Softwarewartung

für Accessler

Karl Donaubauer

AEK 20

Motivation

Relevanz

- mind. 2/3 der Arbeitszeit, täglich, Jahrzehnte, viel Geld

Mauerblümchen

- (fast) Keine Methodenlehre, Literatur, Webseiten, Vorträge…
- Zu vielschichtig, fader Alltag?

Beratung

- Verständnis + Methodik für meine Beratungsprojekte

Quellen

Softwarewartung

Bommer, Spindler, Barr

dpunkt, 2008

http://softwarewartung.net

Wikipedia – SW Maintenance

Blog-Artikel zu engl. Quellen

Relevanz 1

Nach Capers Jones zeigt sich in einer **Industrie nach ihrem 50. Geburtstag** ein kritisches Phänomen: Es werden **mehr Leute zur Pflege** des Produkts **als zur Herstellung** gebraucht. So geschehen mit der Autoindustrie, so geschehen mit der Softwareindustrie 1990. Seither sind mehr Personen mit Weiterentwicklung und Wartung beschäftigt als mit Erstentwicklungen.

(Christian Göhring)

Relevanz 2

Integer 32768… Überlauf… ~~Fehlerbehandlung~~

Coderecycling, Test, Doku, Projektgröße etc.

Softwarewartung

Die **Veränderung** eines Softwareprodukts

nach dessen **Auslieferung**,

um Fehler zu **beheben**,

Performanz oder andere Attribute zu **verbessern**,

oder **Anpassungen** an die veränderte Umgebung vorzunehmen.

(IEEE610)

Warum ist Wartung nötig?

Erbe der Erstentwicklung

- Bugs, Missverständnisse, Schwächen

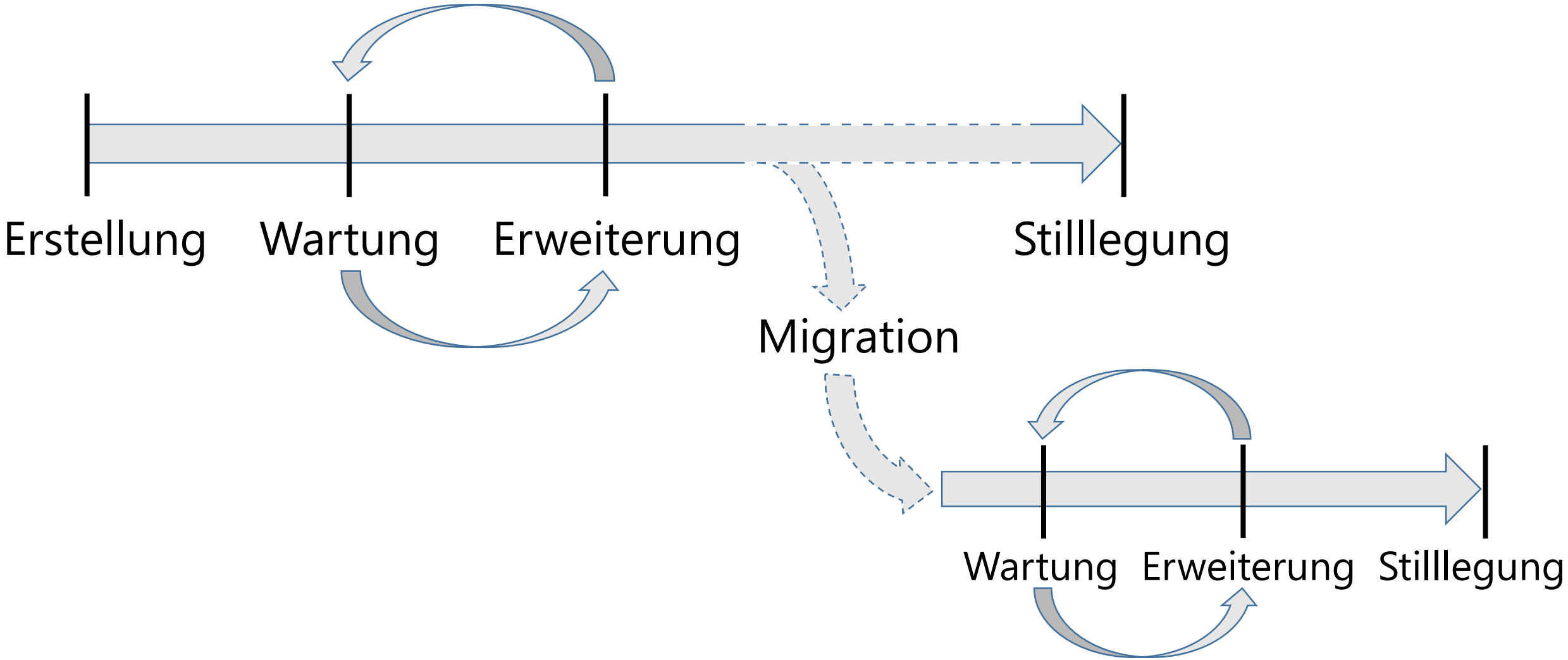
Unbeweglichkeit (Lack of Movement)

- Technische und fachliche Umgebung ändern sich
- System veraltet „von selbst“

Änderungspfusch (Ignorant Surgery)

- Architektur- und Designprinzipien nicht bekannt/beachtet
- Zeitdruck, Unkenntnis, es gab nie Prinzipien
- Führt zu höherer Komplexität des Systems → schwer verständlich, Bugs, Zerfall

SW-Evolution – Lebenszyklus



SW-Evolution: Gründe

Wandel des Umfeldes

- Erweiterungen notwendig
- Risiko des inneren Zerfalls
- Je größer das Wachstum desto intensivere Wartung nötig

Wandel durch die Software selbst

- Nutzung der SW bringt neue Erkenntnisse
- Beeinflusst Umfeld und Weiterentwicklung
- Rückkoppelung

SW-Evolution: 8 Gesetze (Lehman & Belady)

