generic de

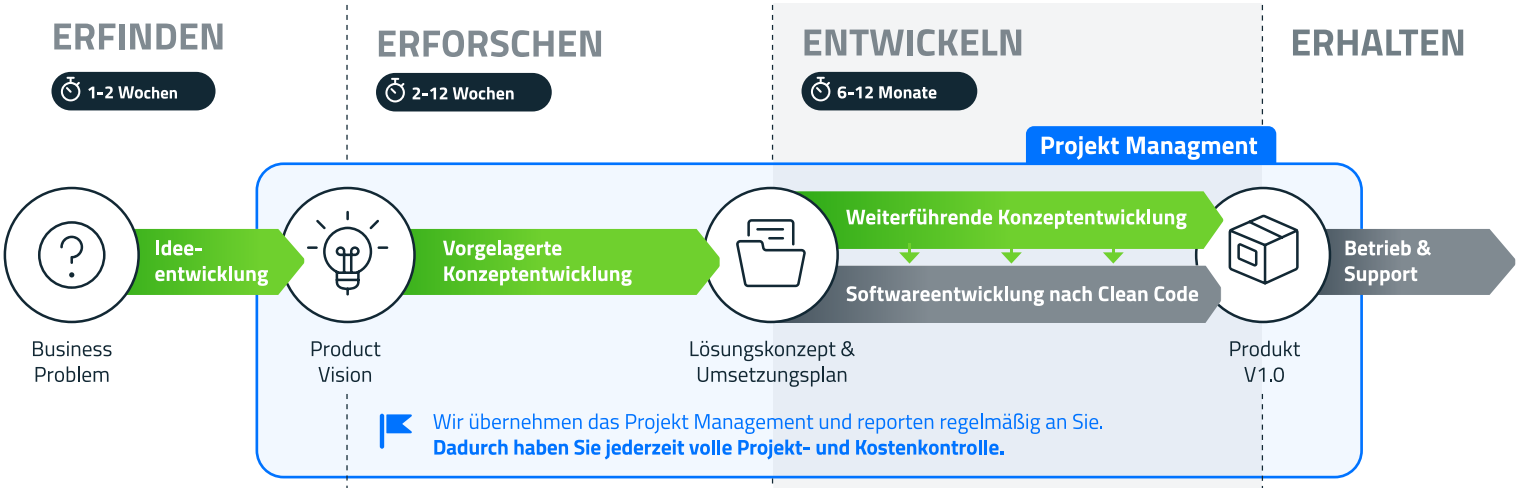
SERVICE-BESCHREIBUNG

Softwareentwicklung

ERFINDEN | ERFORSCHEN | **ENTWICKELN** | ERHALTEN

Das Richtige richtig entwickeln

Über 25 Jahre haben wir unseren Entwicklungsprozess verfeinert und jede Projektphase iterativ optimiert. Unser Ziel: Wir wollen Sie dabei unterstützen das perfekt zu Ihrem Business Case, zu Ihren Anwendern und zu Ihrer Systemlandschaft passende Softwareprodukt zu entwickeln. Und das auf effiziente, nachhaltige und transparente Art und Weise.



Business Probleme identifizieren, priorisieren und Lösungsideen generieren

Ihre Mehrwerte

- **Ihre Problemstellung ist definiert**
- **Verschiedene Lösungsideen sind erarbeitet**
- **Die vielversprechendsten Ideen sind validiert**

Entwickelbares Lösungskonzept erstellen, Produkt-Risiken minimieren, Projekt aufsetzen und Umsetzung planen

Ihre Mehrwerte

- **Ihr Produkt ist so konzipiert, dass es perfekt zum Business Case, den Usern und in Ihre Systemlandschaft passt**
- **Entwickelbare Arbeitspakete für die erste Iteration sind erstellt und eingetaktet**
- **Die Entwicklung ist durchgeplant und Sie erhalten eine realitätsnahe Einschätzung der Gesamtkosten**

Produktentwicklung im Dual Track Agile (konzeptionelle Arbeiten und Softwareentwicklung laufen parallel in Iterationen), kontinuierlicher Release von Produktinkrementen

Ihre Mehrwerte

- **Hohe Anforderungsqualität durch kontinuierliches Benutzerfeedback**
- **Investitionssicherheit durch Clean Code Development**
- **Hohe Flexibilität durch agile Projektführung**
- **Keine versteckten Lock-In-Effekte – der entwickelte Quellcode ist Ihr Eigentum**

Betrieb , ggf. Hosting und Support auf verschiedenen Leveln

Ihre Mehrwerte

- **Falls gewünscht, benötigen Sie keine eigene Infrastruktur**
- **Kontinuierliche Empfehlungen für neue Technologien**
- **Kontinuierliche Sicherheitsupdates**

Wie wir im Detail vorgehen, erfahren Sie in diesem Dokument

Inhalt

1 Warum mit generic.de entwickeln?

Woher wissen Sie, ob Ihre Softwareproduktidee die Wünsche Ihrer Kunden, Ihrer Mitarbeiter und Ihrer IT sowie Ihre eigentlichen Businessziele erfüllt? Wie können Sie die Produktentwicklung trotz neuer Anforderungen und Erkenntnisse auf Ziel halten, ohne dass es Ihren Zeitplan oder Ihr Budget sprengt? Wie schützen Sie Ihre Investition in die individuelle Softwarelösung?

Wir entwickeln für Sie das Richtige richtig – durch die Kombination aus Dual Track Agile und Clean Code Development.

1.1 Dual Track Agile: das Richtige entwickeln

Verschiedene Studien und Reports belegen, dass Softwareentwicklungsprojekte nach dem Wasserfallmodell oft scheitern – insbesondere, wenn es sich um komplexe Unternehmenssoftware mit unbeständigen Anforderungen handelt.[1] Agile Entwicklungsmethoden führen häufiger zum Erfolg. Allerdings bergen agil geführte Projekte, je größer und komplexer sie werden, die Gefahr aus dem Ruder zu laufen, was Entwicklungsdauer und Entwicklungskosten angeht.

Wir verbinden die Vorteile beider Herangehensweisen und entwickeln nach Dual Track Agile mit begleitendem Delivery Management.

		Wasserfall-Projektführung	Agile Software-entwicklung	Dual Track Agile mit Delivery Management
PROJEKT	Kostenkontrolle	✓	✗	✓
	Planungssicherheit	✓	~	✓
	Risikomanagement	~	~	✓
	Entwicklungseffizienz	✓	~	✓
	Flexibilität	✗	✓	✓
	Transparenz	✗	~	✓
PRODUKT	Kontinuierliches Stakeholder-Feedback	✗	✓	✓
	Anforderungsqualität	~	✓	✓
	Früher Markteintritt	✗	~	✓
	Innovationsfähigkeit	✗	✓	✓
	Anpassungsfähigkeit	✗	✓	✓
	Skalierbarkeit	~	~	✓

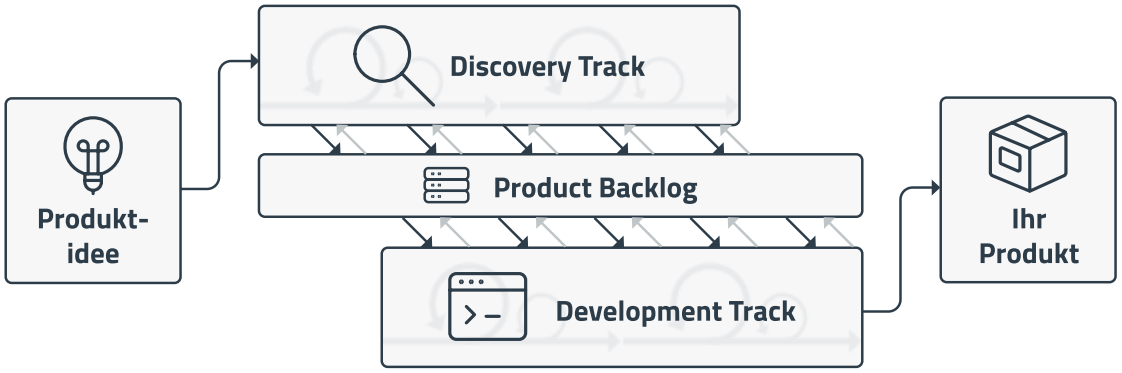
✗ nicht/kaum gewährleistet ~ bedingt gewährleistet ✓ gewährleistet

[1] Vgl. u. a.: https://doi.org/10.24368/jates.v11i4.279; http://doi.org/10.29322/IJSRP.13.08.2023.p14015; https://doi.org/10.1016/j.procs.2021.01.227

Dual Track Agile parallelisiert die Konzeptions- und Entwicklungsphase Ihres Produkts, wobei die Konzeption im sogenannten Discovery Track und die Umsetzung im Development Track abläuft. Beide Tracks arbeiten agil und damit iterativ. Die Art der Arbeit ist verschiedenartig:

- **Discovery Track:** Erheben der Anforderungen an das Softwareprodukt, Übersetzen in entwickelbare Arbeitspakete (Inkremente) und Planen von deren Umsetzung.
- **Development Track:** Umsetzen der im Discovery Track geplanten Inkremente in Quellcode, und Ausliefern des einsatzbereiten Produkts.

