Besonders wichtig ist dabei die Frage, welche fachlichen Aspekte und Workflows künftig in A12-Modellen abgebildet werden und wie die verantwortlichen Fachbereiche dabei eingebunden werden. Gerade im öffentlichen Sektor können zum Beispiel Fachexpert:innen, die für den digitalen Vollzug von fachspezifischen Gesetzen verantwortlich sind, in völlig anderen Abteilungen oder Behörden sitzen. Hier ist zu prüfen, wie sie langfristig in den Entwicklungs- und Wartungsprozess der Software einbezogen werden können, um eine Ende-zu-Ende-Digitalisierung zu ermöglichen.

Mitarbeitende befähigen und einbeziehen

Durch den Einsatz der Low Code-Werkzeuge von A12 werden betroffene Mitarbeitende frühzeitig in den Wandel einbezogen. Dies fördert ihre Akzeptanz für neue Arbeitsweisen und hilft dabei, Widerstände aktiv zu adressieren. mgm bietet zielgruppenspezifische A12 Trainingspfade an, um Nutzende zu befähigen und sie entlang des Entwicklungsprozesses kontinuierlich zu schulen und einzubinden. Die Mitarbeitenden werden nicht nur mit den neuen Werkzeugen und Systemen vertraut gemacht, sondern zu aktiven Mitgestaltern der Veränderung. Dadurch entsteht eine Kultur der Offenheit und des gemeinsamen Fortschritts.

Entwicklungsstände im Rahmen der Kommunikation bereitstellen

Eine klare und zielgerichtete Kommunikationsstrategie bildet die Grundlage für den Projekterfolg. Sie schafft Transparenz, informiert frühzeitig über Veränderungen und bietet Betroffenen die Möglichkeit, sich einzubringen. Unterschiedliche Stakeholder-Gruppen benötigen dabei individuell zugeschnittene Kommunikationsformate, die ihre Perspektiven und Bedürfnisse berücksichtigen. Die deutlichste Sprache spricht aber die Anwendung selbst. Hier bietet A12 den Vorteil, dass lauffähige Entwicklungsstände sehr schnell bereitstehen und den Stakeholdern über Testplattformen zugänglich gemacht werden können. Sie können die Anwendung in einem frühen Stadium testen und mit Feedback an das Entwicklungsteam die weitere Entwicklung beeinflussen.

A12-Enablement / Trainings

Um im Projektteam das notwendige Wissen für die Arbeit mit der Entwicklungsplattform A12 sicherzustellen, bietet mgm verschiedene Schulungen und zugehörige Zertifizierungen an. So stellen wir sicher, dass auch die Teammitglieder des Auftraggebers und der Partner sehr schnell produktiv mit den Werkzeugen und technischen Komponenten von A12 arbeiten können.

Das A12 Enablement besteht aus Trainingspfaden mit Basic- und Specialist-Trainings für die unterschiedlichen Rollen und wird durch optionale Schulungen (zur Barrierefreiheit) ergänzt.

Darüber hinaus umfasst das Trainingsprogramm von mgm eine Reihe von Security-Trainings. Im Rahmen von A12-Projekt sind vor allem folgende Angebote relevant:

- Best Practices für sichere Webanwendungen
- Secure Coding für Java
- DevSecOps: Sicherheit in der CI/CD Pipeline
- Best-Practices der Cloud Security

Abbildung 2 – *Der A12 Enablement Path umfasst eine Reihe von Schulungen und Zertifizierungen für verschiedene Rollen*

Unterstützung des Projektteams durch Querschnittsteams

Neben der Onboarding-Unterstützung durch das Trainingsteam und ausgehend von mehreren langjährigen Großprojekten haben sich innerhalb von mgm projektübergreifende Organisations- und Support-Strukturen etabliert. Sie helfen dabei, die Erfahrungen und Best Practices aus erfolgreichen Enterprise-Projekten auch in neue A12-Projekte einfließen zu lassen.

Neben Projektteams setzt mgm auf eine Reihe von Querschnittsteams, die in zentralen Disziplinen des Software Engineerings wie technische Projekt-Infrastruktur (TPI), Security, Data Science & AI, Qualitätssicherung und UI/UX spezialisiert sind. Sie unterstützen die Projektteams punktuell bei Bedarf, stehen bei spezielleren technischen Fragen zur Verfügung und sind ein wichtiger Garant für den Projekterfolg.

Neben den Querschnittsteams ist innerhalb von mgm außerdem das Professional Services (PS) Team etabliert (siehe S. 24). Es fungiert als Schnittstelle zwischen den Projektteams und dem A12 Team, das die Plattform weiterentwickelt und betreut.

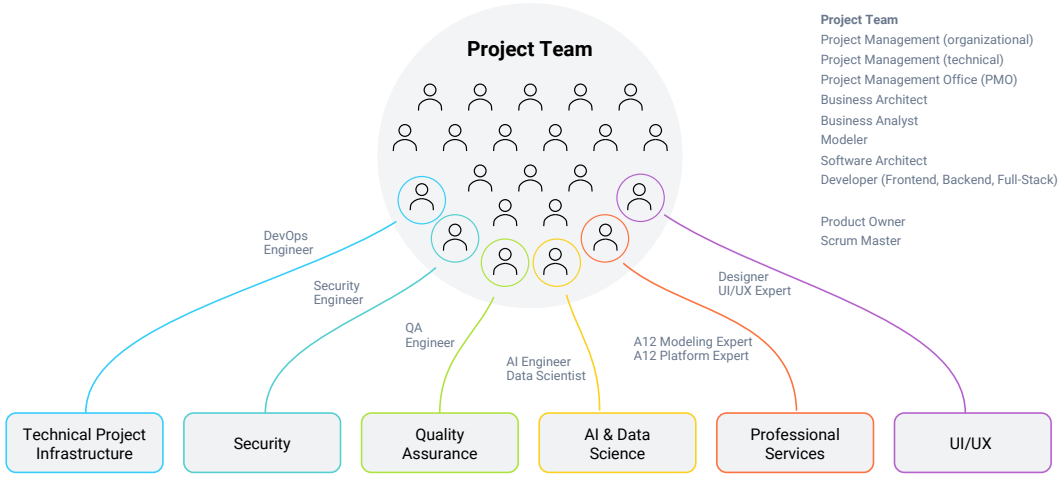


Abbildung 3 – *Spezialist:innen aus den Querschnittsteams bringen als Mitglieder des Projektteams bedarfsgerecht ihre Expertise ein*

Partnerschaftliche Zusammenarbeit

Das wesentliche Ziel eines A12-Projekts ist die effektive Umsetzung der Anforderungen in einer effizient verwendbaren, qualitativ hochwertigen Lösung. Die enge Zusammenarbeit mit dem Auftraggeber ist eine wichtige Voraussetzung, um dieses Ziel zu erreichen. mgm versteht sich als flexibler Entwicklungspartner, der sich in verschiedenen Konstellationen bedarfsgerecht einbringt. Das Spektrum reicht von Projekten, in denen der Auftraggeber und mgm jeweils mit eigenen Entwicklungsteams ein A12-Projekt gemeinsam umsetzen, bis hin zu Projekten, in denen mgm die komplette Entwicklungsverantwortung übernimmt.

Wenn mgm das A12-Projekt verantwortet, orientiert sich der minimale Umfang der Mitwirkung des Auftraggebers am Reifegrad der Detaillierung der Anforderungen zu Projektbeginn. Je mehr Details noch gemeinsam zu erarbeiten sind und je stärker der Fokus auf die Optimierung der abzubildenden Geschäftsprozesse gerichtet werden kann oder soll, desto umfangreicher ist die Mitwirkung auszugestalten.

Durch die Nutzung von A12 kann der Auftraggeber optional bei der Modellierung mitwirken. Folgende Konstellationen sind möglich und bieten sich je nach Projektphase und Modellierungsumfang an:

- mgm übernimmt die komplette Modellierung. Dies ist vor allem zu Beginn des Projekts empfehlenswert, da erfahrene Modellierer den initialen Aufbau der Anwendung beschleunigen können.
- Der Auftraggeber und mgm teilen sich die Modellierungsaufgaben.
- Der Auftraggeber verantwortet eigenständig die fachliche Modellierung. Diese Konstellation ist besonders sinnvoll, wenn die Anwendung mit vielen und regelmäßigen fachlichen Änderungen konfrontiert ist - zum Beispiel aufgrund von häufigen gesetzliche Änderungen.

Im Fall der Einbindung schult mgm die Fachexperten/Business Analysten des Auftraggebers in der Erstellung von A12-Modellen. Diese Aufgabenteilung ist sehr sinnvoll, weil die Modelle immer die Fachlichkeit der Geschäftsdomäne abbilden und die Fachexperten des Auftraggebers diese Geschäftsdomäne am besten kennen. mgm bietet bei dieser Aufgabenverteilung Support bei Modellierungsfragen und verantwortet bei Bedarf die gesamten Entwicklungs- und Technik-Aufgaben. Im Bereich der Steuer hat sich dieses Vorgehen bewährt. Hier definieren fachliche Teams eigenständig Felder, Plausibilitätsregeln und Formulare auf Basis der rechtlichen Vorgaben.

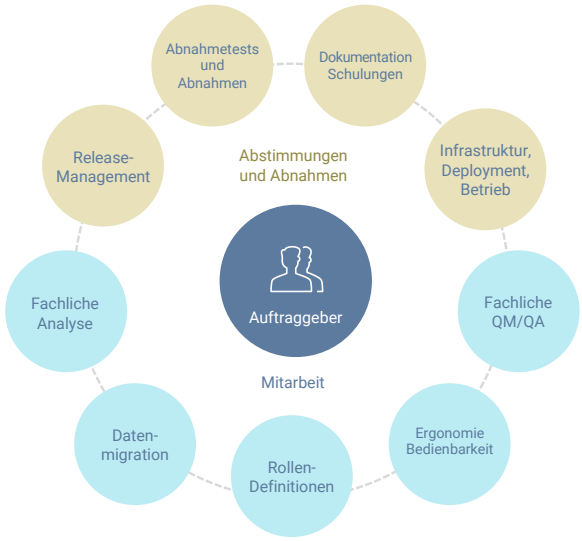


Abbildung 4 – *Beispiele für Mitwirkungen und Übergabepunkte der Zusammenarbeit*

Zeit- und Meilensteinplanung

Durch den Einsatz von Low-Code lassen sich drastische Geschwindigkeitsvorteile bei der Softwareentwicklung erzielen. Die erste lauffähige Version einer Anwendung und das Minimum Viable Product (MVP) einer Business Applikation stehen sehr schnell bereit. Wie hoch die Zeitersparnisse sind, hängt letztlich davon ab, wie intensiv das Projekt auf Standardkomponenten zurückgreift. Je mehr Standardlösungen von A12 zum Einsatz kommen, desto straffer kann die Zeit- und Meilensteinplanung ausfallen.
In einem voll agilen A12 Projekt werden neue Funktionen und Kriterien direkt teamübergreifend und qualitätssteigernd implementiert. Dabei entwickeln der Auftraggeber und mgm zu Beginn einen groben Fahrplan mit Priorisierungen, welcher kontinuierlich proaktiv auf wechselnde Anforderungen angepasst wird. Basierend auf langjähriger Erfahrung erachtet mgm das Plan – Do – Check –Act Prinzip (PDCA) in Kombination mit einem strukturierten und durchgehend priorisierten Backlog als den passendsten iterativen Prozess. Die frühzeitige Erstellung eines Sprint Burndowns ermöglicht es, anhand der Vergabe von Story Points einzuschätzen, wie viele Sprints bis zum Abschluss der jeweiligen Aufgaben notwendig sein werden. Somit kann die reduzierte Planbarkeit eines agilen Projektes ausgeglichen und ein realitätsnaher Zeit- und Meilensteinplan erstellt werden.

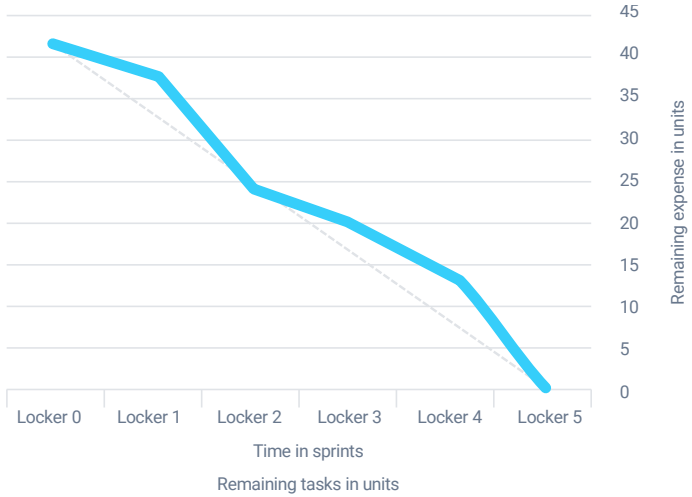


Abbildung 5 – *Beispiel für einen idealen Sprint Burndown*

04

Kick-Off - Booster zum Projektstart

Gerade zu Beginn eines Projekts werden wichtige Weichen gestellt, die den gesamten Projektverlauf beeinflussen. mgm hat einen strukturierten Kick-Off Prozess entwickelt, der Projekte schnell und gezielt in die richtigen Bahnen lenkt. Durch die Einbindung von Expertenteams berücksichtigen wir von Anfang an zentrale Themen wie technische Projektinfrastruktur, Datenschutz, KI, Security, sowie Change & Kommunikation.

Einrichtung der Projektinfrastruktur mit Expertenteams

Um neuen Projekten auf Basis von A12 zu einem reibungslosen Start zu verhelfen, hat mgm einen strukturierten Kick-Off-Prozess entwickelt. Die Projektleitung bekommt von Beginn an wertvolles Feedback von erfahrenen Fachleuten, um die Weichen des Projekts richtig zu stellen. In den Prozess sind sowohl alle spezialisierten Querschnittsteams von mgm sowie das A12-PS-Team eingebunden. Er soll unter anderem Klarheit verschaffen,

- welche Infrastruktur das Projekt benötigt,
- wie die Software dem Kunden bereitgestellt wird, und
- wie Arbeitsabläufe im Projekt organisiert sind (JIRA Workflow, agile Entwicklung, etc.).

Mit Hilfe eines Fragebogens werden systematisch alle wesentlichen Bereiche abgeklopft – von der ISMS Risikoeinschätzung und der QS-Methodologie über Datenschutzfragen bis hin zu sicherheitsrelevanten Aspekten.

Darauf aufbauend richtet mgm die Infrastruktur sowie damit verbundene Prozesse ein - vom Ticketsystem über die Bereitstellung eines Wikis für die Dokumentation bis zur Konfiguration einer selbstentwickelten Lösung für die Zeiterfassung und das Controlling der Projektaufwände.

Darüber hinaus wird das Projekt innerhalb von mgm in die A12 Community aufgenommen. Es bekommt dadurch zum Beispiel die Möglichkeit, über die Teilnahme an Umfragen die Weiterentwicklung der Plattform mitzuprägen.

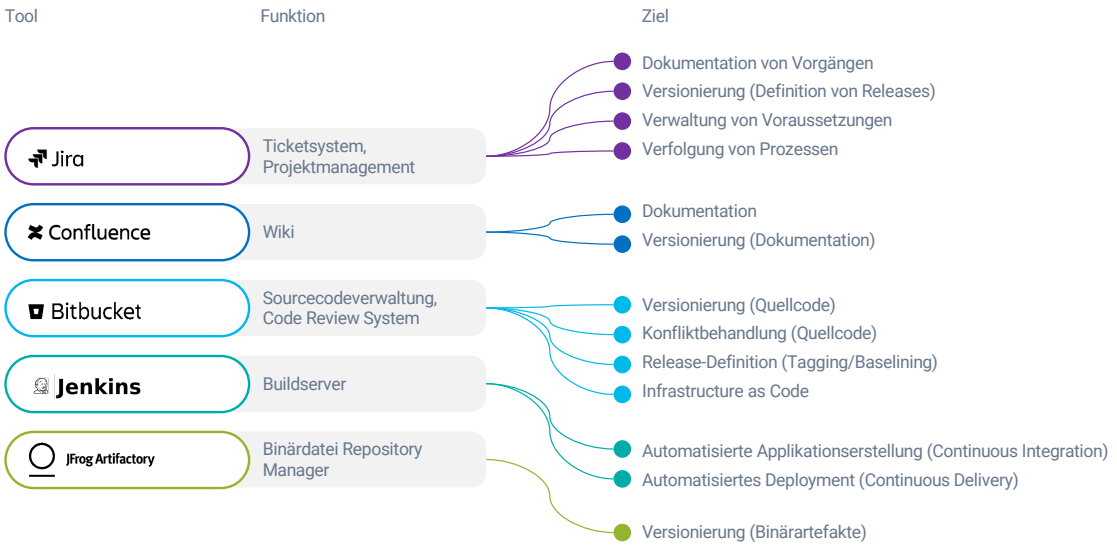


Abbildung 6 – *Standardmäßig eingesetzte Tools und ihre Kernfunktionen*

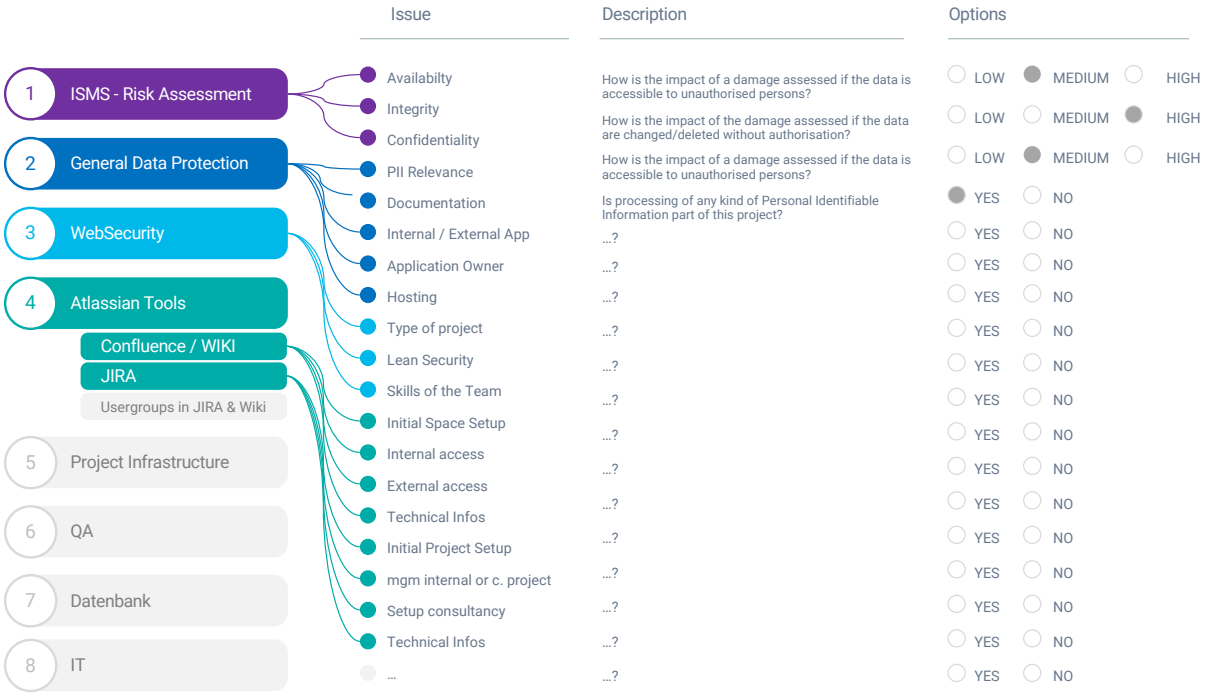


Abbildung 7 – *mgm bietet eine Reihe von Templates, um in komplexen A12-Projekten systematisch wesentliche Herausforderungen zu erfassen*

Technische Initialisierung eines A12-Projekts

In A12-Projekten sind durch den Einsatz der A12-Plattform eine Reihe von Rahmenbedingungen gegeben. Der Low Code-Ansatz impliziert, dass neben den benötigten Anteilen an Individualentwicklung möglichst weite Teile der Anwendung modelliert werden. Um diese Entwicklungsstränge von Anfang an richtig zu verzahnen, empfiehlt sich auch in agilen Projekten eine Grobplanung des architektonischen und fachlichen Konzepts der Anwendung. Sie legt initial fest, wie entwickelte und modellierte Artefakte im Projektverlauf ineinanderfließen.