1. Kontinuierliche Veränderung
   - Kontinuierliche Anpassungen an Umfeld oder Zufriedenheit sinkt (Lack of Movement, Rückkoppelung)
2. Zunehmende Komplexität
   - Erweiterungen, Änderungen, Ignorant Surgery
6. Kontinuierliches Wachstum
   - Funktionsumfang für Benutzer ständig erweitert oder Zufriedenheit sinkt
7. Abnehmende Qualität
   - Benutzer empfinden sinkende Qualität, da sich Ansprüche ändern → Kompromisse, Unschärfen

→ Rigorose Wartung, Refactoring, planvolle Anpassung, permanentes Anforderungsmanagement

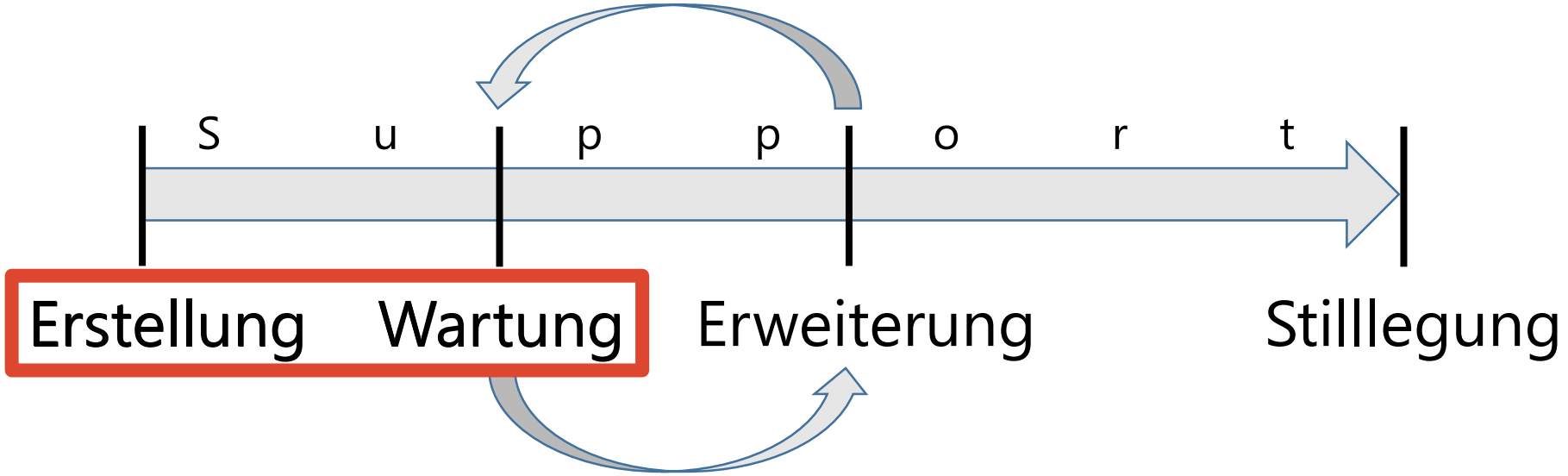
Wartbarkeit

Die **Einfachheit**, mit der ein Softwaresystem oder eine Komponente **modifiziert** werden kann,

um Fehler zu beheben,
Performanz oder andere Attribute zu verbessern oder
Anpassungen an die veränderte Umgebung vorzunehmen.

(IEEE610)

SW-Evolution – Lebenszyklus



Erstentwicklung – Wartbarkeit

Architektur und Design

- Erweiterbar, modular, übersichtlich, nutzt Entwurfsmuster

Codierrichtlinien, Dokumentation, Tests

Mechanismen zur Unterstützung der Wartbarkeit

- Updates im laufenden Betrieb, Automatische Verteilung der FEs
- Fachliche+Technische Ansprechpartner bei Kunden/Abteilungen
- Kommunikation: Fernwartung, Teamviewer etc.

→ Standards für Neuentwicklungen

Erstentwicklung – Wartbarkeit

Qualitätssicherung in der Entwicklung

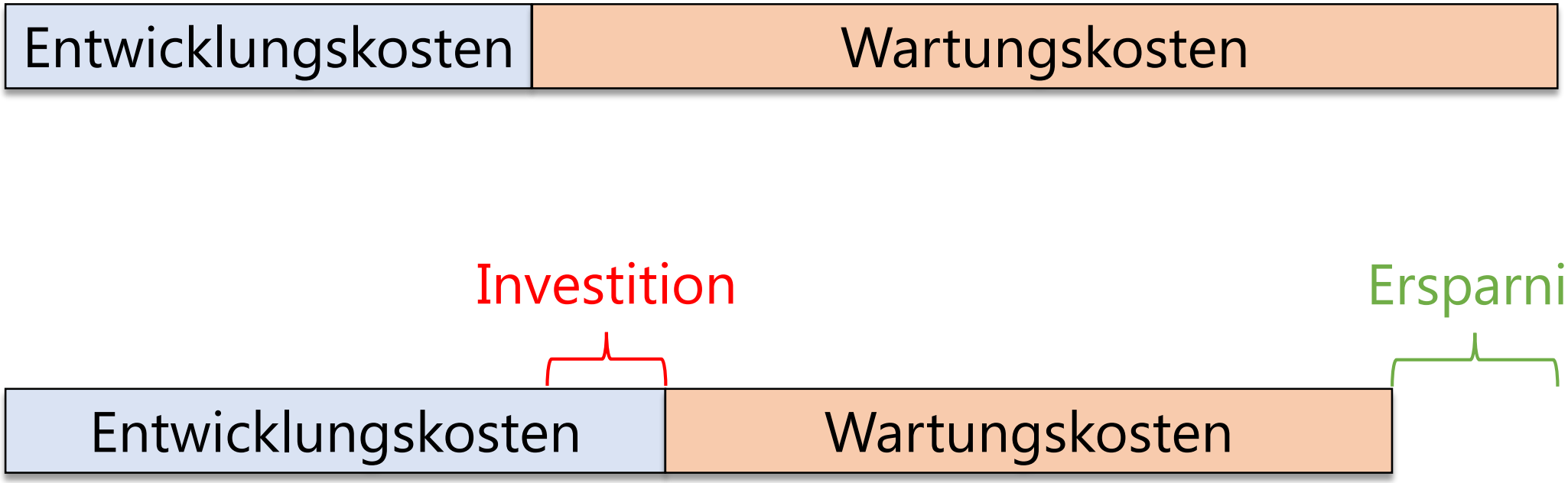
- später oft nicht mehr nachzuholen

Termin- und Kostendruck

- „Lastabwurf“ gegen Ende des Projektes
- Richtlinien, Logging, Doku, Fehlerbehandlung (!)