Im Discovery Track wird ein Arbeitspaket so lange erforscht und konzipiert, bis es „gut genug" ist. Erst wenn wir sicher sind, das Richtige zu bauen, kommt das Arbeitspaket in den Development Track. Während dort das Arbeitspaket entwickelt wird, konzipiert der Discovery Track die nächsten Pakete. Ist ein Arbeitspaket fertiggestellt, wird es in den produktiven Betrieb genommen. Dort prüfen wir, ob es den gewünschten Mehrwert bietet. Die Erkenntnisse hieraus fließen zurück in den Discovery Track.



Durch die iterative und inkrementelle Arbeit bleiben Sie flexibel und können auf sich verändernde Anforderungen reagieren. Das Ergebnis ist ein Softwareprodukt, das sowohl Ihre Businessziele erfüllt als auch eine sehr hohe Nutzerakzeptanz aufweist.

Gleichzeitig schafft die Methode ein besser kontrollierbares Umfeld und ist planungssicherer, was Entwicklungszeit und -kosten angeht. Dies stellen wir durch unser Delivery Management sicher, welches das Projekt kontinuierlich kontrolliert und überwacht.

Sie geben die Vision für Ihr Produkt vor, wir übernehmen den Rest und führen Sie methodisch zu Ihren Zielen. Natürlich arbeiten wir dabei sehr eng zusammen.

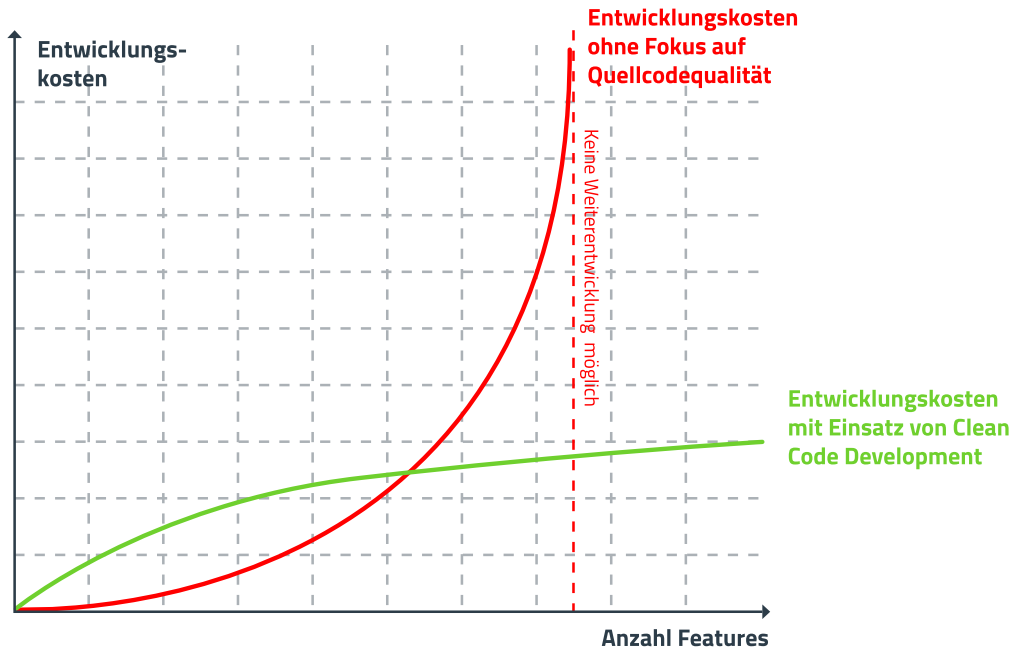
1.2 Clean Code Development: richtig entwickeln

Um agil, aber dennoch effizient entwickeln zu können, muss der Quellcode Ihres Softwareprodukts flexibel anpassbar sein. Wie erwähnt, läuft Konzeption und Programmierung Ihres Produkts parallel. Ändern sich Anforderungen oder wird festgestellt, dass bereits entwickelte Features nicht den erwarteten Mehrwert bieten, wäre es fatal, wenn sich der Code nicht flexibel verändern ließe.

Um maximal wandelbaren Quellcode sicherzustellen, setzen wir auf Clean Code Development. Dabei handelt es sich um eine Entwicklungsmethode, die darauf ausgelegt ist, sauberen und evolvierbaren Quellcode zu schreiben sowie technische Schulden zu vermeiden.

Der große Vorteil: Der Quellcode Ihrer Software bleibt auch nach der initialen Entwicklung in hohem Maße wandelbar. Das bedeutet, Sie können Ihr Produkt auch noch nach fünf oder zehn Jahren flexibel verändern. Konventionell geschriebener Code kann das nur selten gewährleisten und birgt sogar die Gefahr eines Entwicklungsstopps.

Mit Clean Code bleiben Sie agil was Weiterentwicklungen, Optimierungen und Veränderungen angeht. Damit schützen Sie die Investition in Ihre Softwarelösung über den gesamten Produktlebenszyklus.



1.3 Ihre Vorteile

1.3.1 Sie erhalten das perfekt auf Ihr Business, Ihre IT und Ihre Nutzer zugeschnittene Produkt

Der Dual-Track-Agile-Ansatz garantiert, dass nur Lösungen umgesetzt werden, die tatsächlich Mehrwert bringen. Durch die parallele Arbeit in Discovery und Development sichern wir ab, dass jede entwickelte Funktion sowohl wirtschaftlich rentabel, technisch machbar sowie benutzerfreundlich ist. Dies minimiert das Risiko von Fehlinvestitionen und stellt sicher, dass das Endprodukt alle Anforderungen erfüllt.

1.3.2 Sie profitieren von hoher Kosteneffizienz und schützen Ihre Investition

Durch die Kombination aus Dual Track Agile und Clean Code Development holen Sie mittel- bis langfristig den größtmöglichen Nutzen aus Ihrer Investition. Dual Track Agile sorgt dafür, dass Sie frühzeitig die richtige Lösung entwickeln und kontinuierlich Mehrwert daraus ziehen können. Clean Code Development stellt sicher, dass Ihre Software über den gesamten Produktlebenszyklus effizient weiterentwickelbar und wartungsfreundlich bleibt.

1.3.3 Ihr Softwareprodukt wird maximal nutzerzentriert entwickelt

Unsere UX-Designer sind von Anfang bis Ende in den Entwicklungsprozess integriert und stellen sicher, dass die Software nahtlos zu den Nutzern, ihren Arbeitsmitteln und -abläufen passt. Dies reduziert den Schulungs- und Supportaufwand, minimiert Nutzungsfehler und steigert die Effizienz und Zufriedenheit der Anwender.

1.3.4 Sie können die Produktentwicklung jederzeit flexibel anpassen

Die enge Zusammenarbeit in den beiden Tracks und regelmäßige Feedback-Schleifen ermöglichen es uns, flexibel auf Veränderungen zu reagieren und das Produkt kontinuierlich zu optimieren. Dies sichert eine dynamische und agile Entwicklung, die sich schnell an neue Erkenntnisse und geänderte Anforderungen anpasst.

1.3.5 Ihr Produkt gewinnt kontinuierlich an Mehrwert und ist frühzeitig einsatzbereit

Anstatt umfangreiche Spezifikationen im Vorfeld zu erstellen, konzentrieren wir uns darauf, so schnell wie möglich mit der Entwicklung zu beginnen. Priorisierung nach Wichtigkeit ermöglicht es, das Produkt schnell zu etablieren und kontinuierlich Mehrwert zu generieren. Dadurch ist Ihr Produkt bereits während der Entwicklung einsatzbereit.

1.3.6 Produktrisiken werden kontinuierlich minimiert

Durch schnelle Iterationen in den Phasen des Verstehens, Erforschens, Entwerfens und Testens reduzieren wir das Risiko, auf ungeprüfte Ideen zu setzen. Wir validieren und verfeinern Ideen im Discovery Track, bevor sie im Development Track umgesetzt werden, um sicherzustellen, dass nur die besten Lösungen weiterverfolgt werden.

1.3.7 Keine versteckten Lock-in-Effekte – Sie bleiben vollkommen unabhängig

Das Produkt und damit der entwickelte Quellcode gehört zu jeder Zeit Ihnen. Sie haben uneingeschränkte Eigentums- und Nutzungsrechte an der Software und können diese, unabhängig von uns, selbst weiterentwickeln oder weiterentwickeln lassen. Dank Clean Code zeichnet den Quellcode dabei eine hohe Lesbarkeit und Nachvollziehbarkeit aus – auch für Dritte.

