Kanban

Kanban: visuelle Karte / Anschlagbrett, Transparez schaffen, von Toyota, visualisiert die maximal erlaubte WIP (Work in Progress) –> jeden Punkt auf eine Karte schreiben & aufhängen, Spaltentitel -> Position Packet im ARbeitsablauf, Limit Work inn Progress (Begrenzung Arbeitspakete pro Arbeitsschritt), Messung Durchlaufzeit (Lead Time) -> mittlere Durchlaufzeit, Optimierung Prozess um Durchlaufzeit so kurz und vorhersehbar wie möglich zu machen

To do	Dev	Test	Release	Done
5	3	2	3	
Task1	Task2	Task3	Task 5	Task 7
Task 4	Task9	Task10	Task6	Task8
Task20	Task 37		Task 23	Task40

SCRUM und KANBAN

Frameworks / Werkzeuge für Ausgestaltung Prozesse, Mittel um Aufgabe zu bewerkstelligen, Prozesswerkezuge –> effektiver Arbeiten, kein Werkzeug ist komplett, kein Werkzeug ist perfekt, basieren auf ständiger Entwicklung, Arbeit in kleine Stücke runterbrechen, Scrum schreibt Rollen / zeitbegrenzte Iterationen vor, ist standardisierter als Kanban, schreibt Schätzung vor - Kanban gibt keine Rollen und fixen Iterationen vor, Ziel: Minimierung Durchlaufzeiten, Arbeitsverlauf gleichmässig halten :arrow_right: Anreiz Zerlegung Arbeitspakete in kleine Teile, keine zwingende Schätzung,keine Daily Standups vorgeschrieben - ,empirische Kontrollinstumente (Scrum, Kanban), mit Vorgehen exmperimentieren und ihrer Umgebung anpassen, Kontrollknöpfe: Kapazität (hoch), Durchlaufzeit (kurz), Qualität (hoch), Vorhersehbarkeit (hoch) - funktionsübergreifende Teams mit wem? Exp. - Wieviel Arbeit pro Sprint? Exp. - WIP-Grenze? Exp.

Unterschied Scrumboard & Kanbanboard: Beide: Verfolgung Bündel einzelner Arbeitspakete :arrow_right: flow, Bsp. 3 zustände: to do, ongoing, done - Scrum: WiP begrenzt pro Zeiteinheit, Board pro Sprint und Team - Kanban: WIP begrenzt pro Zustand entlang des Arbeitsablaufes, Board bleibt bestehen, Board auch über mehrere Teams hinweg,

Rückkoppelungsschleifen: Kaizen (kontinuierlicher Verbesserungsprozess in Leansprache), Inspierieren & Adaptieren (Inspect & Adapt, Scrumsprache), Scrum :arrow_right: zugrundeliegende Rückkoppelungsschleife :arrow_right: Sprint, insbesondere wenn kombiniert mit XP

Neue Anforderung: Scrum: Aufnahme ins Backlog - Kanban: ins ToDo :arrow_right: Grenze erreicht _arrow_right: entweder ein anderes Element rausnehmen, sobald freie Kapazität: oberstes Element aus dem Stapel

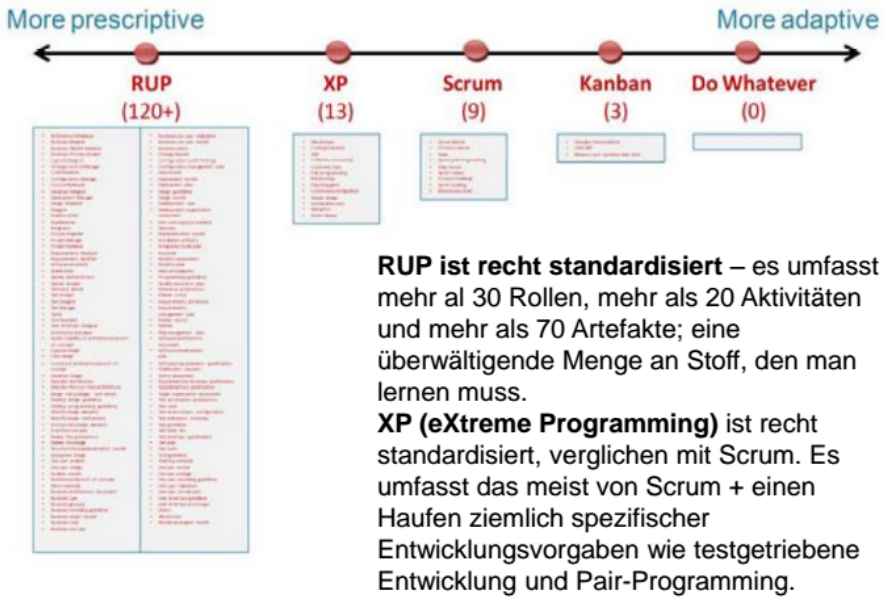


Figure 1: Standardisierung

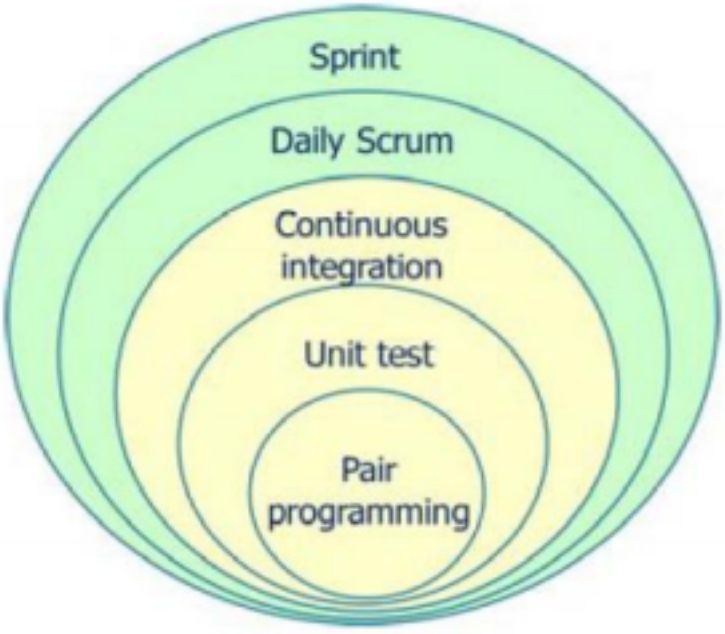


Figure 2: Rückkoppelungsschleife

Arbeit an mehreren Produkten: Scrum: je ein Board, was wenn nur ein Team? Pro Sprint ein Produkt, oder pro Sprint Komponeten von beiden Produkten (gemeinsames Board) - Kanban: analog

Überblick: - Lean & Agil - Scrum: Push Prinzip, Kanban: Pull Prinzip - Nutzung Transparenz für Prozessverbesserungen - Konzentration auf schnell und oft ausführbare SW bereitzustellen - Selbstorganisierende Teams - Aufteilung arbeit - Releaseplan auf Bais empirischer Daten kontinuierlich verbessern

Version Control, Git and Github

Grösstes Problem: DB wird kopiert, nicht gescriptet (z.B. mit Liquibase)

Version control System

- Repository
- Working Copy (local, private)
- Checkout
- Commit
- Update (Git: merge / rebase)

Typen

- Local Version Control Local, separate repository dirs to keep track of changes, single point of failure, only individual files
- Centralized VC one authoritative server, one complete repo, multiple working copies, one official truth, fine graiend access control, single point of failure, slow
- Distributed VC local copy on each computer, no single authoritative repo, so spf, offline, fast, good changing & merging, many possibilities for workflows, collaboration, no real latest version, no reall revision numbers

Tracking Changeset vs. Snapshot

Changeset: stores files in form of differences (deltas) bethween either previous / next version - **Snapshot:** stores each changed file entirely, uses directed acyclic graph to store file-trees

Directed acyclic graphs

consists of nodes and edges, edges are directed, contain no circles, often called trees, efficient algorithms

Revision ID & Tagging

Continuous Numbers or Hash-Values, for usability: User can define Tags for commit

Git

Design Goals: Speed, simple design, strong support for non-linear dev (branching, merging), fully distributed (allows complex workflows, stable sync), ability to handle large projects

Concepts: file system (snapshot storage method), Integrity (checksum for everything, SHA-1, first few chars usually sufficient), no delete (recoverably), tracks contents not files (file with same content :arrow_right: only once uploaded, reference)

Buckets

- Working Directory checked out version of files the user is currently working on
- Staging Area (index) Collect changes for next commit
- Repository
- Stash area storage to temporary keep modified tracked files of a branch while working on different branch, required on branch switch

File Status Lifecycle

tracked: under version control (unmodified, modified, staged)
untracked: new files, ignored files

Commands

around 150

- Local commands
  - git config (–global user.name “Test”)
  - git init (new repo for existing dir)
  - git add
  - git status (status of working dir, current branch, staged files, unstaged changes, untracked files)

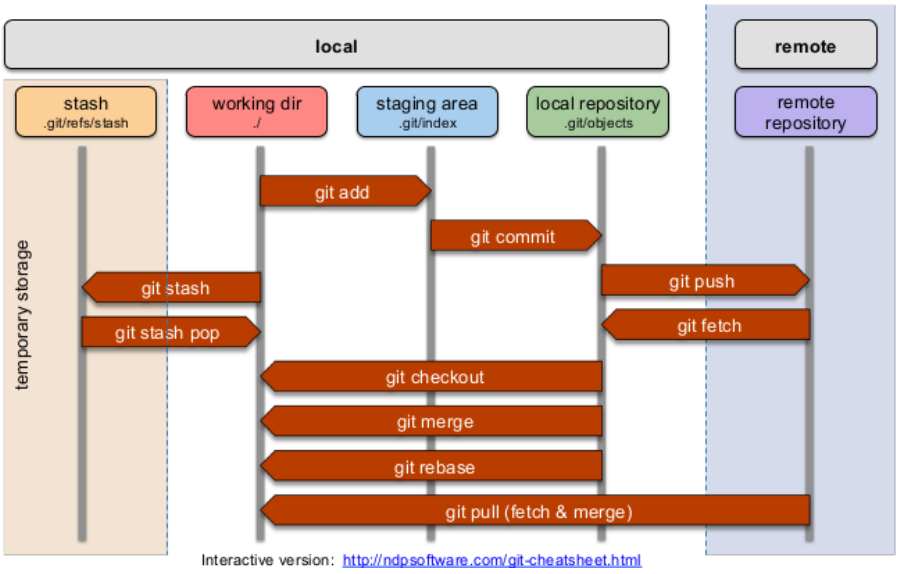


Figure 3:

  - git commit (-am :arrow_right: add and commit) Create commit object, pointer to staged files (snapshot), pointer to previous commit (parent), commit metadata –amend (change commit message)
  - git log (log history, -10, –since=2.weeks, Display: –graph, –decorate, –oneline, –all)
  - git diff (unstaged changes), –cached: staged changes, specific version)
  - git tag
  - git revert (revert (last) commit specific or HEAD)
  - git reset (remove file from staging area, unstage, undo last commit: –soft HEAD~1)
  - git rm –cached (untrack file, but keep in working dir)
  - git rm (untrack and delete)
  - git mv (move reference, rename)

- branch / merge
  - git branch (display branches, last commit: -v, remote tracking branches: -r, local & remote: -av)
  - git merge git checkout master && git merge dev, creates new commit
  - git checkout (checkout branch, -b: replace files in wd with files of branch) – (get file from repo, –force master: get all)
  - git rebase (order commit of the source branch after target branch) git checkout dev && git rebase master, cleaner history, powerful and dangerous
  - git stash (save [message], git stash pop [stashId], list)

- remote commands
  - git remote (show: -v, Details: show , add, rename, remove)
  - git clone (create copy of repo) Mehrere Befehle (init, remote add origin, fetch origin, branch master origin/HEAD, checkout master)
  - git fetch (info from remote not in local repo, git fetch)
  - git pull (=Fetch + Merge, alternative: rebase instead of merge)
  - git push

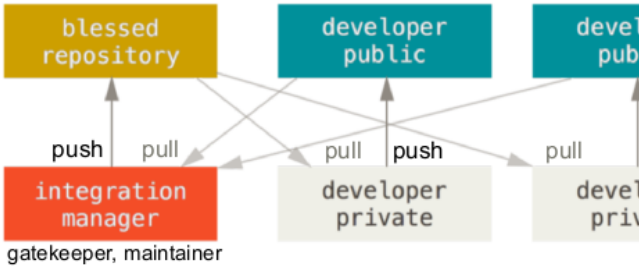
Config: local, global or system
Repo: config, objects, refs, logs, hooks, index

Branching Models

- Progressive stability branching one linear shared flow of commits, multiple branch to express stability
- Feature / topic branching one long running branch (master), stable, new features in separate branches, merge to master
- Git Flow

Collaboration model

- Shared repository (similar centralized VCS, small projects & teams)
- Integration manager each contributor: own private/public repo
  1. project maintainer: push to public main
  2. contributors: clone repo & make changes
  3. contributors: push to their public repo
  4. c.: inform maintainer of a finished feature
  5. m: add c. public repos and merge changes



  6. m: push the merged changes to public main
- Dictator & Lieutenant Extension integration
  1. Dev: work on topic branch, rebase changes on top of blessed master of dictator
  2. Lieutenants merge D. topic branchs into their master branch

3. Dictator merges the l. master branches into the dictators master
4. Dictator pushs his master to the main reference repo, dev: rebase again

Pull-Requests (github Only)

- Maintainer: push to public
- Cont: fork repo and make changes
- Cont: push to public
- Cont: Send pull request for specific feature
- M: merge changes & push to repo

Build Automation & Continuous Integration

Software Automation

- On-Demand (run script or press a button)
- Scheduled (timed)
- Triggered (event)

Types of Automation: - Build Automation (source code, packaging, doc) - Test Automation - Deployment Automation - Operation Automation (provisioning, monitoring, health, scaling)

Goals: - Improve Product Quality - Faster time to market - Minimize risks

Reduce risk by releasing often: minimal changes, less review and test, small improvements, faster bug fixes

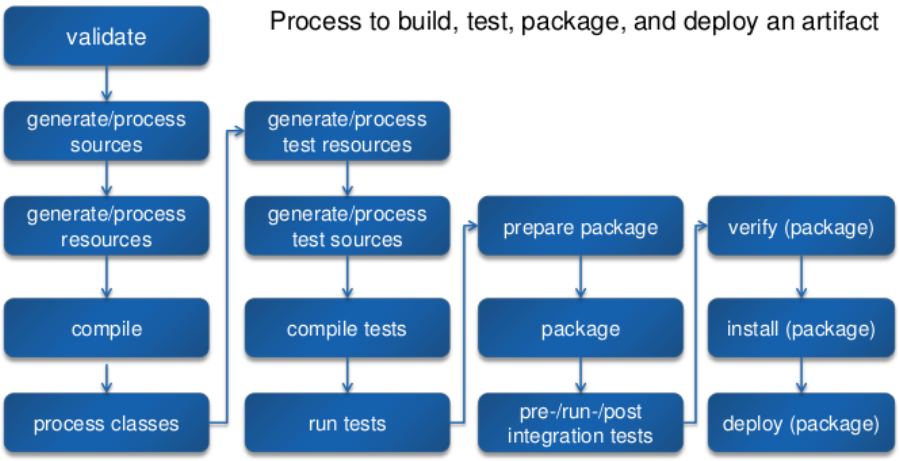
Software Automation Pipeline: Development (Code, Build, Unit Tests), Integration (Integrate, Integr. Tests), QA (Release, Acceptance Tests), Operation (Deploy, Operate) - Build automation (Dev), CI (Dev + Int), Continuous Delivery (Dev, Int, QA), Continuous Deployment (bis und mit Deploy), DevOps (all),

Build Automation

- Resolve dependencies
- Compile source code
- Create documentation
- Run unit tests
- Package SW
- Deploy to runtime systems
- Clean up temp files

Requirements: automated, repeatble, consistent, icnremental, platform independent, seamless integration

Maven




```
${basedir}                      Root folder of the artifact
├── pom.xml                     Project Object Model description
├── src                         Folder containing all sources
│   ├── it                      Integration tests
│   ├── main
│   │   ├── config              Config files
│   │   ├── db                  Database files (e.g. SQL-Scripts)
│   │   ├── java                Java Source files to be compiled
│   │   ├── resources           Non-compiled source files
│   │   ├── scripts             Script files for devs and admins
│   │   └── webapp              Web assets like JSP files, CSS, images
│   ├── site                    Files used for the project site
│   └── test
│       ├── java                unit test source code
│       └── resources           Non-compiled test source files
└── target                      holds the generated artifacts
    ├── classes                 Compiled class files
    ├── example-1.0-SNAPSHOT.jar    packaged project jar
    └── test-classes            Compiled unit tests
```

Archetype: vordefinierte Projektstruktur / Beispiel / Template

Project ID: GAV (GroupId, ArtifactId, Version: {Major}.{Minor}.{Maintenance} + -SNAPSHOT (dev version))

Continuous Integration

Integration: Making different modules working together, modularization, they must be integrated, they do compile, pass tests, run, deploy

Challanges: Modules are built independently, should be reusable, passing modules tests :arrow_right: not automatically pass system test, incompatible dependencies, code quality fails, performance insufficient, expensive integration

CI: reliable builds, fully automated, reproducible build, including tests, runs many times a day, revular intervals

Practices

maintain single source repo, automate build, build self testing, every commit should build on integration machine, keep build fast, test in clone of prod, easy get latest version, everyone can see what's happening, automate deployment

Infrastructure

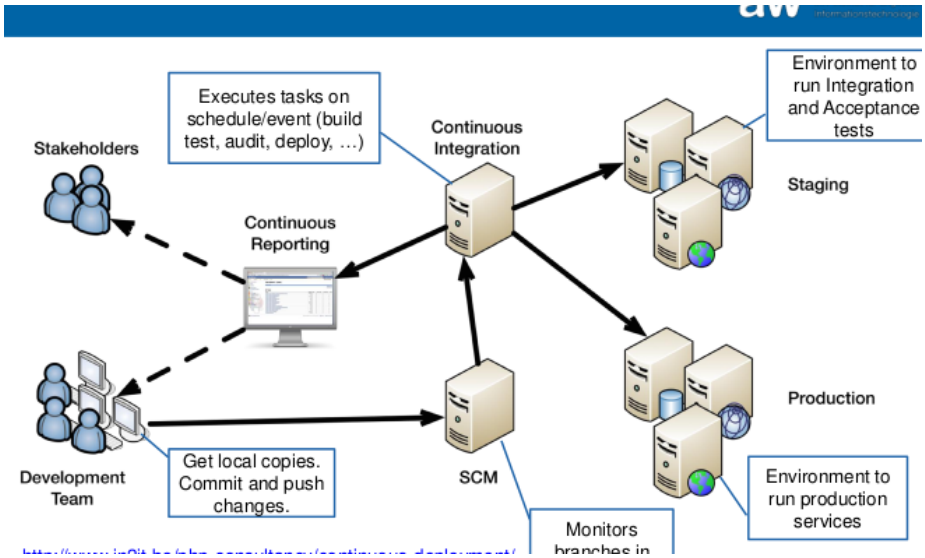


Figure 4:

Workflow

1. Dev: check out into private repo

2. Dev: commit changes to repo
3. CI: monitor for changes, checkout
4. CI: build
5. CI: run unit & integration tests
6. CI: run code audit tools
7. CI: release deployable artifacts for testing on staging
8. CI: assign build label to the version of code
9. CI: inform team about successfull build
10. CI: Publish application (optional)
11. CI: on fail: alert team
12. continue...