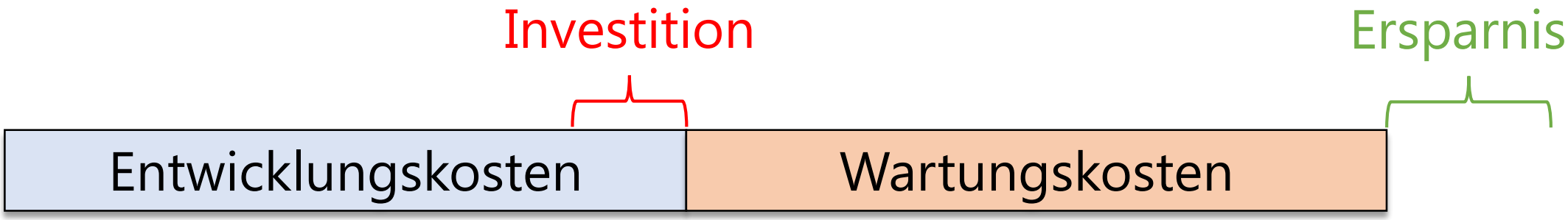
Wette

- Aufwand zahlt sich später aus
- Zeitpunkt und Umfang unbekannt
- Optimum zw. Aufwand und Nutzen


Entwicklungskosten
Wartungskosten
Investition
Ersparnis
Entwicklungskosten
Wartungskosten

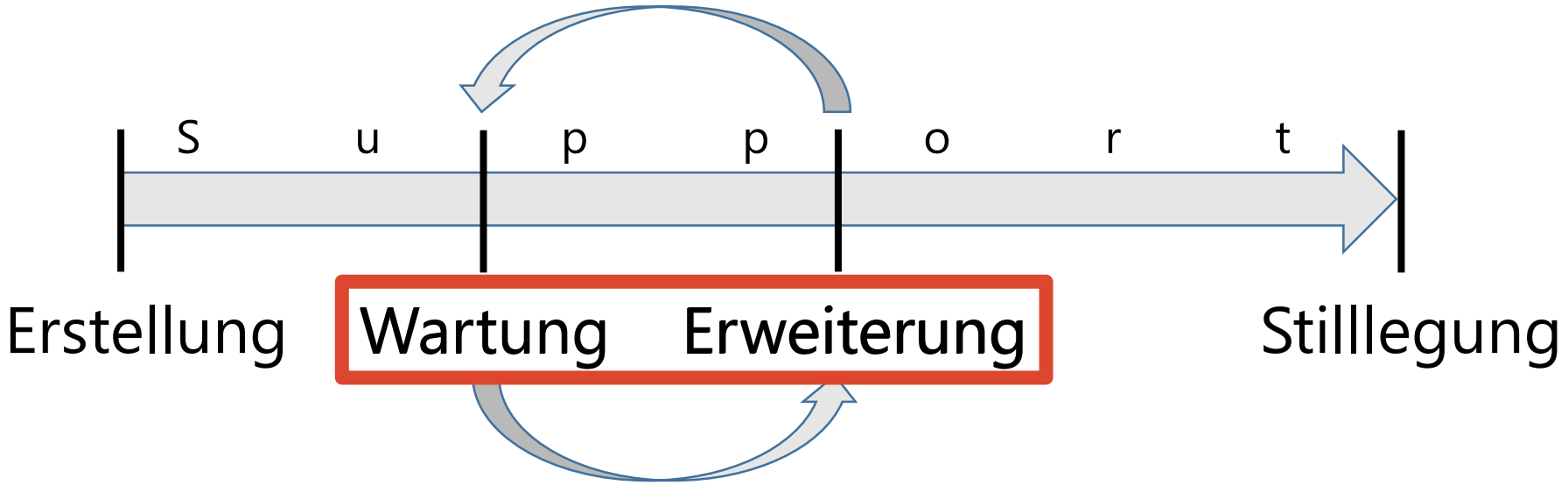
Exkurs – AEK 19

Entwurfsmuster



Access mit Richtlinien+Entwurfsmustern

SW-Evolution – Lebenszyklus



Wartung	Weiterentwicklung
Beheben, verbessern, anpassen	Neue Funktionalität
Neuer Build/Unterversion 1.2.3456 Patch/Bugfix/Update schnell, oft	Neue Version 2.0 selten, umfangreiche Tests
Laufender Prozess (Auslieferung bis Stilllegung) oft unvorhersehbar (Absturz, Bug)	Projektcharakter planbar

- Übergänge fließend z.B. Sanierung, Migration
- Integration umstritten

Kategorien der Wartung

	Korrektur	**Verbesserung**
Reaktiv	**Korrektive** Wartung (Restfehler, Bugs, Abstürze...)	**Adaptive** Wartung (gg. Lack of Movement)
Proaktiv	**Präventive** Wartung (latente Fehler)	**Perfektionierende** Wartung (Performance, Wartbarkeit etc.)