Ein dreiköpfiges Team legt die richtigen Weichen

Wenn in einem Großprojekt alle involvierten Personen von Anfang an gleichzeitig loslaufen, ist die Wahrscheinlichkeit sehr groß, dass sie zunächst in unterschiedliche Richtungen rennen. Um dies zu verhindern, setzten wir initial ganz bewusst auf ein sehr kleines Kernteam. Es besteht typischerweise aus drei projekterfahrenen Personen, die fundierte Erfahrung aus folgenden Bereichen mitbringen:

- Architektur
- Fachliche Modellierung / Business Analyse / A12-Modellierung
- Frontend Development / Backend Development

Idealerweise haben sich diese drei Personen schon vor der Initialisierungsphase in die fachlichen Zusammenhänge der zu erstellenden Anwendung eingearbeitet. Aufbauend auf diesem Wissen analysieren sie eingehend die Anforderungen und entwickeln ein abgestimmtes Konzept sowie eine gemeinsame technische Basis.

Unterstützt wird das Initialisierungsteam durch die spezialisierten Querschnittsteams von mgm. So wird sichergestellt, dass von Anfang an wichtige Aspekte rund um Security, Data Science & AI, Qualitätssicherung, UI/UX-Design, Barrierefreiheit und technische Infrastruktur berücksichtigt werden.

Das Initialisierungsteam bezieht Vertreter aus den Querschnittsteams punktuell bei wichtigen Entscheidungen mit ein. Zum Ende der Initialisierungsphase erfolgt zudem eine Prüfung und Abnahme des Konzepts und der technischen Basis durch die Querschnittsteams.

Abgestimmtes Fach- und Technik-Konzept beantwortet wesentliche Fragen

Aus konzeptioneller Sicht entstehen während der Initialisierungsphase folgende Ergebnisse:

- **Architekturkonzept**
  Ein ausreichend detailliertes Architekturkonzept definiert, welche A12-Komponenten zum Einsatz kommen sowie welche neu zu entwickelnden Komponenten benötigt werden, wie sie geschnitten sind und wie sie miteinander interagieren.
- **Fachliches Modell**
  Das fachliche Modell beschreibt übergreifend, wie die Anwendung fachliche Inhalte abbildet – sowohl mit den Mitteln der A12-Modellierung als auch auf der Code-Ebene. Das Modell beantwortet u.a. die Frage, welche fachlichen Inhalte künftig mit den A12-Werkzeugen erstellt werden können und wie sich die A12-Modelle in die übergeordnete Fachmodellierung der gesamten Anwendung einfügen.

Diese Konzepte bilden eine wichtige Basis für die Entwicklung und schaffen idealerweise ein Fundament, das über den gesamten Projektverlauf Bestand hat. Fakt ist aber auch: Es können in der Initialisierungsphase nicht alle Fragen abschließend beantwortet werden. Ebenso wichtig wie das Treffen richtungsweisender Entscheidungen ist das bewusste Auslassen von Entscheidungen, die zu Projektbeginn noch nicht getroffen werden müssen.

> *Die Initialisierungsphase eines A12-Projekts ist extrem spannend und legt wichtige Weichen für den gesamten Projektverlauf. Es geht darum, mit einem kleinen Team die Anwendung richtig aufzugleisen – und zwar so, dass alle Teilteams und Disziplinen von Anfang an klar verzahnt zusammenarbeiten. Das ist sehr herausfordernd, macht aber enorm viel Spaß. Gerade die Einbindung der Low-Code-Fähigkeiten von A12 in komplexe Individualsoftware bietet ein riesiges Potenzial, erfordert aber auch ganz neue Denkweisen. Jedes Projekt bringt zudem seine eigenen technischen und fachlichen Herausforderungen mit sich.*
>
> Volker Schmitt
> Senior Architekt bei mgm

Deploybarer Durchstich der Anwendung als Startpunkt

Ergänzend zur Erstellung der abgestimmten Konzepte hat das Initialisierungsteam die Aufgabe, darauf aufbauend den Grundstein für eine start- und deploybare Anwendung zu legen und projektspezifische Entwicklungs- und Testumgebungen aufzusetzen. Ein wesentliches Ziel besteht darin, die interdisziplinäre Zusammenarbeit zwischen Business Analyse und Modellierung, Frontend- und Backendentwicklung sowie technischer Projektinfrastruktur zu ermöglichen und zu fördern.

Technisch bedeutet dies, ein Git-Repository auf Basis der A12 Templates zu erstellen. Um das Projekt zu bauen und zu starten, werden Java, Node.js, Gradle und Docker benötigt. Hinzu kommen eine IDE sowie die A12-eigenen Tools zur Modellierung – allen voran der Simple Model Editor (SME). A12 bietet dafür bereits Grundlagen wie das A12 Project Template, den Helm A12 Stack und die A12 Build and Deployment Pipelines. Das Template enthält neben Quellcode für Client und Server auch bereits fertig modellierte Beispielanwendungen und weitere benötigte Komponenten wie Keycloak, die Workflow-Engine oder eine PostgreSQL-Datenbank.

Die zu erstellende technische Basis bildet die konzipierte Architektur und das fachliche Modell ab. Sie stellt sicher, dass einzelne Teilbereiche von Anfang an technisch integrierbar sind. Dazu gehört auch die direkte Einbindung von A12-Modellen, die nicht vom Entwicklungsteam verantwortet werden. Alle Teammitglieder arbeiten ab Tag 1 auf einem gemeinsamen Repository. So wird verhindert, dass Silos entstehen und bestimmte Teile der Anwendung erst im Projektverlauf zusammengeführt werden müssen.

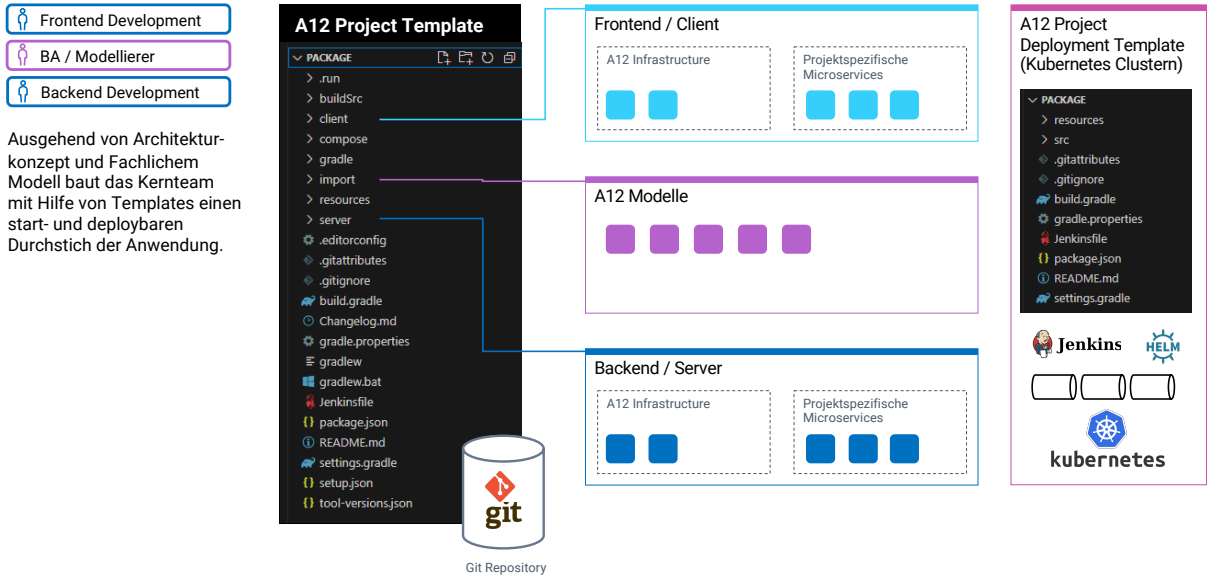


Abbildung 8 – *Templates und eine standardisierte Struktur beschleunigen den Start von A12-Projekten*

Organisatorische Initialisierung des Projekts

Parallel zur technischen Initialisierung findet eine organisatorische Initialisierung des Projekts statt. Hierbei arbeiten vorrangig Projektleitung und PMO zusammen. Sie nutzen die bereits eingerichtete Projektinfrastruktur, um wesentliche Leitplanken für das Team bereitzustellen. Dazu gehören vor allem folgende Aspekte:

- **Zielbeschreibung**
  Ein gemeinsames Zielbild ist eine wesentliche Voraussetzung für eine zielgerichtete Entwicklung. Es dient in Form einer Produktvision im agilen Projekt als übergeordnete Orientierungshilfe. Eine gut formulierte Zielbeschreibung ist langfristig gültig und hilft dem Team, sich auf das Wesentliche zu konzentrieren.

- **Checklisten für On- und Offboarding**
  In keinem langjährigen Enterprise-Softwareprojekt ist das Team von Anfang bis Ende fix. Es gibt immer wieder Momente, in denen neue Teammitglieder hinzukommen oder bestehende das Team verlassen. In solchen Fällen helfen Checklisten dabei, neue Teammitglieder schneller einzuführen sowie Abgänge und Übergaben klar zu strukturieren.

- **Organigramm**
  Wie setzt sich das Team zusammen? Wer übernimmt welche Rollen und Aufgaben? Welche Rechte und Pflichten sind für alle Teammitglieder die Grundlage der gemeinsamen Arbeit? All diese Fragen werden bei der Erstellung eines Organigramms beantwortet, das kontinuierlich auf dem aktuellen Stand gehalten wird.

- **Kommunikationsmatrix**
  Eine Kommunikationsmatrix macht transparent, welche Regeltermine vorgesehen sind und wer an ihnen teilnimmt. Darüber hinaus zeigt es explizit, welche Berichtswege und Kommunikationsketten innerhalb des Teams und in der Zusammenarbeit mit dem Auftraggeber und Partnern vereinbart sind.

- **Dokumentationsstruktur**
  Die Konzeption einer ersten Struktur für die Dokumentation erleichtert es dem Team, Zwischenstände und Projektergebnisse festzuhalten. Außerdem bleibt dadurch transparent, in welchen Bereichen die Dokumentation womöglich noch Lücken aufweist.

- **Roadmap**
  Die Erstellung einer Roadmap hilft dabei, alle bereits vorhersehbaren nötigen Arbeiten und Meilensteine zeitlich darzustellen. Sie unterstützt bei der Planung und der Setzung von Prioritäten.

- **Teamkalender**
  Ein Teamkalender wird aufgesetzt, um eine gemeinsame Sicht auf die Projekttermine zu etablieren. Darüber hinaus werden hier beispielsweise auch Abwesenheiten von Teammitgliedern oder weiteren Stakeholdern eingetragen.

Team an Bord holen auf Basis der erarbeiteten Grundlage

In Summe zielt die Initialisierungsphase darauf ab, zum Projektbeginn grundlegende Entscheidungen zu treffen und Arbeitspakete zu definieren, die möglichst viele Abhängigkeiten auflösen. Anschließend werden alle beteiligten Teilteams an Bord geholt - idealerweise in einem gemeinsamen Team-Kickoff vor Ort. Sie arbeiten von Anfang an auf dem in der Initialisierungsphase entstandenen Grundgerüst der Anwendung und profitieren von dem eingerichteten organisatorischen Rahmen.

Schnittstelle zum A12-Team: Professional Services

Professional Services (PS) bildet innerhalb von mgm die Schnittstelle zwischen den Kundenprojekten und dem A12-Team, das die Weiterentwicklung der Plattform verantwortet. Das PS-Team kümmert sich speziell um die Bedürfnisse der Kundenprojekte und steht ihnen als direkter Ansprechpartner bereit:

Business Professional Services (BPS) ist spezialisiert auf fachliche Anforderungen. Das Team unterstützt Business Analysten und Fachexperten von Kundenprojekten bei der Modellierung fachlicher Inhalte mit A12. Es verhandelt Prioritäten bei Anforderungen. Darüber hinaus bietet es Trainings und Tutorials an, treibt den Aufbau einer Wissensbasis voran und erstellt Standards und Templates für Fachkonzepte, Dokumentation und Qualitätssicherung.

Technical Professional Services (TPS) ist spezialisiert auf technische Anforderungen. Das Team agiert als Ansprechpartner für das Entwicklungsteam. Es managt Bug Reports und koordiniert A12 Patch Releases für Kundenprojekte. Darüber hinaus unterstützt TPS bei der Qualitätssicherung aus technischer Perspektive und stellt technische Dokumentation und Tutorials für Developer bereit.

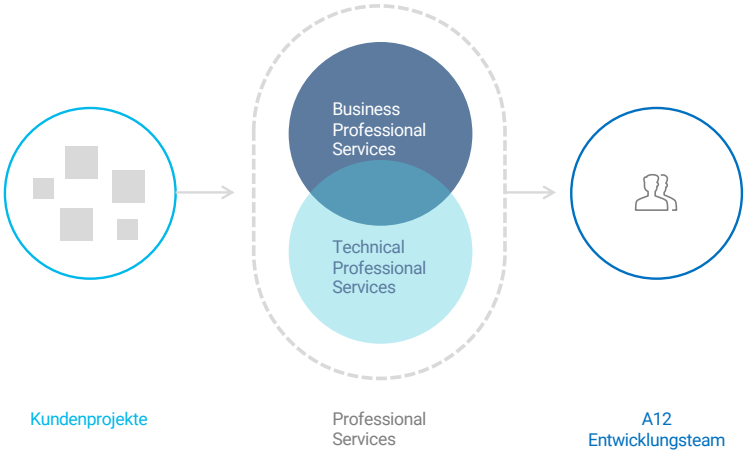


Abbildung 9 – *Die Professional Services-Teams von mgm stehen den A12-Projekten als Ansprechpartner bei Fragen rund um die Plattform bereit und stehen in engem Austausch mit dem A12-Plattform-Team*

Templates für A12-Projekte

Project Template

Das A12 Projekt-Template bildet den technischen Startpunkt für A12-Projekte. Es bietet ein standardisiertes Setup für Anwendungen, die alle wesentlichen Komponenten von A12 nutzen – Client, Data Services und UAA. Es greift Best Practices aus mehreren erfolgreich durchgeführten A12-Projekten auf und bietet eine gut durchdachte, minimale, leicht erweiterbare Konfiguration out-of-the-box.

Templates für Cluster-Umgebungen

Für den Betrieb von Geschäftsanwendungen entwickeln sich Cluster-Umgebungen in Cloud-Infrastrukturen zum de-facto-Standard. A12 bietet deshalb ein standardisiertes und praxiserprobtes Setup für die Entwicklung und den Betrieb von A12-Anwendungen in Kubernetes-Clustern an. Es ermöglicht in Kombination mit dem Project Template das sehr schnelle Aufsetzen von cluster-basierten Entwicklungs- und Produktionsumgebungen. Vordefinierte, vielseitig einsetzbare Build & Deployment Pipelines (siehe Kapitel Continuous Delivery & Operations) automatisieren dabei zahlreiche Prozesse und sorgen dafür, dass sich das Entwicklungsteam auf die eigentliche Wertschöpfung konzentrieren kann und zu Projektbeginn keine Zeit für Konfiguration verliert.

App Factory

Die A12 App Factory ermöglicht die schnelle und automatisierte Bereitstellung von A12-Umgebungen - sprich: der technischen Infrastruktur für die Ausführung von A12-Anwendungen. Sie ist für Entwicklungs-, Tests- und Produktionszwecke ausgelegt, stützt sich auf das A12 Project Template und unterstützt das Deployment in verschiedene Zielumgebungen.

GetA12 - Alle A12-Ressourcen an einem Ort

Die Plattform GetA12 (https://geta12.com/) bildet den zentralen Einstiegspunkt für Projekte, die Software auf Basis von A12 entwickeln. Der Zugang ist beschränkt auf registrierte Nutzende, die in A12-Projekten aktiv sind.

GetA12 stellt unter anderem folgende Ressourcen bereit:

- **Modellierungsumgebung** In GetA12 finden Business Analyst:innen den jeweils aktuellsten Stand der Modellierungswerkzeuge. Sie können dort den jeweiligen Installer downloaden für Windows, MacOS und Ubuntu.
- **Dokumentation** Von der Erklärung grundlegender Konzepte über die Funktionsweise von Modellierungstechniken bis hin zu den APIs der A12 Produkte – die umfangreiche Dokumentation enthält zahlreiche wichtige Informationen und ist das zentrale Nachschlagewerk der A12 Plattform.
- **Release-Übersicht** Neben einer übersichtlichen Darstellung der Release-Linien und der darin verwendeten Versionen einzelner A12 Produkte enthält GetA12 Changelogs und die Ankündigung neuer Releases.
- **A12 Discourse** Die Austauschplattform A12 Discourse fungiert gleichzeitig als Forum und als Wissensbasis für die A12 Community. Sie fördert den projektübergreifenden Wissensaustausch.
- **A12 Artifactory** Für das Entwicklungsteam sind in der A12 Artifactory stets alle aktuellen technischen Artefakte für den Einsatz der A12 Plattform abrufbar. Dazu gehören Repositories für Docker, Helm, Maven und npm.
- **Trainingskatalog** Für alle wesentlichen Rollen innerhalb eines A12 Projekts – von der Front- und Backend-Entwicklung über die Modellierung bis hin zu DevOps und QS – listet der Trainingskatalog auf, welche Schulungen und Tutorials bereitstehen.

05

Ergebnisorientierte Entwicklung

mgm definiert seinen Erfolg durch die Anzahl der in Produktion gebrachten Systeme, die langfristig zum Geschäftserfolg der Kunden beitragen. In der Umsetzungsphase von Softwareprojekten auf Basis von A12 behalten wir deshalb konsequent den Wert für den Nutzer und den Wert für das Geschäft im Fokus.

Anforderungen umsetzen: Modellierung oder Programmierung?

In einem A12-Projekt können Anforderungen grundsätzlich auf drei Arten umgesetzt werden:

- Modellierung mit Hilfe der A12-Editoren
- Implementierung mit Standardkomponenten der Plattform
- Implementierung in Form einer individuellen Lösung

Modellierung von Fachlichkeit

Die mittels Modellierung umsetzbaren Anforderungen lassen sich in der Regel schnell und einfach identifizieren. Ausschlaggebend ist der Modellierungsumfang der Plattform. Mit A12 sind nahezu alle Aspekte rund um die Fachlichkeit modellierbar – von Datenmodellen über die Modellierung von Formularen und Workflows bis hin zum fachlich getriebenen Aufbau der Anwendung.

