Fachhochschule Frankfurt am Main
Fachbereich 2: Informatik
SS 2008

IT Projekt-Management

Vorlesung 8:

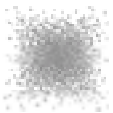
Software Qualitäts- und Defekt-Management

Dr. Erwin Hoffmann

E-Mail: it-pm@fehcom.de

Qualtätsanforderungen für Software

- Software-Projekte (als Teil von IT-Projekt) haben die Erzeugung eines Programm-Codes zur Folge
  - der (nach evtl. Installation) direkt vom Betriebssystem (OS) (wie Windows, UNIX, MacOS) ausgeführt werden kann oder
  - in Form von Source Code ausgeliefert werden, bei dem zusätzliches Kompilieren und Linken vor der Ausführung erforderlich ist.
- Bezüglich der Qualität, hat jede Software spezielle Anforderungen zu erfüllen:
  - Usability: Die Software erfüllt die (veröffentlichten) Anforderungen ('works as designed').
  - Conformance: Die Software entspricht (in Gänze) den dokumentierten funktionalen Aspekten ('works as expected').
  - Absence of bugs: Die Software arbeitet im wesentlichen fehlerfrei ('works under all circumstances').
  - Security: Die (laufende) Software hat keinen direkten oder indirekten (negativen) Einfluss auf den Sicherheitskontext des Anwenders ('works without security impact'), ausser falls dies explizit bezweckt ist.
  - Performance: Ist Leistung kein unmittelbares Ziel der Usability, sollte die Software so wenig Systemressourcen als erforderlich und mit maximaler Leistung arbeiten ('works with little system impact and high performance').

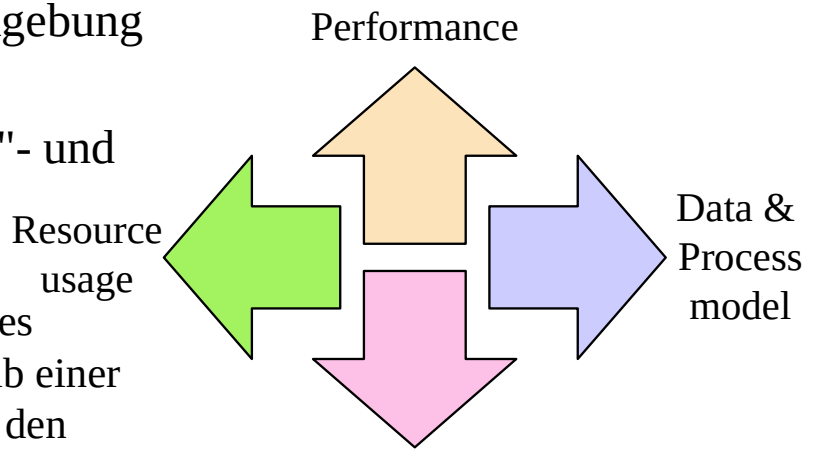


Wie Qualität bei Software erzielt werden kann

- Qualitätsanforderungen an Software können erzielt werden, wenn
  - ein qualifiziertes Software-Design vorliegt, das die Anforderungen in geeigneter Weise umsetzt.
  - durch ein geeignetes OS und ggf. zusätzlich notwendiger Middleware,
  - die Software in einer (qualitäts-) kontrollierten Umgebung mit unterstützender Infrastruktur erstellt wird und
  - unter Einsatz eines qualifizierten "Defect Tracking"- und Dokumentations-System.

Softwarequalität ist unabhängig von der Programmiersprache, d.h. der Code kann mittels eines Interpreters (Skript, Makro), als Byte-code (innerhalb einer Virtuellen Maschine), oder direkt in Binärform (mit den notwendigen OS Loader-Anweisungen) ausgeführt werden.

Selbstverständlich hat die Wahl der Programmiersprache entscheidenden Einfluss auf die Performanz (und auch der Sicherheit).