+ Support
  - Beantwortet Fragen und Problem rund um den SW-Einsatz

Wartungsplan

Bereits während der Entwicklung erstellen

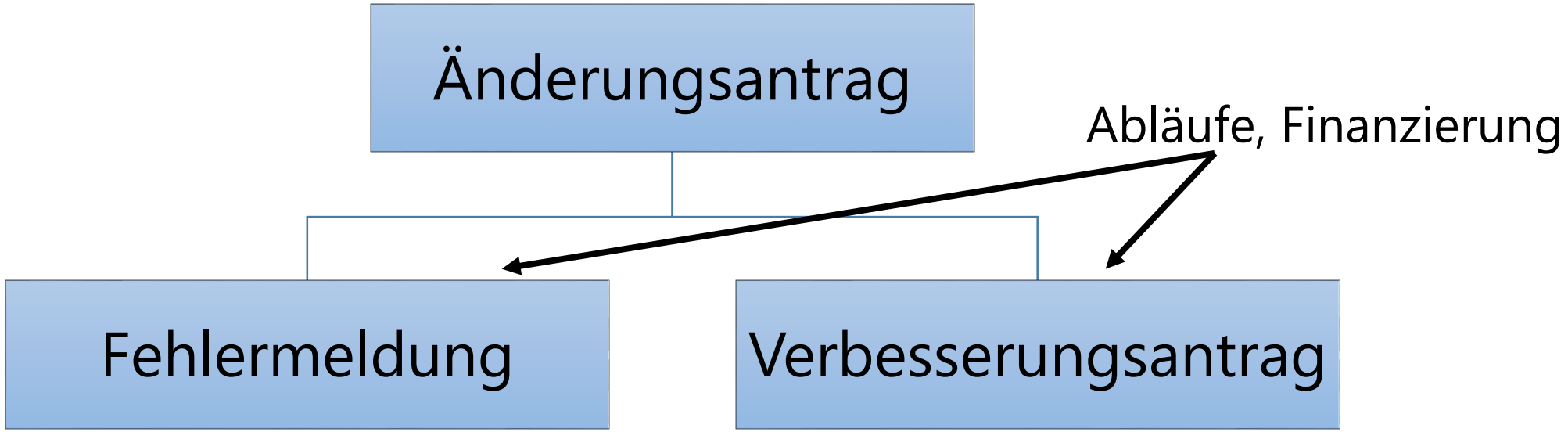
Fragen z.B.

- Welche Arten von Wartungseingriffen wie/durch wen?
- Welches Niveau der Dokumentation aufrecht erhalten?
- Wie/durch wen Support durchführen/unterstützen?

Wartungsfälle

Zeit	Umfang	Kategorie
kurzfristig	klein	korrektiv
mittelfristig	klein-mittel	korrektiv, präventiv, adaptiv, perfektionierend
langfristig	groß	Migration, Sanierung

Änderungsmanagement



- Abläufe: Analyse, Programmierung, Test, Lieferung
- Personen, Technik (Gatekeeper, Bugzilla, OneNote…)
- Paketierung (Update, Upgrade, Release)

Wissensmanagement in der Wartung

Kategorien

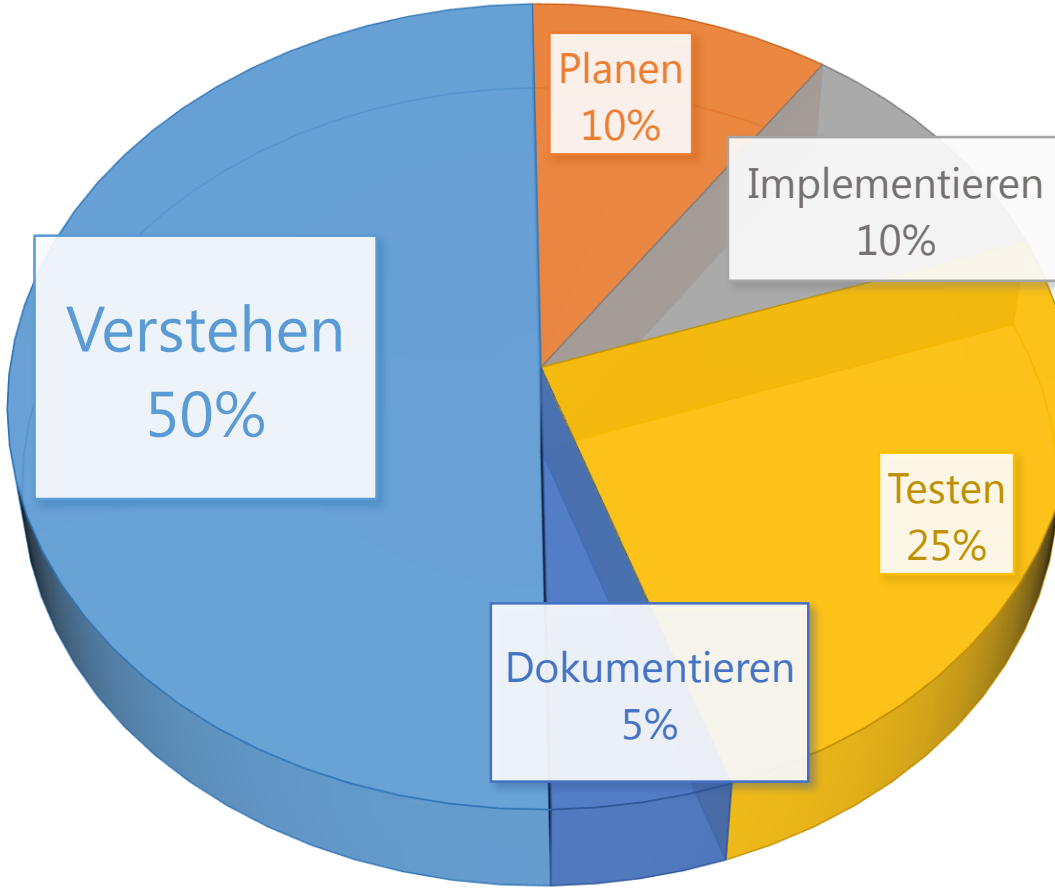
- Domänenwissen (von der SW verwaltetes Problemfeld)
- Prozesswissen (Vorgehen, Wartungsprozesse)
- System- und Technologiewissen (Architektur, Programmiersprache, DB)

Einarbeitung: Wartung > Entwicklung

Wissensinventur, Wissensübergabe

Retrospektive über Wartungsjahr

Software-Analyse

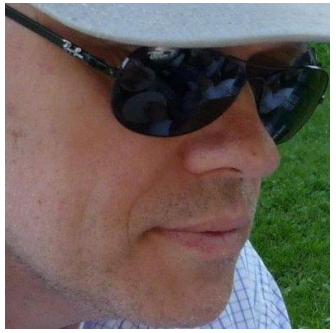


- §1: Verstehen, bevor man ändert
- Bis 50% des Wartungsaufwands
- Betrifft alle Wissenskategorien
- Artefakte, Material
  - Doku, Wiki, Entwurfsmodelle, Skripte (Installation, Konfiguration, Test)
  - Access: Datenbanken, Anwendungsobjekte/GUI, Quellcode
- Architekturanalyse, Codeanalyse
  - Access: Wenige Tools (Prozeduraufrufe in MZ-Tools, VBA-Inspector…)