Die Entscheidungen, welche Anforderungen das Team mit Hilfe von A12 Standardkomponenten oder individuell umsetzt, sind meist komplexer. Sie setzen fundierte Kenntnisse der technischen Komponenten von A12 voraus.

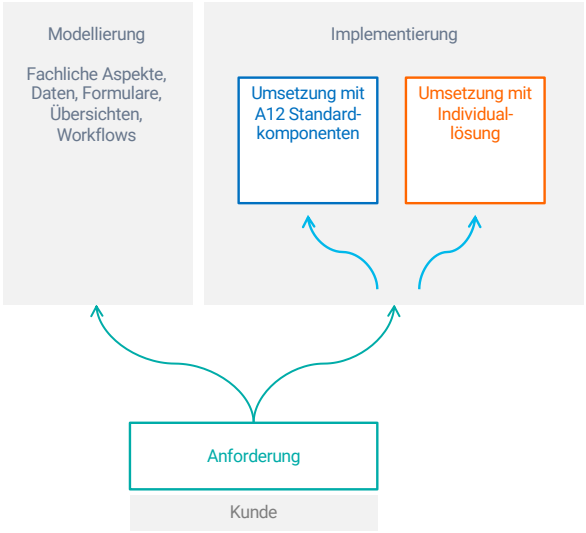


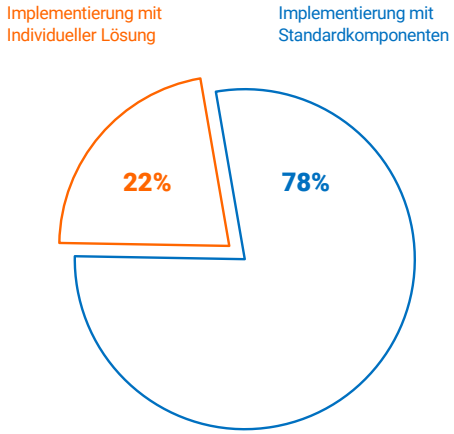
Abbildung 10 – *Kategorisierung von Anforderungen*

Umsetzung mit Standardkomponenten vs. Individualimplementierung

Gerade zu Beginn eines A12-Projekts stellen sich, sofern das Projetteam nicht bereits über umfassende Erfahrung mit der A12-Plattform verfügt, immer wieder die Fragen: Lässt sich eine bestimmte Anforderung mit den Bordmitteln von A12 umsetzen? Oder muss das Team eine individuelle Lösung implementieren, um die Anforderung umzusetzen?

Als modellbasierte Entwicklungsplattform ist A12 explizit für Erweiterungen und Anpassungen durch individuelle Softwareentwicklung ausgelegt. Die APIs der einzelnen technischen Komponenten stellen dafür zahlreiche Extension Points bereit, mit denen sich eigener Code einklinken lässt bzw. sich eigene Lösungen realisieren lassen. Grundsätzlich gilt aber auch: Je stärker ein Projekt auf Standardkomponenten von A12 setzt, desto geringer sind die Implementierungsaufwände. Um sicherzugehen, dass das Entwicklungsteam die Möglichkeiten der A12-Plattform ausreizt und keine unnötigen individuellen Lösungen realisiert, unterstützt das TPS-Team bei Umsetzungsfragen. Es verfügt über umfassendes Fullstack Know-how der A12 Plattform und lotet gemeinsam mit dem Projektteam aus, welche Anforderungen auf welche Weise mit aktuellen und kommenden A12 Features abgedeckt werden können und welche projektseitigen Individuallösungen nötig sind.

Der Anteil der Anforderungen, die mit A12-Standardkomponenten oder individuell implementiert werden, ist von Projekt zu Projekt unterschiedlich. Das Projektteam sollte dabei stets den Aufwand individueller Lösungen ihrer Bedeutung für die Nutzung der Software gegenüberstellen. Ist für ein bestimmtes Feature wirklich eine individuelle Lösung nötig? Oder erfüllt nicht doch eine Standardkomponente den wesentlichen Zweck? Hier gilt es immer wieder, die eigenen Vorstellungen zu hinterfragen und von Fall zu Fall auch Kompromisslösungen einzugehen. Nur so lässt sich der Produktivitätsvorteil des Low Code-Prinzips auch ausnutzen. Idealerweise schwankt der Anteil individuell implementierter Lösungen für Anforderungen zwischen 5 und 25 Prozent.



Idealerweise schwankt der Anteil individuell implementierter Lösungen für Anforderungen zwischen 5 und 25 Prozent.

UI/UX & Barrierefreiheit

Auch von Enterprise Software erwarten User eine erstklassige Bedienbarkeit. Der Anspruch an ein stimmiges und barrierefreies Nutzererlebnis trifft auf eine hohe inhaltliche Komplexität und die typischerweise große Informationsdichte von Geschäftsanwendungen. mgm hat deshalb ein eigenes Designsystem entwickelt: Plasma. Es sorgt zusammen mit einem Leitfaden für Barrierefreiheit für ansprechende und gleichzeitig effektiv nutzbare Software.

Plasma – UI/UX-Standard

Die erzeugten Weboberflächen einer A12-Anwendung sind HTML5-konform, responsive und cross-browser-kompatibel für alle gängigen modernen Webbrowser.

Auf der Client-Seite ermöglicht A12 den schnellen Aufbau modularisierter Front-Ends. Das Client Framework baut auf dem etablierten React/Redux-Ökosystem auf, integriert A12-UI-Komponenten wie Engines und Widgets und ruft die A12-Services im Backend über HTTP/REST-Endpunkte auf.

Zur Gestaltung der Weboberflächen der Anwendung setzen wir Widgets ein. Widgets sind wiederverwendbare UI-Komponenten, die den Plasma-Design-Konventionen und UX-Konzepten folgen. Sie unterstützen Geschäftsanwendungen, die auf Desktops, Tablets und Smartphones mit Tastatur-, Maus- und Touch-Eingabe laufen. Die Komponenten bieten eine einfach zu verwendende, gut dokumentierte, stark typisierte API und sind individuell erweiterbar und anpassbar.

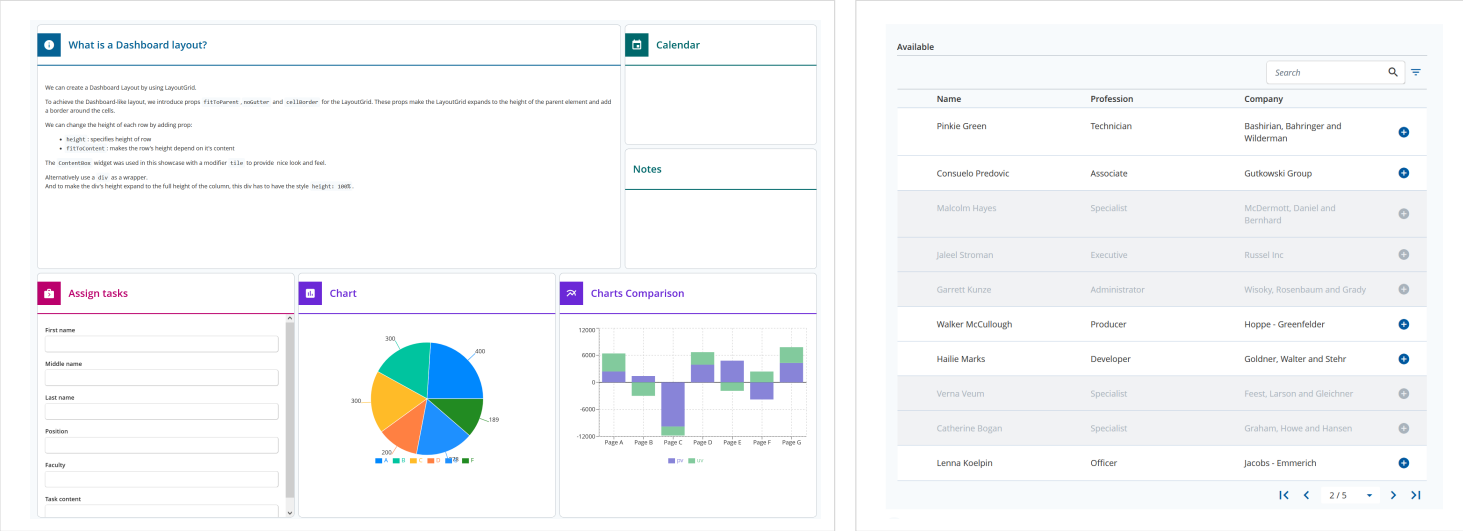


Abbildung 11 – *Das Plasma-Designsystem bietet zahlreiche Lösungsmuster für die Gestaltung von Enterprise-Anwendungen*

Barrierefreiheit - Leitfaden für Projekte

Die Bedeutung von Barrierefreiheit - auch abgekürzt als A11Y, eine Kurzform von Accessibility - nimmt in Web-Anwendungen kontinuierlich zu. Das zeigt nicht zuletzt die verschärfte Gesetzeslage: Bereits seit einigen Jahren sind öffentliche Stellen innerhalb der EU laut der Richtlinie 2016/2102 dazu verpflichtet, Websites und mobile Anwendungen barrierefrei zu gestalten. Ab 2025 müssen laut dem European Accessibility Act bis auf wenige Ausnahmen alle Websites und Web-Anwendungen barrierefrei sein.

Wir empfehlen grundsätzlich, Barrierefreiheit in Projekten zu berücksichtigen, um auch eingeschränkten Nutzer:innen einen diskriminierungsfreien Zugang zu gewährleisten. Bei der Entwicklung barrierefreier Web-Anwendungen mit A12 richtet sich mgm nach den Kriterien des BITV-Tests und den Vorgaben der WCAG.

Viele Funktionen rund um die Barrierefreiheit werden plattformseitig von A12 bereitgestellt und funktionieren out-of-the-box. Die Bausteine der Benutzeroberfläche von A12 wie Widgets und Engines werden kontinuierlich auf Anforderungen an die Barrierefreiheit überprüft und entsprechend weiterentwickelt. Nichtsdestotrotz müssen auch in der Projektpraxis eine Reihe von Aspekten berücksichtigt werden, um die Barrierefreiheit einer A12-Anwendung zu gewährleisten. Es gibt immer spezifische Anforderungen, die nicht von der Plattform abgedeckt werden können. Eine projektseitige Umsetzung ist unabdingbar. Betroffen davon sind nahezu alle Bereiche – vom Design über die Modellierung und Entwicklung bis hin zur Testung.

Um A12-Projekte bei der Umsetzung zu unterstützen, steht in der Dokumentationsplattform GetA12 ein detaillierter Leitfaden für barrierefreie A12-Anwendungen bereit. Er bietet eine praxisorientierte Hilfestellung und umfasst sowohl Hintergrundinformationen zur Barrierefreiheit und ihrer Zertifizierung, als auch Checklisten rund um Design-Vorgaben sowie Anforderungen an die Modellierung und die Entwicklung.

Zentrale A11Y-Standards

Der wichtigste internationale Standard für die Barrierefreiheit von Web-Anwendungen sind die **Web Content Accessibility Guidelines (WCAG)**. Sie werden herausgegeben vom World Wide Web Consortium (W3C). In Deutschland bildet die **Barrierefreie-Informationstechnik-Verordnung (BITV)** eine zentrale gesetzliche Grundlage. Sie basiert im Wesentlichen auf den Vorgaben der WCAG und setzt die EU-Richtlinie 2016/2102 um, der zufolge Websites und Apps offizieller Stellen barrierefrei sein müssen. Für die offizielle Barrierefreiheits-Zertifizierung müssen Anwendungen den BITV-Test bestehen – ein umfangreiches Prüfverfahren, das fast 100 Testschritte umfasst.

06

A12 Upgrades

Mit dem Wechsel auf neue A12-Versionen profitieren A12-Anwendungen von neuen Fähigkeiten der zugrundeliegenden Plattform. Das jährlich empfohlene Upgrade hält die technische Basis der Anwendung aktuell und erhöht ihre Lebensdauer. Um Migrationsaufwände gering zu halten, stellt das Plattform-Team eine Reihe von Hilfsmitteln bereit.

Plattform-Ansatz: Evolvierbare technische Basis

Jedes individuelle Softwareprojekt benötigt eine technische Basis. Bei konventionellen Projekten, die auf der grünen Wiese erstellt werden, wählt das Team den Tech-Stack eigenständig aus. Es entscheidet sich für Technologien, Frameworks und Drittbibliotheken und baut damit gewissermaßen einmalig eine technische Plattform. Dieses technische Fundament bleibt bei klassischen Individualprojekten in der Regel statisch. Der Aufwand für die Aktualisierung lässt sich nur in bestimmten Fällen rechtfertigen, ein Verfall ist vorprogrammiert. Auch wenn die technische Basis zu Projektbeginn State-of-the-Art ist, veraltet sie sukzessive.

A12-Anwendungen bauen hingegen auf einem technischen Fundament auf, das nicht durch das Projektteam erstellt und gepflegt werden muss. Das mehr als hundertköpfige A12-Team von mgm und die A12 Community übernehmen diese Aufgabe und entwickelt die technische Basis kontinuierlich weiter. Kontinuierliche Plattform-Updates erhöhen die Sicherheit der technischen Basis und führen in regelmäßigen Abständen neue Funktionalitäten und Innovationen ein. Während bei einem konventionellen Individualprojekt die Technik einmal ausgewählt und nach dem Go-Live typischerweise nicht mehr angefasst wird, ermöglicht der Plattformansatz eine kontinuierliche technische Evolution der Software.

Die Vorteile im Überblick

- A12 Updates halten die technische Basis aktuell. Der Tech-Stack basiert auf Branchenstandards und wird sukzessive weiterentwickelt.
- Das A12 Plattform-Team managt plattformseitige Drittbibliotheken und schließt mit A12 Security Updates Sicherheitslücken.
- Das Projekt-Team wird entlastet und kann sich auf spezifische Projektherausforderungen konzentrieren.
- Die Lebensdauer der Anwendung wird erhöht, da sich die technische Basis weiterentwickelt und nicht – wie bei konventionellen Individualprojekten – sukzessive veraltet.

A12 Release-Linien

Das A12 Team veröffentlicht einmal pro Jahr eine neue **Release-Linie** mit Long Term Support (LTS). Sie wird zwei Jahre lang unterstützt. Jede neue Release-Linie kann **Breaking Changes** enthalten. Sie erfordern projektseitige Anpassungen, die zum Beispiel durch Änderungen an den APIs notwendig sind.

Für jede Release-Linie veröffentlicht das A12-Team kontinuierlich **Produkt-Updates** in Form von **Minor-Releases** und **Patch-Releases**. Minor-Releases bringen verbesserte Features, während Patch-Releases Bugs beseitigen und neu identifizierte Sicherheitslücken in Drittbibliotheken schließen. Produkt-Updates enthalten keine Breaking Changes. Sie lassen sich ohne manuelle Aufwände installieren.

Als fester Bestandteil der A12 Release Policy werden alle zwei Monate regelmäßige **Security Patches** veröffentlicht. Falls kritische Schwachstellen identifiziert werden, stellt mgm darüber hinaus so schnell wie möglich entsprechende Patches bereit. Darüber hinaus ist die kontinuierliche Überprüfung aller Security-Meldungen der in A12 verwendeten Komponenten Teil unseres Build & Deployment-Prozesses.

Empfehlungen

- A12 Projekte sollten einmal im Jahr auf die jeweils aktuelle Release-Linie upgraden.
- Minor- und Patch-Releases sollten kontinuierlich und zeitnah eingespielt werden, um Sicherheitsrisiken zu minimieren und von stabileren Komponenten zu profitieren.

Migration auf neue A12-Versionen

Beim Wechsel auf eine neue Release-Linie profitieren Projekte von neuen Features der A12-Plattform. In der Regel fallen dafür aber auch Migrationsaufwände an, die in der Planung frühzeitig berücksichtigt werden sollten. Alle relevanten Schritte sind in den Release-Notes dokumentiert. Um die Aufwände zu minimieren, stellt das A12 Plattform-Team eine Reihe an unterstützenden Tools bereit:

- **Modell-Migration**
  Die Migration der Modelle ist in der Regel bei jedem Wechsel auf eine neue Release-Linie erforderlich. Sie erschließt der A12-Anwendung die neuen modellgetriebenen Funktionen der jeweils aktuellen A12-Version. Skripte automatisieren den Vorgang. Der Simple Model Editor erkennt automatisch, wenn ein Workspace veraltete Modell-Versionen beinhaltet und kann sie automatisch updaten.
- **Daten-Migration**
  Die Migration von Daten wird nur erforderlich, wenn bei einer neuen Release-Linie Änderungen bei der Serialisierung, Persistenz-Struktur oder Index-Struktur auftreten. Dies passiert eher selten. Sofern der Fall auftritt, stellt die jeweilige Release-Linie ein Skript bereit, um Daten automatisch in die neue Struktur zu bringen.
- **Code-Migration**
  Im Fall von Breaking Changes (z.B. Änderungen in APIs) in A12-Komponenten, die in einer A12-Anwendung genutzt werden, sind anwendungsseitige Code-Anpassungen nötig. Um sie gering zu halten, setzt A12 auf Auto-Refactoring-Tools wie OpenRewrite und stellt für neue Release-Linien Codemods bereit. Sie helfen dabei, Änderungen wie die Umbenennung von Klassen, Packages und Methoden automatisch vorzunehmen.

Wie hoch die verbleibenden Migrationsaufwände im Detail ausfallen, ist von vielen Faktoren abhängig, grundsätzlich gilt aber der Trade-Off: Je weiter sich das Projekt von den A12-Standards entfernt, desto höher werden die Migrationsaufwände für Plattform-Updates. Um die kontinuierliche Modernisierung der Anwendung sicherzustellen und Folgeaufwände zu verringern, sollte bei A12-Anwendungen nur aus gutem Grund von bereitgestellten Standards abgewichen werden.

Die Zukunft von A12 mitgestalten

A12 wird kontinuierlich weiterentwickelt. Gerade nichtfunktionale Anforderungen, die zum Beispiel durch neue gesetzliche Vorgaben hinzutreten, werden mit sehr hoher Wahrscheinlichkeit zeitnah auf Plattform-Ebene umgesetzt. Darüber hinaus haben Projekte zahlreiche Möglichkeiten, um sich in die Weiterentwicklung der A12-Plattform einzubringen:

- **Bugs melden**
  Für die Meldung von Bugs steht ein Jira Ticket-Template bereit. Es erleichtert durch die einheitliche Struktur und eine Anleitung zum Ausfüllen die systematische Beschreibung von Fehlern, die im Rahmen der regulären Updates behoben werden.
- **Anforderungen stellen**
  Die Möglichkeit, Anforderungen direkt an die A12-Basis stellen zu können, ist eine Besonderheit der A12 Plattform, die bei anderen Low Code Plattformen in der Regel nicht gegeben ist. Projekte erhalten dadurch die Möglichkeit, sehr direkt auf die Weiterentwicklung der Plattform einzuwirken. Für die eingehenden Anforderungstickets ist intern innerhalb von mgm ein detaillierter Konsolidierungs-Prozess etabliert, in dem eine mögliche Umsetzung der Anforderung innerhalb der A12 Plattform geprüft wird.
- **Modell-Repository teilen**
  Indem Projekte ihre fachlichen Modelle offen zur Verfügung stellen, leisten sie einen wichtigen Beitrag zum Ausbau des A12 Ökosystems. Gerade im öffentlichen Sektor eröffnet sich dadurch großes Potenzial, um Doppelaufwände zu vermeiden eine anwendungsübergreifende Interoperabilität zu fördern.
- **Code contributen**
  Im Rahmen von A12-Projekten können Funktionalitäten entstehen, die ggf. in die Plattform zurückfließen können – zum Beispiel als Erweiterungen, die auf bestimmten Extension Points der A12-Komponenten aufsetzen oder langfristig auch als künftige Kernbestandteile der A12 Plattform.

