

LESSONS LEARNED
DER PROTOTYPISCHEN ENTWICKLUNG FÜR KOMMUNEN
KommHUB Talk 20.03.2025



digitalakademie@bw

AGENDA

- KommHUB Auftrag & Kommunales Gewächshaus
- 7 entwickelte Prototypen
- Das Förderprojekt und wie wir bei Prototypen zusammenarbeiten
- Prototyp W-Fragen
- Von Ideen, Bedarfen bis zur Anforderung
- Struktur einer funktionalen Leistungsbeschreibung (beispielhaft für Prototypen)
- Entwicklerkompetenzen
- Technologie-Stack
- Umsetzungsprojekt für Prototypen „in a Nutshell“
- Spiralmodell, Sprints und agile Teams
- Open Source „in a Nutshell“
- 10 Lessons Learned zum mitnehmen

Innovationshub für die Kommunen Baden-Württembergs

- **Prototypische Entwicklung** von innovativen Digitalisierungsideen, bei denen weder auf kommunaler Seite noch auf Seiten einschlägiger kommunaler IT-Dienstleister geeignete Ressourcen zur Verfügung stehen

Die Bereitstellung der Prototypen:

- Durch den **KommHUB bei der Komm.ONE AÖR**
- **Beauftragung geeigneter Entwicklungspartner**
- **Open Source Einsatz** damit eine sinnvolle Weiterentwicklung und Skalierung prinzipiell möglich ist



Kommunales Gewächshaus KommHUB

Wo innovative Ideen-Triebe wirkungsstark für die digitale Verwaltung in Baden-Württemberg wachsen können.

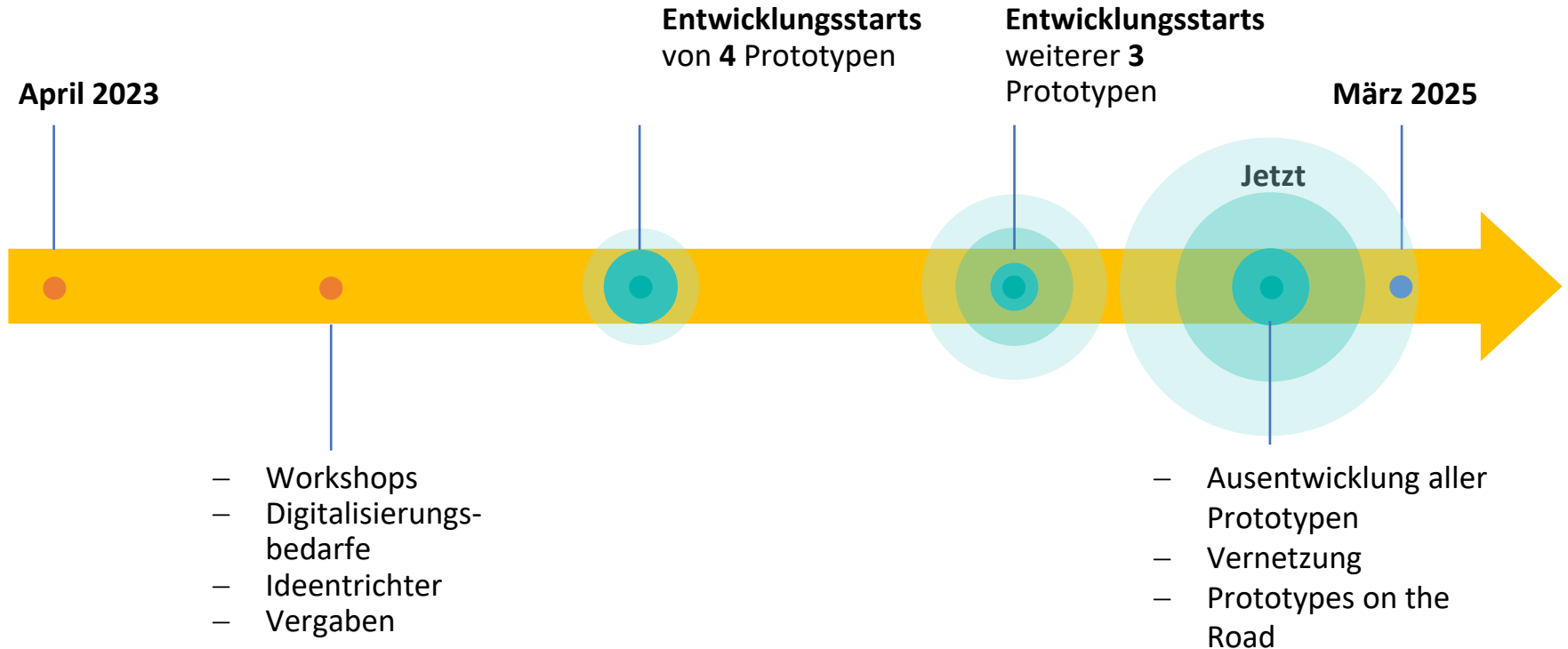


Von der Idee zum
Prototypen.