Typisches Team

Entwicklung

Wartung



Support

Entwicklungsteam = Wartungsteam

Vorteile

- Entwickler kennen das System am besten → Effizienz
- Entwickler berücksichtigen Wartbarkeit in der Erstentwicklung
- Wartungskosten geringer (kein unnötiger Überbau…)

Nachteile

- Projektcharakter gg. langfristig/ad hoc → Spannungen
- Termine der Entwicklungsprojekte in Gefahr

  (Ressourcen geteilt, Probleme im Feld prioritär…)
- Systematik der Wartung leidet (Planung, Prozesse, Metriken, Kostenkontrolle…)
- Änderungen durch Wartung werden weniger dokumentiert (alles im Kopf)

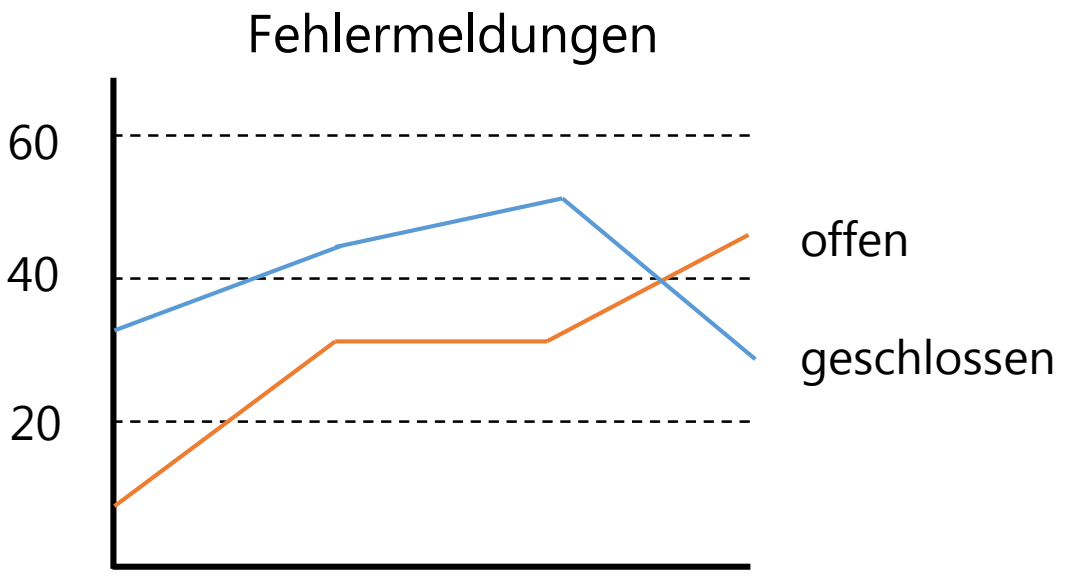
Metriken

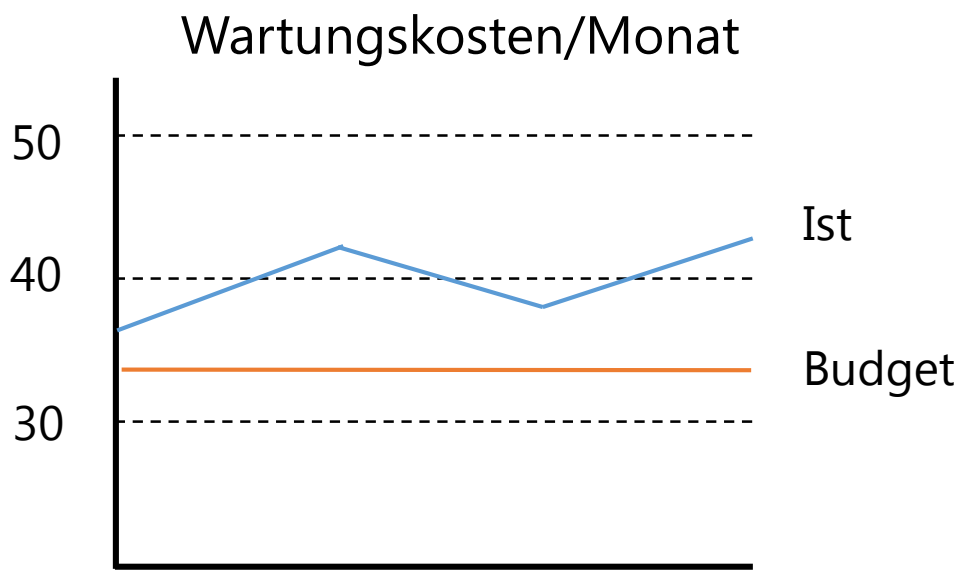
Prozessmetriken

- Anzahl Fehlermeldungen (Logging)
- Fehlerhäufigkeit: $\frac{\text{Anzahl Fehlermeldungen}}{\text{Betriebstage}}$
- Dauer/Kosten Fehlerbehebung
- Häufigkeit/Dauer/Kosten von Anpassungen
- Dauer/Kosten einer neuen Funktion oder eines neuen Release
- . . .

Prozessmetriken

- Vergleichswerte wichtiger als absolute Zahlen
- Erhebung z.B. mittels Kanban oder Ticketsystem
- Probleme: kaum QM, fehlendes Wissen, regelmäßige Erfassung, Disziplin





Metriken

Produktmetriken

- Komplexität
  - Grad, zu dem ein System schwer verständlich und schwer prüfbar ist
- Klassiker: Anzahl Codezeilen (LOC)
  - Ohne Leerzeilen und Kommentare
  - Zusammenhang: Einarbeitung und Testaufwand
- Access-spezifisch: Anzahl Datenbank-Objekte
  - Zusammenhang: Einarbeitung, Überarbeitung, Tote Objekte
- Problem: Viele Metriken in Access/VBA nicht verfügbar
  - Fehlende Werkzeuge

Refactoring

Bessere Struktur von Entwurf und Code

- Verbesserung von Wartbarkeit, Verständlichkeit, Erweiterbarkeit
- Externes Verhalten ändert sich nicht
- Laufend durchzuführen