Successfull build

compiles? tests successfull? released? deployed? or event failed (we were notified about problem)

Team responsibilities

frequent check in, don't: check in broken / untested code, when the build is broken, go home after chcking in until the system builds

Required tools

VCS, Build Server, Deployment Server, Automation tools, CI-Server

Basismodelle

Vorgehensmodelle

beschreibt in abstrakter und idealisierter Form die zeitlich-sachlogische Abfolge von Aktivitäten, Auffassung 1: Jede SWE kann / muss nach def. Prozess ablaufen, Auffassung 2: Jede SWE ist empirischer / dyn. Prozess

Kurz: org. Rhmen für SWE, soollen zu disziplinierten, sichtbaren, kontrollierbaren Entwicklung führen

Basismodelle: - Sequenziells Modell - Waserfall-Modell - V-Modell - Prototypen-Modell - Evooluationäres / Inkrementelles Modell - Objektorientiertes Modell - Nebenläufiges Modell - Spiralmodell

Monumentalmodelle: UP, RUP, V-Modell, Hermes

Rahmenmodelle

Prozessmodelle

Mehr oder weniger starker Bezug zu Softwaretechnik

- ITIL
- EFQM (European Foundation for Quality management)
- Six Sigma (Q-Modell um GP möglichst fehlerfrei zu gestalten)

Legt aus VM konkret fest: - Reihenfolge Arbeitsablauf (Entwicklungsstufen, Phasenkonzepte) - Durchzuführende Aktivitäten - Definition Teilprodukte (inkl. Layout & Inhalt) - Fertigstellungskriterien - Notwendige MQ-Qualifikationen - Verantwortlichkeiten und Kompetenzen - Anzuwendende Standards, Richtlinien, Methoden, Werkzeuge

Rahmenmodelle

Rahmen für Einteilung P in Kategorien, Beschreibung Tätigkeitsmerkmale

- CMMI (Capability Maturity Model Integration)
- Spice Modell / ISO 15504 (Software Prozess Improvment and Capability Determination Modell), Rahmen Bewertung & Verbesserung SWP
- ISO 9000 (Rahmenmodell QMS)
- TQM Modell (Führungsmethode), Qualität im Mittelpunkt

Vorgehensmodell vs. Prozessmodell

Oft synonym verwendet, verwechselt, VM: sagt, was wir zu tun haben, nicht wie - PM: Konkrete Beschreibung, wie die durch ein VM auszuführenden Tätigkeiten durchzuführen sind

Arbeitsschritte und Aktivitäten

Arbeitsschritt: bestimmte Sicht auf Gegenstand Proj. beinhaltet konkrete Aktivitäten Schritte: Analyse, Entwurf, Impl., Test, Inbetriebnahme, Wartung | **Projektphase:** Zeitdauer in welcher AS ausgeführt werden, besteht aus 1 AS

aktivitäten: Arbeitseinheiten, Ziel: Konkretes Produkt oder Erbringung einzelne Leistung, wird durch eine Rolle alleine ausgeführt, Gegenstand von konkreten Arbeitsaufträgen, kann in Arbeitspaketen beschrieben werden.

Code & Fix Prozess-Modelle

1. Code Schreiben, und testen, 2. Code verbssern, 3. GOTO 1 - Ja, wenn Problem klarspezifiziert, und Impl 1 Person - **Nachteile**: Für Behebung Fehler Umstrukturierung, weitere Fehlermeldungen immer teurer :arrow_right: Entwurfsphase vor Programmierung - gut enworfen, SW von Endbenutzer nicht akzeptiert :arrow_right: Definitionsphase vor Entwurf - Fehler schwierig zu finden, Tests schlecht vorbereitet :arrow_right: separate Testphase

Klassisches sequenzielles Phasenmodell

zahlreiche Variantionen | **Phasen:** Problemanalyse und Grobplanung, Systemspezifikation und Planung, System- und Komponentenentwurf, Implementierung und Komponententest, System- und Integrationstest, Betrieb und Wartung - In der Praxis meist so nicht anwendbar, idealmodell

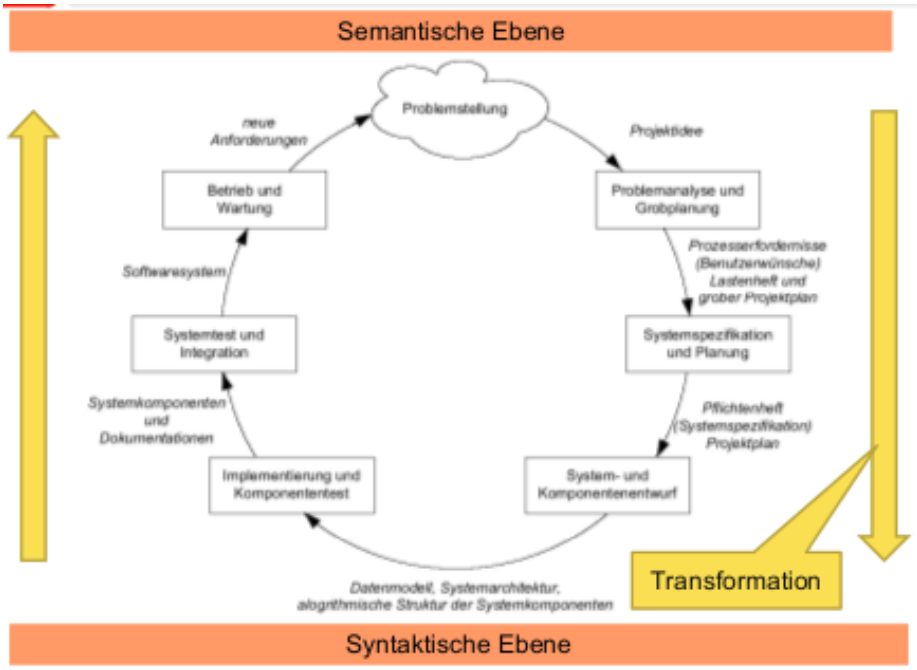


Figure 5: Sequenzielles Phasenmodell

Vorgehensweise Transformation

- Topdown-Dekomposition in Black Boxes

- Ausgangsbasis einzelne Phasen: wohldefinierte Produkte (meist Dokumente)
- Resultat: Input nächste Phase
- Phasen sind klar voneinander abgegrenzt
- Verifikationsschritt: Prüfung ob Phasenergebniss Systemspez. erfüllt.
- Evaluierungsschritt: Prüfung ob Produkt in vorliegender Form den Benutzerwünschen entspricht
- Entwurfsphase: Weitere Zerlegung
- Komponentenspez.
- Synthese Sys.komp.

Wasserfallmodell

Verfeinerung klas.. Phasenmodell, Rückkoppelung zwischen Phasen + Einschränkungen, dass diese nur bei aufeinanderfolgenden Phasen vorkommen, Einbindung Validierung der Phasenergebnisse, optional: iterativer Charakter :arrow_right: eingeschränkt inkrementelle Entwicklungsstrategie (mit Prototypen), zweimal-Bauen (Vorwiegend Sys.spez. und Sys.Arch. - zuuerst Prototyp, dann sauber) :arrow_right: Reduktion Fehler Sys.Spez. und Design-Fehler.

Phasen: Problemanalyse, Sys.spez, Grobentwurf, Feinentwurf, Impl, Integration, Installation, Betrieb & Wartung

Charakteristika nach Balzert: Aktivität in richtigen Reihenfolge, volle Breite durchführen, nach jeder Aktivität: fertiges Dokument (Dokument-getriebenes Modell), Entwicklungsablauf: sequentiell (jede Aktivität abgeschlossen vor der nächsten), Orientierung top-down, einfach, verständlich, wenig Management, Benutzerbeteiligung nur in Definitionsphase

V-Modell 97

Erweiterung Wasserfall, integriert QS, Verifikation: Überprüfung Übereinstimmung SW-produkt und Spez, Validation: Eignung / wert Produkt auf einsatzzweck

4 Submodelle - Systemerstellung (SE) - Qualitätssicherung (QS) - Konfigurationsmanagement (KM) - Projektmanagement (PM)

Ai: Aktivitäten, Pi: Proudkte

V-Modell XT (Xtreme Tailoring), anpassbar an Bedürfnisse, Einbindung Auftraggeber, Stärkere Modularisierung, Tedenz: agil, vier Submodelle gibt es in dieser Form nicht mehr.

Tailoring: Ausschreibungsrelevantes Tailoring vor Entwicklungsbeginn: sinnvolle Aktivitäten & Produkte festlegen, Streichen, Wahl vorhabentyp - Technisches Tailoring: Aktivitäten & Produktinhalte bei Beginn jeder Haupt-A

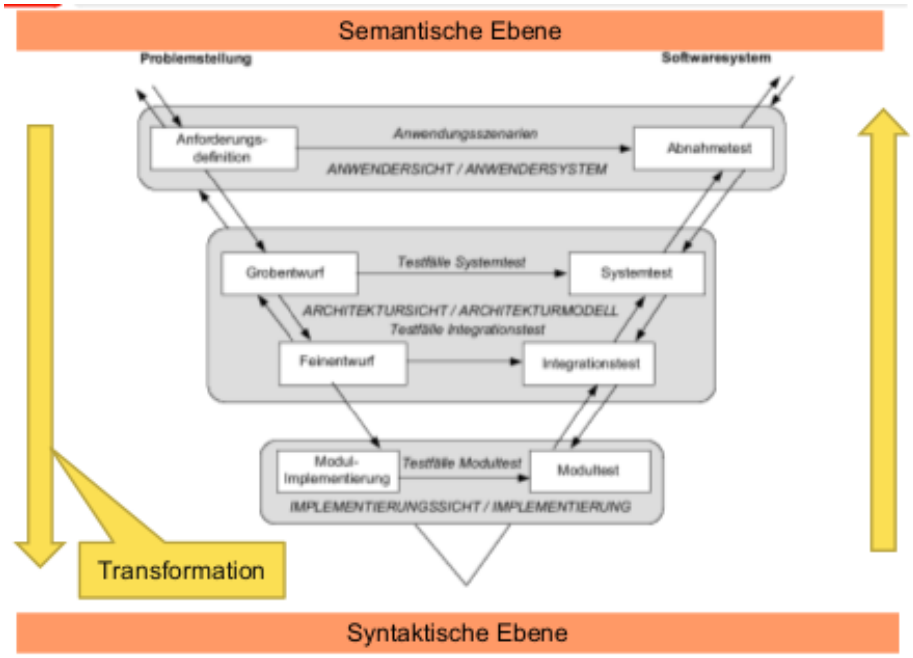


Figure 6: V-Modell

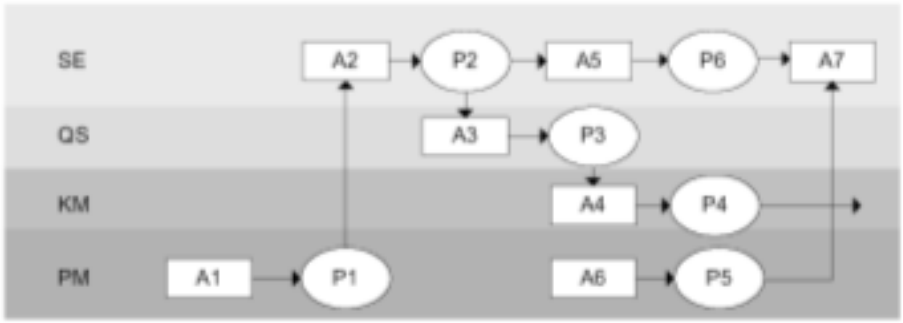


Figure 7: Zusammenwirken

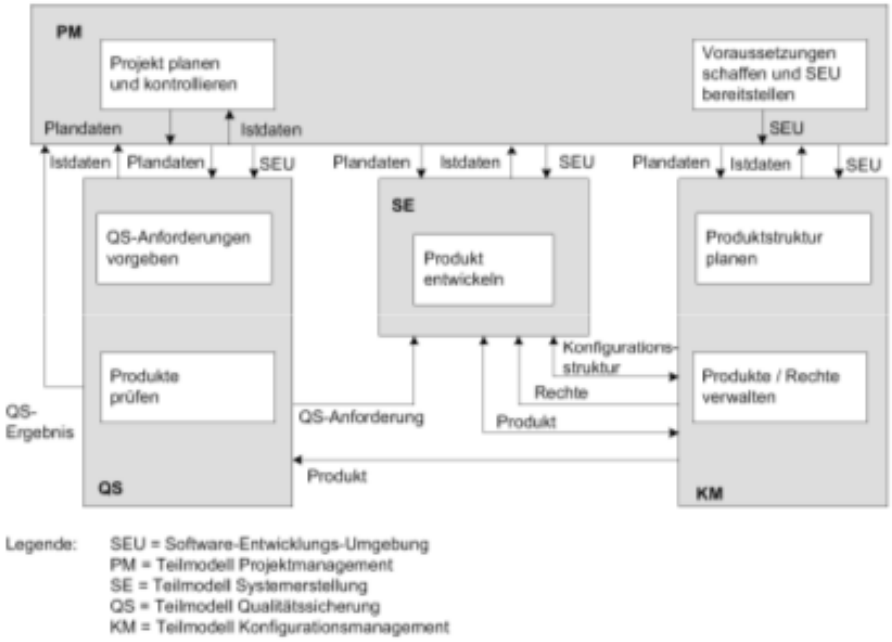


Figure 8: Zusammenwirken

	Manager	Verantwortliche	Durchführende
Submodell PM	Projektmanager	Projektleiter Rechtsverantwortlicher Controller	Projektadministrator
Submodell SE	Projektmanager IT-Beauftrager Anwender	Projektleiter	Systemanalytiker Systemdesigner SW-Entwickler HW-Entwickler Technischer Autor SEU-Betreuer Datenadministrator IT-Sicherheitsbeauftragter Datenschutzbeauftragter Systembetreuer
Submodell QS	Q-Manager	QS-Verantwortlicher	Prüfer
Submodell KM	KM-Manager	KM-Verantwortlicher	KM-Administrator

Figure 9: Rollen

festgelegt., Entscheidung ob A oder P im konkreten Fall sinnvoll ist (Vermeidung Papierflut)

Prototypen-Modell

Traditionelle Probleme: Auftraggeber kann Anforderungen nicht explizit / vollständig formulieren, keine Kooperation nach dem Anforderungen erstelllt wurden.

Unterschiedliche Lösungsmöglichkeiten, experimentell mit Auftraggeber erproben / diskutieren., Realisierbarkeit theoretisch nicht immer garantierbar, Während Akquisitionsphase: Auftraggeber von prinzipieller Durchführbarkeit überzeugen.

Prototyp: Ausführbares Modell (einfach änder- / erweiterbar)
Prototyping: Tätigkeiten welche für Prototyp notwendig sind.