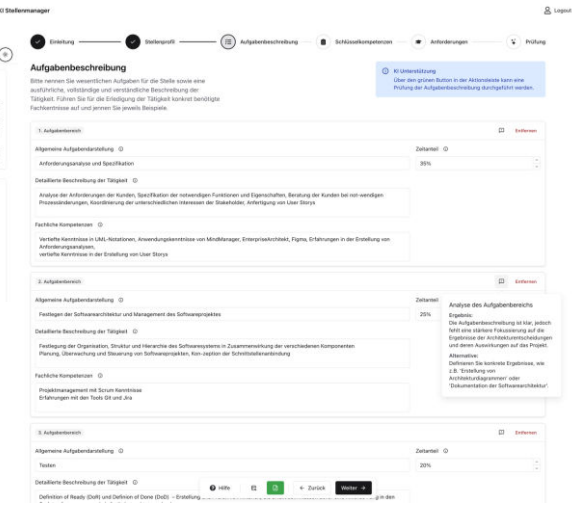
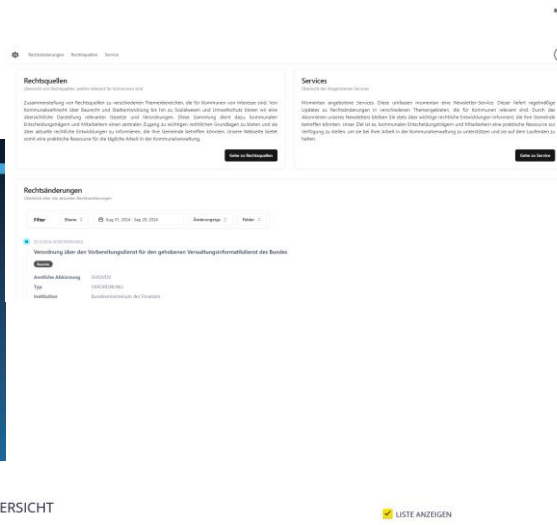
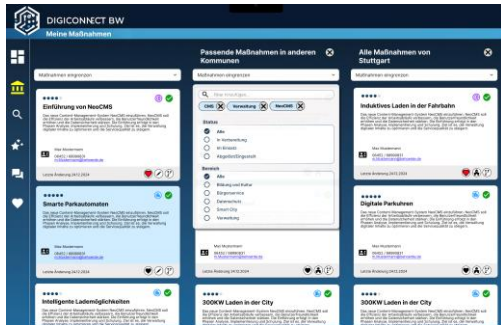
Gemeinsam
und praxisnah.



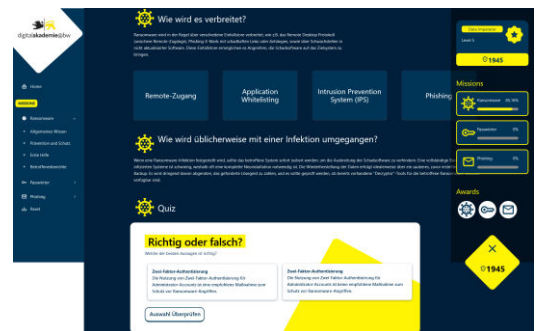
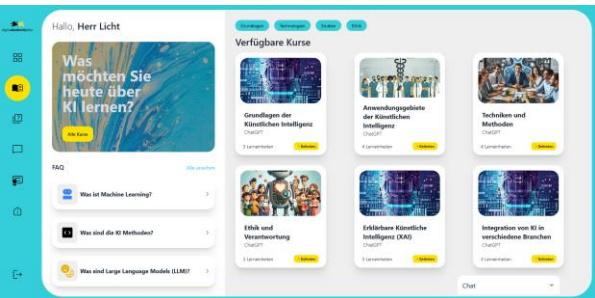
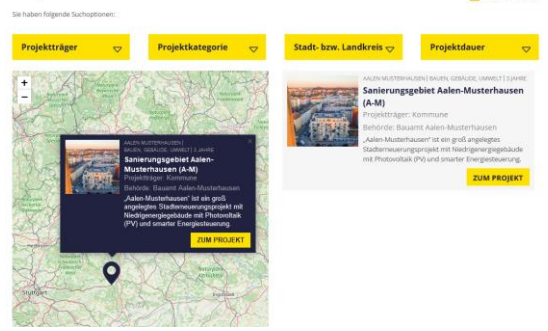
Aktuelle Förderperiode



7 WEB-APP KOMMHUB PROTOTYPEN (2023-2025)



PROJEKTÜBERSICHT



AUSZUG: KOMMHUB FÖRDERPROJEKT

Der KommHUB ist vom Land Baden-Württemberg im Rahmen des Gesamtprojekts „Digitalakademie@bw“ bis März 2025 gefördert. Der KommHUB verfolgt das **Hauptziel kommunale Innovationen mit Mehrwert** zu entwickeln. Dafür werden die wichtigsten digitalen Trendthemen für die kommunale Verwaltung aufgegriffen.

Erreicht wird dies durch eine **gemeinsame prototypische Entwicklung** und experimentelle Pilotierung, die eine landesweite Etablierung in der kommunalen Praxis ermöglichen soll. Weiter getestet, genutzt und weiterentwickelt werden sollen die Entwicklungen insbesondere durch die kommunalen Mitarbeiter der Fachbereiche. Je nach Entwicklungsschwerpunkt der Prototypen liegen die **Hauptnutzergruppen innerhalb der Kommunen bei Digitalisierungsmanagern, Digitallotsen, Fachmitarbeitern, IT-Verantwortlichen, Personalverwaltern**. Die entwickelten Prototypen sollen eine breite Nutzung unter kommunalen Mitarbeiterinnen und Mitarbeitern (in Dezernaten und Ämtern) erreichen.

AUSZUG: „WIE ARBEITEN WIR ZUSAMMEN“

KOMMHUB / ENTWICKLERFIRMEN / KOMMUNEN

Die Entwicklung erfolgt in agiler Zusammenarbeit zwischen dem KommHUB-Team und externen Softwareentwicklern bzw. Entwicklerteams. Das KommHUB-Team fungiert dabei als direkter Ansprechpartner und Koordinator in den einzelnen Entwicklungsphasen. Eine nutzerzentrierte Arbeitskultur steht im Vordergrund. Auch Kommunen und die Digitalakademie können aktiv Input zur Entwicklung geben, der vom KommHUB-Team gesammelt und in die Entwicklung integriert wird. Für offline Workshops am Standort Stuttgart, Hauptsitz der Komm.ONE, steht grundsätzlich der Kreativraum „Innovation-Hub“ zur Verfügung. Funktionen werden in Zyklen spezifiziert, entworfen, implementiert und getestet. Das System wird in Inkrementen erstellt.

Sobald wesentliche Entwicklungsfortschritte (Meilensteine) beim Prototypen erreicht werden, stellt das KommHUB-Team diese kompakt auf der Digitalakademie-Modulwebseite des KommHUB zur Verfügung <https://digitalakademie-bw.de/>. **Der Fokus liegt stets auf der Identifizierung und Umsetzung der geeignetsten Lösung, um den größtmöglichen Beitrag zur Digitalisierung für Kommunen zu erzielen.** Dies geschieht in Abstimmung mit allen Beteiligten, um sicherzustellen, dass die entwickelten Prototypen den Bedürfnissen der Nutzer entsprechen.