ISO 9000 Software Qualitäts-Management

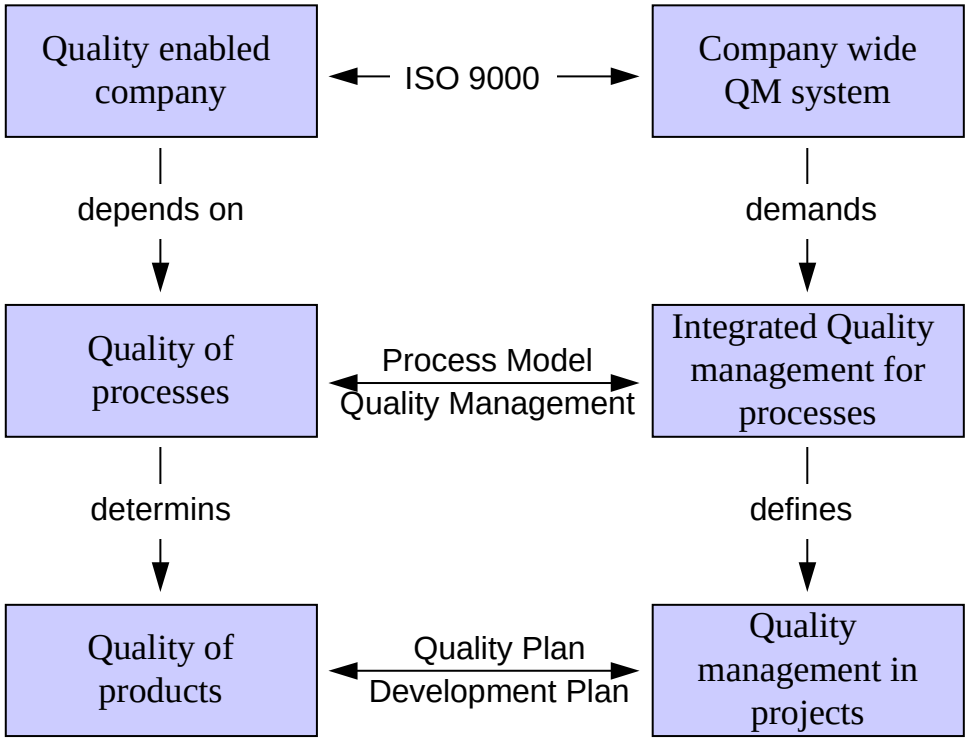
- Die Qualitätskette

  Design/Planning -> Coding/Development -> Control/Improvement

  ist in der Praxis durch Budget-Beschränkung und "time-to-market" Bedingungen eingeschränkt.
  - Nicht durchgeführt Schritte resultieren in einem Qualitätsverlust für das Softwareprodukt.
- Die Norm ISO 9000-1 stellt drei Fragen hinsichtlich der Softwarequalität für 'Informations Systeme" IS im Kapitel A.3 in den Raum:
  - [Management Unterstützung] Sind die verantwortlichen IT Manager für das IS benannt und können diese notwendige Änderungen beauftragen und genehmigen?
  - [Qualitäts Management System] Wurden die Anforderungen an das IS nachvollziehbar dokumentiert?
  - [Audits] Werden die Anforderungen durch Verfahren ergänzt, wie deren konkreter Einsatz überprüft werden kann?

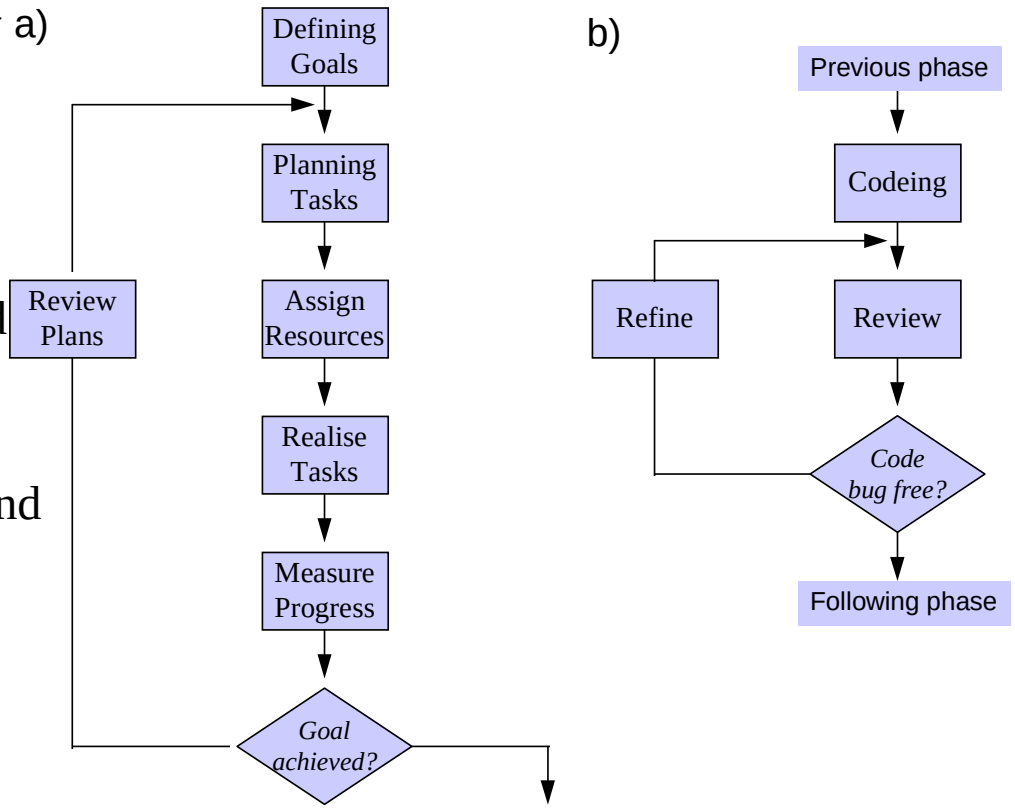
Qualitäts Management Systeme

- ISO 9000 sieht in der Unterstützung des obersten Management als wesentlichen Faktor für ein erfolgreiches Qualitätsmanagement.
- Das eigentliche Qualitäts Management System QMS kann von zwei Seiten betrachtet werden:
  - (a) aus der Sicht der Dokumentation
  - (b) von der operativen Perspektive.



Ergänzende ISO 9000 Standards

- Der Umfang eines QMS hängt von der Branche ab, für das das QMS eingesetzt werden soll und ist in folgenden ISO Standards beschrieben:
  - ISO 9001: Standards für das Qualitätsmanagement im Entwurf, Entwicklung, Produktion, Montage und im Dienstleistungsbereich.
  - ISO 9002: Standards für das Qualitätsmanagement für Produktion und Montage.
  - ISO 9003: Standards für das Qualitätsmanagement für Abnahmeprüfungen und -kontrolle.
- Für die Softwareentwicklung ist die Norm ISO 9001 massgeblich.