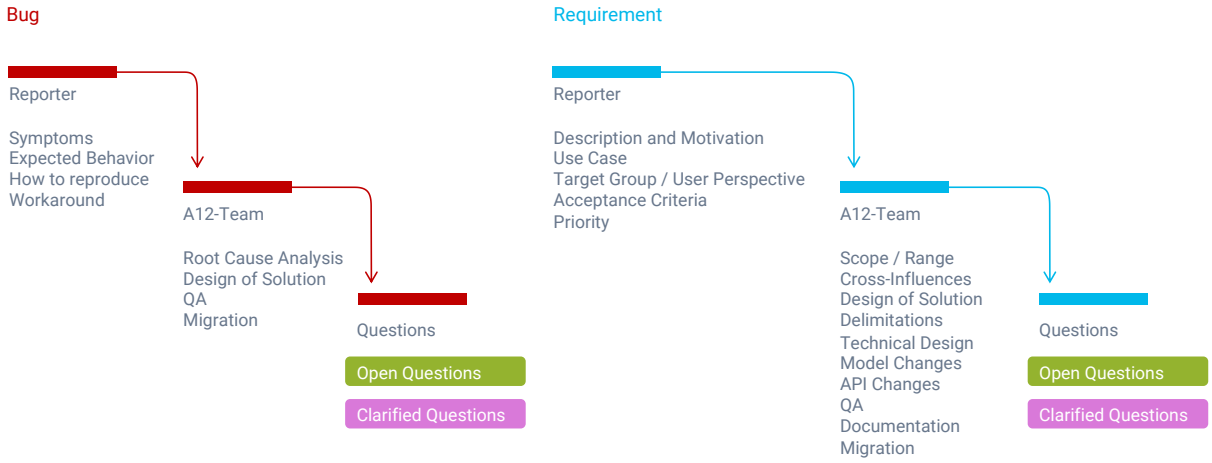


Abbildung 12 – *Standardisierte Struktur von Bug- und Requirements-Tickets*

07

Continuous Delivery & Cloud Operations

Enterprise Software ist lebendig und immer wieder Anpassungen ausgesetzt. Sowohl während der Entwicklung als auch nach Produktivsetzung müssen Änderungen schnell bereitgestellt werden. mgm setzt dafür auf standardisierte Methoden und Umgebungen entlang der gesamten Entwicklungs- sowie Build- und Deployment-Prozesse. Sie sind speziell für den Betrieb containerisierter Anwendungen in modernen Cloud-Infrastrukturen ausgelegt.

Wie lässt sich Quellcode verlässlich in ausführbare Software überführen? Diese Frage steht im Zentrum des Build Managements, das in den vergangenen Jahren einen enormen Zuwachs an Komplexität verzeichnet hat. An die Stelle von Monolithen treten skalierbare und flexibel deploybare Microservices. Wesentliche Treiber dafür waren der Siegeszug agiler Entwicklungsmethoden und der Business-getriebene Anspruch, Software-Neuerungen schneller in Produktion zu bringen. Die zunehmende Verbreitung von DevOps-Praktiken sorgt zudem dafür, dass Entwicklung und Betrieb immer nahtloser ineinander übergehen.

mgm begegnet dieser Entwicklung mit einer Reihe von Standards rund um die benötigten Umgebungen sowie die Abläufe der Build- und Deployment-Prozesse.

Betrieb von A12-Anwendungen auf Kubernetes-Clustern

Auch wenn in der heutigen Enterprise Software-Entwicklung eine Vielzahl an Umgebungen, Systemen und Diensten involviert sind, bleibt eine wesentliche Unterteilung in zwei Bereiche erhalten: Es gibt eine Landschaft, in der Software entwickelt wird, und eine Umgebung, in der die Software ausgeführt wird. Für beide Ebenen setzt mgm bei A12-Projekten auf eine Reihe von Standards, die ein schnelles Setup ermöglichen und wesentliche Build & Deployment-Prozesse automatisieren.

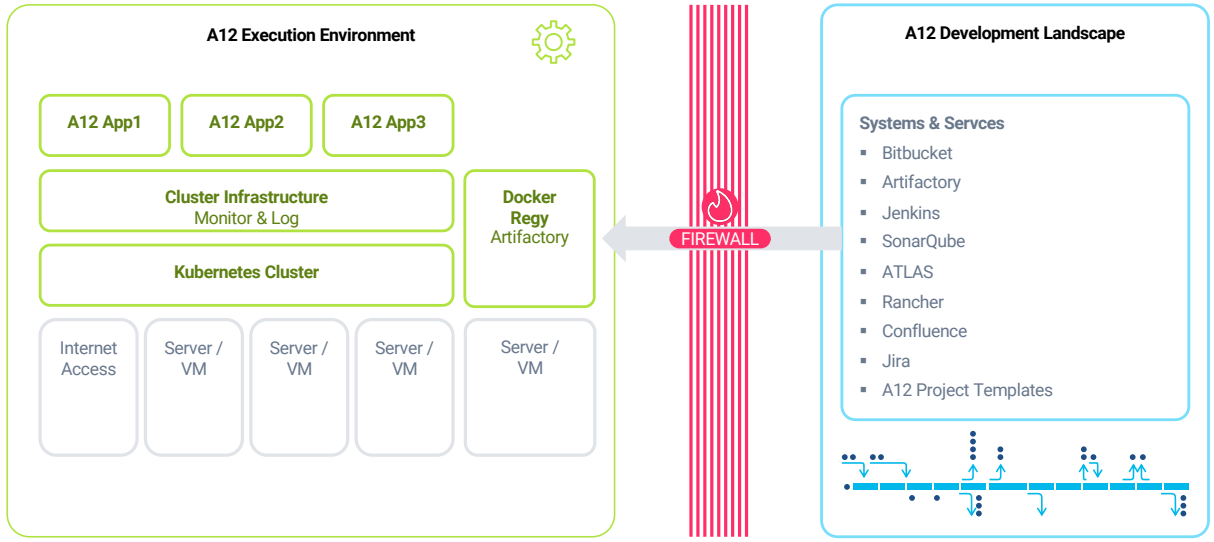


Abbildung 13 – *Schematische Übersicht der Betriebs- und Entwicklungsumgebungen*

Die wichtigsten infrastrukturellen Säulen für die Entwicklungsumgebung (Bitbucket, Jira, etc.) sind auf S. 19 beschrieben. Für die Umgebung zur Ausführung von A12-Anwendungen setzt mgm standardmäßig auf Kubernetes-Cluster. Basierend auf den Erfahrungen aus mehreren großen Softwareprojekten haben wir eine Auswahl an Tools aus dem Kubernetes Ökosystem getroffen, die wir als Standard-Stack empfehlen. Grundsätzlich lassen sich A12-Anwendungen jedoch mit unterschiedlichen Technologie-Stacks betreiben – je nach den Vorgaben des jeweiligen Hosters.

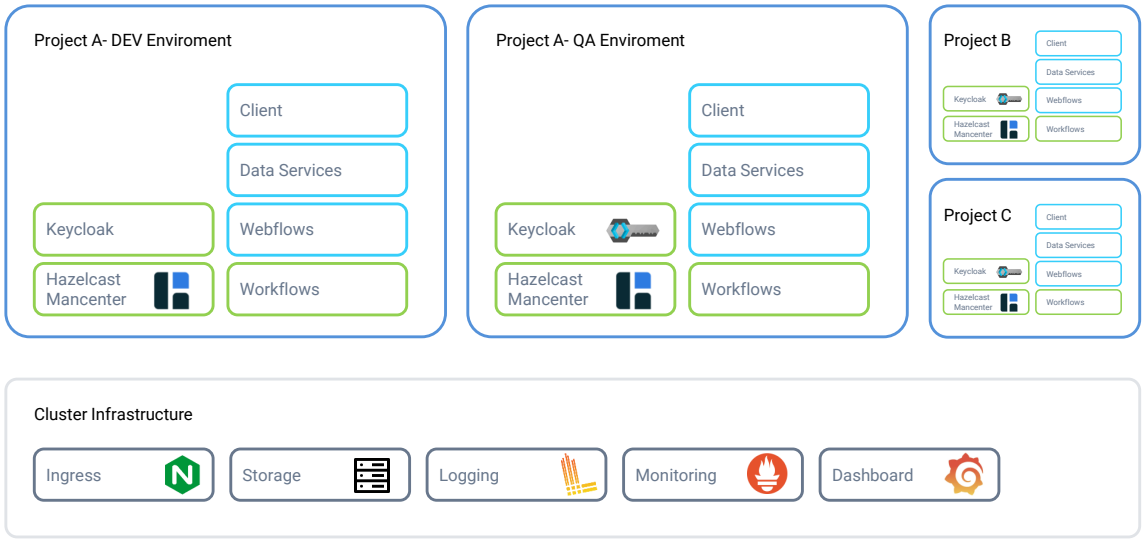


Abbildung 14 – *Standardisierte Projektumgebungen beschleunigen das Setup im Cluster*

Code Management und Versionierung

Ein System zur Versionierung von Quellcode und Modellen ist eine wesentliche Voraussetzung für die Implementierung der CI/CD Pipeline. mgm setzt dabei auf **Git-Repositories**. Als grafische Oberfläche für Code-Reviews ist **Bitbucket** das Tool der Wahl. Für das Versionierungsschema der im Rahmen des Projekts entwickelten Software empfehlen wir **Semantic Versioning** (https://semver.org).

Das konkret zu verwendende Branching-Modell wird in Abstimmung mit dem Releasevorgehen und dem Projektplan festgelegt. Generell ist der Git-flow-Workflow sehr gut für Enterprise Softwareprojekte geeignet, da er die typischerweise benötigten Feature-Branches und Release-Branches unterstützt.

Die Struktur der Repositories folgt einem festgelegten Standard. Neben einem Repository für den Programmcode gibt es stets ein benachbartes Repository für Deployment-bezogene Inhalte. Dort liegen Konfigurationen, die beschreiben, wie die Software automatisiert gebaut und deployed wird.

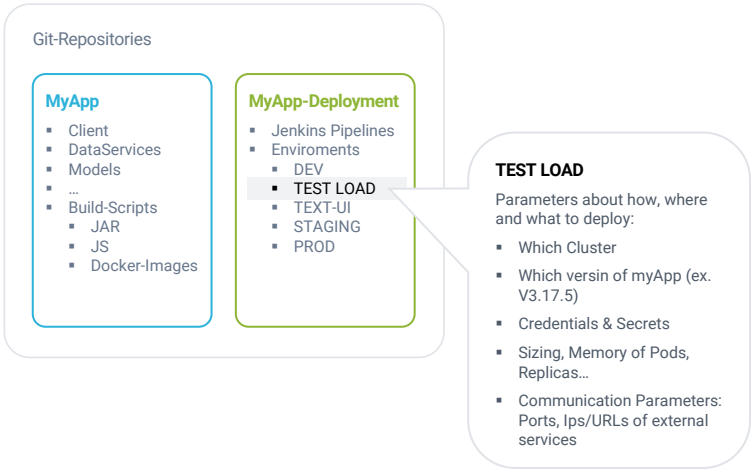


Abbildung 15 – *Standardisierte Repository-Struktur*

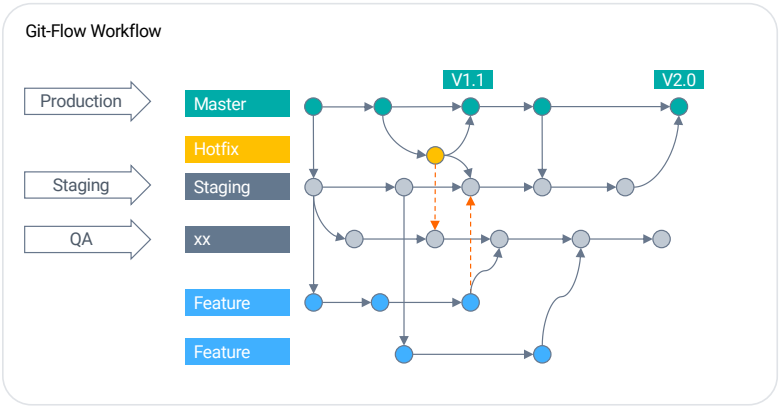


Abbildung 16 – *Beispiel für einen Git-Workflow*

CI/CD Pipelines

Um Quellcode in robuste und qualitätsgesicherte Software zu überführen, nutzt mgm eigens entwickelte und in mehreren Projekten bewährte Build- und Deployment-Pipelines. Die folgende Abbildung skizziert den grundlegenden Aufbau. Die Pipeline skizziert von links nach rechts den Weg vom Code zur produktionsreifen Software. Die Röhrensegmente kennzeichnen jeweils bestimmte Operationen mit den Entwicklungsständen, die weitestgehend automatisiert durchgeführt werden. Führendes System ist der Jenkins Build Server – ein weit verbreitetes Automatisierungstool auf Open Source Basis. Es startet und orchestriert die Build- und Deployment Jobs.

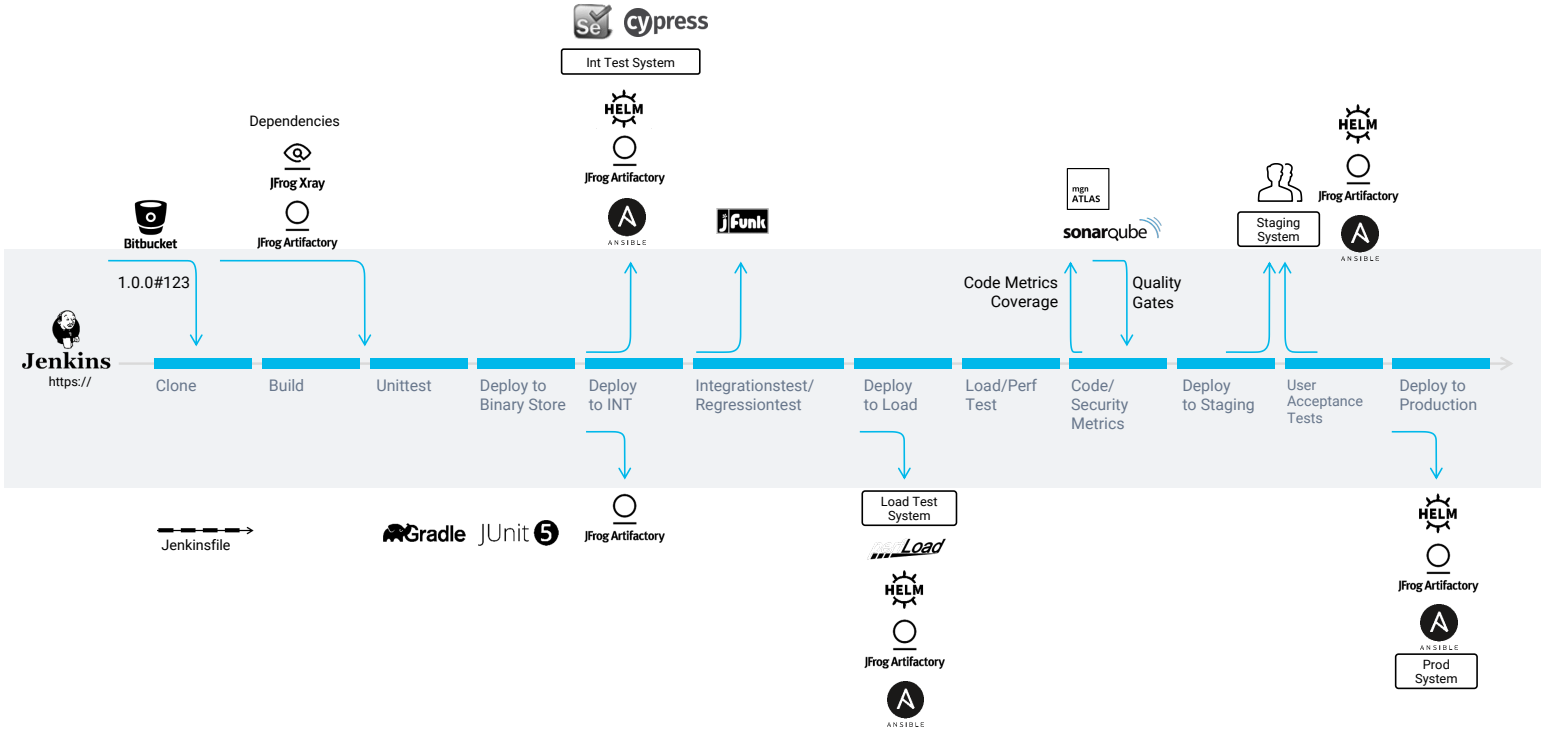


Abbildung 17 – *Jenkins orchestriert die einzelnen Schritte der CI/CD-Pipeline*

In den ersten beiden Röhrensegmenten werden A12-Modelle und Quellcode aus der Quellcodeverwaltung (Git) gezogen und gebaut. Nach dem Bauen werden die generierten Artefakte in einer Container Registry abgelegt. Von hier können die Images in weiteren Schritten auf Testsystemen installiert und mit entsprechenden Testwerkzeugen (Selenium, JFunk, Perfload, Cypress) getestet werden. Bei Bedarf können in einem weiteren Schritt Lasttests (z.B. mit Perfload) durchgeführt werden. Nach erfolgreichem Build werden die Artefakte einer statischen Codeanalyse unterzogen, um potenzielle Fehler und Bugs automatisch aufzuspüren. Im letzten Röhrensegment erfolgt schließlich das Deployment der Software auf ein Produktionssystem.

Im Rahmen des Entwicklungsprozesses gibt es mehrere Anlässe, um die Software zu bauen. Für jeden dieser Anlässe ist der Build- und Deployment-Prozess klar definiert.

	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

Abbildung 18 – *Übersicht der verschiedenen Ausprägungen von Builds*

Dependency Management

Ein Effizienztreiber der Entwicklung moderner Geschäftsanwendungen ist das Wiederverwenden von Programmcode. Man muss das Rad nicht ständig neu erfinden: In der Praxis nutzen Entwicklungsteams deshalb immer wieder bereits bestehende Bibliotheken und Programmpakete. Sie können aus eigener Entwicklung oder aus externen Quellen – typischerweise aus Open-Source-Initiativen – stammen. Wenn der eigene Code solche Bibliotheken aufruft, spricht man von Abhängigkeiten (engl. „dependencies").

Das sorgfältige Management dieser Abhängigkeiten sowie der Versionsstände und Lizenzbedingungen der verwendeten Bibliotheken ist wichtig, um Risiken zu minimieren und nicht in der „Dependency Hell" zu landen. Denn wenn bei einer Third-Party-Library eine Schwachstelle bekannt wird, ist auch die eigene Software potenziell gefährdet. Und wenn Lizenzbedingungen nicht eingehalten werden, drohen folgenschwere Urheberrechtsverletzungen.