PROTOTYP KURZDEFINITION

Ein Prototyp ist eine frühe Version eines Produkts, um Ideen zu validieren.

Unterschiede

- **Prototyp:** Frühe Version mit eingeschränkter Funktionalität
- **Proof of Concept (PoC):** Zeigt, ob eine Idee technisch machbar ist
- **MVP (Minimum Viable Product):** Nutzbares Produkt mit minimalen Funktionen

Quelle: Ries, E. (2011): The Lean Startup.

PROTOTYP vs POC vs MVP vs PILOT

Ein **Prototyp** in der Digitalisierung ist ein verwendbares Model und physisch oder in Form von Software „anfassbar“.

- In der Entwicklung von Softwareprodukten verwenden Teams häufig folgende Formen:
- Vertikaler Prototyp oder Durchstich: Mehrere Schichten, Komponenten oder Anwendungen werden miteinander verbunden, um die technischen Schwierigkeiten früh zu erkennen und zu lösen.
- Horizontaler Prototyp, meist für User Interfaces (UI): Darstellung der Oberflächen, beispielsweise für eine Web-Anwendung oder für eine Smartphone-App.

Ein **Proof of Concept (PoC)** ist eine Aktivität, die häufig mit einem Meilenstein als Projektabschnitt beendet wird. In der Regel erstellen Entwicklungsteams innerhalb des PoC Prototypen zur Demonstration der Machbarkeit relevanter Funktionalitäten oder Eigenschaften.

Ein **MVP** ist ein minimal funktionsfähiges Produkt. Ein MVP ist im Gegensatz zu einer Anwendung in der Pilotierung noch sehr vielen, auch wesentlichen Änderungen unterworfen. Häufig basiert die MVP-Erstellung auf den Erkenntnissen aus PoC und Prototypen.

Unter **Pilot oder Pilotierung** versteht man meistens einen zeitlich beschränkten Echtbetrieb eines Produktes mit einer ausgewählten Anwendergruppe.

Quelle: <https://conciso.de/wie-sich-prototyp-proof-of-concept-mvp-und-pilot-unterscheiden/>

VOM BEDARF BIS SKALIERUNG

Vergaberecht Mindestanforderungen
Ausschreibung Prototyp
Stakeholder Eignungskriterien
Bedarfsermittlung
Skalierung Proof of Concept
Vergabe Markterkundung
Ideenbewertung Agil Verhandlungsvergabe

BEDARF -> ANALYSE -> ANFORDERUNGEN

Methoden für Einblicke in Bedarfe und Erwartungen

- Nutzerforschung
- Personas
- Agile Methoden
- User Stories
- Use Journey Mapping
- Usability Tests
- Kano-Methode
- Good Practices / Best Practices
- ...

Ziele bei der Erhebung von Anforderungen

Ziel sollte es u.a. sein,

- korrekte,
- vollständige,
- eindeutige,
- widerspruchsfreie,
- nach Wichtigkeit und/oder Stabilität bewertete,
- prüfbare
- und verfolgbare

Anforderungen zu ermitteln.

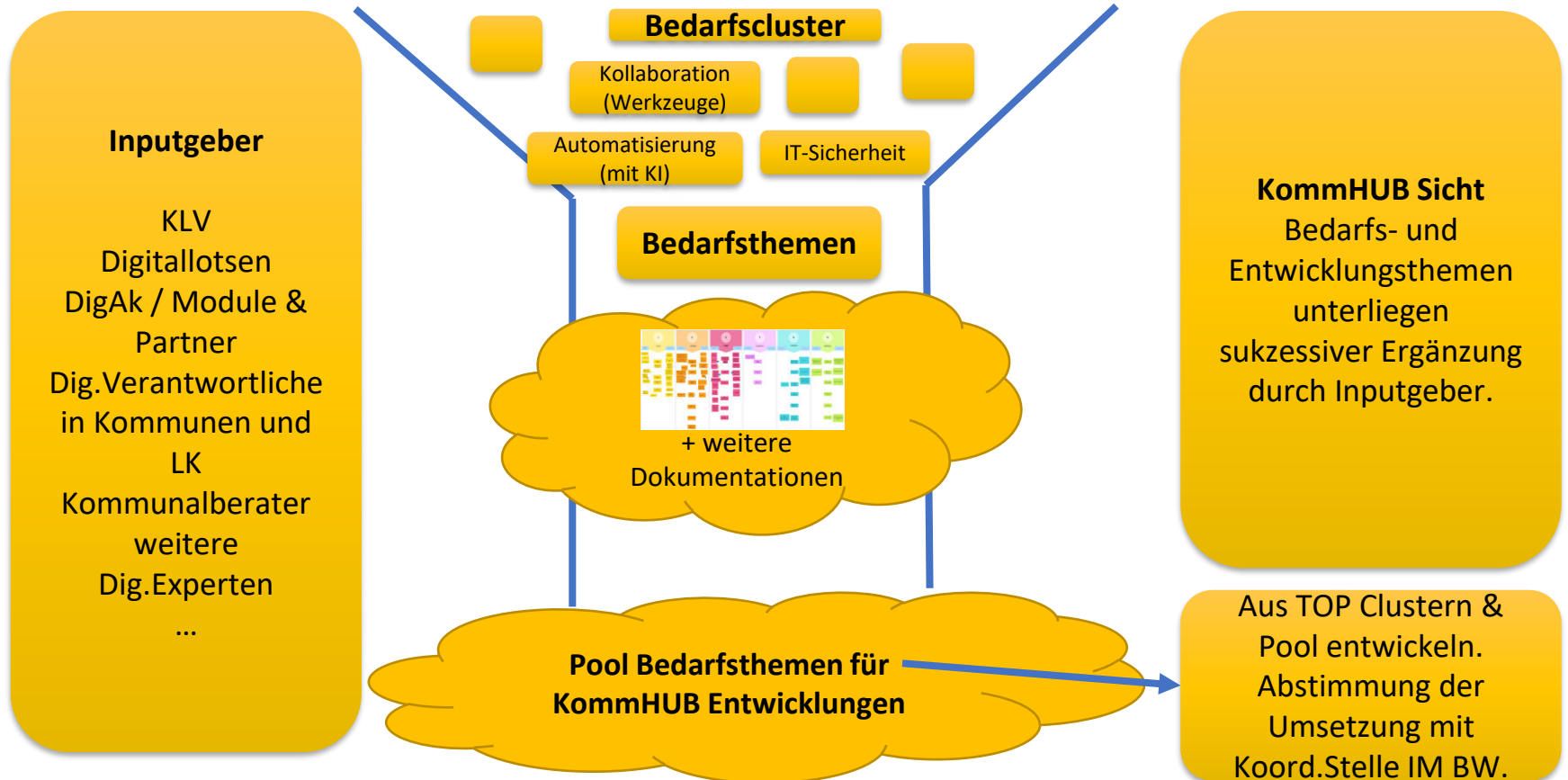
Damit dies gelingt, ist die [Verwendung eines definierten Prozesses sehr empfehlenswert](#). Es gilt bspw. den Systemkontext zu definieren, die Stakeholder zu managen, Szenarien zu verwenden und natürlich Erkenntnisse zu dokumentieren und zu prüfen.