ISO 10013 und QM Dokumente

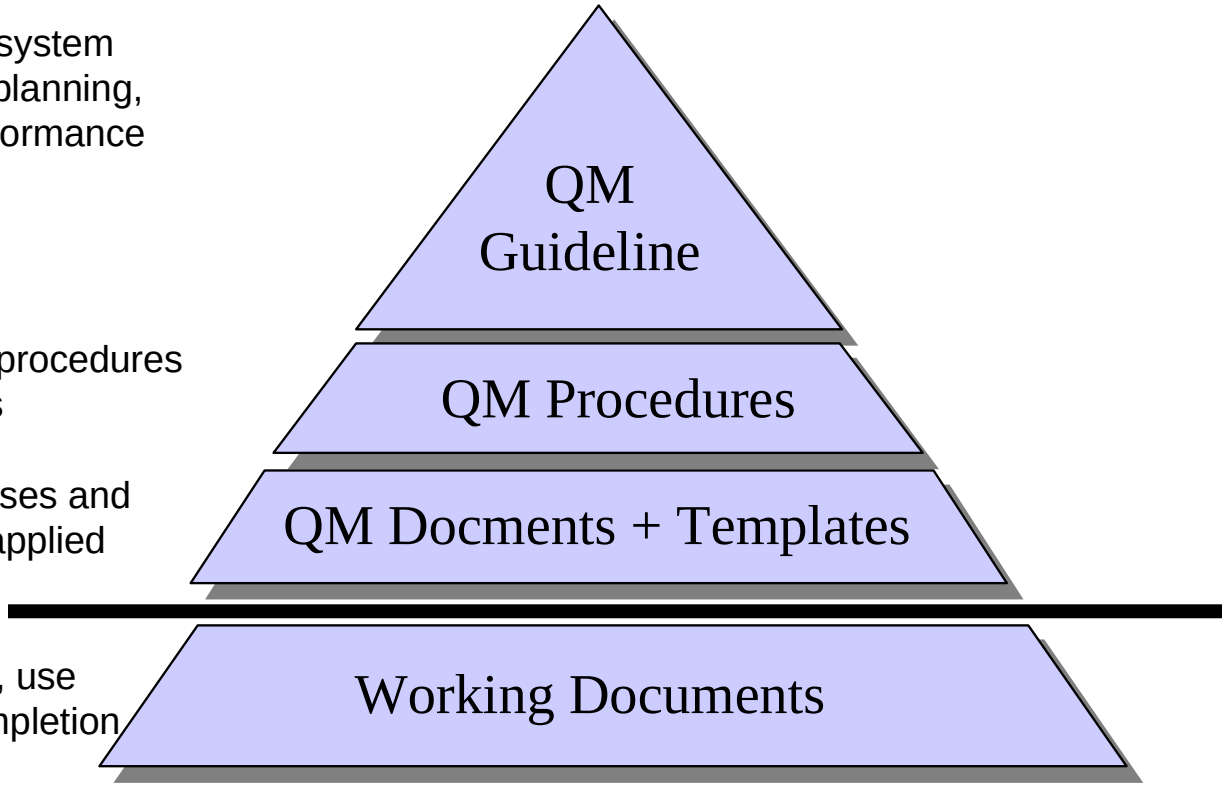
- Die Norm ISO 10013 macht Vorgaben über den Aufbau des Qualitäts Management Systems (QMS) und detaillierte die Rolle der notwendigen Dokumente zur Umsetzung der Qualitätsvorgaben.

Description of QM system according the QM planning, QM goals and conformance with ISO 9000 ff.

Description of QM procedures and responsibilities

Detail list of test cases and how they shall be applied

Design documents, use cases, letter of completion

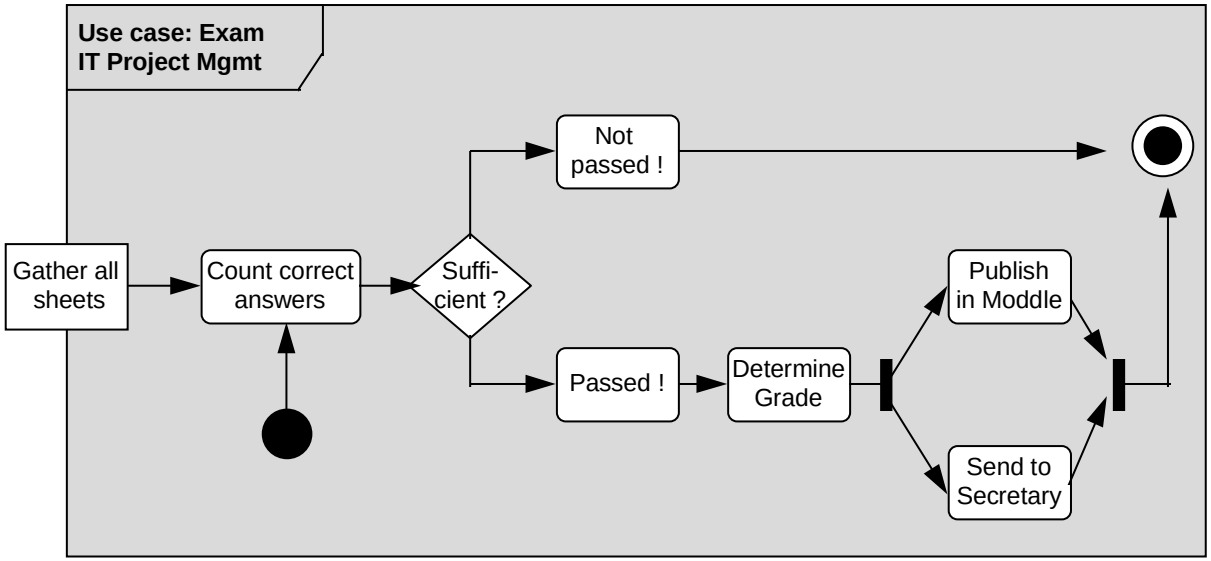


Audits im Qualitäts Management

- Der Aufbau eines QMS entsprechend ISO 9001 für die Software-Entwicklung beinhaltet die Einhaltung der zusätzlichen ISO Standards 9001, 9002 und 9003.
- Neben der Einführung und des Einsatzes eines QMS zur Sicherstellung der des Qualitäts Management für die Produktion, verlangt ISO 9000-1 zwingend die Kontrolle mittels Audits.
- Audits sind Teil der ISO 9000 Zertifizierung und beinhalten:
  - Inhaltliche Übereinstimmung des QMS Handbuchs mit den ISO Normen.
  - Betriebliche Umsetzung des QMS Handbuchs.
  - Überprüfung der Produktqualität.
  - Feststellung der Eignung des Projektteams hinsichtlich des Managements der QM Prozesse.

Anwendungsfälle (Use Cases)

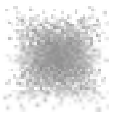
- Die heute Softwareentwicklung macht sich in der Design-Phase der Produkte oder ihrer Komponenten Anwendungsfälle *Use Case* zu nutze.
  - Der Use Case ist eine funktionale Beschreibung der Art und Weise, wie das Produkt bzw. die Komponente arbeiten soll.
- Auf dem Hintergrund der Object Orientierten (OO) Programmierung hat sich eine formale Beschreibung der Use Cases entwickelt, die heute in Form der standardisierten *Unified Model Language UML* vorliegt.
  - UML Use Cases Diagramme sind ein de-facto Standard, um Abhängigkeiten darzustellen.