Im Rahmen der Build- und Deployment-Pipeline nutzt mgm deshalb u.a. auch die sogenannte **Software Composition Analysis (SCA)** – einen automatisierten Prozess für die Analyse der eingesetzten Open Source-Komponenten. So lässt sich von vornherein verhindern, dass Software-Stände mit schwerwiegenden Sicherheitsproblemen überhaupt den gesamten Build-Prozess durchlaufen. Aktuell setzen wir dafür auf unser Produkt **ATLAS** (https://www.mgm-sp.com/mgm-atlas) mit **JFrog Xray**.

Auch der Gesetzgeber hat die Bedeutung erkannt, Software-Abhängigkeiten von Drittkomponenten transparenter zu machen. Im Rahmen des Cyber Resilience Act werden Hersteller europaweit verpflichtet, künftig mit **Software Bills of Material (SBOM)** zu dokumentieren, welche Software-Bestandteile in ihren Software-Produkten enthalten sind. Für die A12-Plattform wird diese Anforderung mit Hilfe von ATLAS schon heute erfüllt. Kundenprojekten steht diese Option auch für A12-Anwendungen zur Verfügung.

Von Repositories in Umgebungen

Während die Entwicklungsstände durch die Pipeline fließen, werden sie an vier Stationen auf bestimmte Umgebungen aufgespielt. Wenn wir die Pipeline mit einer Fertigungsstraße vergleichen, sind diese Stationen wichtige Punkte für die Bearbeitung des Werkstücks. Die Umgebungen INT, TEST-LOAD und STAGING hängen eng mit Qualitätssicherungsmaßnahmen zusammen, die im nachfolgenden Kapitel genauer vorgestellt werden.

UMGEBUNG	ZWECK
INT TEST	Integrationstests, Regressionstests
LOAD TEST	weitere Qualitätssicherungsmaßnahmen wie Last-Tests
STAGING	finale Qualitätssicherung; User Acceptance Tests; fachliche Abnahme-Tests von Release-Kandidaten
PROD	Produktiver Betrieb in Kubernetes Cluster

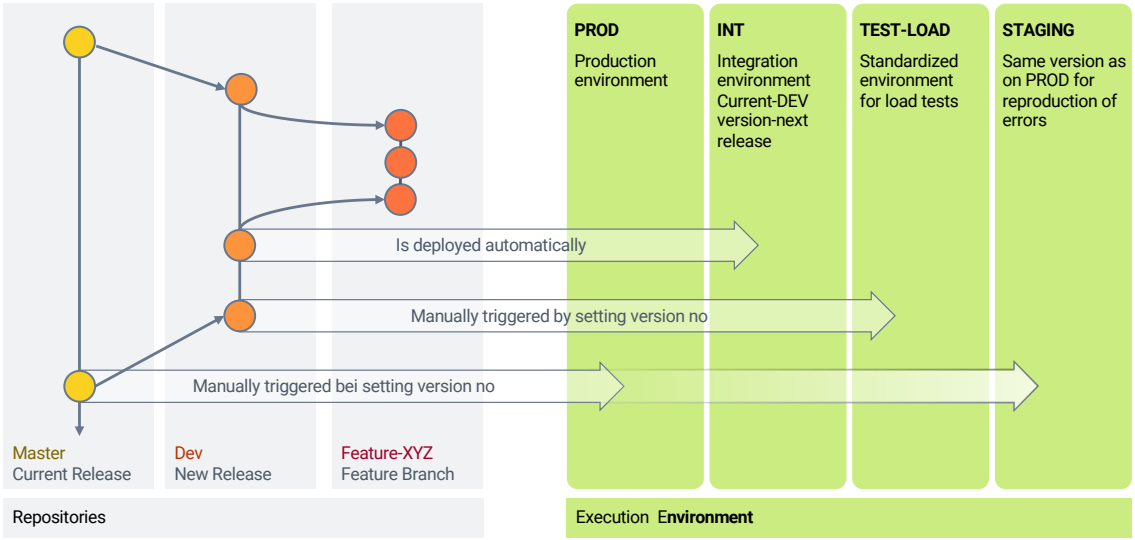


Abbildung 19 – *Übersicht der einzelnen Deployments*

08

Qualitätssicherung

A12 Projekte setzen einerseits auf qualitätsgesicherte Komponenten der A12 Plattform auf. Andererseits gilt es aber auch, QS-Maßnahmen für projektseitig implementierte Lösungen einzurichten. mgm setzt dabei auf Prozesse gemäß internationaler Standards und nutzt eine Reihe von Tools aus eigener Forschung und Entwicklung – vom Testdatengenerator über ein Testmanagementtool bis hin zu weitergehenden Automatisierungswerkzeugen. In Kombination mit einer bewährten Methodik entsteht kompromisslos hochqualitative Software.

Methodik und Vorgehensweise

Einordnung der Qualitätssicherung im Software-Lebenszyklus

Qualitätssicherung begleitet den gesamten Software-Lebenszyklus – von der ersten Idee bis zum Produktivbetrieb. Sie ist integraler Bestandteil des Entwicklungsprozesses, der in jeder Phase unterschiedliche Schwerpunkte setzt und so einen nachhaltigen Qualitätsgewinn sicherstellt.

- **Anforderungs- und Konzeptphase**
  Bereits bei der Erhebung von Anforderungen beginnt Qualitätssicherung. Durch präzise Dokumentation, Reviews unter Einbezug des Kunden und die frühzeitige Validierung von Modellen wird sichergestellt, dass Fehlinterpretationen vermieden und eine belastbare Basis für die Entwicklung geschaffen wird.
- **Design- und Entwicklungsphase**
  Während der Implementierung ist die Qualitätssicherung eng mit der Entwicklung verzahnt. Code-Reviews, statische Analysen und kontinuierliche Integration sichern die Qualität auf technischer Ebene. Fachliche Umsetzungen werden durch Verifizierungen und Regressionstests zeitnah dediziert getestet. Für Applikationen mit hohen Lastanforderungen werden erste Tests an performanzrelevanten Komponenten durchgeführt. Qualität wird nicht nachgelagert geprüft, sondern von Beginn an mitentwickelt.
- **Test- und Integrationsphase**
  In dieser Phase sichern System- und Integrationstests - manuell wie automatisiert - das Zusammenspiel der Funktionen. Ergänzend werden nicht-funktionale Tests durchgeführt, um alle relevanten Qualitätsaspekte abzudecken.
- **Übergabe und Abnahme**
  Mit Abschluss der Entwicklung wird die Software samt aller Nachweise an den Kunden übergeben. Auf dieser Grundlage werden Abnahmetests durchgeführt. Erst mit formaler Abnahme ist die Lösung freigegeben und kann in den Produktivbetrieb überführt werden.
- **Betrieb und Wartung**
  Auch nach der Produktivsetzung endet Qualitätssicherung nicht. Rückmeldungen aus Support und Betrieb ermöglichen es, Prüfungen und Maßnahmen für kommende Versionen gezielt weiterzuentwickeln.

Very Early Testing

mgm setzt generell auf die eigens entwickelte und in vielen Projekten erfolgreich eingesetzte Methodik „Very Early Testing“. Sie zielt darauf ab, Probleme im Projekt möglichst früh im Prozess zu finden, wenn die Beseitigung noch einfach und günstig ist. Tests werden dabei häufiger durchgeführt als bei der klassischen Methodik, bei der in der Regel erst am Ende der Entwicklung getestet wird. Deshalb ist es nötig, einen hohen Automatisierungsgrad der Tests anzustreben und eng mit der Entwicklung zusammenzuarbeiten.

Dieses Paradigma des „Very Early Testing“ kommt bereits bei der Entwicklung der A12-Plattform zum Einsatz und wird auch projektseitig dringend angeraten.

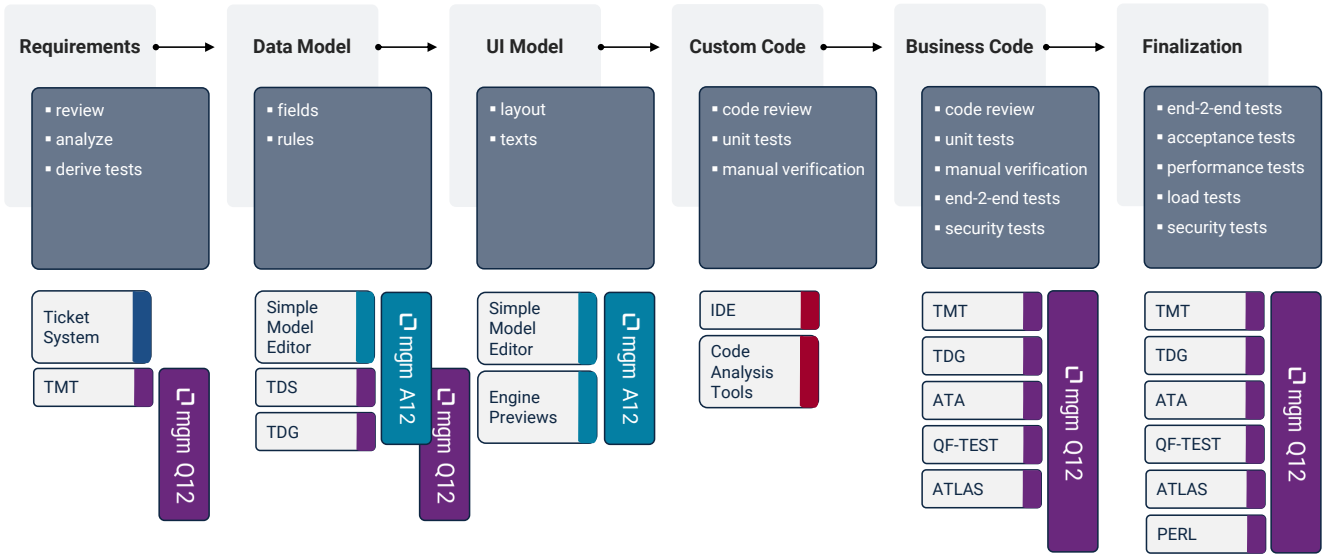


Abbildung 20 – *Übersicht, welche Tools in welchen Phasen des modellbasierten Entwicklungsprozesses zum Einsatz kommen*

Softwarequalitätssicherung im Kontext von A12-Projekten

Qualitätssicherung in Projekten auf Basis der mgm A12-Plattform unterscheidet sich in zentralen Aspekten von klassischen Softwareentwicklungsprojekten. Die modellbasierte Low-Code-Architektur verlagert Testschwerpunkte, verändert Aufwandsstrukturen und eröffnet neue Möglichkeiten für Automatisierung. Die Qualitätssicherung muss diese Besonderheiten gezielt aufgreifen, um den Mehrwert der Plattform optimal zu nutzen.

Schwerpunkte der QS

⊕ **Qualitätsgesicherte A12-Komponenten**
Sowohl die Modellierungskomponenten als auch die Runtime-Komponenten der A12-Plattform sind bereits auf mgm-Seite qualitätsgesichert – funktional wie nichtfunktional, manuell wie automatisiert. Für Projekte bedeutet das: Wiederholte Prüfungen grundlegender Plattformfunktionen entfallen, Testaufwände fokussieren sich auf fachliche Modellierung und Individual-Entwicklung.

⊕ **Verschiebung der Testschwerpunkte**
Durch Modell-Editoren und Preview-Engines können Anforderungen daten- und UI-seitig bereits in frühen Entwicklungsphasen geprüft und validiert werden. Testfälle orientieren sich damit stärker an Modellartefakten und deren fachlicher Korrektheit. Dies ermöglicht unmittelbare Rückmeldungen zur Umsetzbarkeit und Konsistenz, lange Schleifen bis nach Abschluss der Implementierung werden vermieden.

⊕ **Qualitätssicherung für Individual-Code**
Der Anteil individuell entwickelter Funktionen ist in A12-Projekten stark reduziert. Für diese Bereiche greifen die etablierten QS-Maßnahmen. Auf diese Weise bleibt die notwendige Sorgfalt für maßgeschneiderte Erweiterungen gewährleistet, ohne die Effizienzgewinne des modellbasierten Vorgehens zu schmälern.

⊕ **Automatisierungspotenzial & A12-spezialisiertes Tooling**
A12 bietet durch seine Architektur sowie A12-integrierte Werkzeuge eine besondere Eignung für die Automatisierung von QS-Aktivitäten. Testdatengenerierung oder UI-Testautomatisierung lassen sich auf Basis der standardisierten A12-Komponenten unmittelbar aufsetzen. So entsteht eine deutliche Reduktion QS-vorbereitender Aufwände.

QS in A12-Projekten profitiert von einer qualitätsgesicherten Plattformbasis, früher fachlicher Validierung durch Modelle und einem hohen Automatisierungsgrad. Diese Vorteile gilt es, konsequent auszuschöpfen. Der verbleibende Individual-Code wird mit etablierten QS-Verfahren abgesichert.

Verantwortlichkeiten und Rollen in der QS

Im Software-QS-Prozess arbeiten verschiedene Rollen eng zusammen, um die Qualität von Software über alle Phasen des Entwicklungszyklus hinweg abzubilden.

Zentrale Rollen im Software-QS-Prozess sind:

- **Testexperten**
  Testexperten übernehmen im gesamten QS-Prozess die Konzeption, Spezifikation, Durchführung und Dokumentation von Tests – sowohl funktionaler als auch nicht-funktionaler Art. In der Fachlichkeit des Kunden bzw. Projekts kennen sie sich im Detail aus. Für die Durchführung nicht-funktionaler Tests (z. B. Last-, Performance-, Sicherheits- oder Barrierefreiheitstests) ist gesondertes Fachwissen zu den jeweiligen Testverfahren und Tools notwendig, das in der Regel durch Spezialisten abgedeckt wird.
  - **Manuell/Automatisiert**
    Testexperten agieren je nach Projektanforderung in der Disziplin des manuellen Testens oder der Testautomatisierung. Beide Bereiche gehören zum Rollenprofil, werden jedoch nicht zwangsläufig durch dieselbe Person abgedeckt.
  - **In mgm-Projekten arbeiten häufig manuelle Testexperten und Testexperten mit Automatisierungskenntnissen eng zusammen;** je nach Aufgabenstellung werden Testdesign inkl. manueller Testausführung einerseits und Testautomatisierung andererseits arbeitsteilig oder in Personalunion abgedeckt. Die sorgfältige Testanalyse, der Testentwurf und die initiale manuelle Testdurchführung bilden die Basis für eine anschließende Testautomatisierung. Während im ersten Schritt stabile Testfälle identifiziert und fachlich geprüft werden, bringen Testautomatisierer ihr Know-how ein, um diese Prüfungen effizient und wartbar zu automatisieren – insbesondere für wiederkehrende Regressionstests. So entsteht ein integrierter QS-Prozess, in dem beide Rollen aufeinander aufbauen und gemeinsam zu einer nachhaltigen Softwarequalität beitragen.
- **Testmanager**
  In größeren oder besonders komplexen Vorhaben kann der Einsatz eines Testmanagers sinnvoll und notwendig sein. Der Testmanager ist verantwortlich für die Planung, Steuerung und Überwachung sämtlicher QS-Aktivitäten im Projekt. Dazu gehören vornehmlich die Entwicklung der QS-Strategie, das Ressourcenmanagement, die Auswahl geeigneter Methoden und Werkzeuge sowie die Berichterstattung an Stakeholder.
- **Ergänzende QS-Rollen**
  Je nach Projektgröße und Komplexität können zusätzliche Rollen wie TestOps Engineers oder externe Prüfer und Auditoren eingebunden werden. Sie übernehmen spezifische Aufgaben, die über das Kernprofil von Testexperten und Testmanagern hinausgehen.

Softwarequalitätssicherung als separate Dienstleistung

Neben der entwicklungsbegleitenden Qualitätssicherung in Entwicklungsprojekten bietet mgm Testing-as-a-Service sowie QS-Expertenberatung an. Hierbei können Kunden flexibel auf erfahrene Testexperten, Testmanager oder Automatisierungsspezialisten zugreifen – sei es für einzelne Phasen wie Abnahmetests oder als kontinuierliche Begleitung in eigenen Entwicklungsvorhaben.

- Testing-as-a-Service
- QS-Expertenberatung

Darüber hinaus steht das mgm-eigene UI/UX-Kompetenzteam für Fragen und Unterstützung u.a. im Bereich Barrierefreiheitstests zur Verfügung:

- Barrierefreiheit/A11Y

Q12 Qualitätssicherungs-landschaft

Was ist Q12?

Effiziente Qualitätssicherung von Enterprise Software ist ohne spezialisierte Werkzeuge kaum vorstellbar. Sie strukturieren Prozesse, reduzieren manuelle Aufwände und schaffen die Grundlage für reproduzierbare Ergebnisse. Bei mgm bildet die Qualitätssicherungslandschaft Q12 das Herzstück unserer QS-Werkzeuge. Die firmeneigene Toolchain wurde aus den Anforderungen realer Projekte heraus entwickelt und deckt alle zentralen Bereiche moderner QS ab – von der Testplanung über die Automatisierung bis hin zu Sicherheit, Performance und Barrierefreiheit.

Wie ist Q12 strukturiert?

Q12 umfasst mehrere spezialisierte Werkzeuge, die sich einzeln einsetzen oder flexibel kombinieren lassen. Einige davon (TDS, TDG und ATA) sind speziell für Anwendungen auf Basis von A12 ausgelegt, die übrigen sind in allen Enterprise-Anwendungen (A12 und nicht-A12) einsetzbar.