Quelle: <https://t2informatik.de/wissen-kompakt/anforderung/>

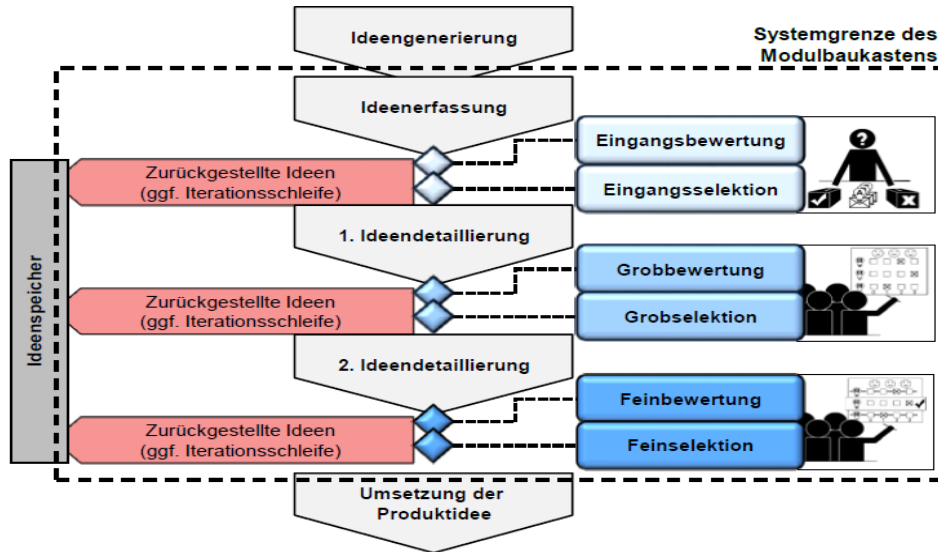


Anforderungen sind die konkreten Kriterien oder Bedingungen, die erfüllt werden müssen, um den Bedarf zu decken → Anforderungen sind präziser als der Bedarf.

KommHUB Trichter: Dringliche Digitalisierungsbedarfe Ideeneingang → Bewertung → Auswahl für Entwicklung



EINSCHUB: GENERISCHER IDEENBEWERTUNGSPROZESS & IDEENBEWERTUNGSDIMENSIONEN



Quelle: Generischer Ideenbewertungsprozess als Modulbaukasten (Messerle, Mathias (2016): Methodik zur Identifizierung der erfolgversprechendsten Produktideen in den frühen Phasen des Produktentwicklungsprozesses. Universität Stuttgart. Stuttgart. Seite 63).

Beispielhaft: Grundlegende Bewertungsdimensionen

- Attraktivität/Relevanz der Idee für die Kommunen
- Mehrwert der Idee für die Kommune
- Attraktivität der Idee für die Bürger:innen
- Umsetzbarkeit der Idee mit existierenden technologischen Möglichkeiten
- Wirtschaftliche Realisierbarkeit und Rentabilität
- Gesamtbewertung der Idee
- > Alleinstellungsmerkmal der Idee
- > Strategie-Fit zu den Projektzielen

BEISPIELHAFT: FUNKTIONALE LB FÜR EIN AGILES IT-SOFTWARE-ENTWICKLUNGSPROJEKT [PROTOTYP] TEIL 1

1. Projektübersicht (ca. 0,5 Seite)

- Projektname und Kurzbeschreibung: Eine prägnante Einführung, die den Gesamtzweck des Projekts darstellt.
- Ziele des Prototypen: Erläuterung der Hauptziele und des gewünschten Endzustands. Hier kann z. B. auf die Problemstellung eingegangen werden, die der Prototyp lösen soll.
- Stakeholder und Zielgruppe: Erwähnung der Hauptnutzer und Beteiligten, für die der Prototyp entwickelt wird.

2. Projektanforderungen und -umfang (ca. 1 Seite)

- Funktionale Anforderungen: Auflistung der wichtigsten Funktionen, die der Prototyp abdecken soll. Jedes Feature sollte kurz beschrieben werden und auf die spezifische Problemstellung eingehen, die es adressiert.
- Nicht-funktionale Anforderungen: Angabe von Anforderungen, die die Systemqualität betreffen, wie z. B. Benutzerfreundlichkeit, Performance und Sicherheitsaspekte.
- Abgrenzungen: Kurze Klarstellung, welche Funktionalitäten bewusst nicht Teil des Prototyps sind, um Missverständnisse zu vermeiden und die Zielsetzung einzugrenzen.