Anwendungs- und Testfälle

- Zusätzlich der Diagrammform der Use Cases, können diese auch in Form von Aktivitätstabelle unter Zuhilfenahme von Templates darstellt werden.
- Im Rahmen der Software-Entwicklung für jede Work Unit (PMBoK) durch einen zugehörigen Use Case beschrieben.
  - Der Use Case kann somit als fester Bestandteil der des Software Designs und der Entwicklung beschrieben werden.
- Als Bestandteil des Qualitäts Plan (QP) entwickelt die QA Abteilung eine Sammlung von Testfällen (*Test Case).*
  - Hierbei werden die funktionalen Abhängigkeiten der Komponenten benutzt, um in Übereinstimmung mit den Use Cases Testfälle zu erstellen.
  - Alle wesentlichen Tests sind in den Test Cases zu beschreiben, die wiederum von den Use Cases abgeleitet sind.
  - Komplexe Software Produkte benötigen umfangreiche Use und Test Cases, mit hohem Aufwand in der Erstellung und späteren Tests.

    Hieraus wird generell abgeleitet, dass die Qualität von Software und die Komplexität gegensätzliche Attribute sind.



Inhalte von Testfällen

- Nur gut-dokumentierte Software kann sinnvollerweise hinsichtlich ihrer Qualität bestimmt werden
  - Wenig oder schlecht dokumentierte Software lässt sich kaum sinnvoll testen und führt sowohl zu einem frustrierten Anwender wie Tester. Der Anwender kann in diesem Fall nur Teile der Software benutzen, obwohl er für die ganze bezahlt hat.
- Jeder Test Case sollte die folgernde Schritte beinhalten (abgeleitet vom Use Case):
  - Default-Verhalten: Liefert die Softwarekomponente die erwarteten Resultate bei den Default-Einstellungen und Eingabe-Werten?
  - Konditionelles Verhalten: Falls die Komponente zusätzliche Konfigurationsmöglichkeiten in Form von 'Schaltern'/'Optionen' bzw. 'Argumente'/'Parametern' bietet, arbeiten diese so wie dokumentiert?
  - Extrem-Verhalten: Wie verhält sich die Komponente falls die Eingabewerte ausserhalb der Spezifikation sind?

Testfälle und Dokumentation

- Jeder qualifizierte Testfall dokumentiert die erwarteten und getesteten Ergebnisse im Hinblick auf die funktionellen Komponenten der Software und beschreibt ausführlich die vorgenommen Testverfahren.
  - Diese Abhängigkeiten unterstreichen die Notwendigkeit eines qualifizierten Software-Designs und der dazu gehörigen Dokumentation.
  - Im speziellen gehört zu einem Produkt eine ausführliche Liste der Return-Codes und Fehlerausgaben
- Da die Erstellung der Dokumentation zeitraubend und aufwändig ist, werden bei der Entwicklung zwei Wege eingeschlagen, dies für den Entwickler zu vereinfachen:
  - In-Line Dokumentation mit dem Code: Hierbei wird im Quell-Code mit einer speziellen Mark-Up Sprache dokumentiert, die später einfach ausgelesen und zusammengefasst werden kann.

    Die Programmiersprache PERL führt das beispielhaft im Rahmen der "Plain Old Documentation" POD durch.
  - Dokumentation im IDE: Einige IDE's, so z.B. Eclipse, führen kontext-sensitive Information mittels eines Plug-in zusammen.

    Hierdurch werden die vorgenommenen Änderungen gleich mit dokumentiert und in das Versions-Kontrollsystem übernommen.

Defect (Fehler) Management

- Was ist ein Defect ?

  Ein Defect ist die Abweichung bei einer Softwarekomponente zwischen dem dokumentieren (Soll) und dem festgestellten (Ist) Verhalten.

- Eigenschaften eines Defects:
  - Quelle

    Die Ursache eines Defects kann bestimmt sein durch:

    Ein Programmierfehler, üblicherweise als *Bug* bezeichnet.

    Ein Fehler, der aus der aktuellen Laufzeit-Umgebung herrührt und zu einem abweichenden Verhalten führt; auch als *Flaw* bezeichnet.

    Ein Designfehler; der Programmierer hat den Code korrekt umgesetzt, aber das Design war fehlerhaft für die Aufgabenstellung.

    Ein Dokumentationsfehler; die Software-Komponente verhält sich nicht entsprechend er vorliegenden Dokumentation.

Die Priorität eines Defects

- Prioritäten

  Prioritäten regeln die 'Wertigkeit' im Defect-Fixing Prozess:

  Undefiniert (-1): Showstopper: Der Defect muss braucht einen sofortigen Fix, da dieser Fehler Tests an weiteren Software-Komponenten verhindert.

  Eins (1): Der Fehler ist so schwerwiegend, dass die Software-Komponente so nicht eingesetzt werden kann.

  Zwei (2): Ein ernster Fehler liegt vor, die erhebliche Funktionsdefizite für die Software-Komponente zur Folge hat.

  Drei (3): Es liegt ein Fehler vor, der aber (durch ein Work-Around) kompensiert werden kann.

  Vier (4): Ein weniger-relevanter Fehler wurde festgestellt, der die bei üblichem Einsatz der Software-Komponente nicht von Bedeutung ist.

  Fünf (5): Es wurden Fehler festgestellt, die sich aber durch ungenügende/fehlerhaft Dokumentation begründet sind.

Kategorien und "Besitzer" eines Defects

- Kategorie

  Ein Defect wird einer Komponenten-Kategorie zugeordnet.
  Häufig beinhaltete ein Software Modul mehrere Komponenten:

  Benutzer-Schnittstelle (Input / Output)

  Middleware, Transportschicht, Schnittstellen/APIs

  Back-end (z.B. Datenbank)

  Im Rahmen des Software-Designs wird jede Komponente einer Kategorie zugeteilt, um speziell auch die Defekt-Zuweisung zu vereinfachen.

- Besitzer

  Zur Fehlerbeseitigung wird jeder Fehler einem *Besitzer* zugewiesen.

  Typischerweise wird jede Software-Komponente von einer bestimmten Person, einem Team, einer Organisation oder von einer Firma entwickelt.

  Daher besteht üblicherweise ein direkter Zusammenhang zwischen *Besitzer* des Defekts und *Entwickler* der Software Komponente.