Einteilung Prototyping

- Exploratives Prototyping (Ziel: Vollständige Systemspez.)
- Experimentelles P. (Ziel: Vollständige Spez. Teilsysteme)
- Evolutionäres P. (Ziel: inkrementelle Systementwicklung)

Prototyparten

- (un)vollständige P
- Wegwerf / wiederverwendbare P
- horizontale P (nur spez. Ebenen des Sys, möglichst Vollständig)
- vertikale P (impl ausgewählte Teile vollständig durch alle Ebenen, geeignet: offene Fkt. / Impl.optionen)
- Demonstrations.P
- Prototyp im engeren Sinne
- Labormuster
- Pilotsysteme

Inkrementelles Modell

Entwicklung stufenweise, Steuerung durch Erfahrung Entwickler / Kunde, Wartung: Erstellung neue Version besth. Produkt, Entwicklung durch Code, Zentrum: lauffähige Systemteile, geeignet: Kunde hat Req. noch nicht vollstänig überblickt, noch unformuliert

Evolutionäres Modell

~agiles Modell, rasch einsatzfähige Produkte, Kombination Prototyp-Modelle, Erfahrung Einsatz P in nächste Version, Zerlegung in überschaubare Arbeitsschritte, bei: WE Standardprodukte

Objektorientiertes Modell

Merkmal: Wiederverwendung Klassen von Klassen-Biblio. und von Frameworks durch Komposition, Vererbung, Polymorphie, vielfältige Aspekte Wiederverwendung (eigen- /fremdentwickelt) müssen im Prozessmodell berücksichtigt werden

Konsequenzen: Entwurf nicht losgelöst von Impl durchgeführt werden, Berücksichtigung Lösungsbausteine während Entwurf, Dauer Impl verkürzt, Lib muss ständig gewartet werden, neue Klassen: PRüfung auf allgemeine Verwendbarkeit

Nebenläufiges Modell

Charakteristika: Aktivitäten parallelisieren (org. / technische Massnahmen), Erfahrung betroffener Personengruppen frühzeitig zusammenbringen, Zeitverzögerungen reduzieren (Parallelisierung, Minimierung Ausprobieren, Reduktion Wartezeiten), Ziel: Vollständiges Produkt ausliefern

Spiralmodell

Kombination bisheriger Modelle (einbetten), Metamodell :arrow_right: Wahl der am besten geeignetsten Vorgehensweise

Schritte: 1. Identifikation Ziele Teilprodukt (leistung, Fkt), Alternative Möglichkeiten, Randbedingungen 2. Evaluierung Alternativen (Berücksichtigung Ziele & Randbedingungen), Aufzeigen Evaluierungsrisiken, kosteneffektive Strategie entwickeln 3. In Abhängigkeit verbleibende Risiken: Prozess-Modell für Schritt festlegen, Kombination mehrere Modell zur Risikominimierung 4. Planung nächster Zyklus + Ressourcen (inkl. Aufteilung Komp.), Review (1-3, Planung nächster Zyklus), Commitment für nächsten Zyklus

Rahmenmodelle

Grundlagen

Rahmen für Prozesse, Einteilung in Kategorien, Tätigkeiten mit Merkmalen beschrieben, oft anhand Best Practicies wie Ziele erreicht, Grundlagen betreffend Anforderungen an Prozesse und Beurteilung, Modelle SW: CMMI, spice, Modelle industrie: ISO 9000, TQM - z.B. als Referenzmodelle

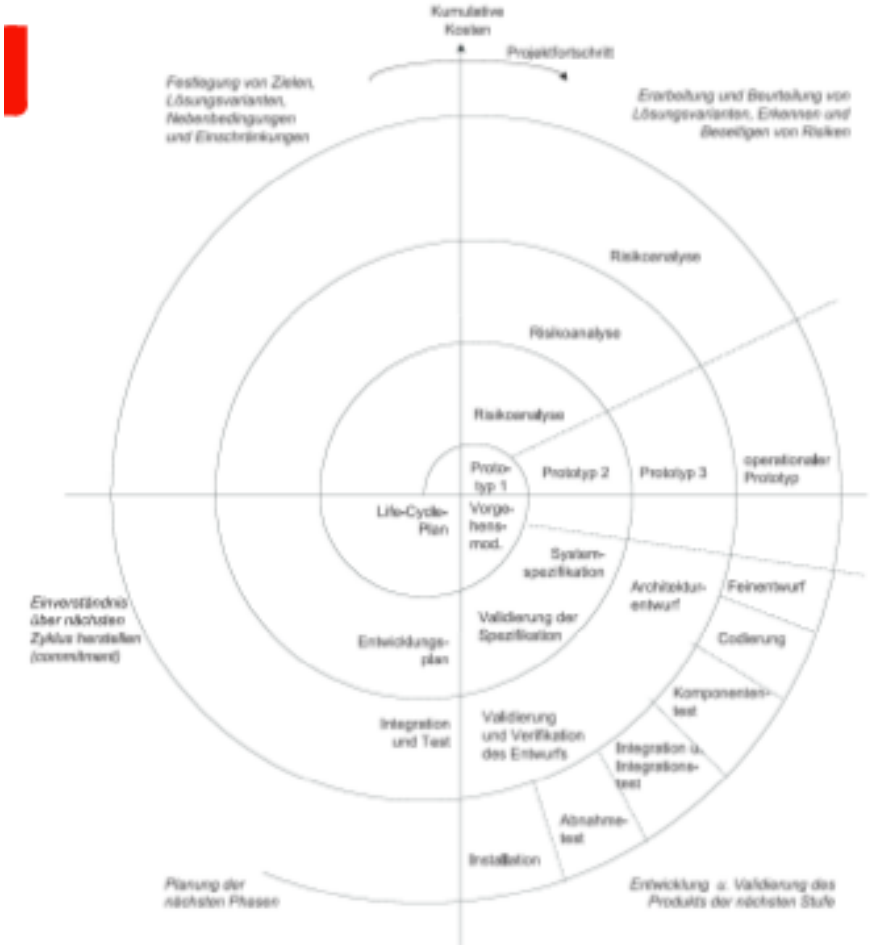


Figure 10: Spiralmodell

Prozessqualität vs. Produktqualität

Prozessmodelle: Produktqualität und Prozessqualität verbessern, Produktqualität eigentliches Ziel, Qualität nicht messbar (via Erfüllung Anforderungen), Erfüllung Anforderungen oft nicht messbar :arrow_right: Entwicklung von Metriken, bei Formulierung Angabe über Messung

Verbesserung Prozessqualität

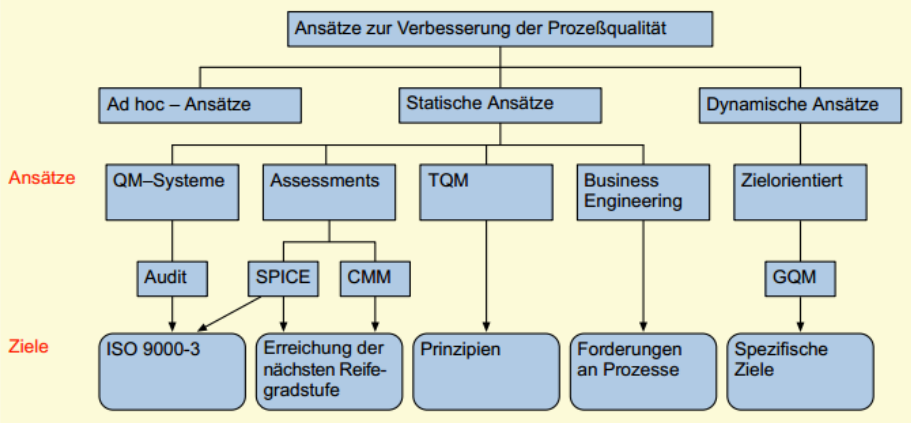


Figure 11:

Software-Prozessverbesserung

PDCA

Modelle aus der SW-Branche

ISO 15504:2004 (Req Assessments und Prozessmodelle): CMMI (Prozesse) + SCAMPI (Assessment) oder ISO 12207 AMD 1 (Prozesse) und SPICE (Assessment)

Assessments

Wo stehen wir? Was können wir verbessern?

Assessmentmethode vs. Prozessmodell

Prozessmodell :arrow_right: Unternehmensspezifische definierte Prozesse :arrow_right: Gelebte / angewendete Prozesse (Überprüfung letzerer)

CMMI

Capability Maturity Model Integrated, Rahmenmodell Prozessmanagement mit betriebsspez. Bezug auf: Prozessdefinition, -schwerpunkte, schulung, -leistung, Innovation und Einführung

Verlaufsdarstellung vs. gestufte Darstellung

Verlaufsdatstellung (Continuous Representation): jeder Prozessbereich einzeln beurteilt, 6 Fähigkeitsstufen (capability levels), Resultat: Fähigkeitsprofil, Prozessverbesserung durch Transition, entspricht Bewertungsmodell von SPICE

Gestufte Darstellung (Staged Representation): Fünf Reifestufen (maturity levels), Fähigkeitsstufen 1-5, Einstufung durch Feststellung welche Prozessbereiche welche Fähigkeitsstufe, Resultat: Zahl 1 - 5, Prozessverbesserung durch schrittweisen Stufenanstieg, entstpricht etwa Bewertungsmodell CMM

Prozessbereiche

22, beurteilt, pro Bereich: specific goals (verbindlich, muss), specific practices (zur Erreichung der Ziele) jeweils Zuordnung von Fähigkeitsstufen (goals and practices), CMMI kann auf spez. Problembereiche zugeschnitten werden (Tailoring)

Fähigkeitsgrade (0-6) - Capability level

Pro Generische Ziele und Generische Praktiken

- 0: Incomplete fachliche Ziele nicht erreicht
- 1: Performed fachliche Ziele erreicht
- 2: Managed Arbeit wird geführt
- 3: Defined Arbeit wird mit Hilfe eines angepassten Standardprozesses durchgeführt und Arbeitsweise verbessert
- 4: Quantitativley Managed Arbeit mit Hilfe statistischen Prozesskontrolle
- 5: Optimizing Arbeit / Arbeitsweise mit hilfe statische Prozesskontrolle verbessert

Kategorien: * Process Management Alle Gebiete mit Mgmt org.weite Prozesse. * Project Management Prozessgebiet mit Mgmt einzelnes Projekt * Engineering technische Entwicklungsthemen, umfassen einfachen Lebenszyklusmodell + Verifikation + Validation Ergebnisse * Support Querschnittsthemen Unterstützung Prozesse

Es entsteht Fähigkeitsprofil

CMMI - Fähigkeitsstufen (capability levels)

charakterisiert durch Generische Ziele die durch Generische Praktiken erreicht werden

Fähigkeitsstufen: * Null
Prozesse Org decken P.-Bereich nur teilweise / nicht ab * 1
Gen. Ziel: Spezifische Ziele erreichen (für jeweiligen Prozessbereich)
Gen. Praktiken: Spezifische Praktiken ausführen (für jeweiligen Prozessbereich) * 2
Gen. Ziel: Geführten Prozess institutionalisieren
Gen. Praktiken:
2.1: Unternehmens-/Organisationspolitik etablieren
2.2: Prozesse planen
2.3: Ressourcen bereitstellen
2.4: Verantwortlichkeiten zuweisen
2.5: Leute schulen
2.6: Konfiguration verwalten
2.7: Relevante Beteiligte (STH) identifizieren und involvieren
2.8: Prozess beobachten und lenken
2.9: Einhaltung des Prozesses objektiv evaluieren
2.10: Status mit höherem Mgmt überprüfen * 3
Gen. Ziel: Definierten Prozess institutionalisieren
Gen. Praktiken: Definierten Prozess etablieren
3.2: Verbesserungsinformation sammeln * 4
Gen. Ziel: Quantitativ geführten Prozess institutionalisieren Gen. Praktiken:
4.1: Quantitative Vorgaben für den Prozess etablieren
4.2: Leistung der Subprozesse stabilisieren * 5
Gen. Ziel: Optimierende Prozesse institutionalisieren
Gen. Praktiken:
5.1: Kontinuierliche Prozessverbesserung sicherstellen
5.2: Probleme an ihren Wurzeln beheben

Reifegrade (maturity levels) 1. Initial (Keine Anforderungen, automatisch jede Org. 2. Managed (Projekte geführt, ähnliches Projekt kann erfolgreich wiederholt werden) 3. Definied (Projekte werden nach angepassten Standardprozessen durchgeführt, org.weite kontinuierliche Prozessverbesserung) 4. Quantitatively managed (Durchführung statistische Prozesskontrolle) 5. Optimizing (Arbeit / Arbeitsweise mit Hilfe eienr statistischen Prozesskontrolle verbessert)

Bestimmung Reifestufe

Reifegrad 2: Mind auf Fähigkeitsstufe 2 * REQM (Requirements Management) * MA (Measurement and Analysis) * PMC (Project Monitoring and Control) * PP (Project Planning) * PPQA (Process and Product Quality Assurance) * SAM (Supplier Agreement Management) * CM (Configuration Management)

Reifegrad 3: Alle FS 3 bis auf: * OPP(Organizational Process Performance) * QPM (Quantitative Project Management) * OID (Organizational Innovation and Deployment) * CAR (Causal Analysis and Resoluation)

Gestufte Darstellung der Reifegrade

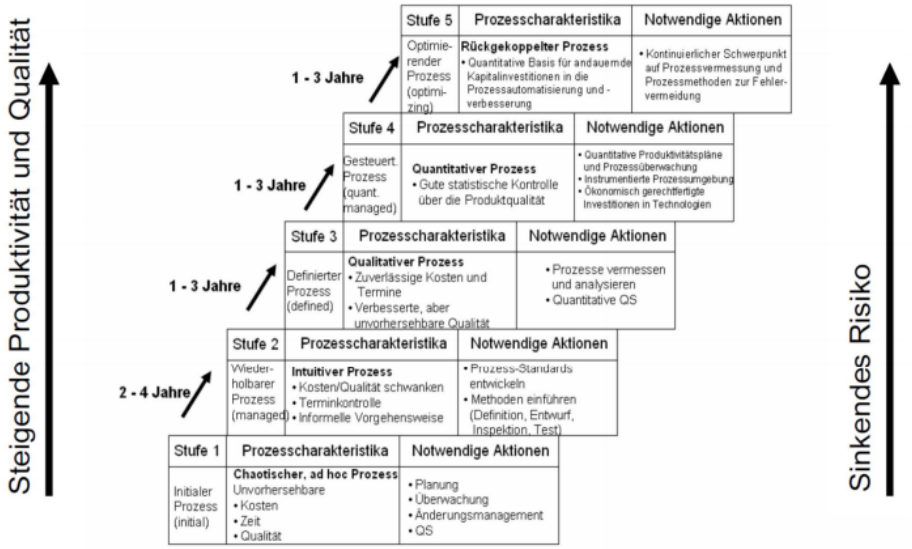


Figure 12:

CMMI Beurteilungsverfahren

Kein vorgeschriebenes Verfahren, sondern vorgaben, die ein Verfahren erfüllen muss (SCAMPI, ARC)

Ziele: Festellen des aktuellen Standes, Identifikation Verbesserungsmöglichkeiten, Überprüfung Wirksamkeit eingeleiteter Massnahmen, Darlegung Prozessqualität der eigenen Org. gegenüber Kunden / Lieferanten

Reifegrad (gestufte Darstellung)	Prozessgruppe (Verlaufsdarstellung)			
	Projekt-management (project management)	Unterstützung (support)	Technik (engineering)	Prozess-management (process management)
5 Optimierend (optimizing)		CAR		OID
4 Quantitativ geführt	QPM			OPP
3 Definiert (defined)	IPM RSKM	DAR	PI RD TS VAL VER	OPD OPF OT
2 Geführt (managed)	PMC PP SAM	CM MA PPQA	REQM	

Figure 13:

ISO 9000

Klassische Fertigungsbetriebe bezieheun Zulieferteile, Zulieferteile beeinflussen wesentlich Qualität Endprodukt, Teile qualitativ hochwertige, Auftraggeber prüft Produktqualität Teilprodukte + Qualität Herstellungsprozess

Normwerk

Auftraggeber-Lieferanten-Verhältnis: allgemeinen, übergeordneten, organisatorischen Rahmen zur QS von materiellen und immateriellen Produkten fest. Teilnormen: 9000-1 (Allgemein, Überblick normen, Leitfaden Auswahl, Anwendung in Bezug auf QM, QS-Nachweisstufe), -2, -3 (Richtlinie wie ISO 9001 für Entwicklung, Lieferung, Wartung SW), -4, 9001 (Modelle Darlegung QS Design/Entwicklung, Produktion, Montage, Kundendienst), 9002 (Def Modelle Darlegung QS in Produktion / Montage), 9003 (Beschränkt Darlegung QS Endprüfung), 9004 (Erläutert die von Norm def. QS-Elemente)

ISO 9000-Ansatz

ISO 9000-3 Relevant Software-QS, Erleichterung Anwendung ISO 9001 für SWE, vollständige Realisierung ISO 9000-3 :arrow_right: automatisch ISO 9001, Zertifizierbar, bestimmte Req an Güte oder Sicherheit eines Produktes werden nicht verlangt (produktzertifikate), 3 Hauptartikel, Inhalt: Entwicklung, Lieferung, Wartung SW, Gliederung: einmalig, periodisch, pro Projekt, vorgehensmodell: kein spzeifisches, Voraussetzungen: SWE in Phasen, Vorgaben für Phasen

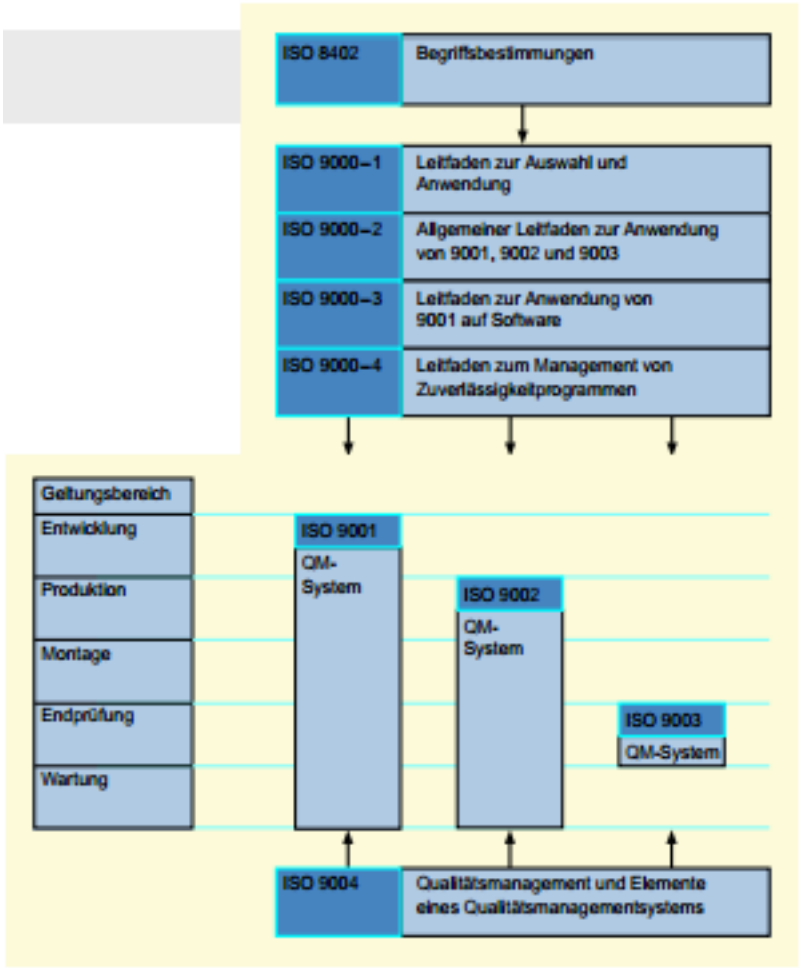

ISO 8402
Begriffsbestimmungen
ISO 9000–1
Leitfaden zur Auswahl und Anwendung
ISO 9000–2
Allgemeiner Leitfaden zur Anwendung von 9001, 9002 und 9003
ISO 9000–3
Leitfaden zur Anwendung von 9001 auf Software
ISO 9000–4
Leitfaden zum Management von Zuverlässigkeitprogrammen
Geltungsbereich
Entwicklung
Produktion
Montage
Endprüfung
Wartung
ISO 9001
QM-System
ISO 9002
QM-System
ISO 9003
QM-System
ISO 9004
Qualitätsmanagement und Elemente eines Qualitätsmanagementsystems