Funktionale LB dienen als Rahmen für die Entwicklung eines Prototypen. Insbesondere in agilen IT-Softwareprojekten. Grundsätzliche Informationen beispielsweise hier:

„So gestalten Sie agile IT-Projektverträge richtig!“ <https://comp-lex.de/agile-it-vertragsgestaltung/>

BEISPIELHAFT: FUNKTIONALE LB FÜR EIN AGILES IT-SOFTWARE-ENTWICKLUNGSPROJEKT [PROTOTYP] TEIL 2

3. Technische Rahmenbedingungen und Architekturvorgaben (ca. 1 Seite)

- Technologie-Stack und Plattform: Beschreibung der Haupttechnologien, die im Prototyp zum Einsatz kommen sollen (z. B. Programmiersprachen, Frameworks, Datenbanken, APIs).
- Architekturmodell: Ein einfaches Schema oder Diagramm, das die geplante Architektur des Prototyps aufzeigt, z. B. eine Client-Server-Struktur, Microservices-Ansatz etc.
- Integrationsanforderungen: Definition der Schnittstellen zu externen Systemen oder Datenquellen, falls notwendig.

4. Agiler Entwicklungsprozess und Rollenverteilung (ca. 0,5 Seite)

- Vorgehensweise: Erklärung, wie der agile Entwicklungsprozess gestaltet wird (z. B. Scrum, Kanban) und welche iterativen Entwicklungsstufen geplant sind.
- Rollen und Verantwortlichkeiten: Zuweisung von Kernrollen wie Product Owner, Scrum Master und Entwicklerteam, um die Verantwortung im Team klar zu definieren.

BEISPIELHAFT: FUNKTIONALE LB FÜR EIN AGILES IT-SOFTWARE-ENTWICKLUNGSPROJEKT [PROTOTYP] TEIL 3

5. Zeitplan und Meilensteine (ca. 0,5 Seite)

- Iterationen und Sprints: Ein grober Zeitplan, der den Ablauf der geplanten Sprints/Iterationen darstellt, idealerweise mit der Definition von Zwischenzielen (Meilensteine) für jede Phase.
- Übergang von Prototyp zu Produktentwicklung: Hinweise darauf, wie die Ergebnisse des Prototypen in die potenzielle Weiterentwicklung fließen können.

6. Abnahmekriterien und Erfolgskriterien (ca. 0,5 Seite)

- Definition der Abnahmekriterien: Detaillierte, messbare Kriterien, die der Prototyp erfüllen muss, um als erfolgreich abgeschlossen zu gelten (z. B. Funktionsumfang, Benutzerfeedback).
- Tests und Feedback-Schleifen: Kurze Erläuterung, wie die Tests (z. B. Usability-Tests, technische Tests) ablaufen sollen und welche Feedback-Schleifen integriert sind.

KURZER BLICK AUF ENTWICKLER-KOMPETENZFELDER (BEISPIELHAFT)

Kompetenzfelder und Erfahrung in IT-Entwicklung (beispielhaft)

- Entwicklung von Web-Apps
- Konzeptentwicklung Applikationen
- Anforderungsanalyse (anhand von User Stories)
- Architektur
- UI Design
- UX Design
- Agile Softwareentwicklung
- Qualitätssicherung
- Deployment und Go Live
- Einsatz Open Source

Weitere Kompetenzfelder und Erfahrungen (beispielhaft):

- UX für Bürger: Nutzerfreundlichkeit und Barrierefreiheit
- App Design: Modulares, ansprechendes Design
- App Entwicklung: Sichere und fehlerfreie Entwicklung
- Schnittstellen (APIs): Standardisierte und sichere APIs
- Öffentlicher Sektor: Compliance und rechtliche Standards

„TECH-STACK“ (BEISPIELHAFT)

Übersicht (zentraler) Programmiersprachen und Technologien für **Web-App-Entwicklung (interaktive browserbasierte Apps)**:

Frontend:

- HTML/CSS, JavaScript, TypeScript
- Frameworks: React, Angular
- UI-Bibliotheken: Bootstrap, Tailwind CSS

Backend:

- Node.js, Python (FastAPI), Ruby on Rails, Java (Spring Boot), C# (ASP.NET), Go, Rust
- APIs: REST, GraphQL, WebSockets (Echtzeit-Kommunikation)
- Datenbanken: SQL (PostgreSQL), NoSQL (MongoDB, DynamoDB)

TECHNOLOGIEN



<https://moguru.de/web-anwendungsentwicklung/>

UNSER TECH STACK

— Programmiersprachen

- C#
- Java
- Swift
- XML, HTML, CSS
- SQL

— Betriebssysteme

- Microsoft Windows
- MacOS, IOS
- Android
- Linux

— Datenbanken

- Microsoft SQL-Server
- MySQL

— Entwicklungsumgebungen

- Microsoft Visual Studio
- JetBrains ReSharper
- JetBrains Rider
- XCode

— Methoden und Techniken

- Scrum
- Pair programming

<https://www.horaios.de/leistungen/#VorteilevonIndividualsoftware>

UMSETZUNG PROTOTYP IN KURZPROJEKT | TEIL 1 (BEISPIELHAFT)

Projektziele

- **Bedarfsanalyse und Priorisierung** kommunaler Digitalisierungsprojekte.
- Prototypische Entwicklung von **Web-Applikationen**, die intensiv getestet und weiterentwickelt werden können.
- Vorbereitung eines **Minimum Viable Products (MVP)** als Basis für die Entscheidung zur Vollausentwicklung in späteren Projektphasen.
- Sicherstellung der **Nachhaltigkeit und Flexibilität** durch den Einsatz von Open-Source-Technologien.