  Ein *Revision Control System* (RCS) oder aber auch ein *Integrated Development Environment* (IDE) fügt automatisch die *Autoren* Information hinzu.

Versionen, Projekte, Status und Due-Date von Defecten

- Version

  Der *Versionierung* der Software Komponente kann entweder explizit vom Entwickler oder aber automatisch vom RCS/IDE erfolgen.

  Unabhängig von der Version der Komponente kann die Identifikation des Release erfolgen, was entweder über einen numerischen oder aussagekräftigen Namen im Rahmen des Projektes erfolgt.

- Projekt

  Die Entwicklung grösserer Software-Komponenten findet normalerweise im Rahmen eines *Projektes* statt.

  Projekte können in *Unterprojekte* unterteilt werden.

  Die Aufteilung hängt vom Projektstrukturplan PSP (entsprechend PMBoK) ab und kann frei gewählt werden, z.B. als Marketing Name.

- Status

  Der Status beschreibt die Einordnung des Defekts im Verarbeitungszyklus (z.B. New, Offen, Geschlossen)

- Due-Date

  Der *Due-Date* ist der voraussichtliche Termin, wann der Defekt geschlossen sein wird.

Defect Tracking Systeme

- Essentieller Bestandteil der Software Entwicklung (und Teil des QM Systems) ist das "Bug-tracking" mittels einer *Defect Management Software*.
  - Eines der gebräuchlichsten Systeme ist *Bugzilla* (public domain).
  - Üblicherweise nutzen Defect Management System als Back-End eine Datenbank und ein grafisches (Web-) Front-End.
- Diese Systeme erlauben das Aufsetzen eines logischen Defect *Lifecycle*.
  - Wie der Defekt-Lebenszyklus abgebildet wird, kann bei einigen Produkten definiert werden; bei anderen ist er fester Bestandteil der Lösung.
- Firmen sind normalerweise darin interessiert, sowohl für das *Defect Management* als auch für das sog. *Incident und Problem Management* die gleiche Software zu nutzen, da die Lifecycle Idee die gleiche ist.
  - Solche *Trouble Ticket Systeme* erlauben mitunter eine sog. Klassen (Class) Definition für den zu berichtenden Fehler oder Vorfall:

    Defekt - Software Entwicklung, Bug tracking.

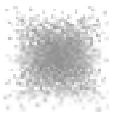
    Incident - *Vorfall*, der eine Abweichung eines Prozesses vom vordefinierten Schema beschreibt (*erratically behaviour*); ggf. einmalig oder wiederholend.

    Problem - Satz zumammengehöriger Vorfälle mit teilweise bekannter Ursache.