Gründe

- Analysierte Schwächen, Bad Smells

Agiles Vorgehen

- Leichtgewichtige, kontinuierliche Veränderung

Sanierung

Kostengünstige Alternative zur Neuentwicklung

- >=50% von Entwurf und Code betroffen

Bessere Struktur von Entwurf und Code

- Verbesserung von Wartbarkeit, Verständlichkeit, Erweiterbarkeit
- Wirschaftlichkeit/Leistungsfähigkeit (wieder-)herstellen
- Lebensverlängernd bzw. Überlebensfähigkeit sichern

Projektcharakter

- Analyse, Ziele, Aufwand festlegen
- Änderungen spezifizieren und durchführen

Wartung – Testen

…und die Access-Wirklichkeit

- Automatisiert: 0,x % AccUnit, Rubberduck…, ~~Regressionstests~~
- Funktionstests aller Art, max. kleine manuelle Regressionstests

Testaufwand reduzieren

- Auswirkungsanalyse
  - Änderungsantrag verstehen, reproduzieren
  - Auswirkungen abschätzen, prüfen, testen
- Statische Analyse (Konventionen einhalten etc.)
- Reviews

Reviews

Strukturierte Gruppenprüfung

- Lesen und Nachvollziehen des Quellcodes
- Fehler, Defekte, Inkonsistenzen finden
- Varianten: Stellungnahme, Fagan, Walkthrough

Walkthrough

- Dokument verteilen = Vorbereitung
- Autor präsentiert – Teilnehmer fragen, kommentieren → Protokoll
- nach dem Review: Überarbeitung → evtl. kurze Review-Wiederholung

Wartungsaufwand – Finanzierung

Korrektive und präventive Wartung

- Gewährleistungspflicht → Kosten müssen im Verkaufspreis enthalten sein
- danach evtl. Wartungsvertrag

Adaptive Wartung

- Wartungsvertrag oder direkt dem Kunden verrechnet

Perfektionierende Wartung

- senkt künftige Wartungskosten
- evtl. Zuschlag auf die Wartungskosten

Support

- Wartungsvertrag bzw. Service Level Agreement (SLA)

Wartungsaufwand – Kalkulation

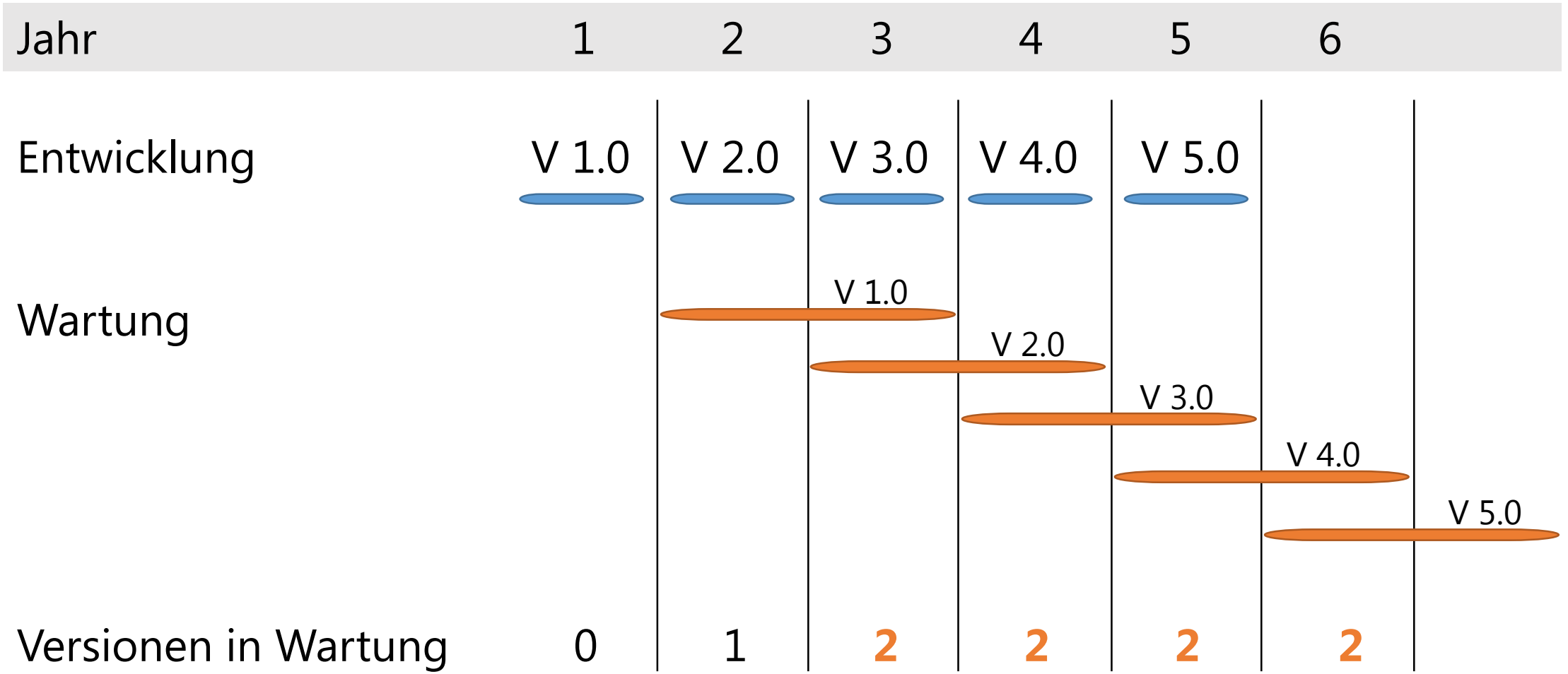
Schätzung 1. Jahr

- Faustregel: pro Jahr 15% der Entwicklungskosten
- div. Schätzverfahren nach Code-Änderungrate, Fehlerdichte…