UMSETZUNG PROTOTYP IN KURZPROJEKT | TEIL 2 (BEISPIELHAFT)

Rahmenbedingungen

Projektlaufzeit: 6 Monate

Personalaufwand:

- Innovationsmanagement/Projektleitung: 12 Personentage (PT)
- Entwicklung: 30 PT
- Gesamtaufwand: 42 PT

Umsetzung:

- Option A: Entwicklung durch externe Partner (nach Ausschreibung)
- Option B: Nutzung interner Entwicklungskapazitäten

UMSETZUNG PROTOTYP IN KURZPROJEKT | TEIL 3 (BEISPIELHAFT)

Phase 1: Bedarfsermittlung und Priorisierung (Monat 1)

Ergebnis:

- Priorisierte Liste der Digitalisierungsprojekte

Phase 2: Anforderungsmanagement und Konzeptentwicklung (Monat 2–3)

Ergebnisse:

- Anforderungsspezifikation
- Dokumentation der funktionalen Leistungsbeschreibung

Phase 3: Prototypentwicklung (Monat 4–5)

Ergebnisse:

- Funktionaler Prototyp, der als Basis für einen MVP geeignet ist

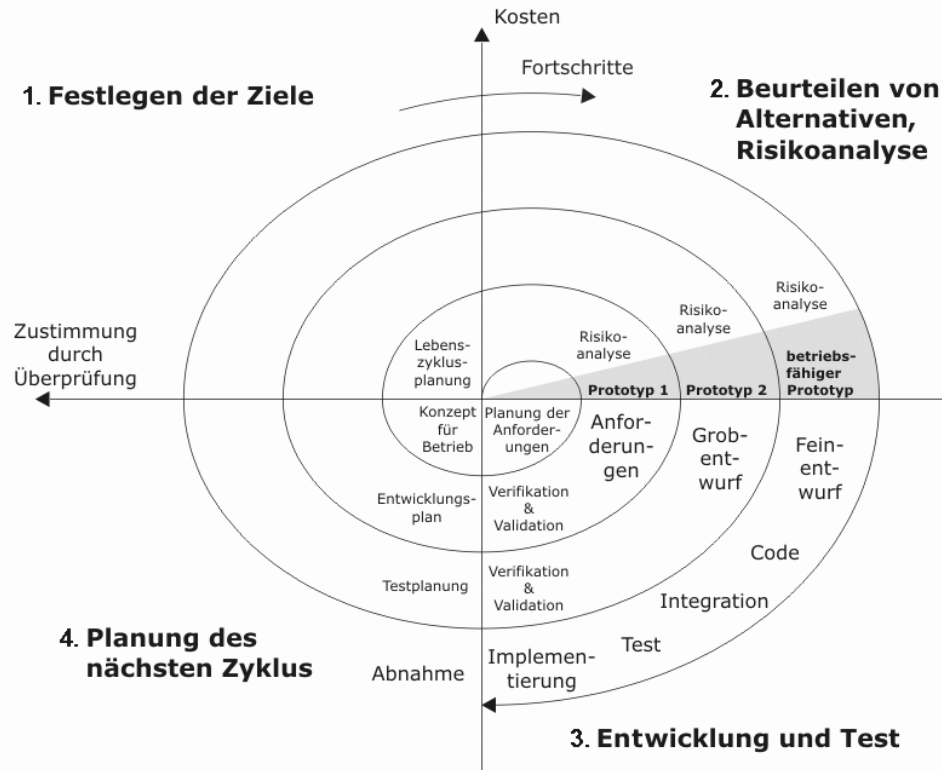
Phase 4: Übergabe und Abschluss (Monat 6)

Ergebnisse:

- Dokumentation für die Weiterentwicklung
- Klare Entscheidungsgrundlage für die MVP-Phase

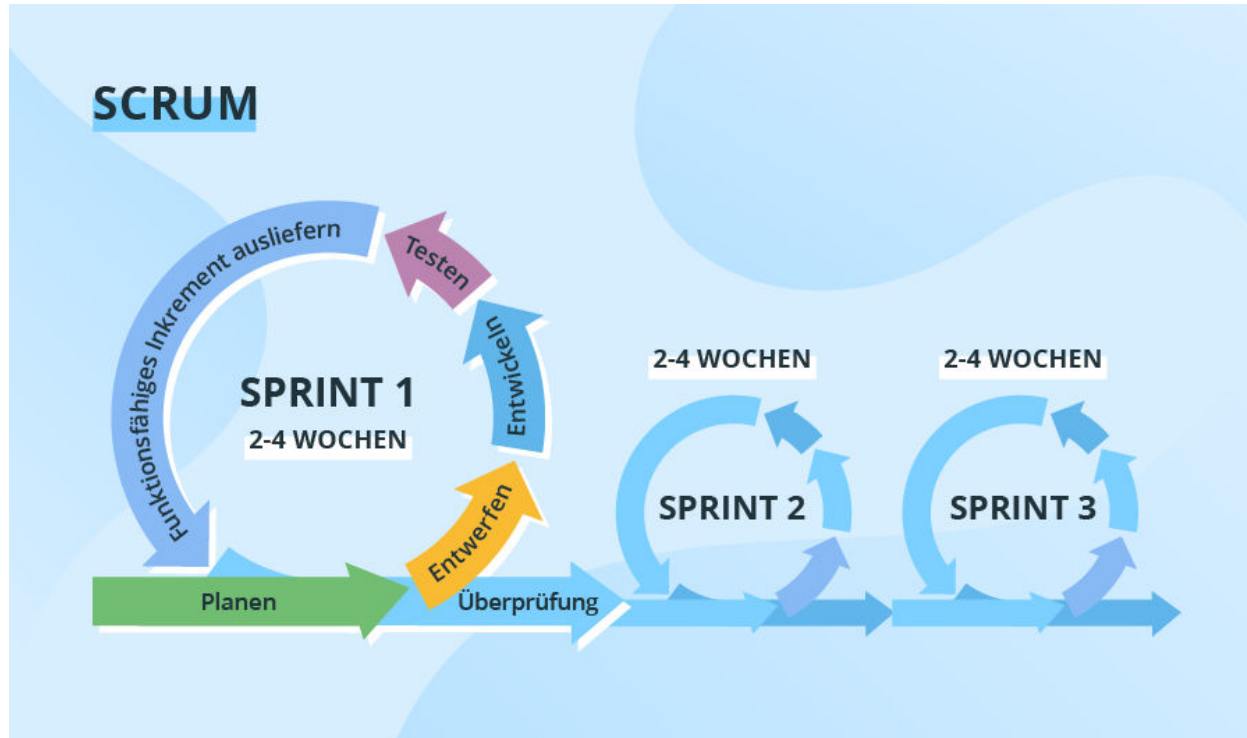
Übergang Phase 2 in Phase 3: Option A und B berücksichtigen. Bei Option A:
Dauer für Vergabe zusätzlich einplanen.