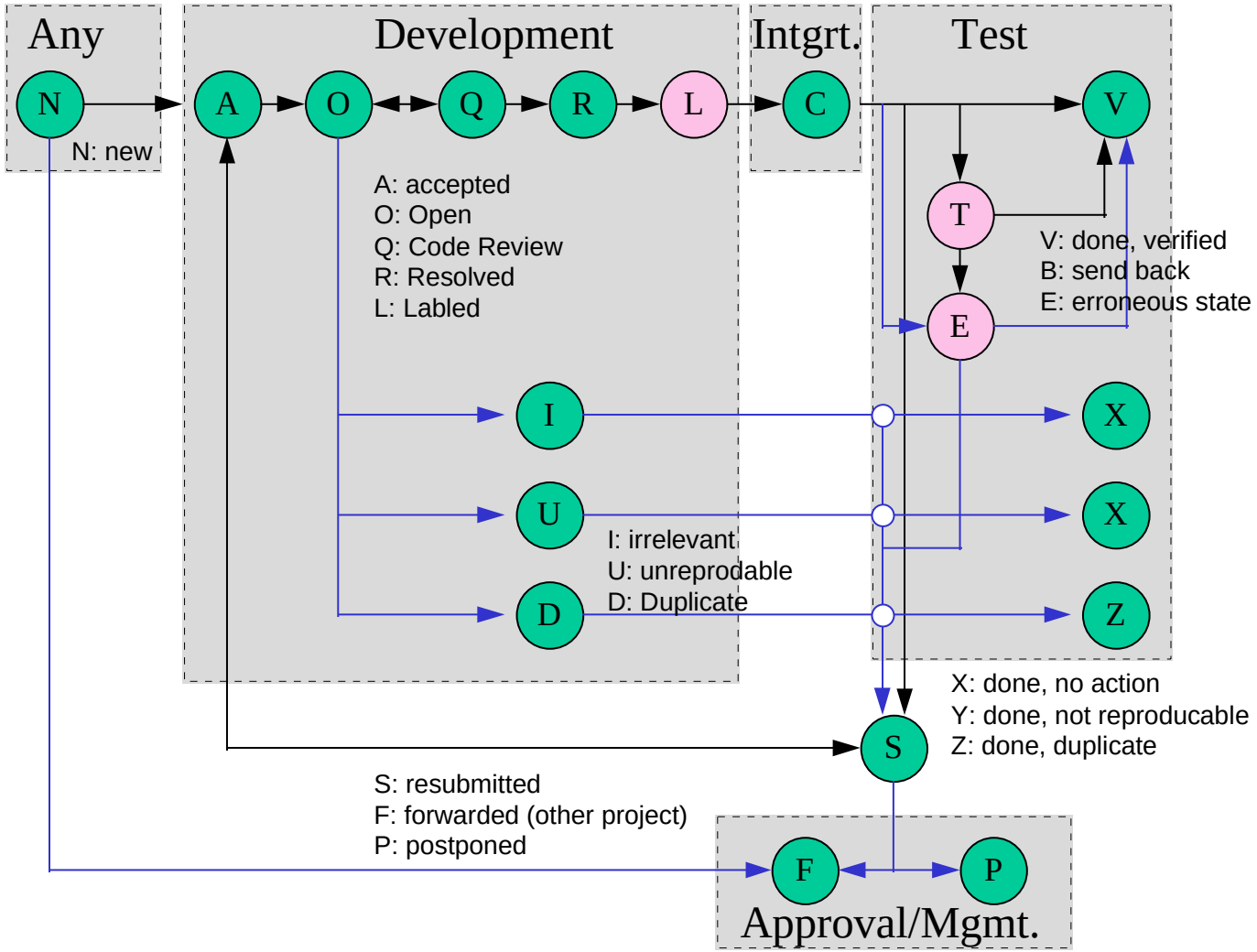
Bugzilla's Defect und Fix Stati

Defect State	Bedeutung
Unconfirmed	Der Defekt wurde eingestellt, konnte nicht bestätigt werden
New	Der Defekt wurde eingestellt, wurde aber noch nicht zugewiesen
Assigned	Der Defekt wurde zum verantwortlichen Entwickler zugewiesen
Reopened	Der Defekt wurde geschlossen, aber benötigt aber zusätzliche Behandlung
Resolved	Der Defekt wurde gelöst und die Lösung implementiert
Verified	Der Defekt wurde gelöst und die Lösung wurde erfolgreich überprüft
Closed	Der Defekt wurde gelöst und die Lösung ins aktuelle Release übernommen

Fix State	Bedeutung
Fixed	Ein Bug-Fix wurde angewandt
Invalid	Der eingestellte Defekt war keiner; ergänzende Information wird benötigt
Wontfix	Der Defekt kann aktuell nicht gelöst werden
Later	Die Lösung des Defekts wird (auf das nächste Release) verschoben
Remind	Es wird keine Lösung bereit gestellt; der Defekt wird auf Wiedervorlage gestellt
Duplicate	Der Defekt ist ein Duplikat eines bereits bekannten
Worksforme	Der Defekt kann nicht reproduziert werden

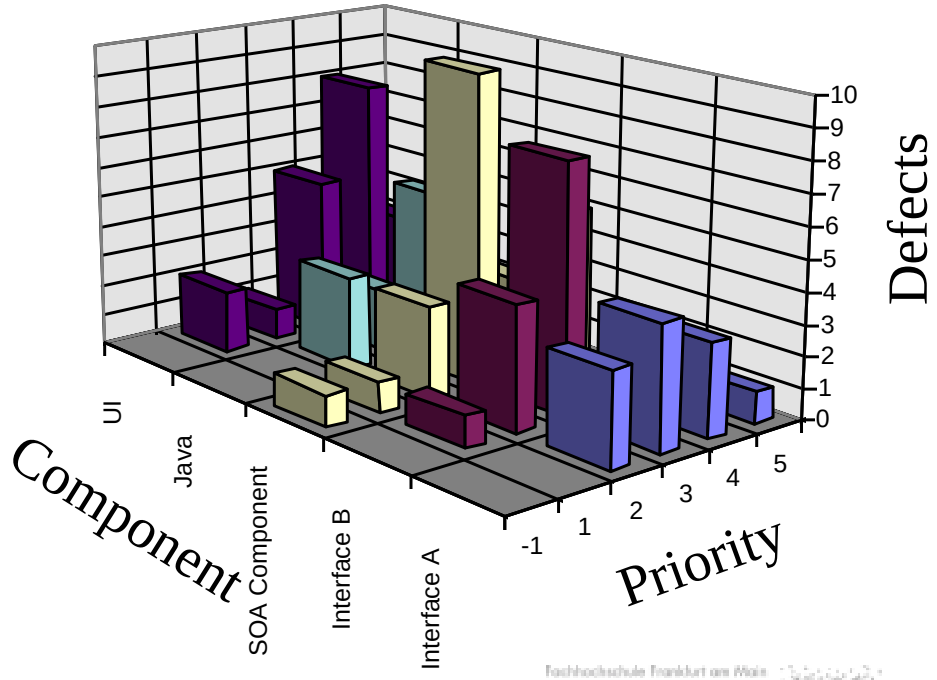


Komplexe Übergange beim Defekt-Management



Aufgaben des Qualitäts-Managers

- Qualitäts Managements bzw. Quality Assignment (QA) Reports sind regelmässig zu erstellen (jede Woche) und beinhaltet folgende Darstellungen:
  - eine *statistische Auswertung*, die die Anzahl der Defekte pro Projekt unter Bezugnahme auf Priorität und Komponente darstellt
  - eine *Liste der Defekte*, ausgehend von den wichtigsten mit kurzer Darstellung des aktuellen (Bearbeitungs) Standes und der geplanten Lösung und des Zeitplan.
- Die statistischen Reports werden häufig als "Lego" Diagramme dargestellt, wobei die Anzahl der Defekt nach Priorität und die Komponente dargestellt werden.
- Um eine qualitative Aussage zu erzielen, müssen folgende Fragen beantwortet werden:
  - Wie viele Defekt von Priorität X wurden seit dem letzten Bericht behoben ?
  - Wie viele Defekt von Priorität X wurden seitdem neu geöffnet ?



Qualitäts Reports

- Zentrale Aufgabe des Qualitäts Managers besteht darin, die einzelnen Defekte zu bewerten und die wichtigen in den Vordergrund zu rücken.
- Ein Qualitäts Report muss die folgenden Defect Attribute berücksichtigen:
  - *Showstopper*: Der Defekt behindert die weiteren Testläufe.
  - *Kritikalität*: Der Defekt ist entscheidend für die Nutzung einer zentralen Anforderung des Produktes und somit für das Gesamtprojekt relevant.
  - *Due-Date*: Die Lösung für diesen Defekt wurde wiederholt verschoben und der Defekt behindert somit den Fortlauf der übrigen Entwicklung.

    Zusammen mit den Entwicklungs-Chef, wird der Qualitäts Manager die Defekte neu bewerten und diese entweder re-priorisieren oder diese einer besonderen Untersuchung zuweisen.

    Der Entwicklungs-Chef wird ggf. die Defekte hinsichtlich der Ownernship neu zuweisen und die Entwicklungs-Aufwände neu bewerten bzw. einplanen.