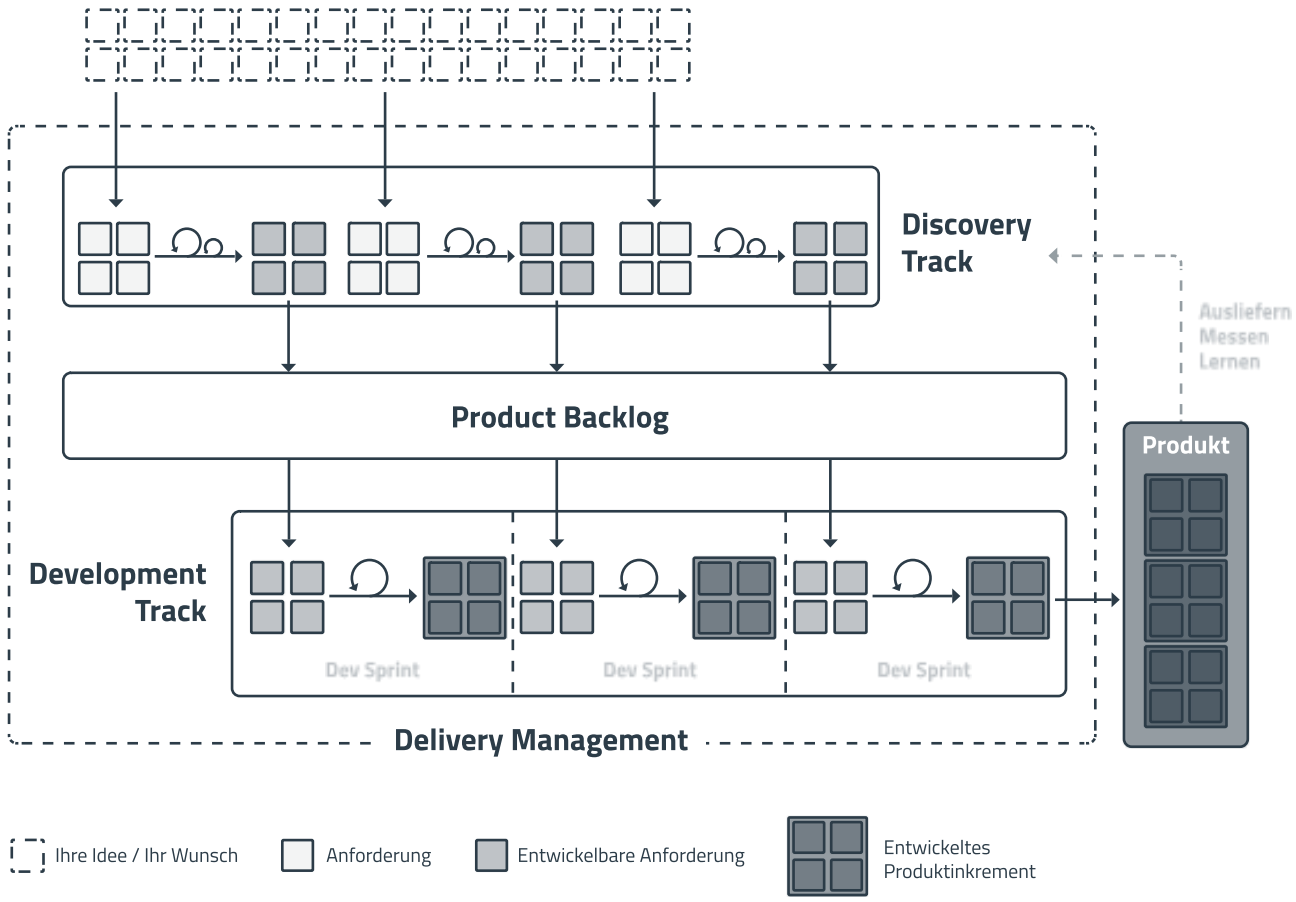
1.3.8 Sie haben jederzeit volle Projekt- und Kostenkontrolle

Wir sorgen für volle Transparenz in jedem Schritt des Projekts, indem wir ständig die Risiken, Annahmen, Schwierigkeiten und Abhängigkeiten überwachen und stets das Auge auf Budget, Zeit und Fortschritt haben. Durch regelmäßige Berichte behalten Sie die volle Kontrolle und Übersicht über Ihr Projekt. Dies ermöglicht rechtzeitige Anpassungen und Kurskorrekturen, falls nötig.

2 Unser Vorgehen im Detail

Im Folgenden möchten wir Ihnen zeigen, wie wir die Entwicklung Ihres Softwareprodukts im Detail angehen. Dafür beleuchten wir zunächst das Thema Agilität, da es Grundvoraussetzung für unsere Arbeitsweise darstellt. Anschließend untersuchen wir die beiden Tracks der Dual Track Methode: den Discovery Track und den Development Track. Wie das Projekt geführt wird und wie wir zusammenarbeiten werden, erfahren Sie im abschließenden Kapitel Delivery Management.

Folgende Grafik zeigt den Dual Track Agile im Detail und dient zur Navigation in diesem Dokument:



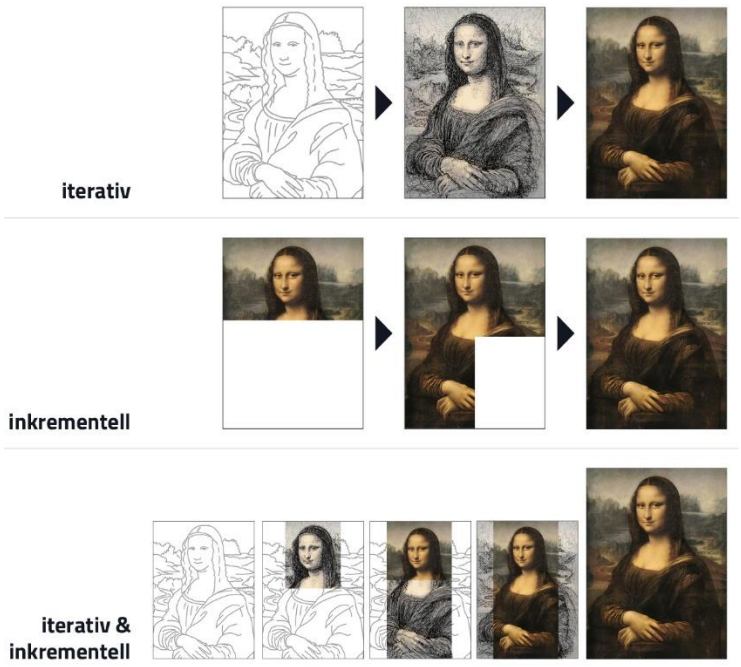
2.1 Agile Arbeitsweise

Agilität ist die Grundvoraussetzung unserer Art der Produktentwicklung. Daher ist es elementar für unser gemeinsames Projekt eine Arbeitsweise zu schaffen, die geprägt ist von den agilen Werten **Offenheit**, **Mut**, **Respekt**, **Fokus** und **Verbindlichkeit**. Dabei helfen uns unter anderem:

- Kurze Zyklen (Sprints)
- Agile Methoden (Scrum, Kanban, OKR etc.)
- Gegenseitige Unterstützung

Die grundlegende Tätigkeit besteht darin, aus Ideen und Anforderungen Produktbestandteile zu entwickeln, zu messen, wie gut die Nutzer damit ihre Aufgaben erledigen können und dann zu validieren, ob man auf Kurs ist oder korrigieren muss. Wir erarbeiten Ihr Produkt dabei gemeinsam in interdisziplinären Teams iterativ und inkrementell:

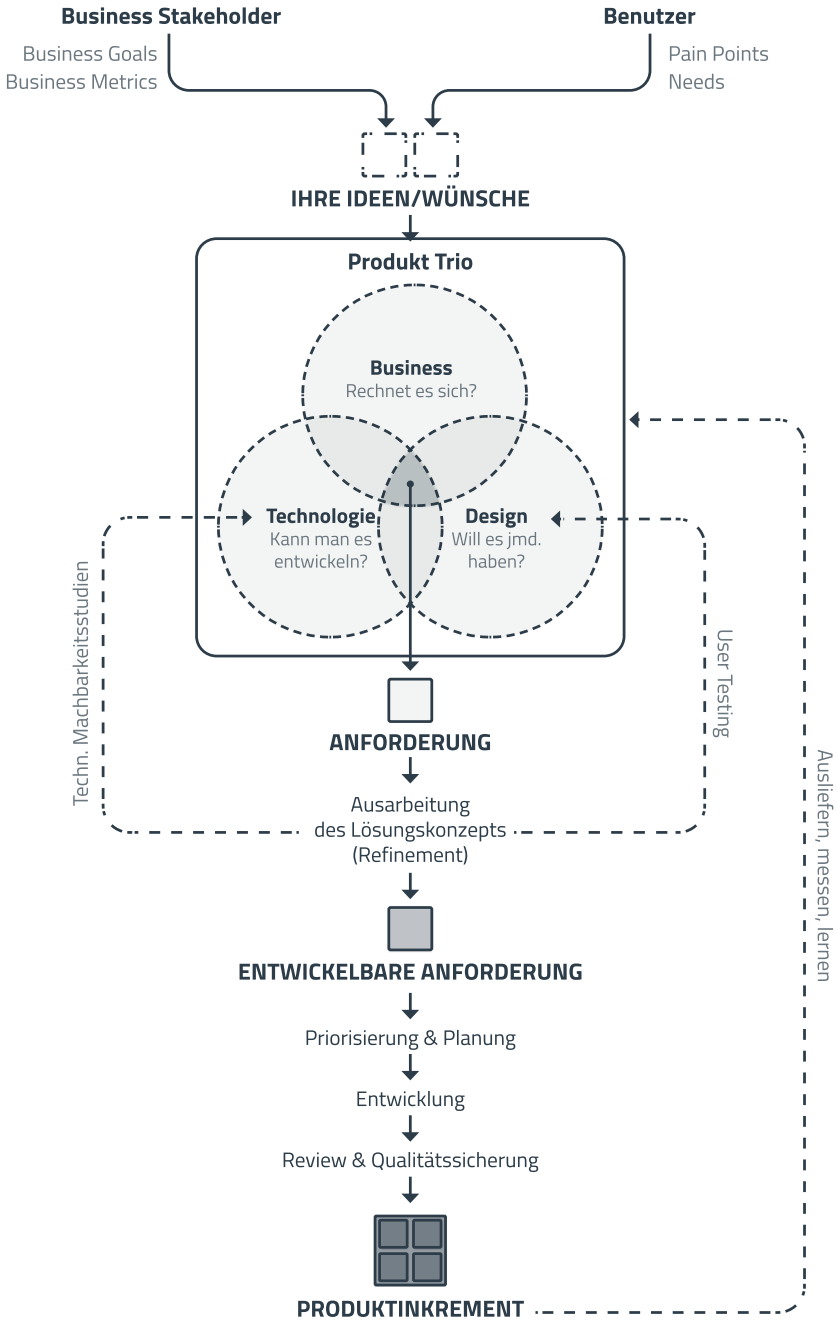
- **Inkrementell** bedeutet, dass wir schrittweise funktionale Erweiterungen des Produkts entwickeln (Inkrement). Dafür wird das Produkt zunächst in sinnvolle Arbeitspakete zerlegt, die im Product Backlog gesammelt und priorisiert werden.
- **Iterativ** bedeutet, dass wir in Feedbackschleifen oder Sprints arbeiten. Die Sprintlänge legen wir zu Beginn des Projekts fest. Erfahrungsgemäß liegt sie zwischen einer und vier Wochen. Pro Sprint entwickeln wir ein oder mehrere Produktinkremente, testen diese mit den entsprechenden Stakeholdern und ziehen Erkenntnisse aus den Tests. Diese fließen in den nächsten Sprint ein. So kommen wir Sprint für Sprint zum perfekten Ergebnis.



2.2 Der Discovery Track

Der Discovery Track ist ein explorativer Prozess, bei dem man viel ausprobiert, schnell und kostengünstig Prototypen generiert. Dadurch kann das Produkt noch vor der Entwicklung mit den realen Anwendern getestet und so Risiken minimiert werden. Konzepte, die nicht zielführend sind, werden herausgefiltert, sodass nur die vielversprechendsten an den Development Track übergeben werden.

Dank des Dual Tracks kümmern wir uns nicht nur zu Projektbeginn um die Discovery, sondern machen daraus eine kontinuierliche und iterative Tätigkeit während der Produktentwicklung.



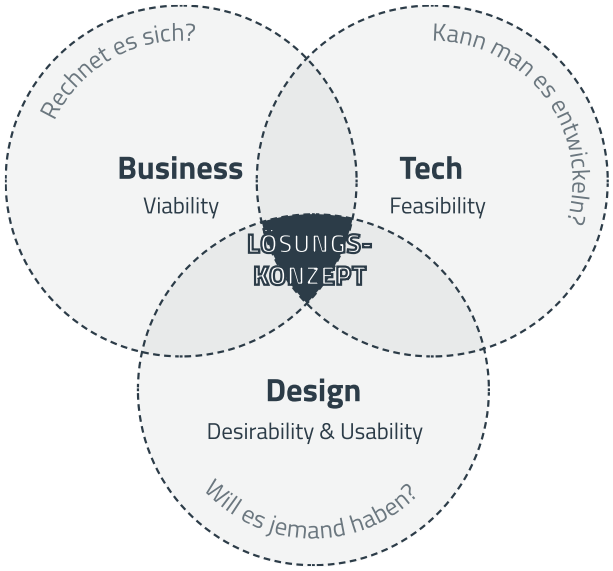
2.2.1 Definieren der Outcomes

Zu Beginn eines Projekts wissen wir in der Regel, welches Problem wir lösen müssen, aber wir wissen nicht, wie wir es am besten lösen können. Daher beginnen wir mit der Definition eines klaren Outcomes für das Business – also dem Mehrwert, der bspw. durch ein neues Feature erzielt wird (mehr registrierte Benutzer, weniger Fehler, weniger Zeitaufwand etc.). Die definierten Outcomes legen den Rahmen für die Discovery fest.

Zu den priorisierten Outcomes erforschen wir die relevanten Prozesse, Nutzer und Systeme und identifizieren Optionen, wie die Outcomes am besten erreicht werden können.

2.2.2 Kontinuierliches Requirements Engineering im Produkt Trio

Als Business-Stakeholder sind Sie Teil des sogenannten Produkt Trios, das die drei Aspekte der Softwareentwicklung abdeckt:



Das Produkt Trio trägt Anforderungen zusammen und findet konstruktiv und zielgerichtet die geeignetste Lösung. Das übrige Entwicklungsteam und die Stakeholder agieren als Berater, um auftauchende Fragen unmittelbar beantworten zu können.

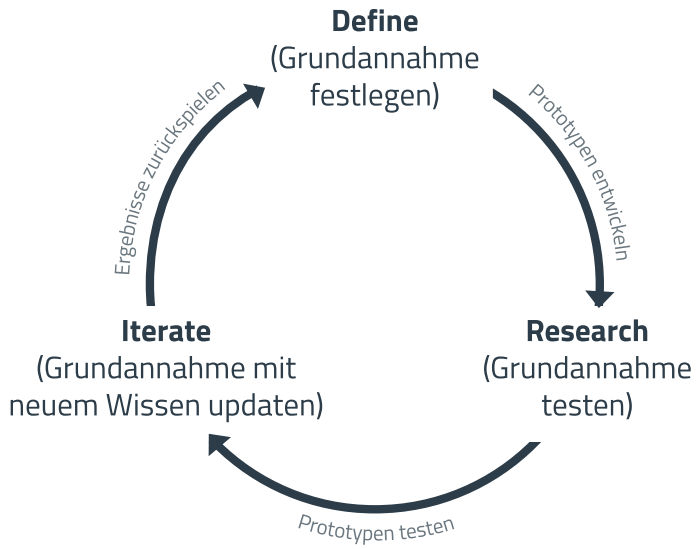
Ausgehend von den Business-Anforderungen identifizieren wir, welche Nutzerbedürfnisse erfüllt werden müssen, um die Geschäftsziele zu erreichen. Durch User Research verstehen wir die Anwender, deren Aufgaben, den Nutzungskontext und welche Defizite die bestehende Lösung hat.

Die funktionalen und qualitativen Anforderungen an das System werden so formuliert, dass sie sowohl Nutzerbedürfnisse erfüllen als auch Business-Ziele erreichen und sich nahtlos in Ihre bestehende Systemlandschaft integriert.

2.2.3 Identifizieren und minimieren von Risiken

Die Anforderungen, die das Produkt Trio definiert, basieren oft auf Annahmen, die wir in Bezug auf die Nutzer oder technische Realisierbarkeit haben. Diese sind dem Produktteam oft nicht bewusst, stellen aber ein hohes Produktrisiko dar. Im Discovery Track werden die Annahmen systematisch ans Tageslicht gebracht, um sie zu validieren. Dadurch werden Risiken minimiert und wir stellen sicher, dass wir keine Zeit verschwenden, ein Produkt zu bauen, welches z.B. die Nutzerbedürfnisse verfehlt.

Tests von Annahmen sind kleine, eingeschränkte Experimente, meist in Form schnell gebauter Prototypen, bspw. für Benutzertests oder technische Machbarkeitsstudien.



2.2.4 Requirements Refinement

Nach Validierung der kritischsten Annahmen werden die Anforderungen kontinuierlich angereicht, bis die Entwickler genügend Informationen haben, um mit der Entwicklung zu beginnen. Dieser Vorgang nennt sich „Refinement“.

- **Technikexperten** entwerfen Lösungsarchitekturen, die sowohl die funktionalen Bedürfnisse als auch anspruchsvolle nicht-funktionale Anforderungen wie Sicherheit und Skalierbarkeit berücksichtigen.
- **UX-Designer** erstellen Interaktionskonzepte, Style Guides und Design-Spezifikation und sorgen für Benutzerfreundlichkeit und eine gute User Experience.

Durch die enge Zusammenarbeit von UX-Designern mit Entwicklern entwerfen wir nur, was machbar ist. UX-Designer begleiten die Entwicklung und sichern damit die präzise technische Umsetzung.