DAS SPIRALMODELL: VORGEHENSMODELL IN DER SOFTWAREENTWICKLUNG



Quelle: Conny, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=3697528>
<https://de.wikipedia.org/wiki/Spiralmodell>

SCRUM UND SPRINTS: AGILE ENTWICKLUNGSMETHODE

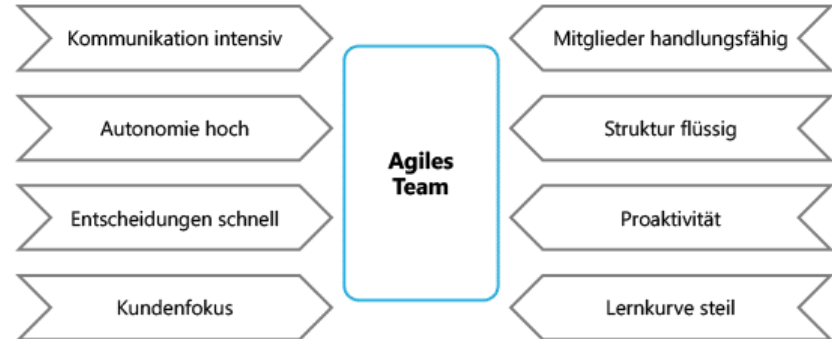


Quelle: <https://www.scnsoft.de/blog/vorgehensmodelle-der-softwareentwicklung#agile-modelle>

AGILITÄT UND AGILE TEAMS



© Dr. Florian Becker | www.wpgs.de



© Dr. Florian Becker | www.wpgs.de

OPEN SOURCE IN A NUTSHELL

„**Open Source** bedeutet, dass der Quellcode einer Software für **jede Person und Institution frei zugänglich** ist. Der Code ist frei verfügbar und kann beliebig oft **kopiert, modifiziert, verbreitet** und **genutzt** werden. So können neue Softwarelösungen auf diesem Code aufbauen. Die **Weiterentwicklung** bestehender Lösungen wird durch die Offenheit und Transparenz des Codes und die Mitwirkung vieler ermöglicht. Verschiedene **Open-Source-Lizenzen** regeln dabei die Art und Weise, wie auf einer OSS neue Lösungen aufgebaut werden können und wie Veränderungen weitergegeben werden dürfen.“

Quelle: The Open Source Initiative: <https://opensource.org>, zusammengefasst von Fraunhofer IESE, im Rahmen des 13. Themenwerkstatt Open-Source-Software.

Warum Open Source für Kommunen wichtig ist:

Nachhaltigkeit, Transparenz, Kosteneinsparung.

Nutzen
Projekt Open CoDE
Community
Open Source
Lizenzen
Anmeldung
Diskussionsforum
GitLab
Softwareverzeichnis

Lizenzen

Wie unterscheiden sich die verschiedenen Open Source Lizenzen? ^

Historisch haben sich zahlreiche Open Source Lizenzen entwickelt, die sich oftmals nur in wenigen Details unterscheiden. Wesentlich ist jedoch die Differenzierung danach, ob die Lizenz ein sog. „Copyleft“ enthält. Unter einem Copyleft versteht man eine Lizenzvertragsklausel, wonach zwar die Modifikation der Software jedermann gestattet, aber zugleich an die Pflicht geknüpft wird, diese Modifikationen bei der Weitergabe ebenfalls wieder der Ursprunglizenz zu unterstellen. Auf diese Weise wird sichergestellt, dass eine Art „Software-Allmende“ entsteht, weil auch Weiterentwicklungen der Software stets als Open Source Software zur Verfügung stehen. Beispiele für Copyleft-Lizenzen sind die GNU General Public License (GPL), die European Public License (EURL) und die Eclipse Public License (EPL).

Lizenzen ohne ein solches Copyleft werden „Permissive Licenses“ genannt und erlauben auch die Verwendung der Software in rein proprietären Lösungen. Beispiele für Permissive Licenses sind die Apache License 2.0, die MIT License und die BSD License.

Was sind gängige Open Source Lizenzen? ^

Es empfiehlt sich eine anerkannte und verbreitete Open Source Lizenz zu verwenden, weil dies den Nutzenden das Verständnis vereinfacht und weniger Kompatibilitätsprobleme erzeugt. Zu den anerkannten Open Source Lizenzen gehören insbesondere folgende Lizenzen:

(1) Lizenzen mit einem strengen Copyleft:

- GNU General Public License, Versionen 2 und 3 (GPL-2.0, GPL-3.0)
- GNU Affero General Public License, Version 3 (AGPL-3.0)

(2) Lizenzen mit einem schwachen Copyleft:

- Mozilla Public License, Version 2 (MPL-2.0)
- Eclipse Public License, Version 2 (EPL-2.0)
- GNU Lesser General Public License, Versionen 2.1 und 3 (LGPL-2.0, LGPL-3.0)

(3) Lizenzen ohne Copyleft (Permissive Licenses):

- MIT License (MIT)
- Apache License 2.0 (Apache-2.0)
- BSD 2 Clause License (BSD-2-Clause)



Handlungsempfehlung

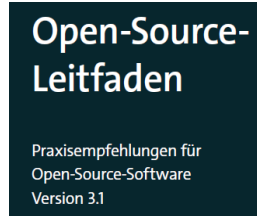
*„**Lizenzangabe:** Es wird empfohlen, in der Ausschreibung für die Entwicklung von OSS die exakte Bezeichnung der gewünschten Lizenz anzugeben. Welche Lizenz konkret verwendet wird, sollte der Auftragnehmerin beziehungsweise dem Auftragnehmer explizit mitgeteilt werden. Bei der Vertragsgestaltung ist weiterhin zu klären, wer welche Entscheidungen trifft und wer welche Rechte an der entwickelten Software hat (Regelung der Nutzungsrechte).“*