Figure 14:

festgelegt, geforderte Ergebnisse pro Phase festgelegt, Verifizierungsverfahren pro Phase festgelegt, dokumente: Vertrag Auftraggeber - Lieferant (Qualitätsrelevante Vertragspunkte), Spezifikation, entwicklungsplan, Qualitätsicherungsplan, Testplan, Wartungsplan, Konfigurationsmanagementplan

Zertifizierung Qualitätsaudit: systematische, unabhängige Untersuchung **Systemzertifikat:** Zertifizierungsstelle, Audit, bescheinigt Qualitätsfähigkeit, 3 Jahre gültig, jährlich Überwachungsaudits, nach 3 Jahren: Wiederholungsaudit (+ 3 Jahre)

TQM (Total Qualitlity Management)

Ziel: magsiches Dreieck (Qualität - Zeit - Kosten)

Qualitätsmanagementkonzept

Qualitätsphilosophie, operationalisierung in betrieblichem Alltag, allgemeine Modelle, abstrakte Schablonen

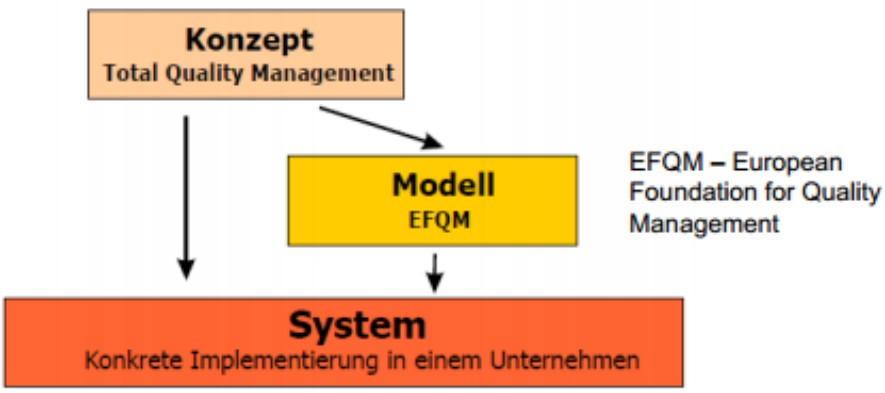


Figure 15:

Herkunft: Autoindustrie Japan

Definition: Mitwirkung aller Mitglieder, führungsmethode einer Organisation, qualität in Mittelpunkt durch Zufriedenheit der Kunden, langfristiger Geschäftserfolg

Zielsetzung: Integriert werden Interessen: Kunden, MA, Unternehmen, Lieferanten - Zentrales Ziel: Qualität aus Sicht Kunden, Kunde entscheidet über Qualität, TQM soll führen zu: höherer MA-Zufriedenheit, gesteigerter Produktivität, reduzierte Kosten, kürere Entwicklungszeiten - TQM muss: aktiv gestaltet, eingeführt, aufrecht erhalten, gelebt werden

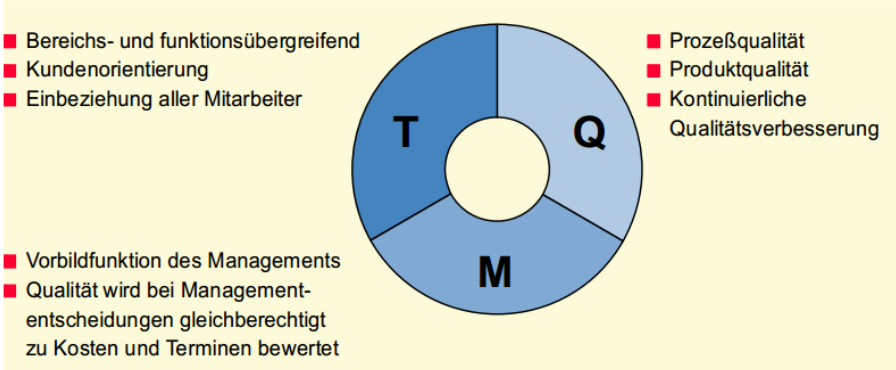


Figure 16:

EFQM-Modell

European foundation for Quality Management, Kern: Bewertungsmodell, Messung Umsetzung QM, Grundlage Preisvergabe & Selbstbewertung

Bewertungskriterien

3 Säulen (Menchen, Prozesse, Ergebnisse)

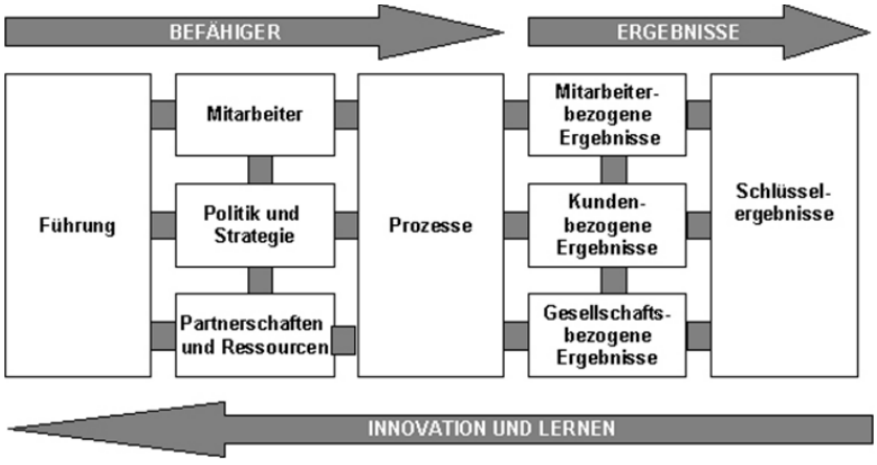


Figure 17:

Elemente "Excellence"

Stufen: Selbstbewertung, Commited to Excellence (Selbstbewertung, Priorisierung Verbesserungspotentiale, durch Validator - EFQM-Prüfer, Zertifikat 2 Jahre gültig), Recogniced for Excellence (umfangreiche Selbstbewerter, Datenerhebung Assessoren, Workshop von Assessoren und Bewerbern vor Ort)

Prinzipien TQM

- Quality first
- Alle Prozesse: Qualitätsprozesse
- Req an Prozesse: 100% erfüllt
- Jeder MA am P beteiligt: sofort beim ersten Mal und jedes Mal erneut richtig tun
- Qualitätsverbesserungen durch Verbesserungen Entwicklungsprozesse
- Verschwendung und Nachtarbeiten vermeiden
- Zuständigkeit aller MA
- Ständige Verbesserung
- Berücksichtigung soziales System mit gewachsenen Strukturen
- Kundenorientierung
- Interne Kunden-Liefernaten-Verhältniss

Einführung TQM erfordert

- Kenntnis der Prozese in einem UN
- Ausreichende Dokumentation
- Ständige Anpassung an neue Erfordernisse
- QS-System zur sätnigen Verbesserung aller Prozesse
- Konsequente Schullung aller MA in Sachen Q und QM

TQM Methoden

Qualitätszirkel

- Primat der Qualität
- Zuständigkeit aller MA
- Ständige Verbesserungen
- Vorgehensweise
  - Problemidentifikation, Problemauswahl
  - Problembearbeitung
  - Ergebnispräsentation
  - Einführung und Erfolgskontrolle

Pareto-Prinzip 80:20 Regel, 80% Aufwand um 20% der Probleme zu lösen, 80% Probleme können mit 20 % Aufwand gelöst werden, Bezogen auf QS, 20% Fehlerursachen erzeugen 80% der Fehler und der Kosten, 80% Fehler können mit 20% des Gesamtaufwands behoben werden.

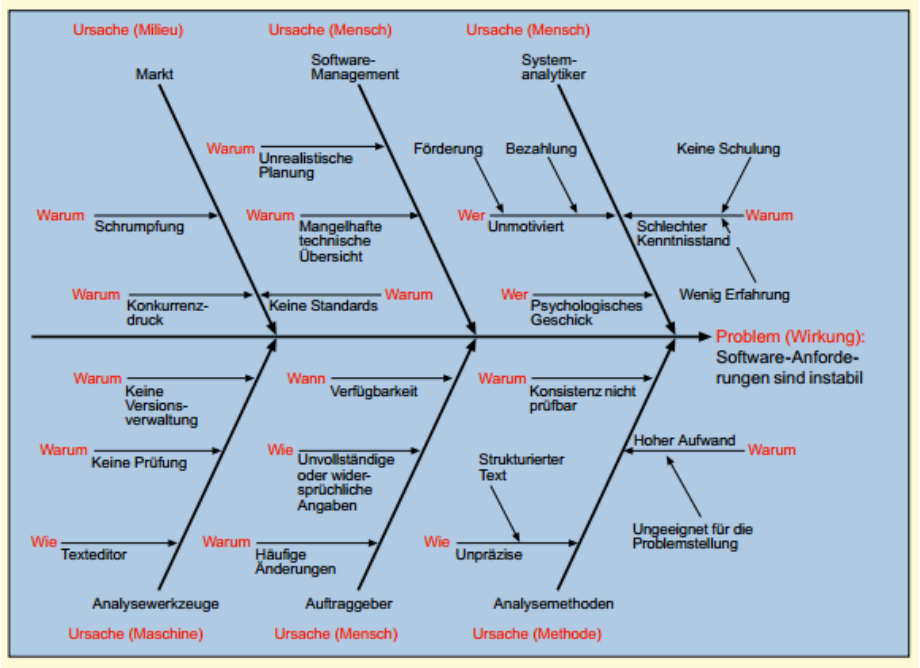


Figure 18:

Ursache-Wirkungs-Diagramm

Risiko Management

Risikomanagement Fester Bestandteil in Managementplänen in Unternehmen, Basis für gewisse Geschäftsbereiche (Versicherungen), Tehma in der SWE

Risiko
Neagative (Bestandesgefährdende) Entwicklung, allenfalls Verfehlung geplannter Ziele (Unternehmens- / Bereichsziele, . . .), Risiko :arrow_right: gewichtetes Muster möglicher Ereignisse + verbundene Folgen, potentielles Problem, ein Problem ist ein Risiko, das eingetreten ist.

Risikomanagement
Systematische Vorgehensweise über Korrekturmassnahmen nachzudenen, vevor

ein Problem eintritt, syst. Erfassung, Bewertung, Steuerung Risiken, Bestandteile: identifikation, Analyse, Bekämpfung, Gegenteil: Krisenmanagement

Bezug SWmanagement
Entwicklung SW-Sys historisch junge Disziplin, viele innovative MA

Risikomanagementprozess

SWE: Besonders schwierig Risiken rechtzeitig zu identifizieren, start bei Projekten, Risiken aller Projekte :arrow_right: Zusammenfassung auf Unternehmensebene, Risikoprozess etablieren um Risiken bewertbar zu machen, **Continuous Risk management Paradigma**: Center: Communicate, Circle: Identify - Analyse - Plan - Track - Control - ... | PMA müssen in der Lage sein, erkannte Risiken an zuständige Stellen weiterzuleiten ohne berufliche Konsequenzen fürchten zu müssen, , Vollständige Integration des RiskM in den Entwicklungsprozess

Risikobewertung und Risikobeherrschung

Risikobewertung: 1. Risikoidentifikation, 2. Risikoanalyse, 3. Risikoprioritätenbildung | **Risikoidentifikation:** Brainstorming / Checklisten, Ergebnis: Liste der projektspez. Risikoelmente, die den Projekterfolg gefährden, allenfalls Analyse vergangener / nicht erfolgreicher Projekte | **Risikoanalyse:** Wahrscheinlichkeit (Verständnis Risiko, Festhaltung genauer Umstände und Auswirkungen)+ Einfluss des Risikos (Erfahrungen oder Abschützung via Skala), für quantitative Bewertung Risiko muss Risikofaktor berechnet werden, Risikofaktor = Eintrittswahrscheinlichkeit x Schadenshöhe, nicht alle Risiken müssen gemanaged werden (sehr kleine Eintrittswsk, bei Eintritt wird Produkt nicht mehr benötigt, Folgen Risiko sind minimal, Risiko wird von anderer Partei getragen) | **Risikoprioritätenbildung:** Ordnung Risiken nach Prioriäten, z.B. via Risikofaktor

Risikobeherrschung: 4. Schritt: Planung Risikofall (für kritische Risiken Eventualfallspläne, was tun im Schdenfall), 5. Schritt: Risikoverminderung (wichtig: gute Planung Massnahmen + Verfolgung Durchführung), 6. Risikoüberwachung (Eintrittsindikatoren laufend überwachen)

Unsicherheiten und Risikodiagramme

Unsicherheiten: Anforderungen, Zusammenspiel (Produkt mit Benutzern / anderer SW), Umgebungsveränderungen, Ressourcen (MA, Skills, ..), Management (Zusammenstellung produktiver Teams), Supply Chain (erbringen andere PMA Leistung?), Politik (Machtspiele, unrealistische Ziele, etc..), Konflikt, Innovation (neue Techniken / Methoden), Skalierung (zusätzliche Fkt auf Projektleistung)

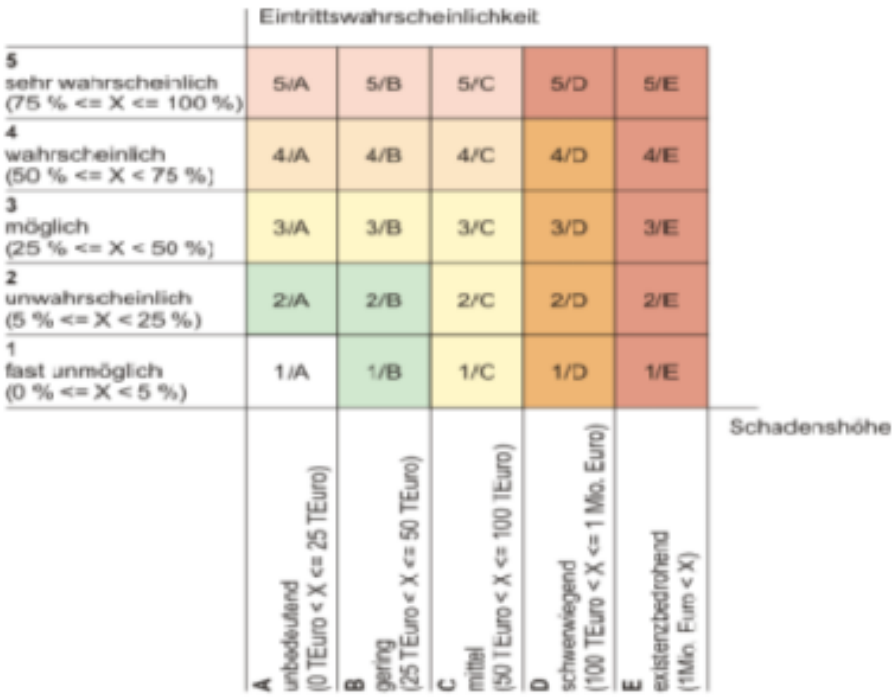
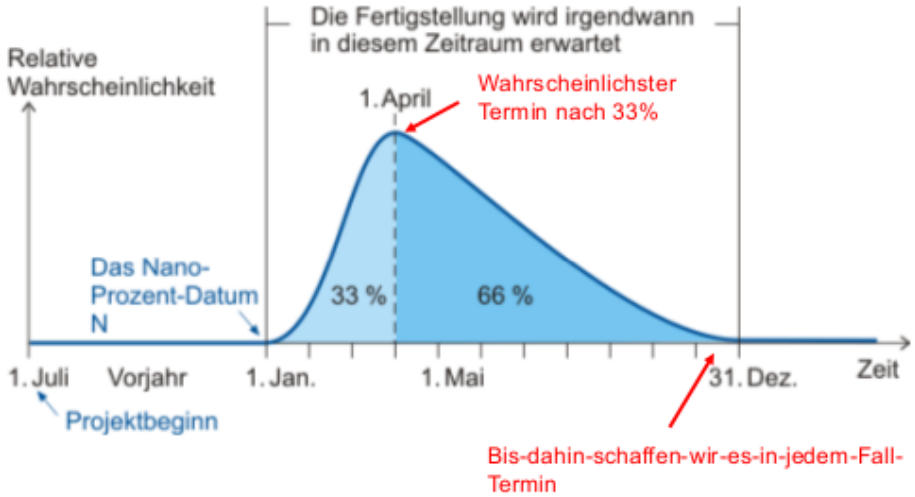


Figure 19: Risikoprioritäten



Risikodiagramm:
Abweichung Graph von x-Achse definiert das erste Datum, dessen WSK nicht bei null, aber dennoch im nanoprozentbereich liegt (Nanoprozentdatum), Regelfall: Ermittlung dieses Wertes bei Schätzung :arrow_right: Fehler: Festlegung Fertigstellungsdatum, Fertigstellungsdatum: liegt zwischen frühstmöglichem Termin und dem Bis-dahin-schaffen-wir-es-in-jedem-Fall-Termin