2.3 Das Product Backlog

Das Product Backlog bildet die Schnittstelle zwischen Discovery und Development Track. Es stellt eine dynamische, ständig aktualisierte Liste mit Anforderungen für die Entwicklung Ihres Produkts dar und enthält alle Ideen, Features und Aufgaben, die für die Weiterentwicklung wichtig sind.

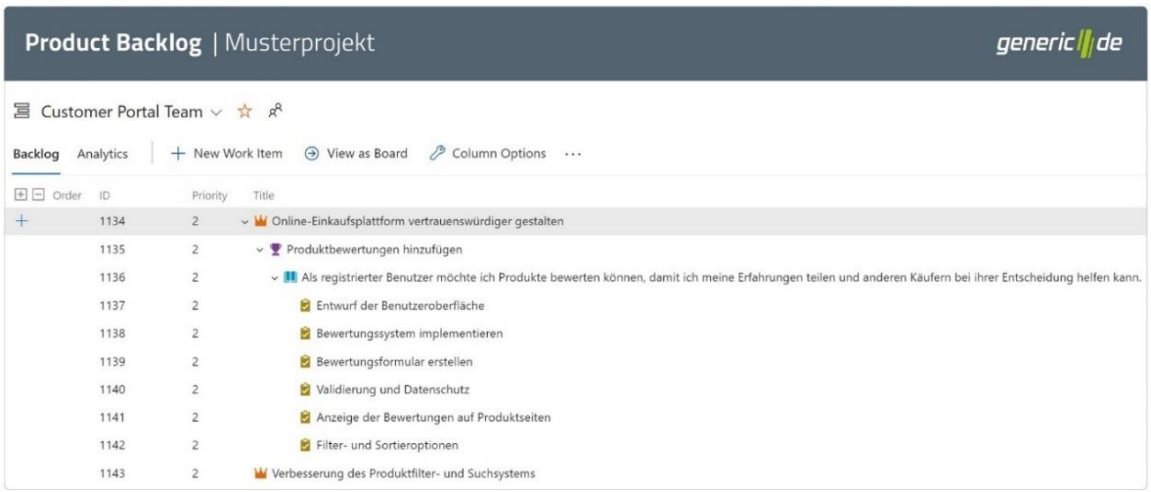
Das Product Backlog beinhaltet die Anforderungen für beide Tracks, Discovery und Development:

- **Discovery:** Im Discovery Track fügen wir dem Product Backlog neue Ideen und Erkenntnisse hinzu. Dies können Benutzeranforderungen, Marktforschungsergebnisse oder Feedback aus Tests sein.
- **Development:** Im Development Track wählen die Entwicklungsteams Aufgaben aus dem Product Backlog aus und setzen sie um. Diese Aufgaben werden in kurzen Entwicklungszyklen (Sprints) bearbeitet.

Die Teams arbeiten zusammen, um sicherzustellen, dass das, was entwickelt wird, nicht nur technisch machbar ist, sondern auch den Bedürfnissen der Benutzer entspricht. Dabei ist das Product Backlog die "Single Source of Truth": Alle Anforderungen an das Produkt und alle Aufgaben, die zu erledigen sind, werden hier erfasst und sind jederzeit transparent nachverfolgbar.

Anforderungen werden in Form von **Backlog Items** erfasst, die alle relevanten Informationen enthalten. Bspw.:

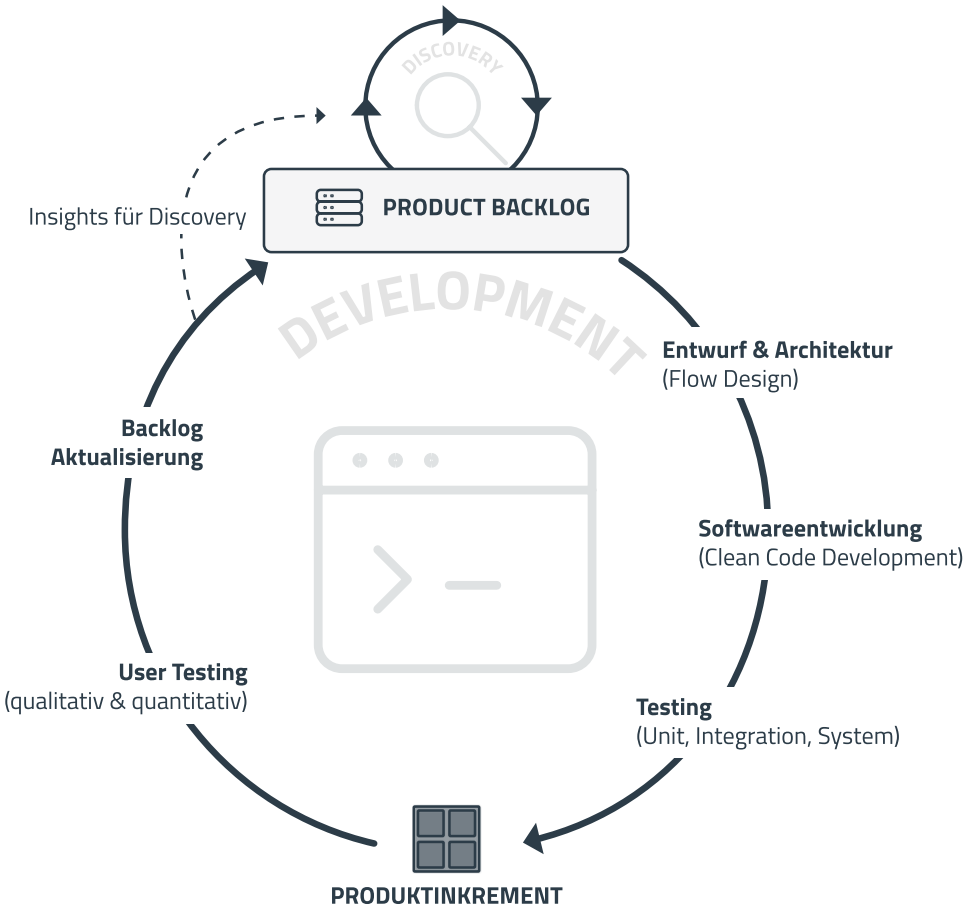
- Kurzbeschreibung (User Story)
- Akteur (Persona, System)
- Hintergrund/Zielsetzung
- Abhängigkeiten
- Akzeptanzkriterien



Exemplarische Darstellung eines Product Backlogs in Azure DevOps

2.4 Der Development Track

Der Development Track beinhaltet die eigentliche Entwicklung der geplanten Features: aus den zuvor abgestimmten Konzepten entsteht das wirkliche Produkt. Auch während dieser Phase unterstützen unsere UX-Designer das Entwicklungsteam. Sind Details unklar oder ist die Umsetzung schwieriger als erwartet, lösen Entwickler und UX-Designer mögliche Blockaden gemeinsam – ohne unnötiges Hin und Her von Tickets oder Warten auf den nächsten Sprint. So stellen wir sicher, dass unsere Entwürfe aus der Discovery zielgerichtet implementiert werden und vermindern ungewollte Änderungen oder Seiteneffekte in der Benutzbarkeit der Software.



2.4.1 Clean Code Development

2.4.1.1 Warum wir auf Clean Code Development setzen

Wir entwickeln Software mit dem Blick nach vorn, sodass sie leicht an sich ändernde Bedingungen und neue Technologien angepasst werden kann. Dabei streben wir eine Balance zwischen gegenwärtigen Anforderungen und zukünftiger Flexibilität an. Entscheidend dafür ist eine hohe innere Softwarequalität.

Während äußere Qualitätsmerkmale bspw. Funktionalität, Benutzbarkeit, Performance oder Skalierbarkeit ausmachen, dreht es sich bei den inneren Qualitätsmerkmalen um die Güte des Quellcodes. Die wichtigsten Merkmale sind **Lesbarkeit**, **Nachvollziehbarkeit**, **Testbarkeit** und **Evolvierbarkeit**. Je stärker diese Qualitätsmerkmale ausgeprägt sind, desto besser kann der Code verändert und angepasst werden – desto nachhaltiger ist demnach das Produkt.

Um die innere Softwarequalität unserer Produkte zu maximieren, setzen wir auf die Entwicklungsmethode Clean Code Development.

2.4.1.2 Was ist Clean Code Development?

Clean Code Development ist ein normgebendes Wertesystem für Softwareentwickler. Über die vier Werte **Korrektheit**, **Wandelbarkeit**, **Produktionseffizienz** und **kontinuierliche Verbesserung**[2] vermittelt Clean Code Development eine perfekt auf die Softwareentwicklung zugeschnittene Arbeitsphilosophie, die von Agilität, konsequentem interdisziplinären Austausch, einem hohen Qualitätsbewusstsein sowie dem Drang nach ständiger Optimierung lebt.

Daneben sind es praktische Handlungsempfehlungen für das eigentliche Coding. In fünf Grade unterteilte Praktiken und Prinzipien, die nicht nur zu hoher innerer Softwarequalität führen, sondern diese auch mess- und prüfbar machen.