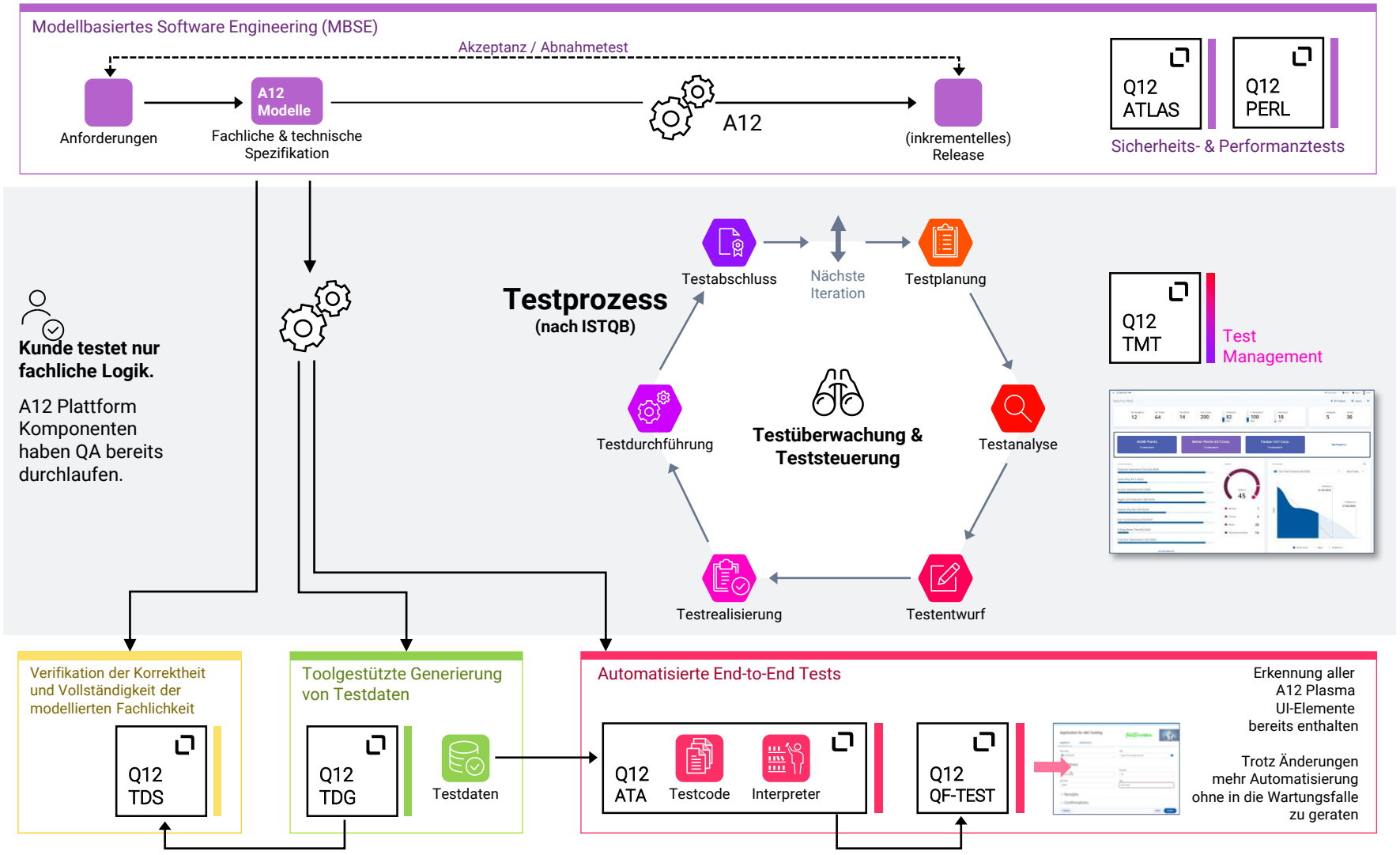
- **Q12-TDS (Test Data Suite)**
  Funktionsprüfung von Datenmodellen mit Schwerpunkt auf Regeln, Felddefinitionen, Mappings und Berechnungen in modellbasierten A12-Anwendungen
- **Q12-TDG (Test Data Generator)**
  Automatische Erzeugung valider Testdaten für komplexe Formulare in modellbasierten A12-Anwendungen
- **Q12-ATA (Automated Test Automation)**
  Automatisierte Erzeugung von Code für die Verwendung in der UI-Testautomatisierung in modellbasierten A12-Anwendungen
- **Q12-TMT (Test Management Tool)**
  Zentrale Plattform zur Dokumentation von Testfällen und zur Berichterstattung über Testdurchführungen
- **Q12-QF-TEST (UI Test Automation)**
  Erstellung, Verwaltung und Ausführung von Tests zur automatisierten Prüfung grafischer Benutzeroberflächen
- **Q12-ATLAS (Automated Toolset for Lean Application Security)**
  Orchestrierung von Analysewerkzeugen für automatisiertes Testen von Applikationssicherheit
- **Q12-PERL (Performance & Load Tests)**
  Analyse, Ausführung und Monitoring von Performance- und Lasttests

Was ist besonders an Q12?

- **Modellgetriebene Automatisierung**
  Modellbasierte Low-Code-Entwicklung verändert den QS-Prozess und erfordert spezialisierte Werkzeuge. Q12 bietet passgenaue Lösungen, die gezielt auf A12-Anwendungen zugeschnitten sind. Sie betten Qualitätssicherung frühzeitig, automatisiert und mit hoher Effizienz in den Entwicklungsprozess ein.
- **Individuell anpassbares Tooling**
  Q12 ist erweiterbar. Die Werkzeuge lassen sich mit Tools von Drittanbietern integrieren. Bei Bedarf kann mgm die Tools an organisatorische oder fachliche Besonderheiten der Kundenorganisation anpassen bzw. um spezifische Features erweitern.
- **Kontrollrahmen für KI-Einsatz**
  KI und Low Code bringen in Kombination enorme Produktivitätsvorteile. Q12 etabliert einen effektiven Kontroll- und Governance-Rahmen für KI-basierte A12-Anwendungen und minimiert die Risiken KI-bedingter Fehler und Halluzinationen.
- **Erprobt in der Praxis von Enterprise-Projekten**
  Alle Werkzeuge in Q12 sind aus den Bedarfen und Anforderungen großer Enterprise-Software-Projekte entstanden. Sie adressieren alle wesentlichen Herausforderungen in der Qualitätssicherung ausgewachsener Geschäftsanwendungen.

Für welche Einsatzszenarien ist Q12 ausgelegt?

- **A12-Anwendungen**
  Dank spezialisierter Werkzeuge erschließt Q12 die Prinzipien der modellbasierten Softwareentwicklung für die Qualitätssicherung. In Verbindung mit den universell einsetzbaren Tools entsteht so eine ganzheitliche QS-Lösung im A12-Umfeld. Die geschickte Ausnutzung von Modellen verschiebt QS-Aktivitäten in frühe Projektphasen, reduziert manuelle Arbeiten und schafft ein in klassischen Projekten nur schwer erreichbares Maß an Automatisierung.
- **Enterprise-Software ohne A12-Bezug**
  Für Enterprise-Anwendungen außerhalb des A12-Umfelds bieten die universellen Q12-Tools ein leistungsstarkes Fundament für moderne Qualitätssicherung über alle Entwicklungsphasen hinweg. Sie unterstützen die Qualitätssicherung komplexer Geschäfts- und Webanwendungen durch integriertes Testmanagement, automatisierte Testdurchführung und intelligente Analysefunktionen im Performance-, Last- und Sicherheitsbereich.


Modellbasiertes Software Engineering (MBSE)
Akzeptanz / Abnahmetest
Anforderungen
A12 Modelle
Fachliche & technische Spezifikation
A12
(inkrementelles) Release
Q12 ATLAS
Q12 PERL
Sicherheits- & Performanztests
Kunde testet nur fachliche Logik.
A12 Plattform Komponenten haben QA bereits durchlaufen.
Testprozess
(nach ISTQB)
Testabschluss
Nächste Iteration
Testplanung
Testanalyse
Testentwurf
Testrealisierung
Testdurchführung
Testüberwachung & Teststeuerung
Q12 TMT
Test Management
Verifikation der Korrektheit und Vollständigkeit der modellierten Fachlichkeit
Q12 TDS
Toolgestützte Generierung von Testdaten
Q12 TDG
Testdaten
Automatisierte End-to-End Tests
Q12 ATA
Testcode
Interpreter
Q12 QF-TEST
Erkennung aller A12 Plasma UI-Elemente bereits enthalten
Trotz Änderungen mehr Automatisierung ohne in die Wartungsfalle zu geraten

Teststufen und -arten

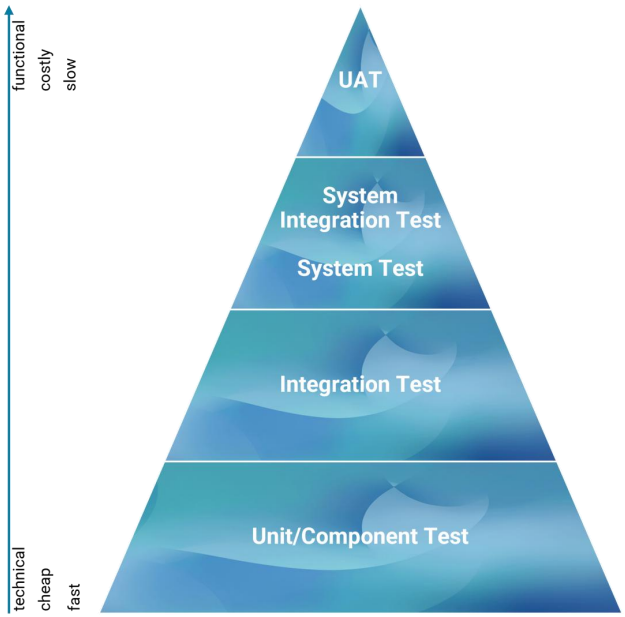
Teststufen

Teststufen ordnen Testaktivitäten entlang der Entwicklungs- und Integrationsschritte eines Systems. Sie legen fest, auf welcher Ebene getestet wird, welche Ziele im Vordergrund stehen und wer typischerweise die Verantwortung trägt. Die Testpyramide wird hier als Visualisierung der Teststufen genutzt. Ursprünglich stammt das Modell aus der Testautomatisierung, lässt sich aber ebenso auf die Gliederung nach Stufen übertragen – unabhängig davon, ob die Tests manuell oder automatisiert ausgeführt werden. Sie verdeutlicht, dass der Schwerpunkt auf den unteren Ebenen liegt, während die höheren Ebenen weniger, aber dafür umfassendere Tests enthalten. Eine Abweichung von diesem Prinzip, wie sie im sogenannten Ice-Cream Cone Anti-Pattern auftritt – zu viele GUI-Tests und zu wenige Unit- und Integrationstests – gilt als Hinweis auf eine unausgewogene Teststrategie.

Zentrale Teststufen sind:

- **Komponenten- bzw. Unit-Tests**
  Prüfung einzelner Programmeinheiten oder Module in Isolation. Ziel ist die frühzeitige Erkennung von Implementierungsfehlern. Sie sind in der Regel automatisiert und werden überwiegend von Entwicklern durchgeführt.
- **Integrationstests**
  Testen der Schnittstellen und Interaktionen zwischen einzelnen Komponenten. Der Schwerpunkt liegt auf Datenübergaben, Kommunikationswegen und dem Zusammenspiel technischer Bausteine.
- **Systemtests**
  Überprüfung des Gesamtsystems als vollständig integrierte Lösung. Getestet wird, ob alle funktionalen und nichtfunktionalen Anforderungen erfüllt sind. Dazu gehören auch Performance-, Sicherheits- und Usability-Tests.
- **Systemintegrationstests**
  Validierung der Interaktion des Systems mit anderen oder externen Systemen. Im Vordergrund steht die Sicherstellung korrekter Schnittstellenfunktionalität und -stabilität über Systemgrenzen hinweg.
- **Abnahmetests (User Acceptance Tests (UAT))** End-to-End-Tests aus Sicht der Anwender oder Business-Stakeholder. Ziel ist die Überprüfung, ob das System die fachlichen Anforderungen und geschäftlichen Bedürfnisse erfüllt. Abnahmetests bilden die Grundlage für die finale Freigabeentscheidung.

Die klare Differenzierung nach Teststufen stellt sicher, dass Qualität sowohl auf technischer Ebene als auch aus fachlicher Perspektive überprüft wird: Von der großen Basis der schnellen, technischen Tests bis zur schmalen Spitze der aufwändigen, geschäftsnahen Tests. Je höher die Stufe, desto funktionaler, kostspieliger und langsamer werden die Tests – gleichzeitig steigt ihre Bedeutung für die fachliche Absicherung.



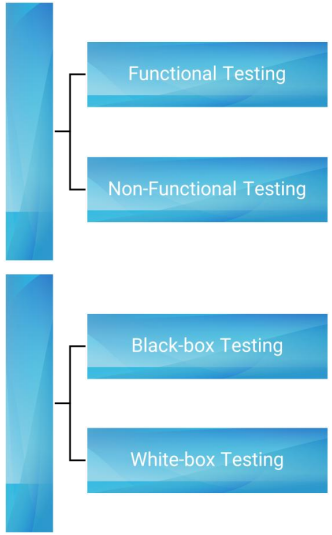
Testarten

Testarten beschreiben unterschiedliche Perspektiven, aus denen ein System geprüft werden kann. Sie sind unabhängig von den Teststufen anwendbar und ergänzen diese durch spezifische Fragestellungen. Während die Teststufen festlegen, wann im Entwicklungsprozess getestet wird, beantworten Testarten die Frage, wie und unter welchem Fokus getestet wird. Eine einheitliche Definition von Testarten existiert in der Praxis jedoch nicht; Begriffe werden oft branchen- oder unternehmensspezifisch unterschiedlich verwendet. Für eine erste Orientierung stützen wir uns daher auf den ISTQB-Standard.

Hier werden zwei Klassifizierungsrichtungen unterschieden:

- **Funktional vs. Nicht-funktional**
  Diese Unterscheidung bezieht sich auf den inhaltlichen Fokus der Prüfung. Funktionale Tests beantworten die Frage, ob das System tut, was spezifiziert wurde. Nichtfunktionale Tests bewerten hingegen Eigenschaften wie Leistung, Sicherheit, Benutzerfreundlichkeit oder Barrierefreiheit. Die Abgrenzung der nicht-funktionalen Eigenschaften orientiert sich an den Qualitätsmerkmalen der ISO/IEC 25010.
- **Black-Box vs. White-Box**
  Diese Perspektive betrachtet die Herleitung der Tests. Black-Box-Tests orientieren sich an Spezifikationen oder User Stories und prüfen ausschließlich über Schnittstellen und beobachtbares Verhalten. White-Box-Tests hingegen greifen auf die innere Struktur von Code oder Architektur zu und leiten daraus ihre Prüfungen ab.

Mit diesen beiden Klassifizierungsrichtungen entsteht ein vielseitiges Raster, das unterschiedliche Blickwinkel auf Qualität vereint. Es ermöglicht, Tests sowohl nach ihrem inhaltlichen Fokus (funktional oder nicht-funktional) als auch nach ihrer Herleitungsmethode (Black-Box oder White-Box) einzuordnen. Damit wird sichergestellt, dass Testaktivitäten systematisch geplant und durchgeführt werden können – unabhängig von Stufe, Vorgehensmodell oder technischer Umsetzung.



09

Security & Datenschutz

Durch einen höheren Grad an Vernetzung ist Enterprise Software heute viel exponierter als vor wenigen Jahren. Die Risiken von Cyber-Angriffen nehmen zu. Um ihnen gezielt zu begegnen, sind mgm Expert:innen einer auf Security spezialisierten Unternehmenseinheit von Anfang an im Projekt involviert. Mit dem Security Toolset ATLAS sorgen außerdem automatisierte Security-Analysen dafür, dass Schwachstellen und Konfigurationsprobleme schnell erkannt und beseitigt werden können.

Security by Design und Lean Application Security

A12 folgt dem Grundsatz Security by Design. Sicherheitsanforderungen werden von Anfang an berücksichtigt, um potenziellen Schwachstellen präventiv vorzubeugen. Security-Experten begleiten alle Phasen der Entwicklung – von frühen Anforderungen über Architekturentscheidungen bis hin zu Abnahmetests.

Im Rahmen von A12-Projekten setzen wir auf den selbst entwickelten Ansatz **Lean Application Security**. Die Basis bildet die Security-Awareness aller Stakeholder sowie das Know-how des Entwicklungsteams. Alle Softwareentwickler und Projektleiter bei mgm erhalten entsprechende Schulungen, die regelmäßig aufgefrischt werden. Security-Experten begleiten die Projekte. Sie überwachen die Einhaltung der selbst auferlegten Qualitätsstandards und erstellen eine iterative Bedrohungsanalyse („Threat Modeling"), die zu jedem Zeitpunkt Auskunft über die Bedrohungen der vorliegenden Anwendung gibt. Die transparente Darstellung dieser Bedrohungen wiederum schärft die Wahrnehmung des Teams bzgl. der notwendigen Schritte zur Vermeidung entsprechender Schwachstellen. Durch „Lean Security" wird eine Anwendung an den abschließenden Penetrationstest gegeben, die bereits von Grund auf sicher gebaut ist, statt, wie oft üblich, allein über den Penetrationstest zu versuchen, die Anwendung sicher zu machen.

mgm ATLAS

Um einen möglichst hohen Grad an Sicherheit zu gewährleisten, setzt mgm auf automatisierte Security-Analysen - sowohl bei der A12-Plattform als auch bei A12-Projekten. Dabei kommt das eigens entwickelte Security Toolset ATLAS zum Einsatz. Als flexible Plattform für Security-Scanner integriert ATLAS eine Reihe von Werkzeugen wie OWASP ZAP und Nikto. Es ermöglicht automatisierte Security Tests und stellt ein konsolidiertes Reporting bereit – auch durch die Integration von Reports in Sonarqube. ATLAS prüft u.a. auf bekannte Schwachstellen in Komponenten von Drittanbietern, erkennt Konfigurationsprobleme wie fehlende HTTP Security Header und testet, wie robust APIs gegenüber Attacken wie Injektionsangriffen sind.

Insgesamt vereint ATLAS Analysen aus folgenden Bereichen:

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

ATLAS wird direkt in die Build-Pipeline integriert. Welche Scans im Detail ausgeführt werden sollen, lässt sich einfach in einer YAML-Datei konfigurieren. Ein Vulnerability Management Dashboard stellt die Ergebnisse der Scans übersichtlich dar.

Die Nutzung von ATLAS fördert in A12-Projekten einen Shift-Left-Entwicklungsansatz für sicherheitsrelevante Aspekte und minimiert die Risiken von Schwachstellen, die erst spät erkannt werden. Da das Security Toolset von Anfang an in den Build-Prozess integriert werden kann, liefert es über den gesamten Entwicklungsprozess wertvolles Feedback über mögliche Sicherheitslücken.

A12-Projekte bekommen im Vulnerability Management Dashboard außerdem eine Einschätzung und Bewertung von Findings, die dem A12-Team bekannt sind. Dies begünstigt eine einheitliche Vorgehensweise im Schließen von Sicherheitslücken.

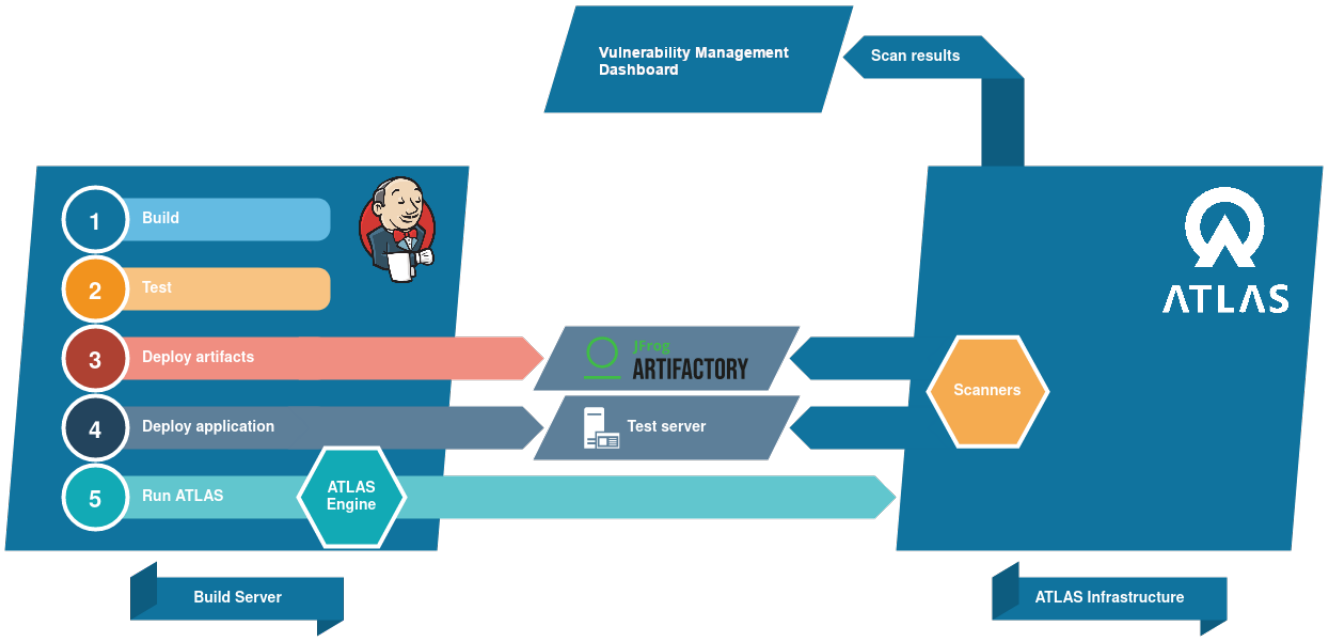


Abbildung 21 – *Grundlegende Architektur von ATLAS*

A12 Security Guidelines

Die A12 Security Guidelines fassen eine Reihe von Richtlinien, Best Practices und Empfehlungen für den sicheren Einsatz von A12 zusammen. Sie stehen auf der Dokumentationsplattform GetA12 bereit und skizzieren ausgehend vom A12 Project Template, wie eine sichere Standardkonfiguration aussieht. Neben der Absicherung von Service Endpoints und dem Ansatz für eine Logging-Strategie unter Berücksichtigung von Vorgaben der Datenschutz-Grundverordnung enthält die Dokumentation Tipps für eine sichere Konfiguration von Keycloak als Identity Provider und Empfehlungen zum Einsatz von Security Headern.