Kernrisiken

1. Fehlerhafter Zeitplan, 2. Inflation der Anforderungen, 3. Mitarbeiterfluktuation, 4. Spezifikationskollaps, 5. Geringe Produktivität

Verfolgung der Top Ten-Risiken

Schritte: Risikoelemente in Reihenfolge bringen, Festlegen Überprüfungstermine, Sitzung beginnt mit Bericht über Fortschritt bei den Top Ten-Risikoelementen, Sitzung: Konzentration Beseitigung RE

Qualitätsmanagement

Begriffe

Qualität: Grad, in dem Satz inhärenter Merkmale Anforderungen erfüllt (ISO 9000:2000), gibt an in welchem Masse ein Produkt (Ware / Dienstleistung) den

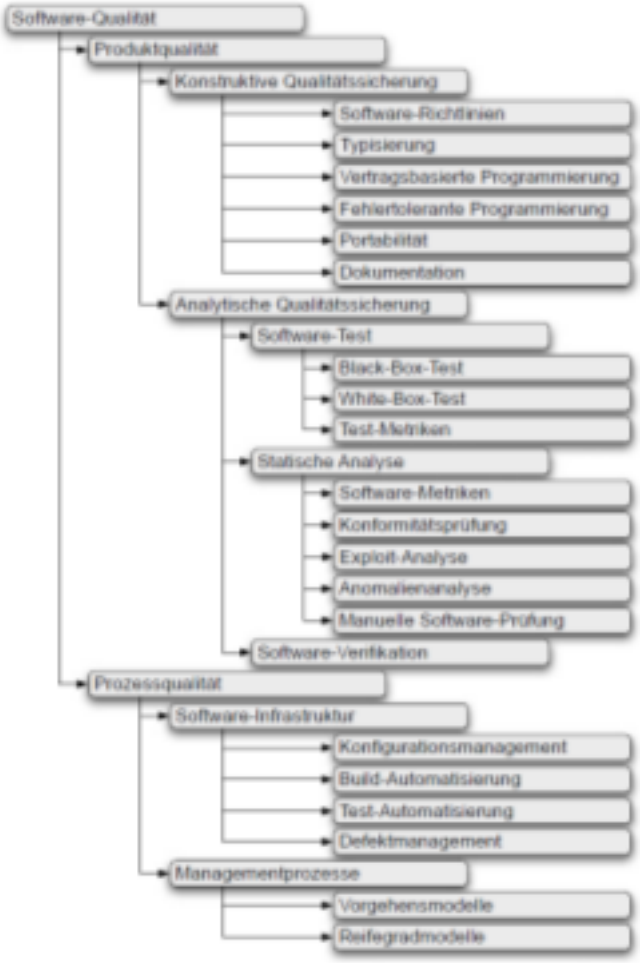
bestehenden REQ entspricht, REQ: Erfordernis / Erwartung vorausgesetzt / festgelegt, Inhärentes Merkmal: Kennzeichnende Eigenschaft einer Einheit, welche diese aus sich selbst heraus hat und die ihr nicht explizit zugeordnet ist | **Qualitätsmanagement:** aufeinander abgestimmte Tätigkeiten zum Leiten / Lenken einer Org. bezüglich Qualität, enhält: Festlegen Qualitätsprolitik, Qualitätsziele, Qualitätsplanung, Qualitätslenkung, Qualitätssicherung, Qualitätsverbesserung | **Qualitätssicherung:** Aktuelle Qualität :arrow_right: Massnahmen zur Korrektur, Abstrakt: Geplante und sys. Tätigkeiten, die innerhalb QM-Sys verwirklicht sind, und die wie erforderlich dargelegt werden, um angemessenes Vertrauen zu schaffen, dass eine EInheit die QA erfüllen wird.

Produkt vs. Prozessorientiertes QM

Produktorientiertes QM: Überprüfung von SW-P und Zwischenergebnissen auf vorher festgelegte Q-merkmale | **Prozessorientiertes QM:** Erstellungsprozess SW-Methoden, beinhlatent Werkzeuge, Richtlinien, Standards

Konstruktive vs. Analytische QM-Massnahmen

Konstruktive Massnahmen: Methoden, Sprachen, Werkzeuge, Standards, Checklisten, die dafür sorgen, dass entstehende Produkte / Erstellungsproz. bestimmte Eigenschaften besitzen. | **Analytische Massnahmen:** diagnostische Vorgehensweise, keine Qualität in Produkt / Proz., Messung existierende Qualitätsniveau



** Hauptgruppen: Prüfungsrelevant, inkl. Beispiele** ###Konstruktives QM Gewährleistung das es dokumentierte Vorgehensweisen für SWE gibt, z.B. Programmierrichtlinien, Reviews, etc.

Bsp. produktorientierte konstruktive Massnahmen: Gliederungsschema Pflichtenheft (Richtlinie), alle wichtigen Infos auf einer Seite (Methode), Einsatz Programmiersprache mit statischer Typprüfung können keine Typfehelr zur Lfz auftreten (Sprache) | **Bsp. prozessorientierte konstruktive Massnahmen**: Festlegung welche Teilprodukte mit welchem Inhalt und welchem Layout wann und von wem erstellt werden müssen, standardisiert den Entwicklungsprozess (Richtlinie), Konfigurationsmanagementsystem zur Identifikation SW-Elemente - Konfiguration (Richtlinie, Werkzeug), Festlegung zulässiger Zustandsübergänge, Festlegung Vorgehensmodell

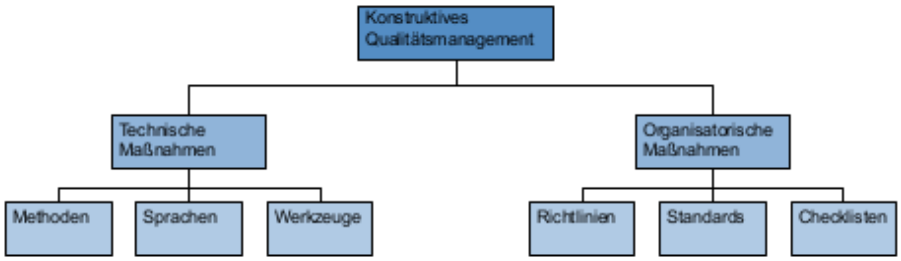


Figure 20: Konstruktive Massnahmen

Analytisches QM

Gliederung: Bezug zur Prüfung (Produkt- / Prozessprüfung), Automatisierungsgrad / Nachvollziehbarkeit / Einsatzbereich Prüfung (Definitions- / Entwurfs- / Impl- / Abnahme / Wartungs- & Pflege-phase)

Grundlagen des Softwaretestens

Begriffe und Motivation

SW immaterielle, nicht einfach prüfbar / testbar, Testen: Verringerung Risiken beim Einsatz

Fehlerbegriff

- Fehler: Nichterfüllung festgelegter Anforderung, Abweichung Ist / soll
- Mangel: festgelegte Anforderung / berechtigte Erwartung nicht angemessen erfüllt
- Fehlerwirkung / failure / Fehlfunktion / Ausfall: Beschreibung Sachverhalt Theoretisch auch durch Umfeld (Magnetismus, Strahlung, . . .)
- Fehlerzustand / fault: Fehlerwirkung hat Ursprung in Fehlerzustand der SW :arrow_right: auch als Defect / Bug bezeichnet
  Fehlerzustand muss nicht immer (am gleichen Ort) zu einer Fehlerwirkung führen
- Fehlermaskierung
- Fehlhandling / error: Einem Fehlzustand / Defect vorausgegangen, Fehlhandlung einer Person (z.B. fehlerhafte Programmierung)

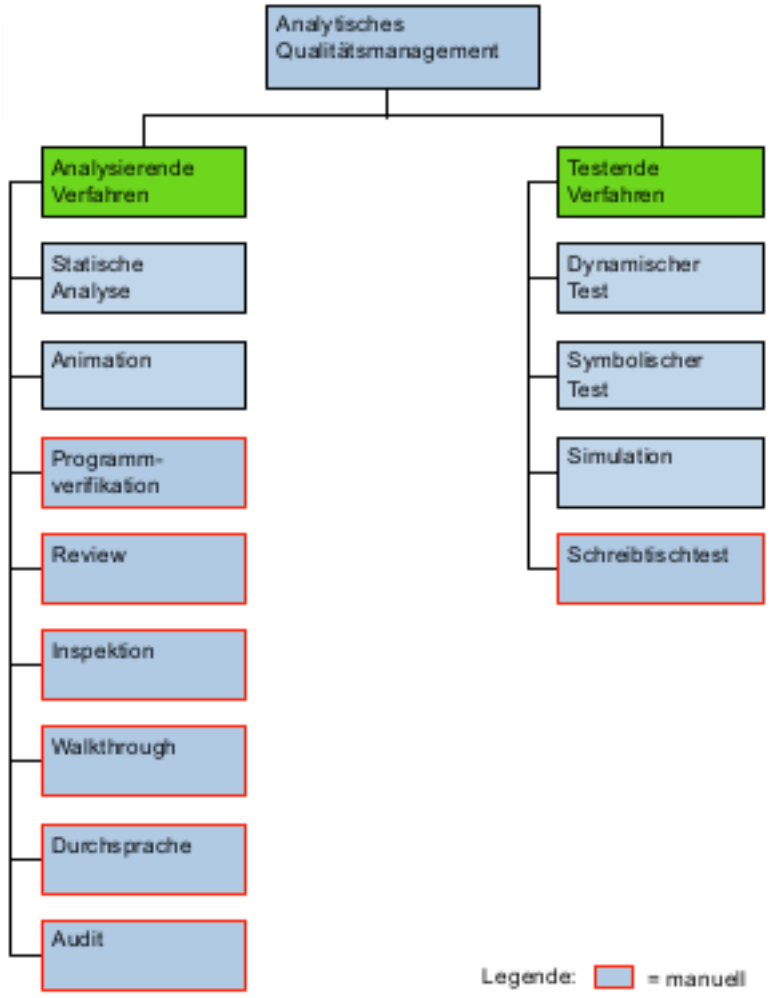


Figure 21: Analytisches QM

Testbegriff

- Debugging / Fehlerbereinigung / Fehlerkorrektur: Lokalisierung Fehler anhand Fehlerwirkung durch SW-Entwickler, Verbesserung Qualität, z.T. auch neue Fehlerzustände :arrow_right: Erneutes Testen
- Test: Fehlerwirkungen gezielt und systematisch aufdecken, jede Ausführung eines Testobjektes, die der Überprüfung dieses Testobjektes dient, Vergleich Soll- / Ist
  Ziele: Ausführung Programm mit Ziel: Fehlerwirkung nachzuweisen, Qualität zu bestimmen, Vertrauen in Programm zu erhöen, Analysieren Programm / Dokumente um Fehlerwirkungen vorzubeugen, statische Analysemethoden werden häufig auch zum Test gezählt
- Testprozess: Ausführen Testobjekt mit Testdaten, Planung, Durchführung, Auswertung Tests (Testmanagement)
- Testlauf: Umfasst Ausführung eines / mehrerer Testfälle, festgelegte Randbedingungen, Voraussetzungen zur Ausführung, Eingabewerte, vorausgesagte Ausgaben / Verhalten
- Testszenarien: Aneinanderreihung Testfälle
- Fehlerfreiheit: Durch Testen nicht erreichbar (ausser bei sehr kleinen Programmen)
- Kategorien Test-Benennung
  - Testziel: Nach Zweck, z.B. Lasttest
  - Testmethode: Nach Methode welche zur Testspezifikation / -Durchführung eingesetzt wird, z.B. geschäftsprozessbasierter Test
  - Testobjekt: Nach Art des Testobjektes, z.B. GUI-Test
  - Teststufe: Nach Teststufe, z.B. Systemtest
  - Testperson: Nach Personenkreis, z.B. Entwiklertest
  - Testumfang: Nach Umfang, z.B. Regressionstest, Volltest

Softwarequalität

Testen dient Steigerung SW-Qualität, nachgewiesene Qualität entspricht dann erwartete Qualität, SWQ umfasst mehr, ISO 9126: Qualitätsmerkmale: Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit, Übertragbarkeit - Prüfung durch geeignete Tests, ISO 25012: Ablösung, 3 Modelle: quality in use model, product quality model, data quality model

- Funktionalität
  Fähigkeiten miest durch Ein-/Ausgabeverhalten oder Reaktion / Wirkung in System beschrieben, Teilmerkmale: Angemessenheit, Richtigkeit, Interoperabilität, Ordnungsmässigkeit, Sicherheit
- Zuverlässigkeit
  Sys Leistungsniveau unter festegelegten Bedingungen über def. Zeitraum

wahren, Unterteilung in Reife (Wie häufig Versagen SW durch Fehlerzustände), Fehlertoleranz, Wiederherstellbarkeit, Fehlertoleranz: (automatische) Wiederaufnahme, Robustheit

- Benutzbarkeit
  Teilaspekte: Verständlichkeit, Erlernbarkeit, Bedienbarkeit, Einhaltung Standards, Konventionen, Style Guides, Überprüfung beim nicht funktionalen Test
- Effizienz
  Messbare Ergebnisse, Zeit / Verbrauch Betriebsmittel für Erfüllung Aufgabe
- Änderbarkeit & Übertragbarkeit Änderbarkeit: Analysierbarkeit, Modifizierbarkeit, Stabilität, Prüfbarkeit - Übertragbarkeit: Anpassbarkeit, Installierbarkeit, Konformität, Austauschbarkeit

Priorisierung Merkmale notwendig

Testaufwand

Vollständiger Test praktisch nicht durchführbar, Verallgemeinerung Aufwand nicht möglich, stark vom Projekt abhängig, Fehler können hohe Kosten verursachen, Testaufwand hoch im Vergleich zu was?, Testaufwand verünftiges Verhältnis zum erzielbaren Ergebnis, “Testen ist ökonomisch sinnvoll, solange die Kosten für das Finden und Beseitigen eines Fehlers im Test niedriger sind als die kosten, die mit dem Auftreten des Fehlers bei der Nutzung verbunden sind”, Sys mit hohem Risiko müssen ausgiebig getestet werden :arrow_right: Testintensität / -umfang in Abhängigkeit Risiko, Auswahl Testverfahren je Aspekt, Test auch auf Fkt., die nicht gefordert ist, **Testfallexplosion:** Schnell hunderte Testfälle / -varianten, Test-Manager: Beschränkung Anzahl, oft wenig Ressourcen, oft schlechte Karten, wenn im Projekt auch Ressourcen knapp sind, :arrow_right: gute Strategie notwendig :arrow_right: fundamentaler Testprozess, Schlüsse aus früheren Projekten

Fundamentaler Testprozess

Testen in jedem Vorgehensmodell, Zusätzlich zu Modellablauf: verfeinerter Ablaufplan für Testarbeiten, Inhalt “Entwicklungaufgabe” “Testen”: Schritte: Testplanung und Steuerung, Testanalyse und-design, Testrealisierung und -durchführung, Testauswertung und Bericht, Abschluss Testaktivitäten - Auffassung als generische Prozessbeschreibung

Testplanung und Steuerung

Planung Testprozess am Anfang SWE-Projekt, analog Projekt: regelmässig Kontrolle Planung, Aktualisierung, Anpassung, Kernaufgabe Planung: Bestimmung

Teststrategie, Prioritäten anhand Risikoeinschätzung

Testkonzept: * Aufgaben & Ziele der Tests festlegen * Ressourcen einplanen (Zeit, Hilfsmittel, Geräte) * ...