	Roter Grad	Oranger Grad	Gelber Grad	Grüner Grad	Blauer Grad
PRINZIPIEN	Don't Repeat Yourself (DRY) Keep it simple, stupid (KISS) Beware of Premature Optimization Favour Composition over Inheritance Integration Operation Segregation Principle (IOSP)	Single Level of Abstraction (SLA) Single Responsibility Principle (SRP) Separation of Concerns (SoC) Source Code Conventions	Interface Segregation Principle (ISP) Dependency Inversion Principle (DIP) Liskov Substitution Principle (LSP) Principle of Least Astonishment Information Hiding Principle	Open Closed Principle (OCP) Tell, don't ask Law of Demeter (LoD)	Design and Implementation Don't Overlapp Implementation Reflects Design You Ain't Gonna Need It (YAGNI)
PRAKTIKEN	Boy Scout Rule Root Cause Analysis Version Control System Simple Refactorings Daily Reflection	Issue Tracking Automated Integrationtests Read, Read, Read Reviews	Automated Unit Tests Mockups Code Coverage Analysis Partizipation in Professional Events Complex Refactorings	Continuous Integration Statical Code Analysis Inversion of Control Container Share Experience Error Measurement	Design before Implementation Continuous Delivery Iterative Development Incremental Development Component Orientation Test First

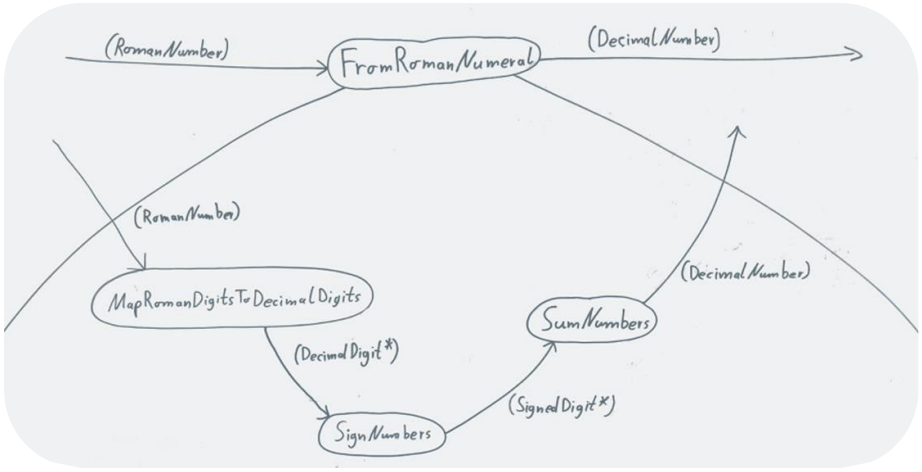
[2] Vgl. https://clean-code-developer.de/das-wertesystem/

2.4.1.3 Wie wir mit Clean Code die innere Softwarequalität maximieren

Entwurf vor der Implementierung mit Flow Design

Wir entwerfen die Code-Struktur unserer Software vor der Implementierung. Dazu setzen wir Flow-Design[3] ein. Bei der Methode werden Anforderungen zunächst hierarchisch zerlegt, um die primären Funktionen herauszuarbeiten. Eine Funktion besteht dabei immer aus einer oder mehreren Interaktionen, die wiederum in Datenflüsse (Flows) zerlegt werden können.

Ziel ist es, das Problem aus der Anforderung mithilfe sinnvoller Datenflüsse bis ins Detail zu lösen. Die komplette Struktur mitsamt den Klassen, Methoden, Zustandsvariablen und Parametern ist damit noch vor der ersten Zeile Code bekannt. Wir schaffen mit Flow Design ein gemeinsames Verständnis und eine gemeinsame Sprache, um Lösungen der Problemdomäne zu entwerfen.



Exemplarisches Flow Design, um römische Zahlen in Dezimalzahlen zu übersetzen.

Durch den Einsatz von Flow Design sind wir in der Lage, Test-First zu entwickeln – wir konzipieren also bereits in der Entwurfsphase eine passende Testsuite. Außerdem ermöglicht uns die Methode eine der Domäne dienliche Architektur zu entwerfen und Over-Engineering zu vermeiden.[4]

[3] Vgl. https://flow-design.info/

[4] YAGNI, vgl. https://clean-code-developer.de/die-grade/blauer-grad/

Prinzipien und Praktiken für das Coding

Folgende Tabelle zeigt einen Auszug von Clean Code Prinzipien und Praktiken, die auf die wichtigsten inneren Softwarequalitätsmerkmale einzahlen:

Qualitätsmerkmal	Clean Code Prinzipien (Auzug)	Clean Code Praktiken (Auzug)
Lesbarkeit	**Source Code Convention:** Nutzen von syntaktischen Konventionen, die ein schnelles Lesen und Erfassen des Codes unterstützen	**Statical Code Analysis:** Automatisierte Tests, die den Quellcode auf formale Fehler überprüfen **Code Reviews:** Quellcode sollte immer von einer zweiten Softwareentwickler:in begutachtet werden
Nachvollziehbarkeit	**Keep It Simple Stupid:** Wenn (auch) eine einfache Lösung funktioniert, ist immer diese zu wählen	**Version Control System:** Ein Versionskontrollsystem ist einzusetzen **Issue Tracking:** „Kundenanfragen" müssen systematisch erfasst, priorisiert, abgearbeitet und nachverfolgt werden
Testbarkeit	**Integration Operation Separation Principle:** Entweder eine Methode enthält nur Logik (Operation) oder keinerlei Logik und nur Aufrufe anderer Methoden (Integration)	**Test First:** Zuerst werden Tests für eine bestimmte Schnittstelle geschrieben und erst danach wird diese implementiert **Test Automation:** Egal ob Unit oder Integration Testing: Tests müssen automatisiert werden, um konsequent Mehrwert zu bieten
Evolvierbarkeit	**Single Responsible Principle:** Eine Klasse sollte immer nur eine Aufgabe und/oder Verantwortlichkeit haben **Open Closed Principle:** Eine Klasse muss offen für Erweiterungen sein, jedoch geschlossen für Modifizierungen	**Component Orientation:** Komponenten können unabhängig voneinander implementiert werden und erfüllen dabei einen gemeinsamen Kontrakt, sodass sie später beliebig integriert werden können

Wir verwenden die Prinzipien und Praktiken der Clean Code Initiative[5] ausschließlich, um die Clean Code Werte einzuhalten, nicht als Selbstzweck.

[5] vgl. https://clean-code-developer.de/

Testing

Zur Einhaltung der Clean Code Werte Korrektheit und Wandelbarkeit testen wir unsere Software kontinuierlich und automatisiert über qualitativ hochwertige Tests, die wir mit Metriken nachweisen können. Wir schreiben Tests auf unterschiedlichen Ebenen (Unit, Integration, System) und legen ein besonderes Augenmerk auf Akzeptanztests. Dabei sind sie unabhängig von der gewählten Technologie. Idealerweise werden die Akzeptanzkriterien von Ihnen definiert.

Automatisierung

Wir automatisieren unsere Builds, Tests und Deployments nach Continuous Integration und Continuous Delivery. Die Infrastruktur unserer Software definieren wir in Code.

Wir alle sind uns der Verantwortung bewusst, die jeder einzelne trägt, um der Einhaltung und Sicherstellung dieser hohen Qualitätsansprüche gerecht zu werden. Wir wollen damit Nachhaltigkeit in unseren Softwareprojekten schaffen und gleichzeitig Rücksicht auf aktuelle und zukünftige Ressourcensituationen nehmen – ein echter Mehrwert für Sie als Kunde, für zukünftige Entwickler und für uns.

2.4.2 Die Ergebnisse des Development Tracks

Lieferbare Software oder Produktinkremente: Das Endresultat des Development Tracks ist lieferbare Software oder Produktinkremente, die den Benutzern zur Verfügung gestellt werden. Diese werden typischerweise in regelmäßigen Intervallen veröffentlicht, um kontinuierlich Mehrwert zu schaffen.

Getestete und einsatzbereite Features: Die im Development Track erstellten Funktionen durchlaufen Tests, um sicherzustellen, dass sie den Anforderungen entsprechen und fehlerfrei funktionieren. Die Ergebnisse dieser Tests bestätigen, dass die Features einsatzbereit sind.

Metriken: Nach der Auslieferung werden, wenn möglich, Daten und Metriken gesammelt, um die Leistung der implementierten Funktionen zu bewerten. Diese Metriken erstrecken sich von objektiven und technischen Parametern, wie Antwortzeiten, Benutzbarkeit oder Ressourcennutzung bis zu subjektiven Kriterien, wie Ladezeiten und Benutzerzufriedenheit.

Benutzerfeedback: Usability Tests und Benutzerfeedback zum ausgerollten und getesteten Produktinkrement oder Feature machen messbar, ob das Produkt den angestrebten Mehrwert bei Benutzern erzeugt. Erkenntnisse daraus fließen in die nächsten Iterationen.