Unerkannte Defekte

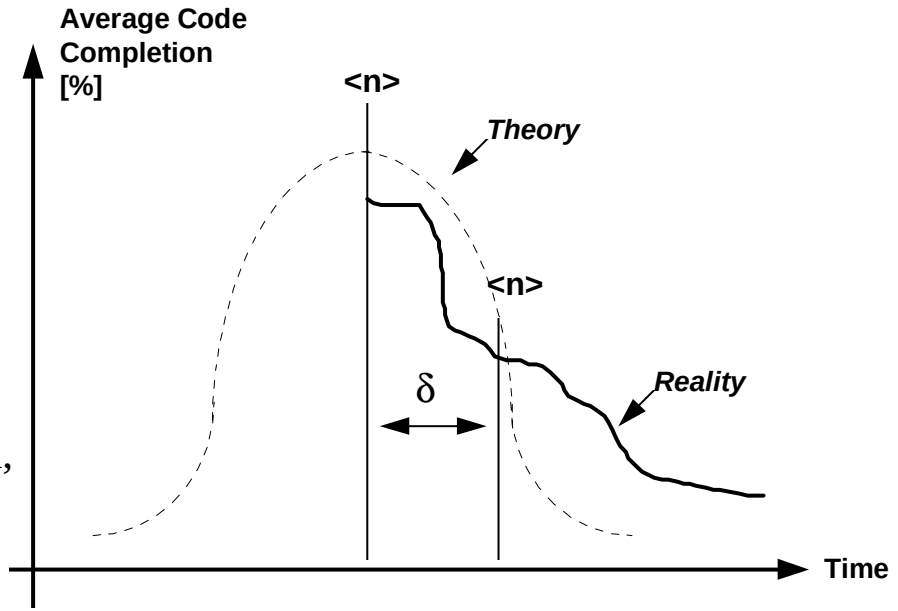
- Unter Einsatz eines geeigneten Quality Management Systems und unter der Annahme, dass die Defekte in kontrollierter Weise bearbeitet werden, besitzen wir folgende QA Informationen.
  - Verteilung der Anzahl *neuer Defekte* unter den Kriterien Priorität und Komponente.
  - Verteilung der Anzahl *gelöster Defekte* in Abhängigkeit von Priorität und Komponente.
- Was wir nicht kennen ist:
  - Anzahl und Verteilung der bislang *nicht-erkannten* Defekt.
- Üblicherweise ist für ein 'Release' eine Reihe weniger-schwerwiegender Fehler durchaus akzeptabel.
  - Dies verlangt jedoch eine gewisse Abschätzung über die im Produkt potentiell auftretenden Fehler unter Einschluss der bekannten.

Entwicklungszyklus des Programmieren

- Der Schlüssle für die Abschätzung unbekannter Defekt liegt in der Hinzuziehung der Entwicklungs-Informationen:
  - Phase 1: Am Beginn der Entwicklung ist die Code-Basis sehr klein und schwerwiegende Defekt (Priorität 1) sind nicht ungewöhnlich.
  - Phase 2: Das Entwicklerteam wird sowohl mit den Werkzeugen als auch mit seinen Aufgaben vertraut, Hilfsroutinen werden erstellt und die Entwicklung optimiert. Die Code-Basis wächst im Verlauf der Zeit und mit der Anzahl der Entwickler.
    Der Qualitäts-Manager stellt fest, dass neue Defekte eingestellt, aber umgehend gelöst werden.
  - Phase 3: Wird das Ende der Entwicklung erreicht, wird häufig festgestellt, das einige Teile noch fehlen, die nachentwickelt werden müssen.

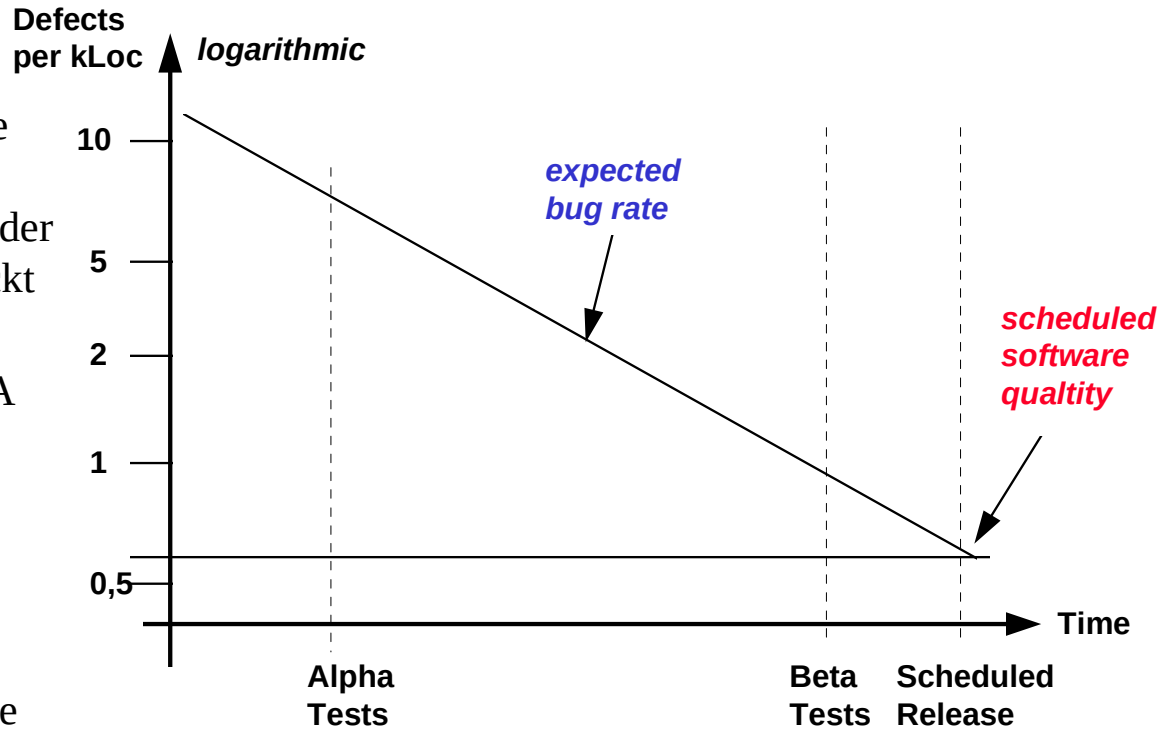
    Die fehlenden Komponente werden schnell und in Eile erstellt.