Threat Modeling (Bedrohungsanalyse)

Die Sicherheit von A12 als Plattform gewährleistet mgm durch die vielfältigen, oben beschriebenen Maßnahmen. Sobald jedoch auf Basis von A12 ein neues Produkt entsteht, ist dieses Produkt individuellen Bedrohungen ausgesetzt, die u.a. von den verarbeiteten Daten, der Exposition z.B. im Internet sowie den Umsystemen abhängen. Diese Bedrohungen gilt es frühzeitig zu identifizieren und geeignet zu behandeln, d.h. ihnen entgegen zu wirken (Prävention), sie zu beobachten (Reaktion) oder dieses Risiko zu akzeptieren, weil beispielsweise das Risiko gering und die Kosten für Maßnahmen unverhältnismäßig hoch wären. Dieser Prozess der Identifikation von individuellen Bedrohungen und der Bewertung möglicher Gegenmaßnahmen heißt Threat Modeling und wird von mgm als Dienstleistung im Rahmen der Entwicklung einer A12-Anwendung angeboten, damit diese Anwendung so viel Sicherheit wie nötig erhält - ohne unnötige Kosten für überflüssige Maßnahmen zu verursachen.

Datenschutz: Verantwortung im Projektkontext

In der Regel verarbeitet eine A12-Anwendung auch personenbezogene Daten – seien dies Namen, E-Mail-Adressen, Anmeldeinformationen oder vergleichbare Angaben. A12 berücksichtigt Datenschutzanforderungen von Beginn an und unterstützt eine gesetzeskonforme Umsetzung in allen Projektphasen. Die Plattform folgt dem Prinzip „Privacy by Design & Default“ und bietet bereits im A12 Project Template datenschutzfreundliche Voreinstellungen.

Die **A12 Security Guidelines** in Verbindung mit den mgm-internen **Secure Software Development Guidelines** enthalten praxisnahe Empfehlungen zur sicheren Konfiguration von Service Endpoints, zur Umsetzung einer Logging-Strategie unter Berücksichtigung geltender Datenschutzvorgaben sowie zur Einbindung von Identity Providern wie Keycloak.

Bereits bei der Konzeption einer Anwendung – insbesondere bei der Aufnahme funktionaler Anforderungen – sollten Aspekte wie Rollen- und Rechtemanagement, die Minimierung personenbezogener Daten, Speicherdauer und Datenlöschung sowie angemessene technische und organisatorische Maßnahmen bei der Verarbeitung berücksichtigt werden.

Die Verantwortung für die Einhaltung datenschutzrechtlicher Anforderungen liegt bei der jeweiligen Kunden- bzw. Projektorganisation. In enger Abstimmung mit den zuständigen Datenschutzverantwortlichen sind dabei auch Themen wie die Rechtmäßigkeit der Verarbeitung, die Wahrung von Betroffenenrechten, die Durchführung von Risikoanalysen und Datenschutz-Folgenabschätzungen sowie die Pflege des Verzeichnisses von Verarbeitungstätigkeiten zu berücksichtigen.

Durch die enge Verzahnung von Security und Datenschutz entsteht ein ganzheitlicher Schutzansatz, der regulatorische Anforderungen erfüllt und gleichzeitig die technische Sicherheit stärkt. A12-Projekte profitieren von klaren Leitlinien, die helfen, personenbezogene Daten verantwortungsvoll zu verarbeiten – effizient, transparent und im Einklang mit den hohen europäischen Standards.

Anhang A

Projektteam - Rollen und Aufgaben

ROLLE	AUFGABEN
Projektleitung organisatorisch	⊕ Planung und Steuerung der Aufgaben ⊕ Verfolgung von Status und Fertigstellungsgrad ⊕ Berichtswesen ⊕ Ansprechpartner für Projektleitung des Kunden ⊕ Ressourcen- & Risikomanagement ⊕ Änderungswesen & Eskalationsverfahren ⊕ Teambuilding & -zusammenhalt
Projektleitung technisch	⊕ Technische Koordination ⊕ Technische Verantwortung ⊕ Delivery Management ⊕ Release Management ⊕ Leitung des Entwicklungsteams ⊕ Berichtswesen ⊕ Risiko- & Problemmanagement

A. Projektteam - Rollen und Aufgaben

ROLLE	AUFGABEN
Project Management Office (PMO)	⊕ Administrative Projektunterstützung ⊕ On-/Offboarding ⊕ Rechnungsvorbereitung
Facharchitekt	⊕ Konzeption einer konsistenten fachlichen Logik ⊕ Übergeordnetes fachliches Anwendungsdesign
Business-Analyst	⊕ Fachlicher Ansprechpartner für den Auftraggeber ⊕ Anforderungsanalyse ⊕ Konzeptkonsolidierung und fachliche Dokumentation
Modellierer	⊕ Umsetzung von fachlichen und technischen Anforderungen in A12-Modellen ⊕ Abgleich der Anforderungen mit dem A12-Featureumfang auf Modellierungsebene ⊕ Fit-Gap-Analyse Entwicklung vs. Modellierung
Software Architekt	⊕ Technischer Ansprechpartner für den Auftraggeber ⊕ Gesamtarchitektur und beteiligte Systeme ⊕ Steuerung der Entwicklungsaufgaben ⊕ Überwachung der nicht-funktionalen Anforderungen
Entwickler (Frontend, Backend, Full-Stack)	⊕ Feinkonzeption, Programmierung und Dokumentation der Umsetzung der fachlichen Anforderungen auf Basis des Architekturentwurfs ⊕ Erstellung und Durchführung von Unit-Tests ⊕ Durchführung von Code-Reviews ⊕ Ergebnispräsentation in Formaten wie Sprint-Reviews
QS-Engineer	⊕ Konzeption der entwicklungsbegleitenden QS-Aktivitäten ⊕ Definition von Testfällen und Testdaten ⊕ Durchführung und Dokumentation der Tests ⊕ Unterstützung für Vorbereitung und Durchführung für die Teilreleases und die Gesamtlösung ⊕ Kundenbegleitung bei Abnahmetests

A. Projektteam - Rollen und Aufgaben

ROLLE	AUFGABEN
Tech-QS-Engineer	⊕ Konzeption der erforderlichen QS-Automatisierung ⊕ Konzeption und Durchführung von Last- und Performance-Tests ⊕ Sicherstellung der Verfügbarkeit der Testumgebung pro Release
DevOps Engineer	⊕ Build- und Deployment-Prozess ⊕ Bereitstellung notwendiger Infrastruktur (Vorbereitung Testsystem, Pflege und gemeinsames Repository) ⊕ Konfigurationsmanagement ⊕ Release Management
Security Engineer	⊕ Sicherstellung der Einhaltung von Security-Richtlinien in der laufenden Entwicklung ⊕ Application Security ⊕ Security Tests ⊕ Penetration Tests
Designer / UI/UX Experte	⊕ Konzeption von Styleguides ⊕ Entwurf der Benutzeroberflächen ⊕ Umsetzung der Barrierefreiheit ⊕ Anwendung grundsätzlicher Bedienparadigmen ⊕ Erarbeitung und Priorisierung der Anforderungen zur Ergonomie
Product Owner* (in Vertretung, i.d.R. vom AG)	⊕ Anforderungsaufnahme sowie Erstellung der Produkt Vision und Roadmap ⊕ Klärung fachlicher Anforderungen ⊕ Priorisierung von Anforderungen ⊕ Regelmäßiges Feedback mit allen Projektbeteiligten ⊕ Abnahme und Verifizierung ⊕ Verantwortlich für den Produkterfolg
Scrum Master*	⊕ Projektstruktur (Jira/Confluence) ⊕ Verantwortung der Prozesse und Workflows inkl. kontinuierlicher Verbesserung ⊕ Metriken & Kennzahlen ⊕ Verantwortung Kollaboration und Moderation ⊕ Teamcoach für Produktivitätssteigerung

Anhang B

Projektmanagement-Methodik

mgm stellt sicher, dass die Zusammenarbeit der im Rahmen der Projektierung Beteiligten reibungslos abläuft und alle Ziele effizient und ressourcenschonend erreicht werden. Welche Projektmanagement-Methodik dabei zum Einsatz kommt, richtet sich im Detail nach den Strukturen und Anforderungen des Auftraggebers. Im Folgenden skizzieren wir eine Reihe wichtiger Faktoren und Ansätze, die sich in unserer Projektpraxis bewährt haben.

- **Stakeholder-Analyse und Management**

  Zur erfolgreichen Projekt-Umsetzung und für die Akzeptanz der Anwendung ist es entscheidend, die Interessen wesentlicher interner und externer Stakeholder zu berücksichtigen – und sie je nach Projekt-Relevanz entsprechend einzubinden. Typische Stakeholder sind Vertreter von System-Schnittstellen, Anwendergruppen, Administratoren und Abteilungsleiter.

- **Definition von Rollen, Zuständigkeiten und Verantwortlichkeiten**

  Sowohl die Rollen rund um technische Entwicklungsleistungen als auch das organisatorische Management müssen klar definiert und transparent sein. Dafür werden zunächst Kriterien festgelegt, die die Verantwortlichkeit im Rahmen einer Aufgabe näher beschreibt – zum Beispiel gemäß der RACI-Technik. Dann werden für jedes Aufgabenfeld die Verantwortlichkeiten nach dem festgelegten Raster jeder Rolle durchdekliniert. mgm setzt hier auf ein Verantwortlichkeits-Zuordnungs-Diagramm. So lassen sich Missverständnisse vermeiden, die sonst zu Konfliktsituationen und einem schwer kalkulierbaren Risiko führen.

- **Risikoanalyse und Vorbereitung des Risikomanagements**

  Zur Identifikation von Projektrisiken empfehlen wir eine ausführliche Risikoanalyse. Sie identifiziert die Projektrisiken in den jeweiligen Implementierungsphasen und schätzt ihre Eintrittswahrscheinlichkeit ein, bewertet das Schadensausmaß und beschreibt Gegenmaßnahmen. Dabei werden sowohl Risiken in der technischen Umsetzung und im Bereich Personal als auch externe Risiken berücksichtigt. Die Analyse sollte übergehen in ein kontinuierliches Risikomanagement.

- **Entwicklung eines Kommunikations- und Berichtsweges**

  Ein Kommunikationsplan beschreibt, welche Stakeholder welche Art von Information wann und mit welcher Methode über welches Medium erhalten. Im Rahmen einer Kommunikationsmatrix wird mit den Stakeholdern (z.B. Projektleiter, Lenkungsausschuss, Betriebsrat) verabredet, welche Medien in welcher Weise genutzt werden sollten. Die Auswahl eines optimalen Kommunikations- und Berichtsweges hängt von den definierten Projektrollen sowie von individuellen und kulturellen Aspekten ab.

- **Anlegen eines Projektglossars**

  Jedes Softwareprojekt bringt eine Vielzahl technischer und fachlicher Aspekte mit, die unterschiedlich ausgelegt werden können. Um möglichst früh ein gemeinsames Verständnis zu fördern, sollte direkt nach dem Projektstart ein projektbezogenes Glossar angelegt werden. Es enthält die wichtigsten Begriffsdefinitionen und hilft, Missverständnisse zu vermeiden. Im Projektverlauf können sukzessive weitere Definitionen hinzutreten oder die anfänglichen Definitionen geschärft werden.

Entscheidungshilfe für die passende Methodik

Manche Situationen sind klar umrissen und vorhersehbar. Wenn wir ein Gartenlabyrinth aus der Vogelperspektive betrachten, können wir einfach einen Weg vom Platz in der Mitte durch verzweigte Wege zum Ausgang finden. Wir können einen Plan entwerfen, der verlässlich zum Ziel führt. In der Enterprise Software-Entwicklung gibt es aber immer wieder Situationen, die dynamisch und unvorhersehbar sind. Hier durchschreiten wir eher einen wilden Dschungel, um ein dahinterliegendes Ziel zu erreichen. Der Weg ist nicht vollständig vorhersehbar. Wettereinflüsse wie Starkregen können Bereiche unpassierbar machen. Hier hangeln wir uns Stück für Stück in Richtung des Ziels, beobachten immer wieder unsere Umgebung und passen die Route entsprechend an.

In der Terminologie von Softwareprojekten sprechen wir in Situationen wie dem Gartenlabyrinth und dem klar definierbaren Plan von einem Wasserfallmodell. Im Fall des Dschungels sind wir in der Welt agiler Methoden, die ein iteratives, schrittweises Vorgehen in einer dynamischen Umgebung vorsehen. Doch wann liegt ein Labyrinth vor uns und wann ein Dschungel? Spoiler Alert: In den meisten Softwareprojekten gibt es beide Fälle in unterschiedlichen Schattierungen. Manche Aspekte sind bekannt und sehr gut planbar, andere sind ungewiss oder unterliegen einer großen Dynamik.

Für die erfolgreiche Projektabwicklung ist es entscheidend, zu wissen, in welcher Situation wir uns gerade befinden – und wie wir in dieser Situation am besten navigieren, um ein angestrebtes Ziel zu erreichen. mgm setzt bei der Wahl einer passenden Methodik auf das Cynefin-Framework. Es unterscheidet vier Situationen, die jeweils verschiedene Handlungsempfehlungen mit sich bringen:

- **Einfach:** Die Situation ist bekannt und es gibt klare kausale Zusammenhänge. Die Anwendung von Best Practices führt verlässlich zum Ziel.
- **Kompliziert:** Die Situation ist nicht vollständig bekannt und erfordert eine eingehende Analyse. Ihr kann aber letztlich mit einem klar strukturierten Vorgehen und einem Plan begegnet werden. Gute Prognosen und valide Schätzungen sind möglich. Entscheidungen müssen aber früh getroffen werden und auf Veränderungen kann nur bedingt eingegangen werden.
- **Komplex:** Die Situation ist unübersichtlich und erfordert ein experimentelles Vorgehen. Die Lösungsfindung erfolgt schrittweise. Während Prognosen schwierig sind, bringt das Vorgehen eine schnelle Reaktionsfähigkeit mit. Entscheidungen können spät getroffen werden.
- **Chaotisch:** Die Situation ist unkontrollierbar, erfordert aber eine direkte Reaktion. Ein genaueres Assessment erfolgt nachgelagert.

Die Einordnung von Situationen in dieses Schema ist dabei nicht fix. Die Einschätzung einer Situation kann sich im Projektverlauf ändern. Wenn das Team zum Beispiel eine komplexe Situation tief genug durchdringt, kann sie zunächst als kompliziert – und später vielleicht als einfach – eingeordnet werden. Je mehr Situationen im Projektverlauf in die Kategorie „Einfach" fallen, desto besser. Hier lassen sich typischerweise Abläufe automatisieren und die digitale Dividende des Projekts erhöhen.

Agile Methodiken

Bei der Umsetzung setzt mgm auf gängige agile Methoden wie **SCRUM** oder **Kanban**.

Beide Methoden beschreiben einen iterativen Prozess, in dem Aufgaben, Funktionen und Kriterien in einem Product Backlog organisiert werden. Innerhalb dieses Backlogs werden sie priorisiert und anschließend an Hand der Priorisierung vom Team bearbeitet. Regelmäßige Feedbackgespräche mit allen Stakeholdern erlauben es, Prozesse und Funktionen zu re-evaluieren und besprochene Änderungen unvermittelt zu implementieren. Ein weiterer Vorteil eines agilen A12 Projektes ist die Vermeidung einer langen Planungsphase. Das Vorgehen sieht eine möglichst schnelle Erstellung einer lauffähigen ersten Produktversion vor, die iterativ erweitert wird - und zwar immer mit einem klaren Blick auf die Anforderungen des Auftraggebers und der künftigen Nutzer. Die Verwendung der A12-Plattform und des Low Code-Ansatzes beschleunigen die Entwicklung und sorgen dafür, dass bereits die erste Version eine vergleichsweise hohe Maturität aufweist.

Kundenspezifische Anpassung und Hybrid-Modelle

Da es im Sinne der Softwareentwicklung keine One-Fits-All-Strategie gibt, werden alle A12 Projekte kundenspezifisch angepasst. Dies kann bedeuten, Ansätze aus agilen sowie klassischen Methodiken zu vereinen. Meist wird dieser Ansatz gewählt, um die Nachteile einer einzelnen Methodik zu umgehen und den Gegebenheiten der Organisation des Auftraggebers gerecht zu werden. Die individuell strukturierte Gewichtung lässt es zu, nur einen Teil des Teams nach agilen Methodiken agieren zu lassen (z.B. die Produktentwicklung), während andere Teile des Teams (das übergeordnete Management, das Controlling oder Marketing) die Ansätze des klassischen Projektmanagements verfolgen. Gründe dafür können klassisch hierarchische Unternehmensstrukturen sein, in denen agiles Arbeiten schwer realisierbar ist. Die endgültige Gewichtung der klassischen und agilen Anteile der hybriden Lösung erarbeiten der Auftraggeber und mgm gemeinsam, wobei individuelle Bereiche anhand des Cynefin-Frameworks auf ihre Komplexität und damit einhergehende Planbarkeit geprüft werden.