Steuerung: * Überwachung (im Vergleich Planung) * Berichterstattung * Aktualisierung Planung

Testmanagement: * Verwaltung Testprozess * Testinfrastruktur * Testmittel

Testintensität stark Abhängig Testmethode und zu erreichenden Überdeckungsgrad (Kriterium Ende Test) + evtl. Erfüllung Anforderung als Endkriterium, z.B.: Alle Fkt mind. einmal getestet, oder 70% aller möglichen Transaktionen, Ausgangs- / Testendkriterien, Priorisierung Testfälle, Auswahl & Beschaffung Werkzeuge, Prüfung Aktualität vorhandene Werkzeuge, z.T müssen Teile der Testinfrastuktur selbst realisiert werden

Testrahmen (test bed): darin werden Systemteile zur Ausführung gebracht, müssen miest selbst programmiert werden

Testanalyse und Testdesign

Testbasis analysieren: Dokument ausreichend Detailliert umd Testfälle abzuleiten (z.B Req-Dok, Architektur-Dok, Risikoanalyse, weitere), Testbarkeit feststellen: Testobjekt prüfen ob leicht zu testen (z.B. Zugang zu SST), Berücksichtigung bei Programmierung, Grundlage für Konkretisierung Testbedingungen, Berücksichtigung Risiko für weiters Vorgehen, **Traceability:* zwischen Req und spez. Testfälle, + Verfolgbarkeit Änderungen der Req / Testfälle - Zuerst logische Testfälle, dann Konkretisierung (d.h. Tatsächliche Eingabewerte festlegen), auch umgekehrt möglich (wenn Testobjekt unzureichend spezifziert), konkrete Testfälle im Prozessschritt Realisierung, Auswahl log. Testfälle Basis Testbasis, Spez. Testfälle zu unterschiedlichen Zeiten Entwicklungsprozess (Black / Whitebox), Testfall: Vorbedingungen, Randbedingungen, Erwartetes Ergebniss / Verhalten, Testorakel: z.B. Spezifikation, Entweder Ableitung aus Spez, oder via Inverse-Funktion nach Test - Testfälle nach 2 Kriterien unterschieden: Prüfung spezifizierte / erwartete Ergebnisse / Reaktionen, Prüfung nicht spezifiziert / unerwartete / ungültige Eingaben - Beschreibung Testinfrastruktur

Testrealisierung und Testdurchführung

Bildung konkrete Testfälle aus log. TF, Realisierung Testinfrastruktur und Testrahmen im Detail, Testläufe durchführen / protokollieren, Priorisierung TF, TF zu Testszenarien, Testrahmen programmiert und geprüft - Nach Übergabe an Test: Prüfung Vollständigkeit Systemteile :arrow_right: prinzipielleStart- / Ablauffähigkeit, Start Test mit Smoketest (Prüfung Hauptfkt.), Testdurchführung: Infos / Dok: Testrahmen, Eingabedaten, Testprotokoll, etc., Re-

produzierung später, Bei allfälligem Fehler: Entscheid Fehler Ja / Nein, evtl. ergänzende TF, Klassifizierung Fehler

Testauswertung und Bericht

Prüfung ob in Planung festgelegte Ausgangskriterien erfüllt sind, kann zur Beendigung Testaktivitäten führen, angemessenes Endkriterium pro Methode, weitere Tests wenn mind. 1 Kriterium nicht erfüllt, evtl. weiterer Tests notwendig (! nicht alle Tests bewirken Annäherung Endkriterium), Berücksichtigung Risiko, , Nichterfüllung Ausgangskriterium (z.B, durch tote Programmanweisungen) :arrow_right: Hinweis auf ungereimtheiten in Spez, evtl. Überarbeitung Planung - Kriterien: Testabdeckungskriterium, Fehlerfindungsrate - Praxis: Testende häufig durch Zeit / Kosten - Alle Abschlusskriterien erfüllt: Zusammenfassender Bericht für Entscheidungsträger, Komponententest: Mitteilung Erfüllungsgrad an Projektmanager

Abschluss der Testaktivitäten

oft vernachlässigt, Abweichungen Planung / Umsetzung, Ermittlung Gründe, Daten: Datum Freigabe SW-Sys, Datum Testende / -abbruch, Meilensteine, Wartungsreleases, wichtige Informationen: Welche geplanten Ergebnisse wurden wann erreicht? unvorhergesehene Ereignisse?, offene Probleme / Änderungswünsche?, Akzeptanz wie hoch? - Kritischer Rückblick, Verbesserungspotential, Konservierung Testmittel für Wartung

Psychologie des Testens

Errare humanum est, Testen nicht destruktiv, sondern extrem kreativ / intellektuell herausfordernd, **Entwicklertest:** genügedn Abstand, eigenes Produkt kritisch prüfen, Gefahr sinnvolle Tests zu vergessen / nur oberflächlich, grundsätzliche Designfehler durch falsch verstandene Aufgabenstellung, Testfall kommt ihm nicht in den Sinn (Blind), Vorteil: keine Einarbeitung notwendig - **Unabhängiges Testteam:** Förderlich Qualität, Schärfe, Unvoreingenommenheit, Aneignung Know-How, dafür Test-Know-How, Art und Weise Mitteilung Fehler sehr wichtig, nicht einfach / leicht, Reaktion Entwickler “It’s not a bug, it’s a feature” nicht hilfreich, Förderlich Zusammenarbeit: Gegenseitige Kenntniss Arbeit, analoges Problem auf Mgmt-Ebene

Allgemeine Prinzipien Softwaretest

1. Testen zeigt Anwesenheit von Fehlern
   Kein Beweis für keine Fehlerzuständie im Testobjekt
2. Vollständiges Testen ist nicht möglich

3. Mit dem Testen fürhzeitg beginnen
4. Häufung von fehlern
   Gleichverteilung Fehlerwirkungen meist nicht gegeben
5. Zunehmende Testresistenz (pesticide paradox)
   Tests lassen in Wirksamkeit nach, TF regelmässig prüfen / modifizieren / ersetzen
6. Testen ist abhängig vom Umfeld
7. Trugschluss: Keine Fehler bedeutet ein brauchbares System

Ethische Leitlinien

Tester häufig Zugang zu vertraulichen / sensiblen Informationen, angemessener Umgang

Ethik-Kodex für "Certified Tester" * Öffentlichkeit
Verhalten in Übereinstimmung öffentlichem Interesse * Kunde und Arbeitgeber
Wahren Interessen Kunde / Arbeitgeber in Übereinstimmung öffentlichem Interesse * Produkt
Sicherstellung glieferte Arbeitsergebnisse erfüllen höchstmögliche fachliche Anforderungen * Urteilsvermögen
Bewahren Rechtschaffenheit und Unabhängigkeit in sachverständiger Beurteilung * Management
SWT-Manager / Testleiter moralisch einwandfreie Einstellung zum Management SW-Test * Berufsbild
Fördern Integrität und Ansehen ihres Berufes in Übereinstimmung öffentliches interesse * Kollegen
Gegenüber Kollegen: fair und hilfsbereit, Förderung Kooperation * Persönliche Einstellung
berufsbegleitend kontinuierlich fort- / weiterbilden, an diesen Leitlinien orientierte ethische Einstellung fördern

Change- und Configuration Management

Durchschnittliches Informationssystem: Lebensdauer 12-15 Jahre
Durschnittliche DB: 20-30 Jahre
Durchschnittliche Bauzeit, Durchschnittlicher SW: Früher: 42 Monate, Aktuell: ca. 27 Monate
Informationsmittel: Betriebsmittel, Unterstützende Instrumente

Change Management als Prozess

RUN-Phase: Im Griff mit Prozessen, Decommit-Phase: Schwierig, Halsbrecherisch, Change / Veränderung: Planung, jemand muss verantwortlich sein,

Einführung neuer Arbeitsweisen / Abläufe via IT, Rollout: Braucht Freunde ("Superuser")

Veränderung: Über Zielsetzungen: Transformationsziele, Reduktionsziele (Zwischenlösung), Applikationsziele (Theorie)

Genereller Ablauf Veränderungen: - Zielformulierung - Verantwortung definieren - Zeitraum - Change Control Board (in Industrie: z.T. zu umfangreich)

Prüfung: Phasen: Create / Build / RUN / Decommit, Problem Rollout

Software Maintenance

Standard: IEEE 1219, Kosten: 90 %, 4 Change-Arten: Adaptive, Perfective, Corrective, Preventive
Projekt: Preis wird gedrückt, nach Abnahme: CR-Zeit (Verrechnung)
Vermeidung Hohe Kosten, Software nicht fehlerfrei / nicht perfekt, Akzeptable Unterhaltskosten, Effekte: Lineare Attraktoren (Systemzustände mit magnetischen Wirkungen, n+1 Änderung –> das Ganze kippt unaufhaltbar, Explosion Maintenance-Kosten), Pflege von KnownErros (Billigste Art von Fehlerbehebung)

CAPEX: Capital Expenses (Investitionen für Plan, build)
OPEX: Operational Expenses (Run)

Maintenance: Nicht mehr alle Wünsche erfüllen, wenn dann eher via Change Request, Diskussionen: Fehler oder CR
Ziel Entwickler: Möglichst geringe Unterhaltskosten, Konflikt: Perfektionismus / Pragmatismus

ITIL

Idee: Betriebsmittel als Service Ziel ITIL (Gegensatz SWE): Es soll so bleiben, Continual Service Improvement: Bringt nichts, wird in der Praxis nicht verwendet.
Service-Bäume: Logischer Aufbau, nicht hierarchisch, Wo ist die Vertragsschnittstelle Ticketing-, Knowledge-, Monitoring-, ITIL-System: REMEDY, Tivoly, HPOV

Relevante ITIL-Prozesse im Betrieb: - Wichtig: - Incident-Mgmt - Problem-Mgmt - Change-Mgmt - Configuration-Mgmt - Release Mgmt - Weniger Wichtig: - Service-Mgmt - Availability-Mgmt - Contingency-Mgmt - Capacity-Mgmt - Continuity-Mgmt - Financial-Mgmt

Beispiel

Es soll so bleiben wie es ist: Etwas ist anders –> Callcenter / Helpdesk –> Trouble Ticket, Max. zur Behebung: Keine Änderung an Konfiguration -

geöffnete und geschlossene Incidents- / Trouble-Tickets, Keine Standardisierte Benennung Second- / Third-Level

Reproduzierbare Fehler:
Nicht-Reproduzierbare Fehler: Aus Betriebssicht keine Fehler, nicht bearbeitbar Configuration-Manager ~= Product Manager (Zuständige Person, nicht aus SCRUM)
3 Eskalationsstufen: 1. Vorfall - Keine sofortige Behebung: 2. Incident (Behebung ohne Systemveränderung, ohne Lösung): Problem (Fachleute, z.B. Reboot, Indexierung, Clean) - 3. Change (Wenn Problem nicht behebbar) | Systemzustand darüber messbar, Anzahl Incidents: egal, Anzahl Probleme: interessant –> Viele Probleme / Wenig Changes –> Plattform etwas schief

Kosten der Informationstechnologie

Allgemeines und Ausgaben für die IT

Entwicklungs-Kosten, Betriebs-Kosten (ca. 20% der Entwicklungskosten pro Jahr), Betriebsnutzen, Entwicklungsnutzen (selten kalkuliert, Nutzeneffekte durch Engineering, z.B. Prozessmodellierung), IT heute nur als Kosten in der Buchhaltung (werden heute nicht korrekt verrechnet), Strukturkapital: Strukturen (Businessregeln, etc.), IT: Wachstumsbranche, starker Einfluss auf technologisch orientierte Wirtschaftssektoren, Wertschöpfung: Hälfte durch interne IT, Hälfte durch externe IT, Ausgaben IT: Durchschnitt über alles: 4% des Umsatzes, Dual-Vendor-Strategie (Speziell Schweiz), 20'000 IT-Unternehmen (ca. 1 -4 Angestellte)

Kennzahlen für die IT

Wie bewertet jemand, der nicht versteht was die IT tut?

- Kalkulatorische Bewertung (Vor Projekt, Projektantrag)
  Rechtfertigung durch Investitionsrechnung (Net Present Value oder vereinfacht: Return On Investment)
- Kalkulatorische Bewertung nach Projekt: unüblich
- Total Cost of Ownership
  Kosten von Erstinstandsetzung bis Abschaltung, Zusammensetzung aus direkten (HW, SW, Operations, Verwaltung) und indirekten Kosten (End-User Operations, Downtime)
- Opportunitätskosten
  Wird selten benutzt

IT-Controlling

Wie gut ist eine IT?, Woher kommen die Zahlen für das Controlling? (Siehe Beilage)

Pricing & Estimation

Einleitung

- Time & Material (Nach Aufwand über Zeit)
  - Pure T&M
  - T&M wire price Limit (Kostendach), sollte nicht gemacht werden, nicht fair, unternehmerisch nicht fair
  - T&M with quantity discount
- Fix-Pricing ("Werkvertrag")
  - Pure Fix-Price
  - Conditional Fix-Price (z.B. Höherer Preis vor best. Datum, Niedrigerer Preis nach best. Datum)
- Value Driven Pricing (Preis durch Wertbeitrag festgelegt, z.B. Entwicklung für Start-Up: Kein Preis, Preis auf Verwendungsbasis)
  - Value Driven Price on client saving / earnings
  - Value Driven Price on number of pieces
- Risk Based Pricing (Preismodell von IBM für Cloud-Services unter Berücksichtigung Risiken)

Preisgestaltung

Marktwirtschaftliche Preisgestaltung (S. 6): Nicht 1:1 in IT, Stundenansatz (~125.-/h in CH), Erwartete Nachfragemenge (Nicht relevant), Analyse des Marktpreises (Schwierig, Ansatzpunkt Stundensatz), Vergleich Mindestpreis-Marktpreis (Marge, wie viel können wir nachgeben? Marge: ~5-15%) Reuse-Faktor von Projekten sehr gering / unwahrscheinliche, Selbstständiger: Ein Drittel der Einnahme sofort weg (Preis: 127.- + 1/3), Preisberechnung für Systeme mit Umsystemen schwierig

Lizenzen

Buch "Das 1x1 des Lizenzmanagements", ca. 1/2 aller Firmen falsch lizenziert (über- / unterlizenziert)

Offertprozesse

Angebote sind im Prinzip Chef-Sache (auf Seite Anbieter), Vereinheitlichung Angebotsprozess schwierig

Öffentliche Vergabeverfahren

Öffentliche Hand: guter Kunde (Ausnahme: Spanischer Staat), Liefer- oder Dienstleistungsauftrag, www.simap.ch (Öffentliche Ausschreibungen), Leistungsbezüger (LB): Auftraggeber, Leistungserbringer (LE): Auftragnahmer, Grenze für öffentliche Ausschreibung: 0.5 Mio. CHF (von WTO festgelegt)

- Offenes Verfahren [S. 12]
  Angebot ausarbeiten und einreichen: Üblich Fragerunde bei Ausschreibung, alle Fragen und Antowrten aller potenziellen LE werden an alle versendet
- Selektives Verfahren [S. 13]
  Zahl der Anbieter aufgrund von Anforderungskriterien einschränken.
- Einladungs-Verfahren [S. 14]

- Freihändiges Verfahren [S. 14]
  Anbieter wird ausgewählt, eher in der Industrie

Heute: Kontingente für offene Verfahren, z.B. Reservation von bestimmter Anzahl Tage in den nächsten x Jahren, Auswahl mit freihändigem Verfahren bei konkretem Bedarf, Dienstleistungslose

Verfahren zur Aufwandschätzung

Algorithmische Verfahren: Verlassen Hochschule nie, keine Verwendung in Praxis, Schützung: Durchschnittliches Projekt (40% Codierung, 20% Projektleitung, 20% Testing, 20% Admin), Puffer auf Basis Risikoeinschätzung: 10-20%, werden oft in Preisverhandlungen wieder weggegeben, Serios wenn Risikoaufschlag ausgewiesen, Höherer Preis durch mehr Testing / bessere Qualität: In Praxis funktioniert das nicht

- Algorithmic (Nicht angewendet)
  - COCOMO: Constructive Cost Model, Produktgrösse als Basis für den Aufwand in Personenmonaten und Laufzeit verwendet
  - COCOMO Intermediate: Erfahrungsfaktor
  - COCOMO II
  - SEER
  - PRICE

  - CoBRA
  - KnowledgePLAN
- Productivity (sehr selten angewendet)
  - Sizing Based Model: (z.B. nach Function Points, nicht geeignet, schwer nachvollziehbar)
  - Function Based Model
  - Object Based Model
- Subjective
  - Experts View (Einzel- / Mehrfachbefragung, Delphi-Methode, Schätzklausur)
    Delphi: Ab gewissem Projektvolumen (5-10 Mio.)
  - Expirience Based
- Others
  - Parkinson
    Klassische Berechnung + Kontrolle durch Berechnung Benötigte / Verfügbare Personen über Projektlaufzeit und Stellenprozent
  - Price to Win Unternehmen will gewinnen, auf dieser Basis wird der Preis festgelegt

Wichtigste Schätzmethode: Expertenschätzung (mehr Experten != bessere Qualität), Prüfen mit Parkinson, Nachvollziehbarer Preis, Budgetierung gemeinsam mit Kunde

Wichtig (Prüfung): Arten Preisgestaltung (Time & Material, Fixed Price, Value Driven), Was ist fair aus Sicht Betreibswirtschaft, Prozesse Auswahl öffentliche Hand, Wieso Subjektive Schätzmethode die wichtigste? Erklären Parkinson / Price to Win, Wie würden Sie zählen? (irgendeine Antwort, einfachste Antwort: So das es jemand versteht –> nachvollziehbar)

Mündliche Prüfung

3 Fragen, Hinweise aus dem Unterricht

Kapitel 2

Grundlagen

Begriffe und Motivation

Test: An activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of

some aspect of the syste mor componente

Validierung: Prüfung, ob Entwicklungsergebnis die individuellen Anforderungen bzgl. einer speziellen beabsichtigten Nutzung erfüllt.

Verrifikation: Prüfung (formaler Beweis), ob Ergebnisse einer Entwicklngsphase die vorgaben der Phaseneingangs-Dokumente erfüllen.