    Die Code-Basis steigt sprunghaft an.



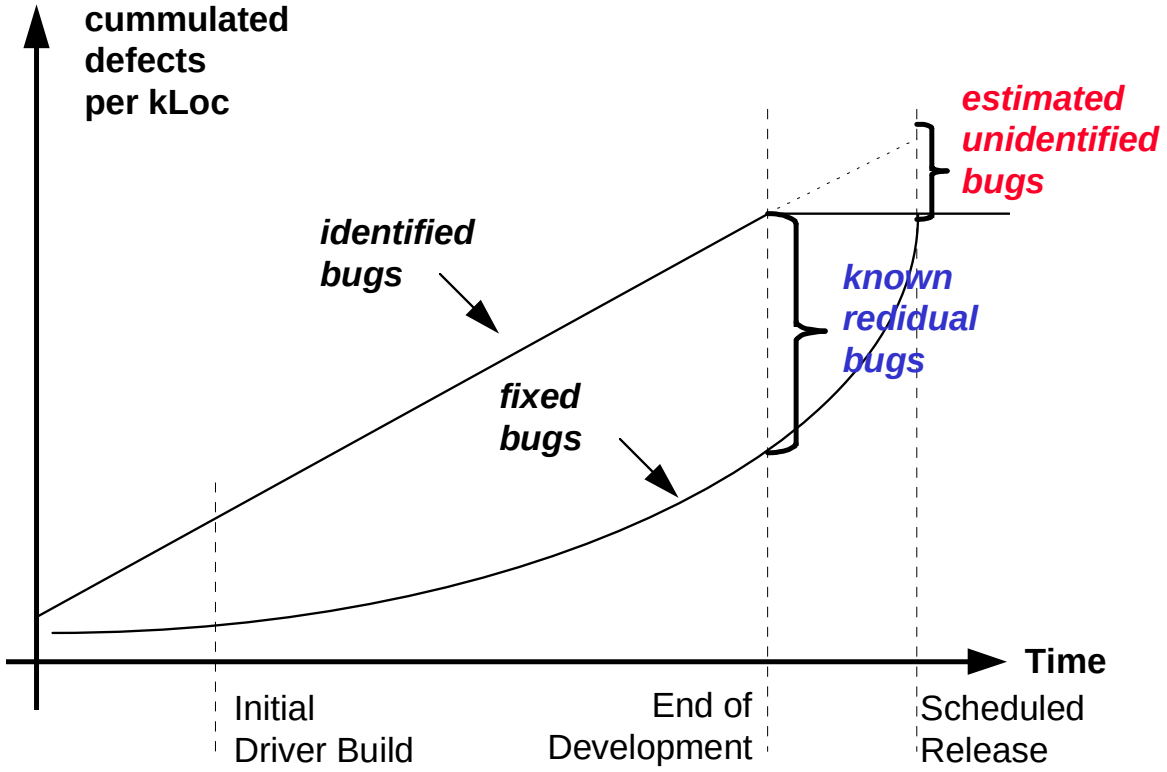
Annehmbare Fehlerrate (Defect Rate)

- Zur Software Qualitätskontrolle muss der Qualitäts Manager die Anzahl der Defekt mit dem eingecheckten Source-Code korrelieren.
- Typischerweise wird davon ausgegangen, dass die Anzahl der Defekt mit der Grösse des eingecheckten Codes korreliert, was üblicherweise als 'Defekte/kLoc' (Anzahl der Defekte pro 1000 Zeilen Code) ausgedrückt wird.
  - Auch mit den fortgeschrittensten QA Methode muss davon ausgegangen werden, dass eine gewisse Zahl von Bugs im Code vorliegt und auch akzeptabel ist.
  - Der Qualitäts Management Plan (QMP) hat auszuweisen, wie sich die akzeptablen Bugs verteilen und welche Schwellwerte einzuhalten sind.



Abschätzung unerkannter Defects

- Aufgabe des Qualitäts Mangers ist, die Anzahl der erkannten und gefixten Fehler pro kLoc zu ermitteln und dies auf einer Zeitachse darzustellen.
- Eine qualifizierte (nicht notwendigerweise lineare) Extrapolation liefert eine Abschätzung über die Anzahl der potentiell offenen Fehler.



Entwicklungs- und Test Umgebungen

- Zur Unterstützung des Software Qualitäts Managementzyklus und als Teil des QMP werden häufig unterschiedliche Umgebungen auf- und eingesetzt:

  Sofern gemeinsam notwendige Ressourcen, einschliessliches des Betriebssystems, der Middleware (Datenbanken) und andere notwendige Applikationen für eine Aufgabe eingesetzt und konfiguriert werden, sprechen wir von einem Environment (Umgebung).

- Üblicherweise werden die folgenden Umgebungen aufgesetzt und genutzt:
  - [DEV] Entwicklungs Umgebungen: Hier sind Compiler, Linker und Makefile Utilities eingeschlossen, als auch *Integrated Development Environment* (IDS) sowie *Source Code Control* Systeme.
    Zumindest eine Entwicklungs-Umgebung wird zur Unterstützung des *Release Management* eingesetzt.
  - [AT] User Acceptance Test Umgebungen: Diese (System) Testumgebungen entsprechen der späteren Produktionsumgebung in gewissem Umfang; jedoch auf kleinerem Niveau. Hier sind spezifische Werkzeuge zum Testen der Software für das Qualitäts Management zu finden.
    Diese Umgebungen sind massgeblich für die Annahme der Software-Komponente.
  - [INT] Integration Test Umgebung: Falls third-party System in den Test eingeschlossen werden müssen dient eine spezielle UAT als Integrations Test Umgebung.
  - [REF] Referenz Umgebung: Dies Umgebung ist vergleichbar der späteren Produktion aufgesetzt Hierauf können Performance- und Regressionstests durchgeführt werden.
    Manchmal fungiert die Umgebungen auch als Backup- bzw. Fail-Over Umgebungen für die Produktion.
  - [PROD] Produktions-Umgebung: Die vorgesehene Produktions-Umgebung, mit ggf. zusätzlichem Hardening, was auch auf die Referenz-Umgebung zu übertragen ist.