2.5 Delivery Management

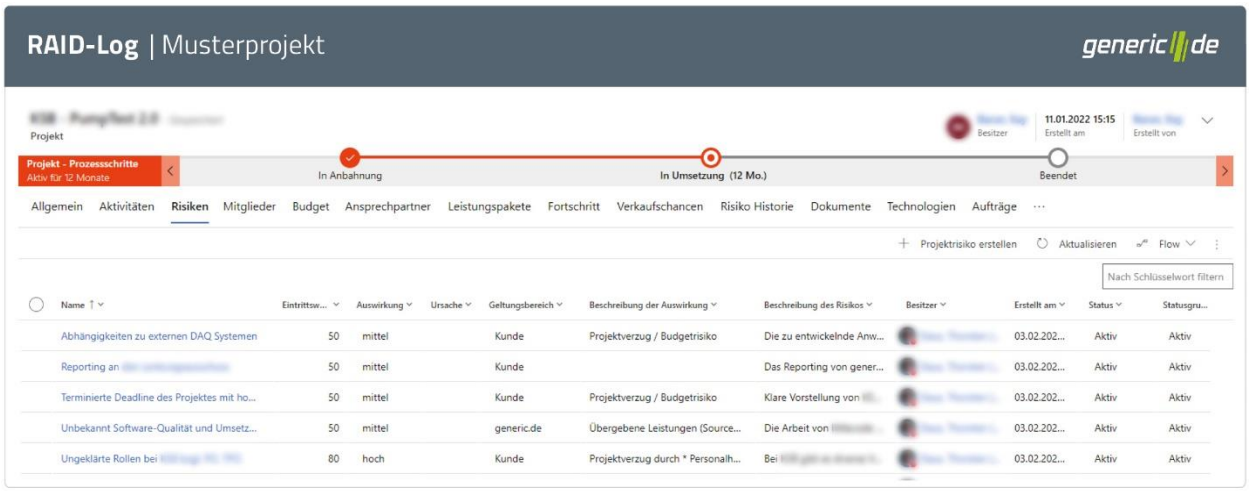
Das Delivery Management spielt eine zentrale Rolle in der agilen Produktentwicklung, indem es sicherstellt, dass wir Ihnen Ihr Softwareprodukt zeitgerecht, effizient und in bestmöglicher Qualität liefern. Außerdem sorgt die Abteilung durch regelmäßige Reports dafür, dass Sie jederzeit volle Projekt- und Kostenkontrolle haben. Im Folgenden wird erläutert, welche Tools wir dafür nutzen, wie Ihre Rolle dabei aussieht und wie sich unsere Zusammenarbeit gestaltet.

2.5.1 Risiko-Management mittels RAID-Log

Ein RAID-Log ist ein effektives Werkzeug zur Identifizierung und Verwaltung von potenziellen Herausforderungen und Chancen im Projekt. Das Akronym steht für: **R**isks (Risiken), **A**ssumptions (Annahmen), **I**ssues (Probleme) und **D**ependencies (Abhängigkeiten). Das Führen eines RAID-Logs ermöglicht es den Projektverantwortlichen, sich systematisch und proaktiv mit Unsicherheiten und potenziellen Hindernissen auseinanderzusetzen. Dadurch können mögliche Risiken frühzeitig erkannt und adressiert werden, was die Wahrscheinlichkeit von Verzögerungen oder Kostenüberschreitungen reduziert.

Das RAID-Log ermöglicht Ihnen:

- Transparenz über aktuelle und potenzielle Herausforderungen
- Aktive Teilnahme am Problemmanagement
- Besseres Verständnis der Projektrisiken
- Fundierte Entscheidungsfindung



Exemplarische Darstellung eines RAID-Logs in Azure DevOps

2.5.2 Unsere Zusammenarbeit

Wie gestalten sich unsere Zusammenarbeit? Alles konsolidiert sich in einem gemeinsamen Project Space. Hierfür arbeiten wir üblicherweise mit Azure DevOps und Microsoft Teams – unsere erste Wahl, um die wichtigsten Aspekte gut zu integrieren. Gerne setzen wir eine entsprechende Instanz für Ihr Projekt auf oder arbeiten in einer von Ihnen bereits verwalteten Umgebung.

Sie haben bereits andere Werkzeuge in Ihren Prozessen etabliert? Selbstverständlich arbeiten wir auch gerne in für Sie vertrauter Tool-Landschaft.

Die erwähnten agilen Grundprinzipien schätzen wir auch in der Kommunikation: wir pflegen kurze Wege und schnelle Reaktionszeiten. Die Erreichbarkeit während üblicher Bürozeiten sind beiderseits gegeben, so dass Anliegen schnell geklärt werden können. Einige sinnvolle Regeltermine, wie Daily, Sprint-Review oder RAID-Reporting geben der Zusammenarbeit die nötige Struktur und Transparenz. So sichern wir die Möglichkeit, regelmäßig aktiv am Prozess teilzunehmen.

Wir stemmen Ihr Projekt nicht allein. Sie bringen die für Ihr Vorhaben erforderliche Zeit für inhaltliche Tätigkeiten mit und schaffen die Rahmenbedingungen für effiziente Arbeit. Sie erledigen zeitnah die Aufgaben im Rahmen des Projekts, die wir nicht erledigen können. Dabei ermöglichen Sie den Kontakt zu Fachexperten und Nutzern, um belastbaren fachlichen Input und direktes Benutzerfeedback zu gewährleisten.

2.5.2.1 Die RACI-Matrix

Damit wir Ihre Erwartungen bestmöglich erfüllen können, definieren wir zu Projektstart gemeinsam mit Ihnen die Rollen und Verantwortlichkeiten von Personen und/oder Teams. Dafür nutzen wir die sog. RACI-Matrix. Das Akronym steht für **R**esponsible (inhaltlich zuständig), **A**ccountable (verantwortlich), **C**onsultant (konsultiert) und **I**nformed (informiert). Die Vorteile dieser Herangehensweise:

- Klare Zuweisung von Verantwortlichkeiten & Verbindlichkeiten
- Förderung der Kommunikation und Zusammenarbeit
- Erhöhung der Effizienz und Transparenz
- Unterstützung bei der Konfliktvermeidung
- Unterstützung bei der Entscheidungsfindung
- Unterstützung bei der Projektsteuerung

Zur besseren Vorstellung, hier eine exemplarische RACI-Matrix:

	Kunde	Projekt Manager	Product Owner	Scrum Master	Dev-Team	UX Designer	Software-architekt
Anlegen Systemzugänge	**A**	**R**	**C**	**I**	**C**	**I**	**C**
Projektsteuerung	**C**	**A & R**	**C**	**C**	**I**	**I**	**I**
Agiler Prozess	**C**	**I**	**C**	**A & R**	**C**	**C**	**I**
Rollout-Unterstützung	**A**	**I**	**R**	**I**	**C**	**I**	**I**
Requirements Engineering	**A**	**C**	**R**	**I**	**C**	**C**	**I**
Testing & Releases	**A**	**C**	**R**	**I**	**C**	**C**	**C**
Qualitätssicherung	**C**	**A & R**	**C**	**C**	**C**	**C**	**C**
UX-Konzepte	**C**	**I**	**C**	**I**	**C**	**A & R**	**C**
Quellcode	**I**	**I**	**I**	**I**	**A & R**	**I**	**C**
...							

R = Responsible (zuständig) | **A** = Accountable (verantwortlich) | **C** = Consultant (konsultiert) | **I** = Informed (informiert)

2.5.2.2 Bereitstellung von Insights, Know-how und Ressourcen

Damit wir Ihnen das Softwareprodukt auf den Leib schneidern können, ist es unerlässlich, einen Konterpart auf Ihrer Seite zu haben. Denn letztendlich kennen Sie sich in Ihrer Fachdomäne am besten aus, wissen über die notwendigen Business Insights Bescheid und können entscheiden, welche Ressourcen Sie für die Produktentwicklung bereitstellen können.

Wir bezeichnen diesen Part als Product Owner auf Kundenseite. Ob es sich dabei um eine Person oder mehrere handelt, ist von Projekt zu Projekt unterschiedlich. Elementar ist, dass diese Rolle unser erster Ansprechpartner für all jene Fragen ist, die wir selbst nicht beantworten können. Außerdem ist sie verantwortlich für die Beistellung der Bedürfnisse und Anforderungen Ihrer Stakeholder.