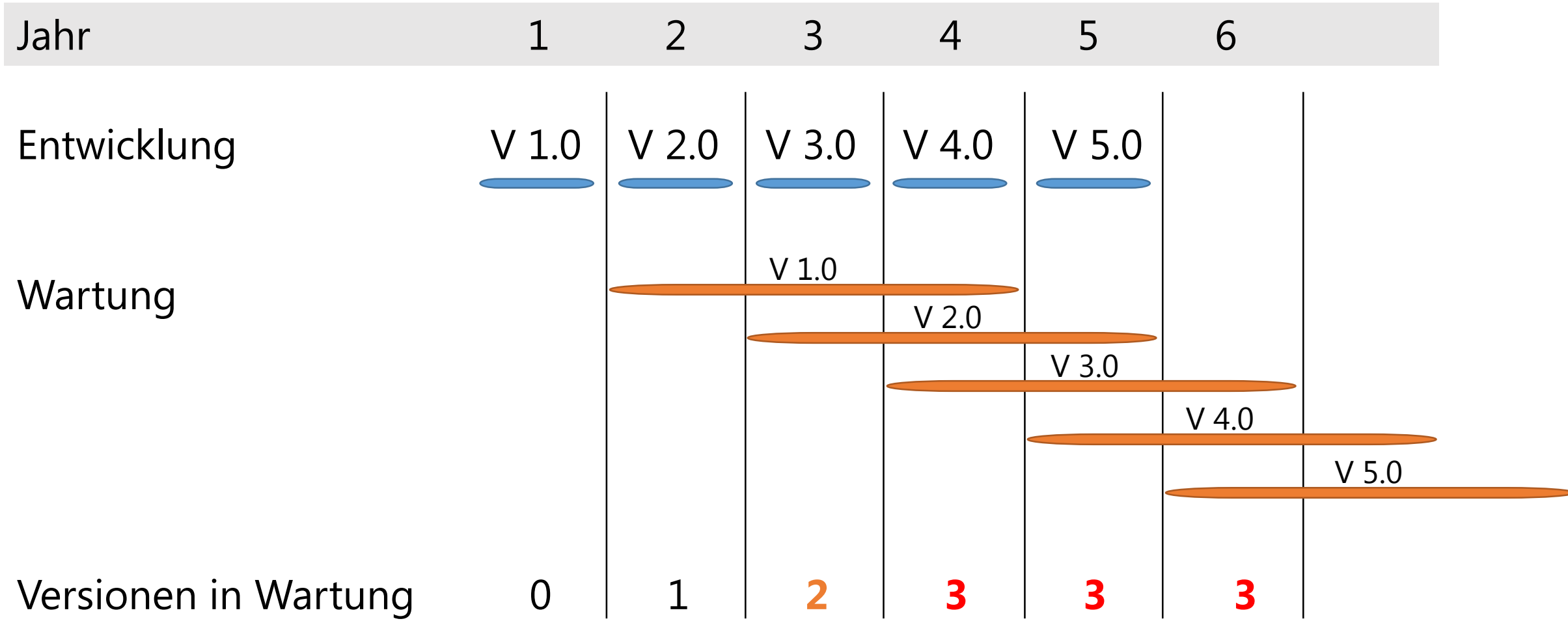
Folgejahre

- Vorjahr mit aktuellen Anpassungen

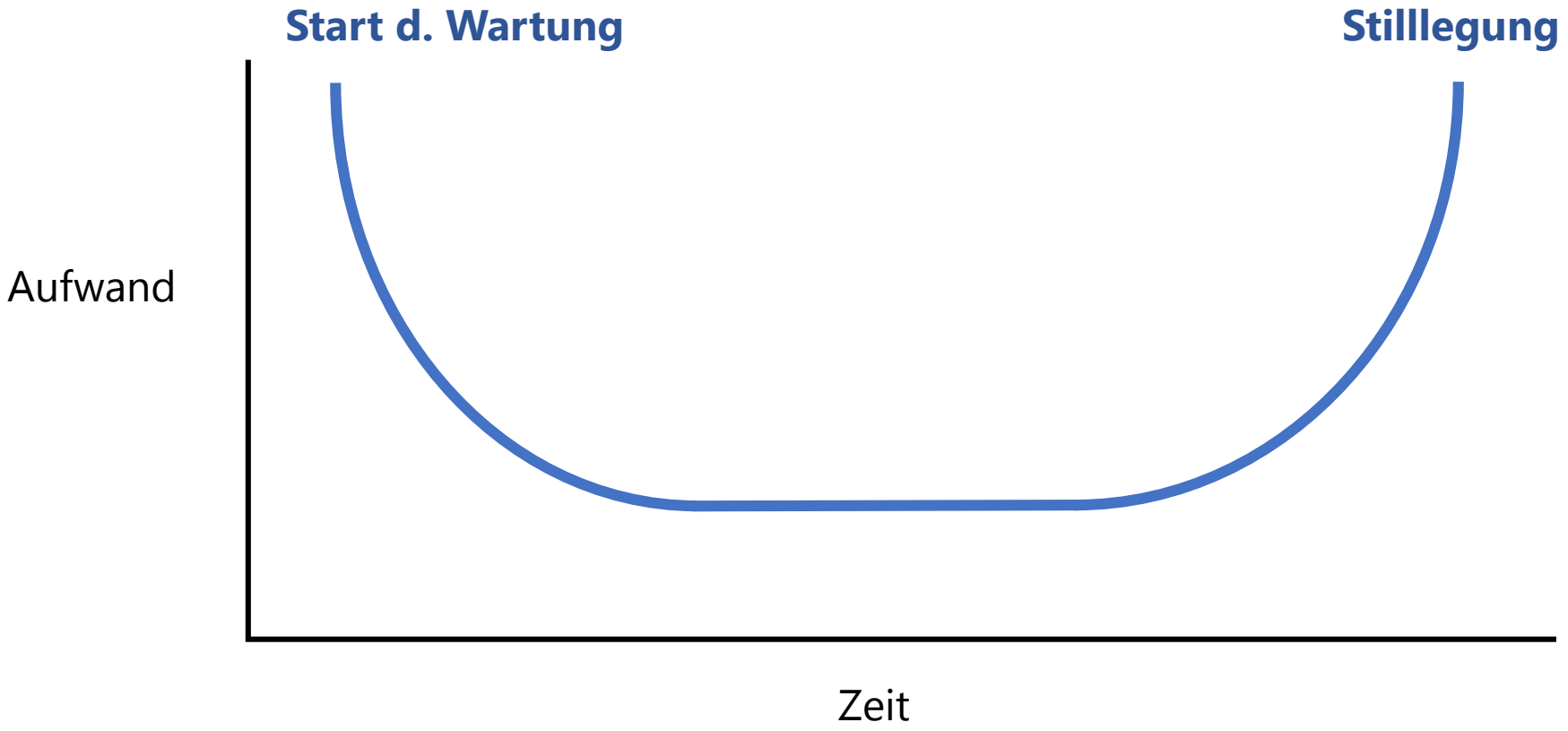
Wartungsaufwand → Wartungsroadmap



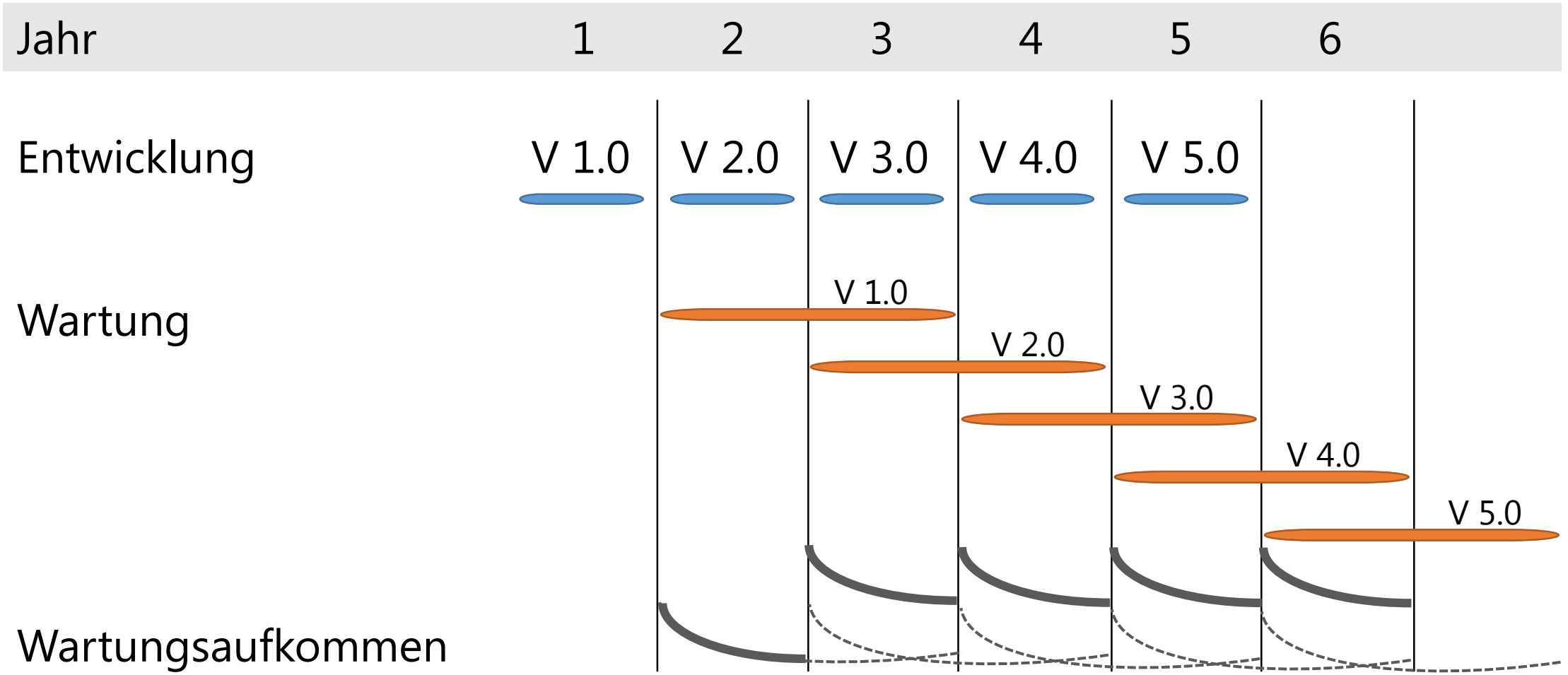
Wartungsaufwand → Wartungsroadmap



Wartungsaufwand – Lebenszyklus



Wartungsaufwand – Prognose



Wartungsaufwand steigt

Mehrere Versionen (Releases)

- Dezidierte Abkündigung!

Mehrere Plattformen

- Mehrere Access-Versionen (mdb/accdb), Komponenten

Sonderlösungen

- Für spezielle Kunden(gruppen)

Wartungsaufwand sinkt

Schrittweises Vorgehen

- Nicht zu viele Änderungen auf einmal

Wartbarkeit aus der Entwicklung

- Architektur, Design, Doku, Richtlinien, Tests…

Übergang Entwicklung → Wartung

- Entwicklung hält Kosten+Termine ein
- Qualität, Wartbarkeit?
- Bewusste Übernahme/Übernahmetests in die Wartung

Wartungsaufwand – Externe Komponenten

Entwicklung profitiert – Wartung evtl. nicht

- Jede SW hat Fehler/Bugs, auch externe
- Eigener Lebenszyklus
  - Versionswechsel, geänderte Anforderungen, Supportstopps...
- Nur Teile genutzt → nur Teile wären selbst zu programmieren
- Abhängigkeiten, je langlebiger desto schwerwiegender
  - Quellcode verfügbar?
- Prinzip aus Sicht der Wartung: Nur soviel Externes wie nötig

Wartungskosten – Kunde/Auftraggeber

Verständnis für die Wartung schaffen

- Kunde: Wartung = Fehlerbehebung
- Wartungstätigkeiten erläutern
- Wartungsleistungen im Wartungsvertrag auflisten

Abgrenzung Fehler – Erweiterung

- Gewährleistung oder Kunde zahlt
- Bug oder Feature? → Saubere Anforderungsspezifikation, gute Kundenbeziehung
- Adaptive Wartung gg. Erweiterung → Klarer Wartungsvertrag
- Wünsche per Zuruf → Striktes Änderungsmanagement

Wartung – Was nehmen wir mit nach Hause?

Relevanz

Ursachen

Methodik

Aufwand

Bewusstsein für Wartung

Wartbarkeit in der Entwicklung fördern!