(Quelle: Open-Source-Software in Kommunen, Seite 27. Smart City Dialog, BBSR)

OPEN SOURCE: QUELLEN | TEIL 1

BITKOM

- Open Source Leitfaden 3.1
(Standard, in Deutsch)



- Open Source Monitor

<https://www.bitkom.org/Bitkom/Publicationen/Studie-Open-Source-Monitor>

KGST

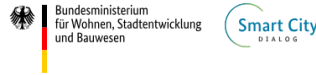
<https://www.kgst.de/>

- Open Source in Kommunen. Ein Baustein für mehr Digitale Souveränität. Teil 1: Grundverständnis, Potenziale und Herausforderungen (5/2021)
- KGST Denkanstoß: Open Source in Kommunen. Ergebnisse einer Umfrage
- Open Source in Kommunen. Teil 2: Aufbau einer Open-Source-Governance (18/2024)

Open Source in Kommunen
Ein Baustein für mehr Digitale Souveränität
Teil 1: Grundverständnis, Potenziale
und Herausforderungen

OPEN SOURCE: QUELLEN | TEIL 2

- **Smart City Dialog**, z.B 13. Themenwerkstatt „Open-Source-Software: Vom öffentlich zugänglichen Code bis zur Nutzung in Kommunen“ 14.03.2024



- **Open-Source-Software in Kommunen (Smart City Dialog, BBSR)**

Open-Source-Software-Lizenzen

Tabella 4. Auswahl von OSS-Lizenzen | Quelle: <https://www.bbsr.de/de/bsg>

Lizenzkategorie	Beispiele
Lizenzen mit strenger Copyleft	• GNU General Public License, Version 2 und 3 (GPL-2, GPL-3) • GNU Affero General Public License, Version 3 (AGPL-3.0)
Lizenzen mit schwächerer Copyleft	• Mozilla Public License, Version 2 (MPL-2.0) • Eclipse Public License, Version 2.0 (EPL-2.0) • GNU Lesser General Public License, Version 2.1 und 3 (LGPL-2.1, LGPL-3.0)
Lizenzen ohne Copyleft (Proprietäre Lizenzen)	• MIT License (MIT) • Apache License 2.0 (Apache 2.0) • BSD 2-Clause license (BSD-2-Clause)

- **VITAKO**: Handreichung: Ausschreibung von Open Source Software <https://vitako.de/archive/6331>
- **Open Source Business Alliance e.V.**: Handreichung: Zur Nutzung der EVB-IT beim Einsatz von Open Source Software <https://osb-alliance.de/publikationen/veroeffentlichungen/handreichungen-zur-nutzung-der-evb-it-beim-einsatz-von-open-source-software>

Einrichtungen

- **Zendis**

Das Zentrum für Digitale Souveränität der Öffentlichen Verwaltung (ZenDiS) unterstützt die Öffentliche Verwaltung in Bund, Ländern und Kommunen dabei, sich aus kritischen Abhängigkeiten von einzelnen Technologieanbietern zu lösen.

<https://www.zendis.de>

- **openCode**

openCode ist die zentrale Plattform der öffentlichen Verwaltung für Open-Source-Software. Sie richtet sich an alle, die die digitale Transformation der Verwaltung mitgestalten möchten: in erster Linie Mitarbeitende der öffentlichen Verwaltung und Entwickler:innen, die an der Verbesserung von Verwaltungssoftware interessiert sind.

<https://opencode.de/de/faq>

1. Frühzeitige Nutzerbeteiligung lohnt sich – Regelmäßiges Feedback von Verwaltungsmitarbeitenden und Bürger:innen verbessert die Entwicklung.

2. Iteratives Vorgehen spart Zeit und Kosten – Kleine, testbare Schritte ermöglichen flexible Anpassungen und verhindern unnötigen Mehraufwand.

3. Klare Anforderungen bringen Sicherheit – Präzise User Stories und Workflows helfen, Missverständnisse und Fehlentwicklungen zu vermeiden.

4. Technische Nachhaltigkeit zahlt sich aus – Open-Source-Technologien und modulare Architekturen erleichtern langfristige Nutzung und Erweiterung.

5. Prototyp ist nicht gleich Endprodukt – Ein Prototyp dient der Erprobung und Validierung, nicht als fertige Lösung – Weiterentwicklung ist essenziell.

6. Agile Methoden unterstützen Anpassungsfähigkeit – Scrum oder Kanban fördern den Austausch zwischen Verwaltung, Entwicklern und Stakeholdern.

7. Interdisziplinäre Teams bringen Mehrwert – Die Zusammenarbeit zwischen Fachabteilungen, IT und Bürger:innen verbessert das Projektergebnis.

8. Datenschutz und Sicherheit von Anfang an bedenken – DSGVO und IT-Sicherheit sollten direkt in die Planung integriert werden.

9. Einfachheit überzeugt – Intuitive Bedienung und klare Prozesse sind wichtiger als eine Vielzahl an Funktionen.

10. Prototypen helfen bei Entscheidungen – Sie bieten eine gute Grundlage für Diskussionen, bevor größere Budgets investiert werden.

Eine Veranstaltung der Digitalakademie@bw

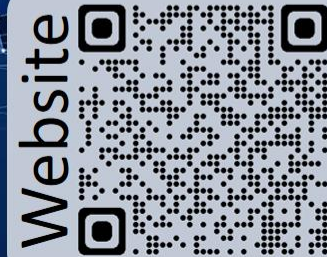
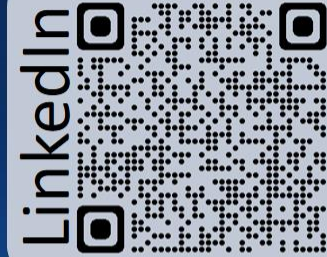
Ein Förderprojekt des Landes



Baden-Württemberg

MINISTERIUM DES INNEREN, FÜR DIGITALISIERUNG UND KOMMUNEN

Immer auf dem
Laufenden bleiben ...



digitalakademie@bw