Error/Mistake: (Fehlhandlung, Versagen, Irrtum), Fehlerhafte Aktion einer Person (irrtum), die zu einer fehlerhaften Programmstelle führt - Eine Person macht einen Fehler - (Schulung, Konvention, Prozessverbesserung)

Fault/Defect/Bug: Fehlerhafte Stelle eines Programms, die ein Fehlverhalten auslösen kann - Programm enthält Fehler - (Inspektion, Review, Programmanalyse)

Failure: (Fehlverhalten, Fehlerwirkung, Mangel) - Fehlverhalten Programm gegenüber spezifikation, das während seiner Ausführung tatsächlich auftritt. - Anwendung zeigt einen Fehler - Test

Wirkung Defect: Lokalisierung, Bekannt: Wirkung, debugging: Dynamische Tests (können Fehlwirkungen zeigen, die durch Fehlerzustände versursache werden), Debugging (Entwicklungsaktivität, die Ursache einer Fehlwirkung identifiziert, analyisert, entfernt)

Ziele: Aufdecken Fehlerzustände (möglichst viele Fehlerwirkungen verursachen), Erzeugen von Vertrauen bzgl. Qualitätsniveaus des Systems, Liefern von Informationen zur Entscheidungsfindung, vorbeugen von Fehlerzuständen

Begriffe

Testfall (Test case): Ausführungsbedingungen, Eingabedaten, Sollresultat

Testsuite: Sammlung von Testfällen

Expected result: Für Eingabe spezifziertes Resultat

Coverage: Testvollständigkeitsabdeckung

Testware: Gesamte Testumgebung (test harness)

Test-Orakel: Modul, dass das Sollresultat für einen Testfall berechnet

Testobjekt: Prüfling

SUT: System under Test

Unsystematischer Test

Laufversuch: Entwickler testet, Test ist beendet, wenn Programm läuft und die Ergebnisse vernünftig aussehen

Wegwerf-Test: Test ohne Systematik, Probieren, Notiz bei Ungereimtheiten, Test endet wenn Tester findet es sei genug getestet

Ziele: Reproduzierbarkeit, Planbarkeit (Aufwand / Nutzen prognostizieren), Wirtschafltichkeit (Optimierung Aufwand / Nutzen), risiko- / Haftungsreduktion

Systematischer Test

spezialisten testen, Test ist geplant (Testvorschrift), Programm gemäss Testvorschrift (Testspez), Ist- / Sol-Vergleich, Dokumentation Ergebnisse, Fehlersuche / -behebung separat, Wiederholung nicht bestandene Tests, Test endet, wenn Ziele erreicht, Laufende Aktualisierung Testspezifikation

Software Qualität nach DIN/ISO 9126

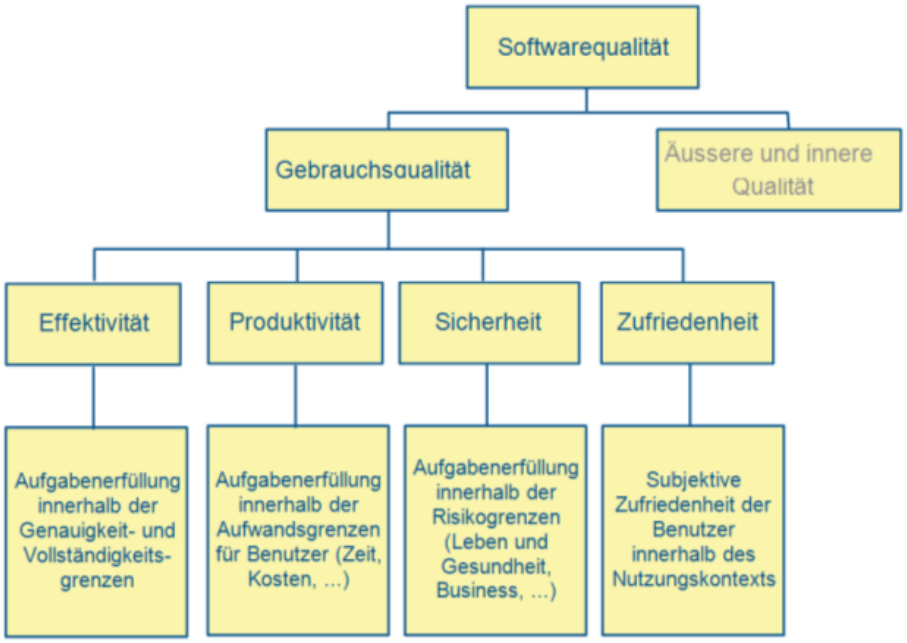


Figure 22:

Qualitätsanforderungen

Vorgabe pro Produkt: Welche Güte (Qualitätsniveau) pro Qualitätsmerkmale, Nicht alle QMM lassen sich gleich gut erfüllen, Prioritäten festlegen (Absprache Auftraggeber, Anwender), Bestandteil NFA

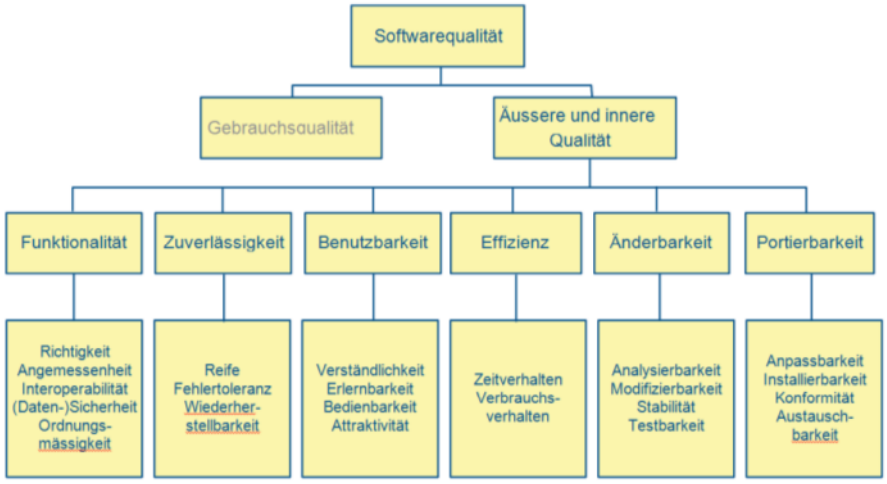


Figure 23:

Testen und Qualität

Testen misst Qualität anhand anzahl gefundender Fehlwirkungen, erhöht indirekt Qualität, erhöht indirekt Prozessqualität (Fehler dokumentiert, analysiert, etc.), erhöhen Vertrauen in Q

Testaufwand

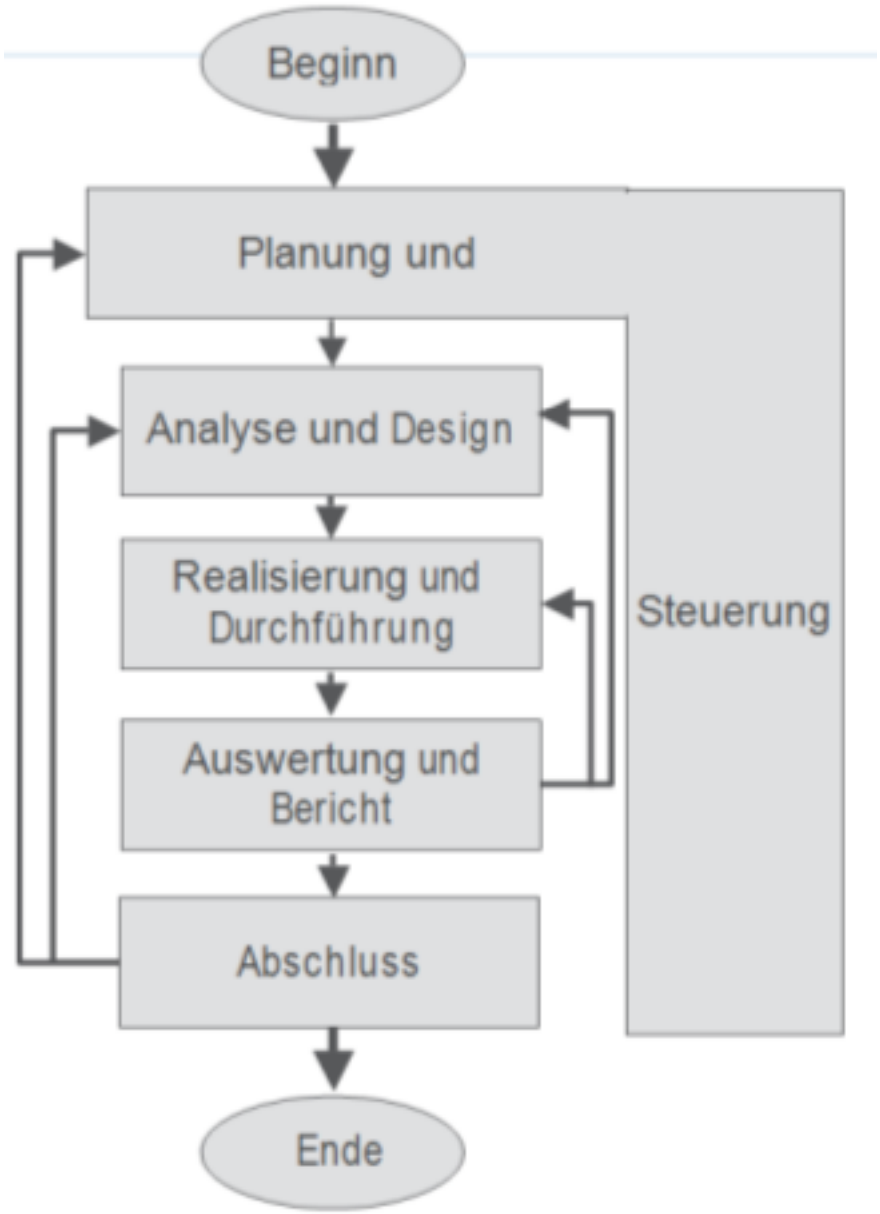
Vollständiges testen eines Programmes nicht möglich, Testaufwand in der Praxis: 25% - 50% des Entwicklungsaufwandes, Testintensität und -umfang in Abhängigkeit Risiko und Kritikalität, 2/3 Testaufwand z.B. auf Komponententests, Testen unterliegt beschränkten Ressourcen, Wichtig: Adäquate Testverfahren auswählen, Unnötige Tests vermeiden, Sowohl Positiv / Negativ-Tests berücksichtigen, Tests auf Fkt, die nicht gefordert sind, können sinnvoll sein

Qualitätssicherung

Analytische QS (Ergebnisse - Dokumente - Software | Prozesse) - Konstruktive QS (Normen, Standards, PL, SW-Technik, . . .)

Fundamentaler Testprozess

In jedem Vorgehensmodell, V-Modell: Trennung konstruktive / prüfende Aktivitäten



z.T. Aktivitäten überlappend / parallel, geeignet für jede Teststufe

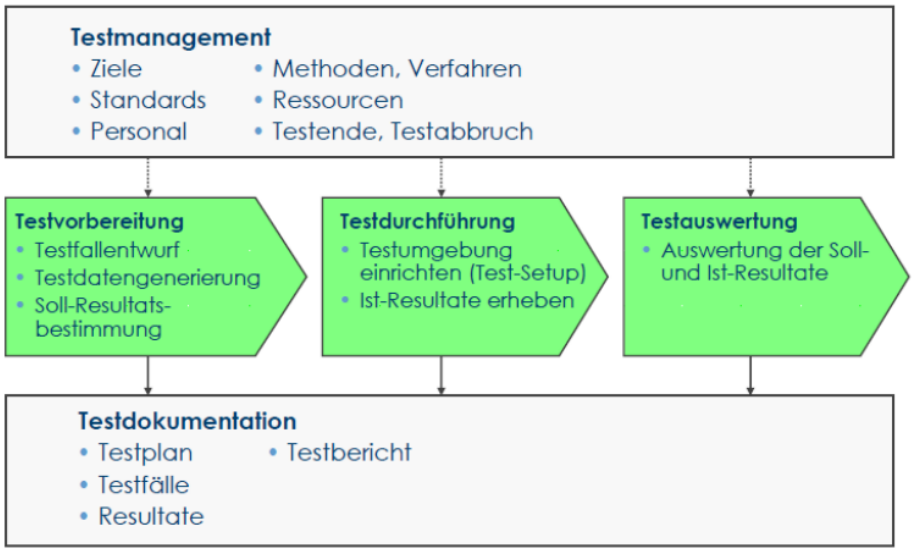


Figure 24:

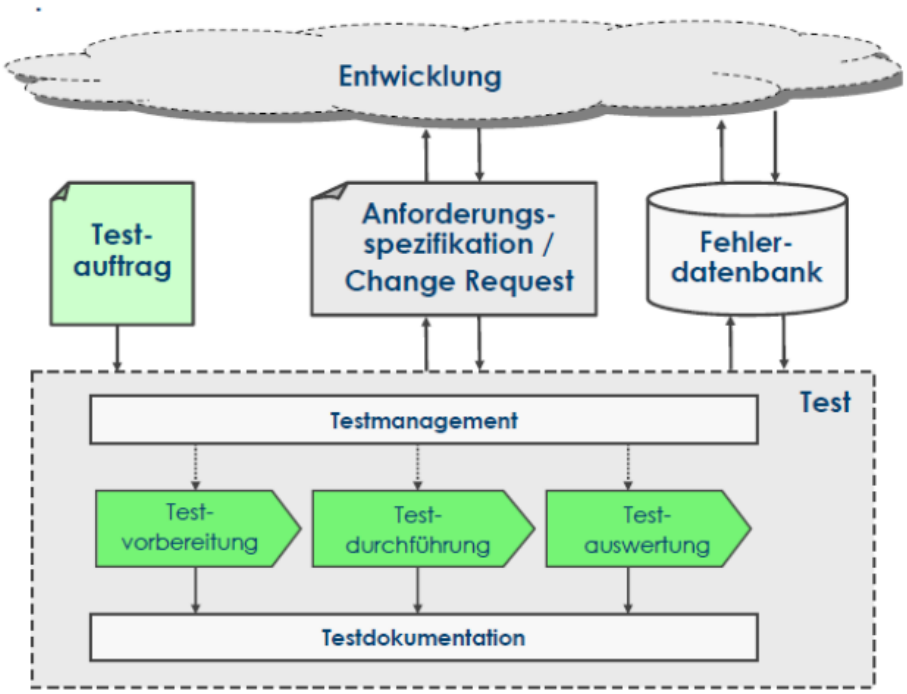


Figure 25:

Testplanung und Steuerung

Testplanung und Steuerung

Testauftrag: Prüfling, Termine, Testaufwand & Ressourcen, Testgüte (Rerun-all, Modifikationstest, Fehlernachtest, Nur Testfälle best. Prio), Testende- / Testabbruchkriterien

Testanalyse und Testdesign

Hauptaufgaben: * Review Testbasis (Req, SW Integrity level 1, Risikoanalysebericht, Architektur, Design, SST-Spez) * Bewertung Testbarkeit von Testbasis und Testobjekten * Identifizierung und Priorisierung Testbedingungen * Entwurf / Priorisierung abstrakte Testfälle * Identifizierung benötigte Testdaten

Beschreibung Testfall: * Beschreibung Ausgangssituation (Beschreibung Systemzustand) * Eingaben / Aktionen * Soll-Resultat

Testrealisierung und Durchführung

- Ablauf Testfälle festlegen
- Wichtigste Testfälle zuerst
- Ausführung von Testabläufen
- Nachvollziehbarkeit / Teproduzierbarkeit
- Vergleich Ist / Soll
- Fehlerwirkungen / Abweichungen festhalten / analysieren um Grund eines Problems festzustellen

Testauswertung und Bericht

Testendkriterien: * Alle spezifzierten Testfälle ausgeführt * Testfälle best. Risikogruppe getestet * best. Zeit kein weiterer Fehler mehr gefunden wird * best. Anzahl an Fehlern gefunden ist * Strukurtest-Kriterien erfüllt sind * vereinbarter Testaufwand geleistet ist

Auswertung Testprotokolle, Entscheidung ob mehr Tests notwendig oder Anpassung ausgangskriterien, Erstellung Testabschlussbericht für STH

Psychologie des Testest

Entwicklung = konstruktiv, Test = destruktiv? :arrow_right: Testen extrem kreative / intellektuell herausfordernde Aufgabe, Entwickler eigens Programm testen? Blindheit gegenüber eigenen Fehlern (grundsätzliche Designfehlern,

Testfall kommt ihm nicht in den Sinn ..), Kein Einarbeitungsaufwand (kenn Testobjekt)

Tester: Unvoreingenommenheit (nicht sein Produkt), Einarbeitung notwendig, Test-Know-How mitgebracht

Testing durch: Entwickler, Kollegen des Entwicklers im Projekt, Personen anderer Abteilungen, Personen anderer Organisationen, Werkzeugeinsatz möglich, Aufteilung produkt- / projektabhängig, wichtig: Mischung und Ausgewogenheit unabhängigen Tests und Entwicklertests

Feedback: Mitteilung Fehlerwirkungen: neutrale, objektive, konstruktive Art und Weise, ungestörte, offene Kommunikation, Fingerspitzengefühl, Reproduzierbarkeit wichtig, eindeutige Req / präzise spez.

Grundsätze: 1. Anwesenheit Fehlerwirkungen achgewiesen, Testen kann nicht zeigen, dass keine Fehlerzustände im Testojbekt vorhanden sind. 2. Vollständiges Testen (Austesten) ist nicht möglich 3. Testen ist keien späte Phase in der SWE, so früh wie möglich 4. Fehlerzustände in Testobjekt nicht gleichmässig verteilt, oft: gehäuft 5. Tests nur wiederholen, bringt keine neuen Erkenntnisse 6. Testen ist abhängig vom Umfeld 7. System ohne Fehlerwirkungen bedeutet nicht, dass das System auch den Vorstellungen der späteren Nutzer entspricht

Kapitel 4

Betrachtung Prüfobjekt durch Mensch / Werkzeug
Reviews: Manuellle Prüfung, 1:n, kann bei allem angewendet werden
Statische Analyse: Automatisierte Prüfun mit Werkzeugen, Dokumente mit formaler Struktur (Code, UML)

Strukturierte Gruppenprüfungen

Review: Oberbegriff verschiedene Prüfverfahren: Inspektion, Walktrough, Technisches Review, Informelles Review, Peer Review - effizientes Mittel zur Sicherung der Qualität, Ende Projektphase: Review

Grundlagen

Analyse Testobjekt durch intensive Betrachtung, Ziel: Ermittlung Fehlerzustände im Dokument, Grundidee: Prävention, so früh als möglich erkennen

Reviews

Nicht-Werkzeuggestutzt: Nutzung menschliche Analyse- / Denkfähigkeit, intensives Lesen / Nachvollziehen, verschiedene Vorgehensweisen

Grundlegende Vorgehensweise **Arbeitschritte:** 1. Planung 2. Kick-Off 3. Individuelle Vorbereitung 4. Review Sitzung 5. Überarbeitung 6. Nachbereitung

Planung Mgmt. bestimmt Dokumente für Review, Schätzung und Einplanung Review, Eingangs- / Austrittskriterien für Durchführung Review, Manager wählt geeignete Person aus / Zusammenstellung Review-Team, Prüfung Kooperation Autor "review-fähiger"-Status, Festlegung Ort und Zeit Kick-Off

Kickoff Informationen an beteiligte Personen durchgeben, Überprüfung Eingangsbedingungen, Schrifltiche Einladung / Sofortiges Treffen, weiter Unterlagen für Beteiligte bereitstellen, Prüfkriterien festlegen

Individuelle Vorbereitung Gutachter mit Dokument auseinandersetzen, Fragekatalog erstellen

Sitzung Moderator kann Sitzung absagen / abbrechen (Gutachter kommt nicht, ungenügend vorbereitet), Resultat (nicht Autor) wird diskutiert, keine Angriffe auf Autor, Autor dar sich / das Resultat nicht verzeitigen, Moderator darf nicht Gutachter sein, keine allgemeinen Stilfragen, Entwicklung / Diskussion Lösung nicht Aufgabe Reviewteam, Jeder Gutachter Befunde angemessen präsentieren, Konses Gutachter laufend protokollieren, Befunde in Form von Anweisungen an Autor, Gewichtung: z.B. Kritischer Fehler, Hauptfehler, nebenfehler, Gut - Review-Team: Empfehlung Annahme (Akzeptieren ohne Änderung, mit Änderung ohne Reviewe, nicht akzeptieren), Protokoll Liste aller Befunde: Informationen zum Prüfobjekt, verwendete Dokumente, Beteiligte Personen, Rollen, Zusammenfassung / Protokoll, Ergebnis + Empfehlung

Überarbeitung Behebung Befunde (durch Autor), evtl. Rücksprache Gutachter

Nachbereitung Kontrolle Durchführung Überarbeitung, Prüfung Korrekturen / Fehlerzustände, Sammeln Metriken, Prüfung Austrittskriterien, Manager entscheidet ob er Empfehlung Gutachter folgt (alleinige Verantwortung beim Manager), evtl. weiteres Review bei nicht Akzeptanz

Rollen

- Manager
  Enscheidet über Durchführung Review, Prüfobjekte, Ressourcen für Review-Team zusammenstellen, Überprüft Review-Ziele, Entscheidet über Vorgehen

- Moderator
  Leitet Sitzung / Entscheidende Person Erfolg Review, Planungsaktivitäten, Vermittlung Standpunkte, Keine Voreingenommenheit / eigene Meinung
- Autor Ersteller Dokument (Hauptverantwortlicher), Kritik nicht als Kritik an Person auffassen
- Gutachter (Reviewer, Inspektor) max. 5, gut befundene Teile entsprechend benennen, Kennzeichnung Mängel, so gewählt, dass verschiedene Sichten / Rollen vertreten
- Protokollant

Reviewarten

- Gruppen von Reviews (Produkte / Teilprodukt, Prozesse / Projektablauf)
- Walktrough (Mittelpunkt: Sitzung, kein formaler Ablauf)
- Inspektion (Formalstes Review)
- Technische Review
- Informelles Review

Statische (werkzeugbasierte) Analyse

Keine Ausführung Testobjekt, Viele Defekte erst bei dynamischen Tests erkennbar, Überprüfung, Vermessung, Darstellung Dokument / Programm, Prüfung Einhaltung Konventionen / Standards, Datenflussanalyse, Kontrollflussanalyse, Metriken

Compiler als statisches Analysewerkzeug

Syntax, Cross Reference List, Typgerechte Verwendung, Ermittlung nicht deklarierte Var, Nicht erreichbarer Code, Über- / Unterschreitung Feldgrenzen, Prüfung Konsistenz SST

Prüfung Einhaltung Konventionen und Standards

Programmierkonventionen durch Analysatoren, Richtlinien: sollten automatisch prüfbar sein.