Im Folgenden finden Sie eine exempl. Auflistung möglicher Aufgaben und Eckdaten dieser Rolle:

Bereitstellung	Aufgaben	Verfügbarkeit
Bereitstellung des Business Case	Sorgt für Zugang zu den Anwendern	Daily: Tägliche Abstimmung
Erarbeitung der Produktvision mit unserer Unterstützung	Sorgt für Zugang zu Fachexperten	Requirements Refinement: 1 x pro Sprint
Verfügbarkeit bei Fragen vom Entwicklungsteam	Definiert und überwacht geschäftliche Kennzahlen	Sprint Planung: 1x pro Sprint
	Verantwortet den kommerziellen Erfolg	Sprint Review: 1x pro Sprint

	Kümmert sich um Kommunikation & Kollaboration mit den Stakeholdern	

2.5.3 Qualitätssicherung auf Basis von ISO 9001

Die Umsetzung folgender Qualitätsmaßnahmen unterstützt die Durchführung des Softwareprojekts nach den ISO 9001-Standards und trägt dazu bei, die Qualität der Ergebnisse sicherzustellen:

Projektplanung	Erstellung eines detaillierten Projektplans unter Berücksichtigung von Ressourcen, Zeitrahmen und Qualitätszielen.
Risikomanagement	Identifikation und Bewertung von Risiken sowie Implementierung von Maßnahmen zu deren Management.
Anforderungsmanagement	Strukturiertes Anforderungsmanagement, um klare, verständliche und nachverfolgbare Anforderungen zu gewährleisten.
Dokumentenlenkung	Implementierung von Prozessen zur effizienten Erstellung, Überprüfung, Genehmigung und Aktualisierung von Projektdokumentation gemäß ISO 9001.
Validierung und Verifizierung	Planung und Durchführung von Validierungs- und Verifizierungsaktivitäten, um sicherzustellen, dass die Software die spezifizierten Anforderungen erfüllt.
Änderungsmanagement	Strukturiertes Änderungsmanagement, um sicherzustellen, dass alle Änderungen ordnungsgemäß dokumentiert, überprüft und genehmigt werden.
Projektüberwachung und Projektsteuerung	Etablierung von Überwachungsmechanismen, um sicherzustellen, dass das Projekt gemäß Plan verläuft, und Implementierung von Korrekturmaßnahmen bei Abweichungen.
Kommunikationsmanagement	Klare Definition von Kommunikationskanälen und -protokollen, um eine effektive und transparente Kommunikation innerhalb des Projektteams sicherzustellen.
Testing und Qualitätssicherung	Implementierung von umfassenden Testprozessen, um die Qualität der Software durch systematische Tests zu gewährleisten.
Schulungen und Kompetenzentwicklung	Bereitstellung von Schulungen, um sicherzustellen, dass das Team über die notwendigen Fähigkeiten für die Projektumsetzung verfügt.
Projektabnahme und Kundenfeedback	Klare Verfahren zur Abnahme des Projekts durch den Kunden und Erfassung von Kundenfeedback für zukünftige Verbesserungen.
Dokumentation	Dokumentation von Erfahrungen und Erkenntnissen aus dem Projektverlauf zur kontinuierlichen Verbesserung künftiger Projekte.
Kontinuierlichen Verbesserung via Retrospektiven	Retrospektiven sind Meetings, bei denen das Team gemeinsam den vergangenen Entwicklungszyklus reflektiert und Verbesserungsmöglichkeiten identifiziert. Dieser Prozess trägt dazu bei, kontinuierlich die Arbeitsweise, Produktqualität und die Projektqualität zu verbessern.

2.5.4 Release

Agile Softwareentwicklung setzt auf iteratives Vorgehen und Fokus auf die aktuelle Iteration. Maßgebliche Voraussetzung ist, dass Release-Planung und damit Entwicklungsplanung entweder inhaltlich oder zeitlich getrieben sind – immer unter der Prämisse, möglichst hohe Qualität zu liefern:

- Ist ein inhaltlicher Umfang gegeben, erfolgt das Release zum Zeitpunkt der Fertigstellung der Implementierung.
- Ist ein zeitlicher Rahmen gegeben, entspricht der Inhalt des Release dem, was zum definierten Zeitpunkt fertiggestellt ist.

Sind sowohl Inhalt als auch Zeitrahmen/Budget vorgegeben, geht dies im Regelfall auf Kosten der Qualität.

2.5.5 Product Roadmap

Die Product Roadmap visualisiert die geplante Entwicklung mitsamt den Projektmeilensteinen sowie der wichtigsten Funktionen Ihres Produkts über einen bestimmten Zeitraum. Dadurch erhalten Sie einen Überblick darüber, wie sich Ihr Produkt im Laufe der Zeit entwickeln soll, um die strategischen Ziele zu erreichen und den Bedürfnissen der Benutzer gerecht zu werden.

2.5.6 Wenn die Zusammenarbeit endet

Die Arbeitsergebnisse sind jederzeit Ihr Eigentum und in Ihrem Zugriff. Das gilt auch für sämtlichen Quellcode, den wir erstellen. Eine physische Übergabe dessen ist daher nicht erforderlich. Um einen sauberen Projektabschluss sicherzustellen, führen wir allerdings folgende Tätigkeiten durch:

- Wir übergeben sämtliche zusätzliche Artefakte, wie z.B. dokumentierte Erkenntnisse und die Arbeitsstände der Ideenbibliotheken zur weiteren Entwicklung Ihres Produkts.
- Wir führen ein oder mehrere Termine durch, darunter:
    - Eine Demo oder Präsentation für Nutzer und Stakeholder.
    - Eine Nachbesprechung oder Retrospektive über das Gesamtprojekt.
- Ein erfolgreiches Projekt beschließen wir gerne mit einer gemeinsamen Feier.

3 Fazit

Werfen wir abschließend nochmals einen Blick auf die eingangs gestellten Fragen:

Woher wissen Sie, ob Ihre Softwareproduktidee die Wünsche Ihrer Kunden, Ihrer Mitarbeiter und Ihrer IT sowie Ihre eigentlichen Businessziele erfüllt?

Während des kontinuierlichen Requirements Engineerings im Discovery Track werden aus Ihren Wünschen, Ideen und Zielen durchdachte und jederzeit anpassbare Anforderungen. Dabei stellt das Produkt Trio sicher, dass alle elementaren Sichtweisen – Business, Nutzer und Technologie – abgebildet werden. **Das Ergebnis ist ein maximal nutzerzentriertes Softwareprodukt, das kontinuierlich Mehrwert liefert und perfekt in Ihre IT-Landschaft passt.**

Wie können Sie die Produktentwicklung trotz neuer Anforderungen und Erkenntnisse auf Ziel halten, ohne dass es Ihren Zeitplan oder Ihr Budget sprengt?

Mit unserer Interpretation von Dual Track Agile kombinieren wir die Vorteile der Wasserfallprojekt-führung mit den Vorteilen agil geführter Projekte. Dadurch bleibt der Entwicklungsprozess maximal flexibel für neue Anforderungen, während das Projekt hinsichtlich Entwicklungsplan und -budget kontrollierbar bleibt. Eine wichtige Rolle spielt dabei unser Delivery Management, das das Projekt durchgehend monitort und Ihnen berichtet. **Damit haben Sie jederzeit volle Projekt- und Kostenkontrolle.**

Wie schützen Sie Ihre Investition in die individuelle Softwarelösung?

Ihre Investition in die individuelle Softwarelösung schützen wir einerseits dadurch, dass wir sicherstellen das richtige Produkt für Ihre Nutzer, Ihre Systemlandschaft sowie Ihre Geschäftsziele zu entwickeln. Auf technischer Seite ist Clean Code Development Ihre Garantie für die langfristige Wandelbarkeit und Weiterentwickelbarkeit Ihres Produkts. **So bleiben Sie für den gesamten Produktlebenszyklus handlungsfähig und unabhängig.**

Dual Track Agile in Kombination mit Clean Code Development ist der Schlüssel zu erfolgreichen und kundenorientierten Softwareprodukten. Wir sind davon überzeugt, dass diese Methode die Qualität und den Mehrwert Ihres Produkts steigert und freuen uns darauf, mit Ihnen gemeinsam innovative Lösungen zu schaffen.

Mit uns entwickeln Sie das Richtige richtig.

Ihr Kontakt bei generic.de

Kontaktieren Sie uns für weitere Informationen und lassen Sie uns gemeinsam die Zukunft Ihrer Softwareentwicklung gestalten.

Jörg Lenz

Key Account Management
Tel.: +49 721 619096-50
E-Mail: joerg.lenz@generic.de

Marcus Schönherr

Sales Manager
Tel.: +49 721 61 90 96-43
Mobil: +49 174 7026360
E-Mail: marcus.schoenherr@generic.de

Sebastian Betzin

CTO & Vorstand
E-Mail: sebastian.betzin@generic.de

generic.de software technologies AG | Johann-Georg-Schlosser-Str. 66 | D - 76149 Karlsruhe
Vorstand: Michael Puder (Vorsitzender), Michael Speer, Sebastian Betzin | Aufsichtsratsvorsitzender: Jürgen Betzin
Telefon: +49 721 619096-0 | info@generic.de | www.generic.de
HRB 705226 | Amtsgericht Mannheim | USt.-IdNr. DE202990883
Volksbank Karlsruhe | IBAN: DE73 6619 0000 0054 4641 00 | BIC: GENODE61KA1
Deutsche Bank | IBAN: DE49 6607 0024 0099 8856 00 | BIC: DEUTDEDB660