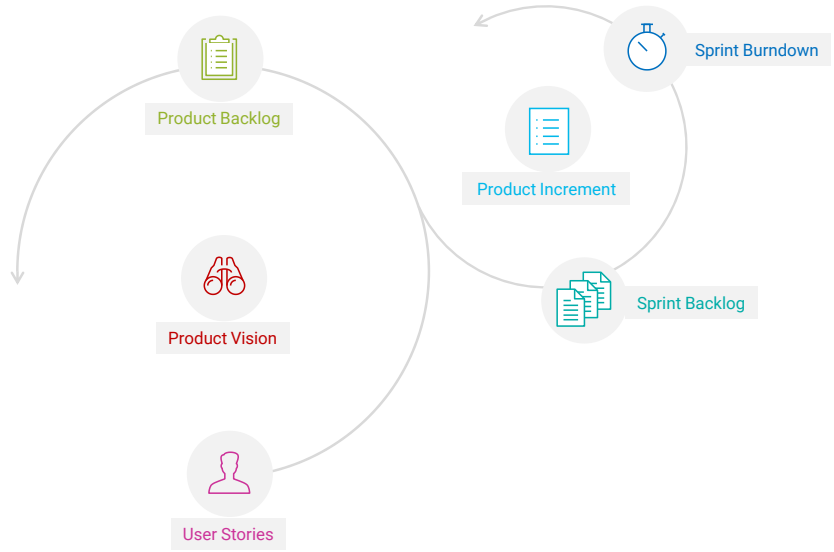


Abbildung B.1 – *Zusammenhänge der wichtigsten Artefakte bei der agilen Vorgehensweise*

ARTEFAKT	ERLÄUTERUNG
Product Backlog	Das Product Backlog ist eine strukturierte und priorisierte Liste aller verbleibenden Funktionen und Aufgaben. Der Product Owner trägt die Verantwortung für die korrekte Pflege des Backlogs, wobei er dazu die Parameter Geschäftswert, Risiko und Lieferdatum heranzieht.
Sprint Backlog	Das Sprint Backlog beinhaltet alle Spezifikationen derjenigen Aufgaben, die das Entwicklungsteam für die Umsetzung in einem Sprint vorsieht.
Produktinkrement	Das Produktinkrement entspricht einer Liste, die alle im aktuellen Sprint fertig gestellten Aufgaben und Funktionen beinhaltet. Die Hürde dieses Artefaktes ist die Definition des Status „Done“ und das heterogene Verständnis aller Teammitglieder. Ziel ist es, die fertig gestellten Aufgaben sobald wie möglich zu einem Minimum Viable Product (MVP) zusammen zu setzen. Sobald ein Produkt mit Kernfunktionen entwickelt wurde, wird dieses an den Nutzer herangetragen, und gemeinsam mit diesem weiterentwickelt. So minimiert sich das Risiko, dass ein Produkt an den Bedürfnissen der Zielgruppe vorbei entwickelt wird.
Sprint Burndown	Im Sprint Burndown wird erläutert, wie schnell die User Stories im Vergleich zur Planung abgearbeitet wurden.
User Stories	User Stories entsprechen untergeordneten Einzelaufgaben, die aus der Sicht des Kunden entwickelt werden. Häufig werden einem Produkt Funktionen oder Möglichkeiten hinzugefügt, obwohl nicht klar ist, ob der Nutzer diese benötigt oder überhaupt will. Diesen Umstand versuchen SCRUM Teams mit User Stories zu umgehen.
Produktvision	Eine gemeinsame Produktvision ermöglicht es einem Team, intrinsische Motivation zu entwickeln und den jeweiligen Arbeitsstand kontinuierlich mit dem Ziel abzugleichen.

Anhang C

Anforderungen systematisch aufnehmen

Ausgehend von langjähriger Erfahrung in großen Enterprise Software-Projekten hat mgm einen strukturierten Prozess für die Analyse und Erfassung von Anforderungen entwickelt. Eine Reihe von Vorlagen und Checklisten helfen dabei, im richtigen Moment die richtigen Fragen zu stellen und Anforderungen in einem angemessenen Detailgrad festzuhalten.

Die wichtigste Quelle für Anforderungen sind die im Projekt involvierten Personen. Mit Hilfe einer Stakeholder-Checkliste stellen wir sicher, alle wesentlichen Ansprechpartner für verschiedene Rollen zu identifizieren. Weitere, nicht-personenbezogene Quellen für Anforderungen können zum Beispiel Altsysteme und deren Dokumentation sowie gesetzliche Vorgaben und Normen sein.

Bei der Erhebung von Anforderungen setzen wir unter anderem auf Fragebögen, Interviews, Anforderungsworkshops, die Beobachtung / Mitarbeit in Prozessen der Anwendergruppe und das Prototyping. Die Beschreibung der Anforderungen erfolgt stets zielgruppenspezifisch und ergebnisorientiert. Gerade bei fachlichen, funktionalen Anforderungen können zudem gemeinsame Modellierungssessions zu einem besseren Verständnis beitragen.

Um Anforderungen zu konkretisieren, definieren wir - ergänzend zu etwaigen Abnahmekriterien des Auftraggebers - bedarfsgerecht auch Erfüllungs- und Akzeptanzkritierien. Sie enthalten wichtige Informationen für die Umsetzung und bilden zudem die Grundlage für Testfälle.

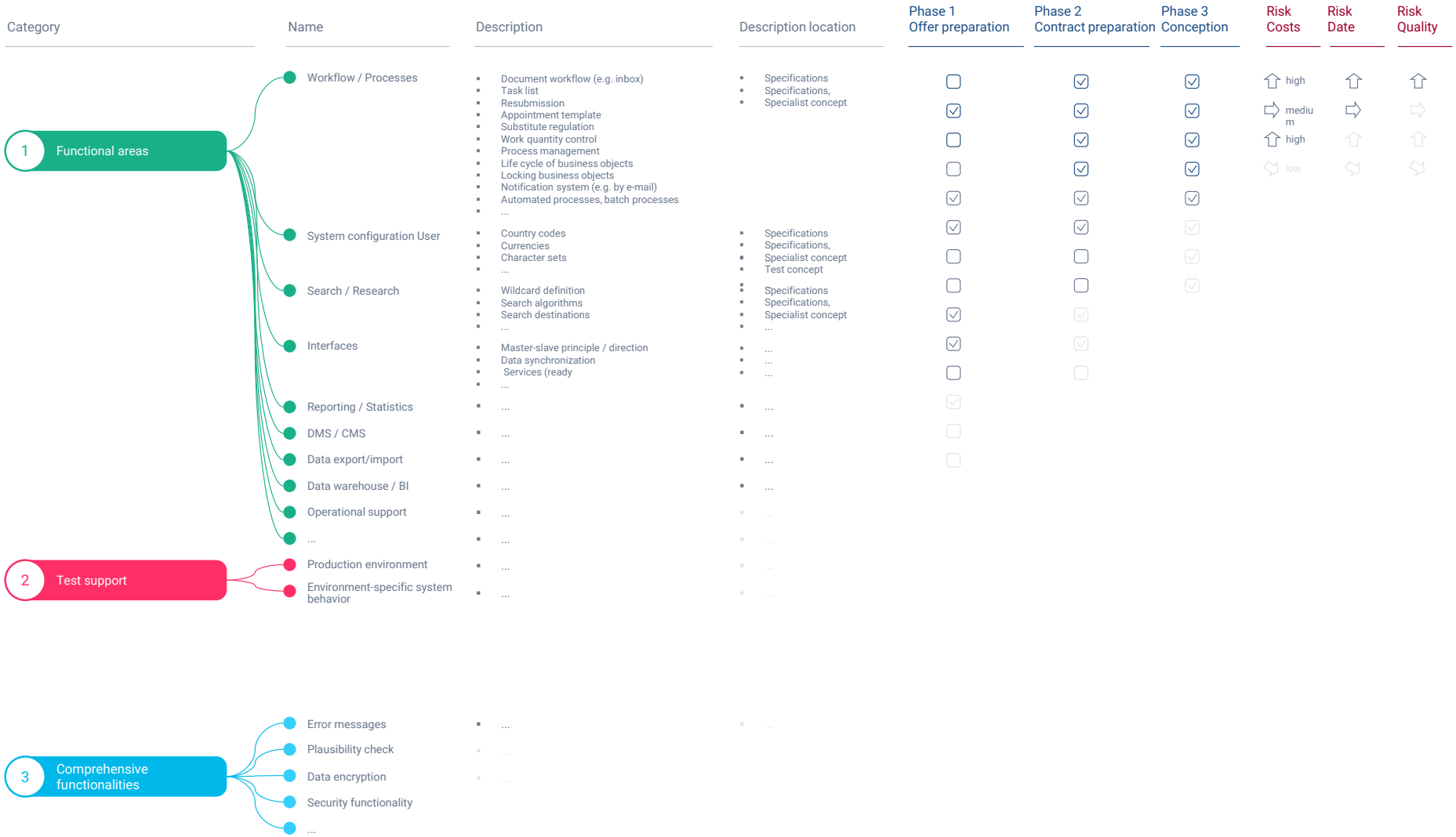


Abbildung C.1 – *Erhebung funktionaler Anforderungen*

C. Anforderungen systematisch aufnehmen

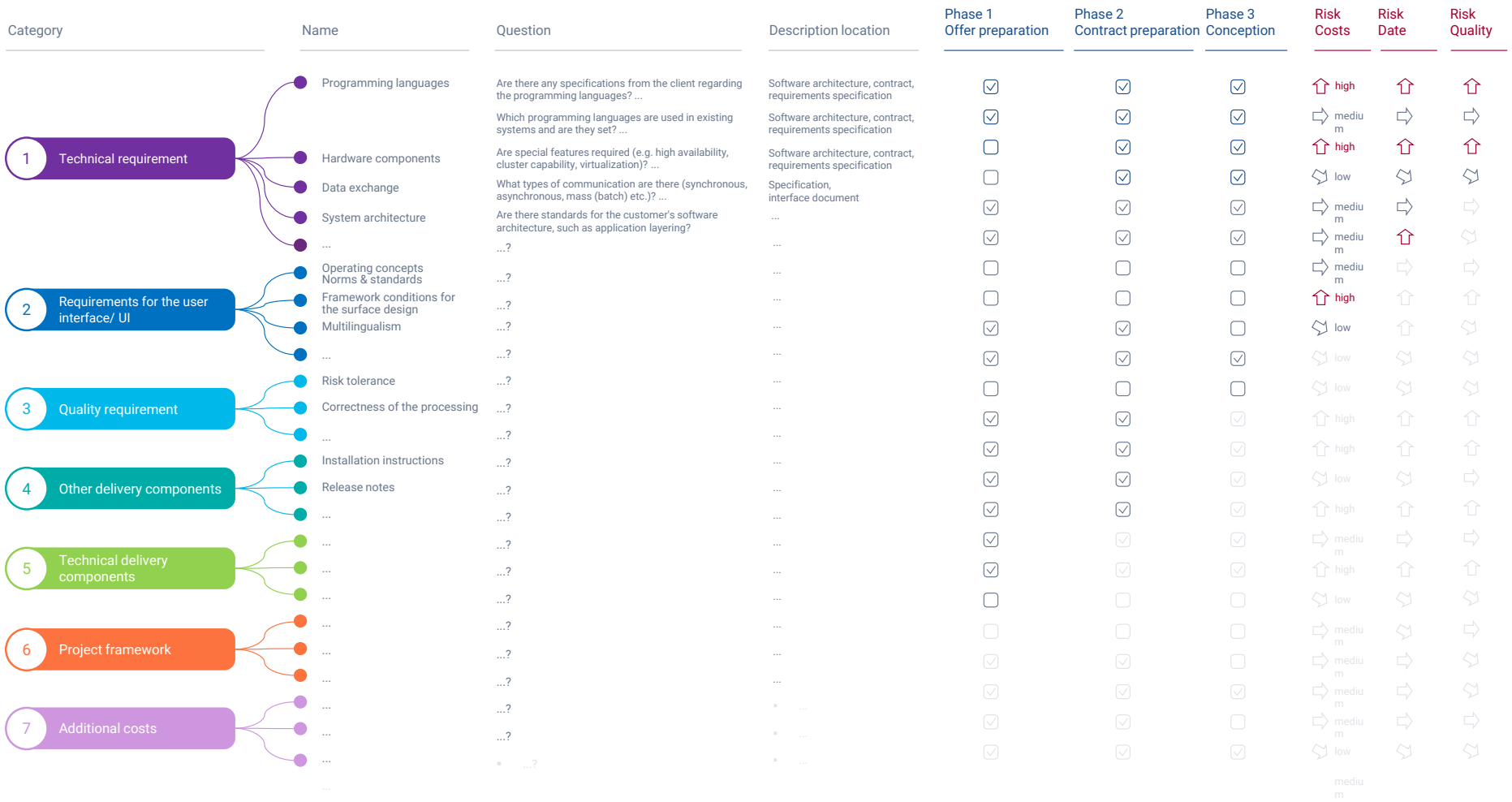


Abbildung C.2 – *Erhebung nichtfunktionaler Anforderungen*

Anforderungsanalyse im agilen Projekt

In agilen A12 Projekten wendet mgm ein fachbereichsübergreifendes Anforderungsmanagement an. Alle Anforderungen werden dabei aus Nutzerperspektive betrachtet und pro Fachbereich übergeordnet priorisiert. Die Priorisierung erfolgt in enger Abstimmung mit dem Auftraggeber. Dem zugrunde liegt die kontinuierliche Bewertung des jeweiligen Beitrags zur Zielerreichung, des Aufwand-Nutzen-Verhältnisses und etwaiger Umsetzungsrisiken. Um den Priorisierungsprozess zu institutionalisieren, hat sich in komplexeren Projekten ein regelmäßiges – z.B. 14-tägiges – Backlog Grooming bewährt.

Alle Anforderungen werden initial als User Stories (Stories) aus Nutzersicht erfasst. Eine weitere Kategorisierung erfolgt basierend auf der Relevanz der jeweiligen Anforderung. Ist diese überdurchschnittlich hoch oder das Thema übermäßig komplex, wird die User Story zum Epic. Ein Epic besteht in der Regel aus mehreren Stories, die zur Erfüllung des Epic abgeschlossen werden müssen. Dieses Schema bietet den Vorteil, dass es ein zweites Level der Priorisierung bietet, nämlich die Priorisierung der Epics, sowie die Priorisierung der einzelnen Stories innerhalb des Epics.

In bestimmten Fällen kann es dazu kommen, dass ein agiles Projekt oder Teilaspekte ein Feinkonzept im klassischen Sinn benötigen. mgm bevorzugt es in diesem Falle, nur den ersten Meilenstein detailliert zu planen. Die darauffolgenden Meilensteine werden zu diesem Zeitpunkt ausschließlich geschätzt und zu Beginn der jeweiligen Projektphase anhand aktueller Ergebnisse und Erkenntnisse weiter ausgeführt. Durch dieses Vorgehen inkludiert mgm den planerischen Aspekt eines klassischen Projektmanagements in ein agiles Vorgehen. Dies bedeutet eine maßgebliche Reduktion des initialen Planungsaufwandes und ermöglicht eine kontinuierliche, präzise Planung des laufenden A12 Projektes mit hoher Inklusion des Auftraggebers.

Alle Anforderungen werden in einem digitalen Anforderungsmanagement-Tool (Jira) zentral in Tickets dokumentiert. So sind sie – ebenso wie der Umsetzungsstatus – für alle Beteiligten transparent einsehbar und können kommentiert werden. Die Beschreibung der Anforderungen erfolgt stets zielgruppenspezifisch und ergebnisorientiert. Ergänzende Konzepte werden in einem Projekt-Wiki abgelegt und sind bidirektional mit den Tickets aus dem Anforderungsmanagement-Tool verknüpft.

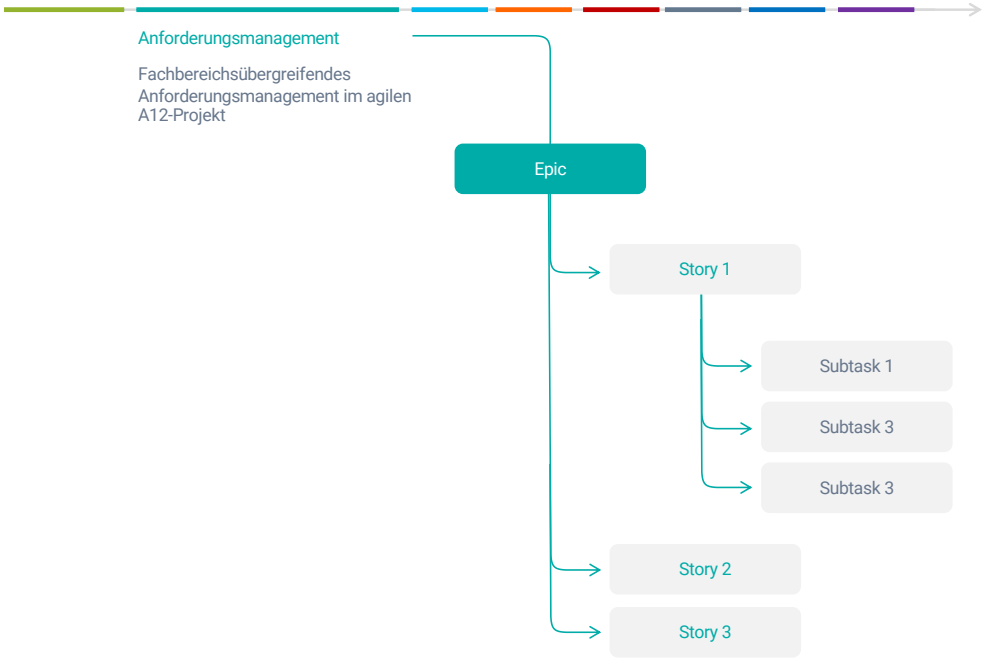


Abbildung C.3 – *Priorisierungsebenen von Anforderungen*

Glossar

A11Y Abkürzung für Accessibility - Barrierefreiheit. 30

A12-Modell Modell, das aus dem modellbasierten Entwicklungsansatz von A12 hervorgegangen ist - z.B. *Document Model* im Bereich der Datenmodellierung und *Form Model* im Bereich der UI-Modellierung. 21, 22, 40

A12-Komponente Technische Komponente, die Teil der A12-Laufzeitplattform ist - z.B. ”Data Services” und ”Workflows”. 21

A12-Anwendung Anwendung, die auf Basis der Plattform A12 entwickelt wird. 2, 25, 29, 30, 36, 37, 41, 53

A12-Projekt Individuelles Softwareprojekt, in dessen Rahmen eine A12-Anwendung erstellt wird. 2, 13, 14, 15, 25, 27, 30, 52

A12-Team Team innerhalb von mgm, das die Entwicklung der A12-Plattform verantwortet. 2, 24, 32, 52

ATLAS von mgm entwickeltes und angebotenes Security-Toolset, siehe auch https://www.mgm-sp.com/mgm-atlas. 2, 41, 50, 52

BPS Business Professional Services - mgm-seitiges Team als Schnittstelle zwischen Projekt-Teams und dem A12-Team, das bei fachlichen Fragen unterstützt. 24

Projektteam Team, das im Rahmen eines A12-Projekts die Entwicklung einer A12-Anwendung verantwortet. 14, 28

PS Professional Services - Überbegriff für BPS und TPS. 15, 19, 24

QS Abkürzung für *Qualitätssicherung*. 25, 42

SME Simple Model Editor - das zentrale Modellierungswerkzeug der A12-Plattform. 22

TPS Technical Professional Services - mgm-seitiges Team als Schnittstelle zwischen Projekt-Teams und dem A12-Team, das bei technischen Fragen unterstützt. 24

Haben wir Ihr Interesse geweckt?
Sprechen Sie mit uns darüber, wie A12 und klare Projektstandards Ihre Projekte nachhaltig voranbringen.

www.mgm-tp.com/a12.html

mgm technology partners GmbH

Taunusstr. 23
80807 München

Tel +49 89 / 35 86 80-0

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

Innovation Implemented.