Datenflussanalyse

Untersuchung Verwendung Daten auf Pfaden: ur-Anomalie (undefiniert Wert Var auf Programmpfad gelesen) - du-Anomalie (Var erhält Wert, der ungültig wird, ohne dass dieser verwendet wurde), dd-Anomalie (Var erhält auf Programmpfad ein zweites Mal einen Wert, ohne dass der erste Wert verwendet wurde)

Kontrollflussanalyse

Programmstück als Kontrollflussgraph, Anweisung + Sequenz von Anweisungen = Knoten, Änderungen durch Verzweigungen und Schleifen, Visualisierung: Einfache Erfassung Anomalien (Sprünge aus Schleifen, Mehrere Ausgänge)

Ermittlung von Metriken

Number of Variables, Number of Methods, Weighted Method Complexity, Lack of Coehsion of Methods, Number of Redifined Methods, Number of childern, Depth of Inheritance tree

Zyklomatische Zahl

Gesucht (g) = e - n + 2, n: Anzahl Knoten, e: Anzahl Kanten, höher als 10: nict tolerabel, für Wartbarkeit von Bedeutung, gibt Auskunft über Testaufwand: obere Grenze Anzahl benötigte Testfälle

Kapitel 5 - Dynamischer Test

Programme: statische Beschreibung von dyn. Prozessen, Dyn. Tests prüfen die durch Interpretation resultierende Prozesse, Testobjekt wird ausgeführt - Bedingungen und Voraussetzungen für die Tests und Ziele festlegen, Testfälle spezifizieren (Eingangs / Ausgangsdaten), Testauführung festlegen

Black-box Verfahren: Keine Informationen über Inneres, Point of Observation, Steuerung Ablauf durch Auswahl Eingabetestdaten (Point of Control liegt ausserhalb Testobjekt)

Spezifikationsorientierte Verfahren ("Funktionale Testverfahren"): Modelle / Anforderungen an Testobjekt werden zur Spezifikation herangezogen, systematische Ableitung Testfälle

Erfahrungsbasierte Verfahren: Nutzen Wissen / Erfahrung von Menschen zur Ableitung Tests, Wissen von Testern, Dev, Anwendern, Betroffenen über SW, Wissen über wahrscheinliche Fehler und Verteilung

White-box Test: Testfälle aufgrund Programmstruktur (strukturorientierte Testverfahren), Informationen über Aufbau SW, Überdeckungsgrad Code für vorhandene Testfälle messen, Ableitung weiterer Testfälle, Point of Observation innerhalb Testobjekt, Point of Control kann inerhalb Testobjekt liegen

Blackbox-Test

Spezifikationsorientiert, wenig Testfälle, trotzdem qualitativ gute Tests /-abdeckung

Äquivalenzklassenbildung

Repräsentative Eingaben, Gültige Eingaben, Ungültige Eingaben
Möglichst wenig Testfälle, möglichst wirkungsvoll, Spezifikation nach Eingabegrössen und deren Gültigkeitsbereiche, pro Gültigkeitsbereich: 1 Äquivalenzklasse, bei Verdacht anderes Verhalten: Unterteilung in neue ÄK, Je Mitte ÄK: Repräsentant als Testfall wählen, Vollständigkeit: Alle Repräsentanten sind getestet, Äquivalenzklassenüberdeckung

Testfälle mit mehreren Parametern Eindeutige Kennzeichnung ÄK, Pro Param mind 2 ÄK: Eine mit gültigen Werten, eine mit ungültigen

Testfälle minimieren Kombinieren Testfälle aus allen Repräsentanten nach Häufigkeit ud Benutzungsprofile sortieren, Priorisierung Reihenfolge, Nur benutzungsrelevanten Testfälle, Sicherstellen, dass jeder Repräsentation einer ÄK mit Repräsentation anderer Klasse zur Ausführung kommt, Minimalkriterium: 1 Repräsentation jeder ÄK, Repräsentanten ungültiger ÄK nicht mit R. von gültigen ÄK kombinieren

Testendkriterium z.B. ÄK-Überdekcung

Vorteile / Nachteile **Vorteile:** Anzahl TF kleiner als bei unsystematischer Fehlersuche, Geeignet für Programme mit vielen Ein- / Ausgabebedingungen
Nachteile: Betrachtet Bedingungen für einzelne EIn- / Ausgabeparameter, Beachtung Wechselwirkungen / Abhänigkeiten aufwändig

Empfehlung Kombination mit fehlerorientiertem Verfahren

Grenzwertanalyse

Wertebereiche, Wertebereichsgrenzen, Bei Verzweigungs- / Schleifenbedingungen: oft Grenzbereiche, Fallunterscheidungen fehlerträchtig (off by one), Kombination andere Verfahren, Zusammenfallen Grenzwerte benachbarter Äk

Testendkriterium Überdeckung = (Anzahl / Gesamtzahl) * 100%

Vor- / Nachteile **Vorteile:** Grenzen ÄK häufiger Fehler als innerhalb, Grenzwertanalyse bei richtiger Anwendung einer der nützlichsten Methoden, Effiziente Kombination andere Verfahren **Nachteile:** Rezepte Auswahl Testdaten schwierig, Bestimmung relevante GW schwierig, Kreativität Findung Testdaten notwendig, oft nicht effizient genug angewendet, da zu einfach erscheinen

Zustandsbezogener Test

Komplexe (innere) Zustände und Zustandsübergänge, Ausgangsbasis: Spezifikation Programm als Zustandsgraph (endlicher Automat), Testfall aus: Ausgangszustand, Ereignis (Eingabedaten), Soll-Folge-zustand (Soll-Resultat)

Zustandsüberdeckung: Jeder Zustand muss mind. einmal erreicht werden
Zustandspaarüberdeckung: Von jedem Zustand muss in jeden möglichen Folgezustand gewechselt werden **Tranisationsüberdeckung:** Alle Zustandsübergänge müssen mindestens einmal wirksam werden

Roundtrip-Folgen Beginnend im Startzustand, Endend im Endzustand oder Zsd der bereits in dieser oder anderen Roundtrip-Folge enthalten war, Zustandsübergangsbaum

Arbeitsschritte

1. Analyse Zustandsdiagramm
2. Prüfung Vollständigkeit
3. Ableiten Übergangsbaum für Zsd-Konformanztest
4. Erweitern Übergangsbaum für Zsd-Robustheitstest
5. Generieren Botschaftssequenzen / Ergänzen der Botschaftsparameter
6. Ausführen der Tests und Überdeckungsmessung

Zustandsbezogene Testfälle Vollständig zustandsbezogener Testfall: Anfangszustand Testobjekt, Eingaben Testobjekt, Erwartete Ausgaben / Verhalten, Erwarteter Endzustand - Pro Testfall: Zustand vor Übergang, Auslösendes Ereignis, Erwartete Reaktion (durch Übergang ausgelöst), nächster erwarteter Zustand

Testendkriterien Minimalkriterien: Z-Überdeckung= (Anzahl / Gesamtzahl) * 100% Weitere Kriterien: Zustandsübergangs-Überdekckung

Entscheidungstabellentest

Bedingungen / Aktionen, Beschreibung Geschäftsregeln, Abhängig mehrere logische Eingangswerte: verschiedene Aktionen, Unterstützen Vollständigkeit Test, pro Spalte: Testfall, je Regel: mind. 1 Testfall

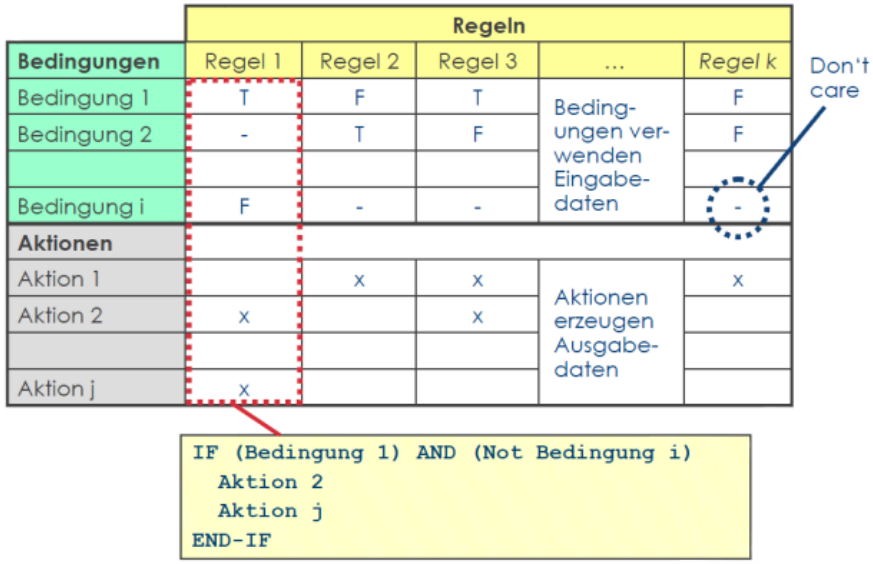


Figure 26:

Testfälle Bsp Überdeckungskriterien: Alle Bedingungen mind. einmal N und J, Alle Aktionen mind. x mal, Alle Spalten, Konkrete Testdaten aus Wertebereichen ableiten

Anwendungsfallbezogener Test Use Cases / Geschäftsszenarien beschreiben Interaktionen zwischen Akteuren - Systemen, Ergebnis / Wert für Anwender, Vorbedingungen, Hauptszenarie, Alternativszenario, Prozessabläufe

Ablauforientierte Testfälle So auswählen, dass geforderte Überdeckung Anwendungsfall erzielt wird: Normale / Alternative / Ausnahme Abläufe, Mögliche Wiederholungen innerhalb Szenarien

Diverse Blackboxtestarten Zufallstest (Zufälliger Repräsentation für TF, evtl. Berücksichtigung statistische Verteilung), Smoke-Test (prinzipielle Lauf-

fähigkeit, kein Testorakel, Versuch offensichtlicher Absturz erzeugen, “Installationstest”), Syntax-Test (Ermittlung Testfälle bei Vorliegen formale Spezifikation Syntax der Eingaben, Regeln syntaktische Beschreibung um Testfälle zu spezifizieren)

Whitebox Verfahren

Nutzung interner Informationen, Gesucht: fehleraufdeckende Stichproben der möglichen Programmabläufe / Datenverwendungen, Testverfahren zur Herleitung / Auswahl TF sowie zur Bestimmung Vollständigkeit Prüfung (Überdeckungsgrad), Daher: strukturorientierte / bezogene / strukturelle TF

Kontrollflussbasiert (Strukturtest)

Anweisungsüberdeckung: Anteil ausgeführte Anweisungen, vStatement Coverage, c0-Test
Zweigüberdeckung: Anteil ausgeführte Programmzweige (Branch coverage), Grundlage: Kontrollflussgraph, c1-Test

Anweisungsüberdeckung

Dyn., kontrollflussbasiert, mind. einmalige Ausführung aller Anweisungen des Testobjekts, Überdeckung = (Anzahl. / Gesamtzahl) * 100%

Testfall anhand Pfad durch Kontrollflussgraph bestimmt, alle Kanten durchlaufen, Berechnung: nur gezählt, ob Anweisung durchlaufen wird, wenn Deckungsgrad erreicht: Beendigung

Diskussion Schwaches Kriterium, 100% Abdeckung in Praxis nicht immer erreichbar (gewisse Ausnahmen nur schwer Testbar), Leere Kante: Überbrückt ein / mehrere Knoten, ohne Bedeutung

Entscheidungs- / Zweigüberdeckung

Kontrollflussbasiert, fordert Überdeckung aller Entscheidungen Testobjekt zu allen Fällen, jede entscheidung ist zu allen Fällen auszuwerten, TF anhand Pfad im Kontrollflussgraph, Häufigkeit Entscheidungsergebnisse spielt keine Rolle, Überdeckung = Anz. / Gesamt * 100%

Entscheidungs- vs. Zweigüberdeckung

EÜ: Betrachtet Entscheidungen, mind. einmal zu allen Möglichkeiten auswerten, Testfall auf Basis Entscheidungen
ZÜ: Alle Zweige abdecken, jeder Testfall anhand Pfad durch Graph

Test der Bedingungen

Wahrheitswerte, Atomare Teilbedingungen, Zusammengesetzte Bedingungen (Verknüpfungen atomare TB mit booleschen Operatoren), Entscheidungen: zusammengesetzte Bedingungen

EÜ: ausschliesslich Ergebnis-Wahrheitswert berücksichtigt, Problem: Bedingung aus TB: im Test strukturelle Koplexität Bedingung berücksichtigen - unterschiedliche Anforderungen, Abstufung Testintensität

ÜK: Verhältnisse zwischen erreichten / geforderten Wahrheitswerten, Verfahren Komplexität im Mittelpunkt Anstreben 100% Abdeckung

Einfache Bedingungsüberdeckung

Kontrollflussbasiert, dyn, Jede Entscheidung auf Wahr und Falsch prüfen, max. 2n Testfälle

Mehrfachbedingungsüberdeckung

Teste jede Kombination der Wahrheitswerte aller atomaren Ausdrücke, 2n Testfälle, wächst exponentiell mit Anzahl unterschiedlicher atomarer Ausdrücke, Auswertung Gesamtbedingung: beide Wahrheitswerte, auch Kriterien EB

Minimale Mehrfachbedingungsüberdeckung

Teste Gesamtausdruck einmal zu wahr und einmal zu falsch, + jede Kombination Wahrheitswerte bei denen Änderung des Wharheitswertes eines atomaren Ausdrucks den Wahrheitswert des gesamten Ausdrucks ändern kann (MM-Kombination),

Instrumentierung

Auswertung Test: Ermittlung welche programmteile ausgeführt wurden, Testobjekt an strategisch wichtgien Stellen instrumentieren, zusätzliche Anweisungen einbauen, grössere Programme: nur Werkzeuge

Erfahrungsbasierte Testfallermittlung

Erfolg syst. Test stark von Qualität Dokumente Abhängig, Q: Lesbarkeit, Testbarkeit, Vollständigkeit, Eindeutigkeit, Widerspruchsfreiheit, oft Formalisierungsgrad / Granularität nicht adäquat - Grundidee: Erfahrung / Wissen Tester bei Definition einbeziehen, Kompensation Qualität, Bsp: Error-Guessing

Error Guessing

Testentwurfsverfahren, Erfahrung / Wissen Tester, Vorhersage Fehlerzustände in Komponente,

Prüfungsfragen

Kanban

- Was bedeutet das Wort Kanban?
- Wie funktioniert Kanban?
- Was ist WiP?
- Was passiert wenn WiP zu gross oder zu klein gewählt wird?
- Wie ist ein Kanban Board aufgebaut
- Was sind die wichtigsten Unterschiede zu SCRUM?

Kapitel 5

- Was regelt ein Vorgehensmodell?
- Was ist der Unterschied zwischen einem Vorgehens- und Prozessmodell?
- Was ist der Vor- und Nachteil eines sequenziellen Modells?
- Welche Verbesserungen bringt das Wasserfallmodell gegenüber einem sequenziellen Modell?
- Erklären Sie die Begriffe: Phase, Arbeitsschritt, Aktivität
- Was ist speziell am V-modell
- Was ist ein Tailoring? Vorteile?
- Welche Ausprägungen des Prototypenmodells kennen Sie?
- Inkrementelles und Evolutionäres Modell
- Was ist die Zielsetzung des Spiralmodells