Software Quality is like beauty *difficult to messure*

Make software quality visible with a signet

Prof. Radomski

HfTL

University of Applied Science

Courses:

- Applied Computer Science
- Business Informatics
- Computer Science and Telecommunications
- Information and Communication Technology
- Information and Communication Technology (Master)

Full-time programs, co-operative programs, extra-occupational programs, continuing education

HfTL

Software Quality is like beauty

- **Name**: Profn. Dr.-Ing. Sabine Radomski
- **Vocation**: Information- and Communication Technology
- **Research topics**: Software Quality, Cloud Computing, IT Security
- Courses:
  - Software Engineering
  - Software Management
  - Distributed Systems
- Patent 2017
- Award: Professor des Jahres 2015

HfTL

Agenda

- Software Quality
- How good is software quality today?
- Digitalization
- Objectives of the quality label
- Open topics
- Conclusion

Software Quality

SW- Quality is like beauty

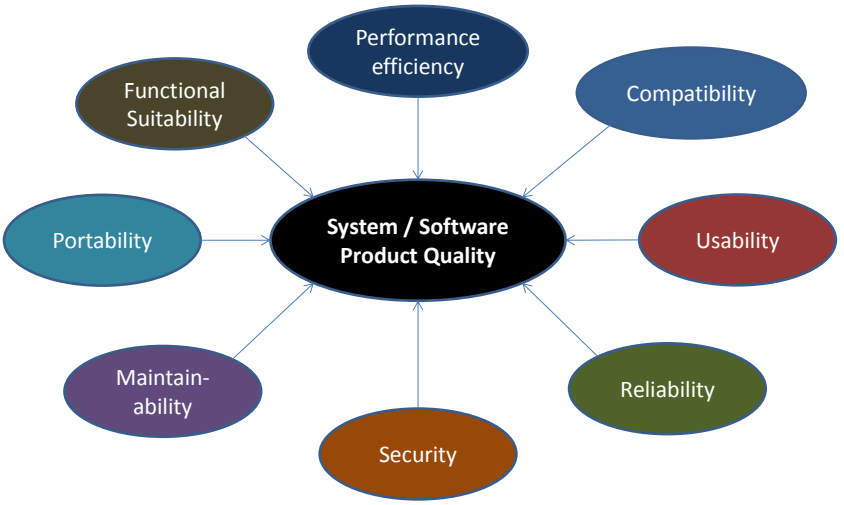
- Depend – in the eye of the visitor: everybody use SW in different way
- Depend on product
- Depend on process
- SW works longer than the developer think!

Software Quality

Software

- is immaterial and inhomogeneous
- Software are developed like a work of art –unique
- Don't abrade by using
- Requirements are changing all the time

What is Software quality? ISO 25.010; Product Quality

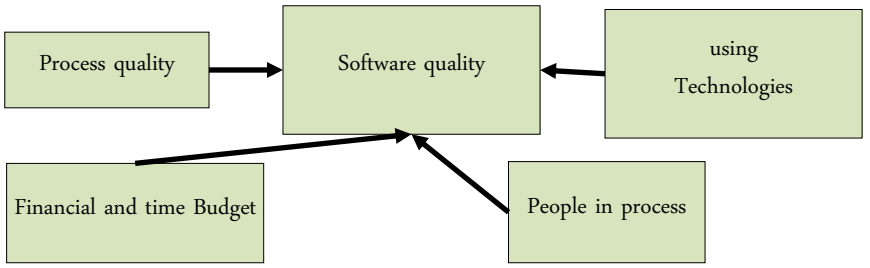


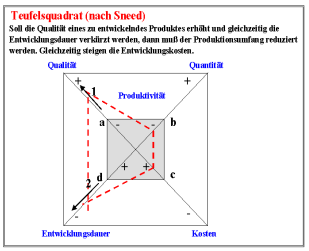
Similarities of Phoenix and Software Engineering

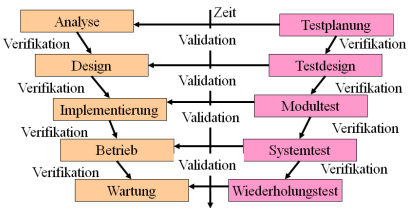
Phoenix

- Barren Environment: desert – changing of requirements
- Not enough water – not enough budget
- Team is fighting every time - misunderstandig of stakeholder
- Every time backlash by storms – changings in environment
- **but: the Team has one target: come out of the desert!**

How develop Software Quality?







9

Vulnerabilities in Software

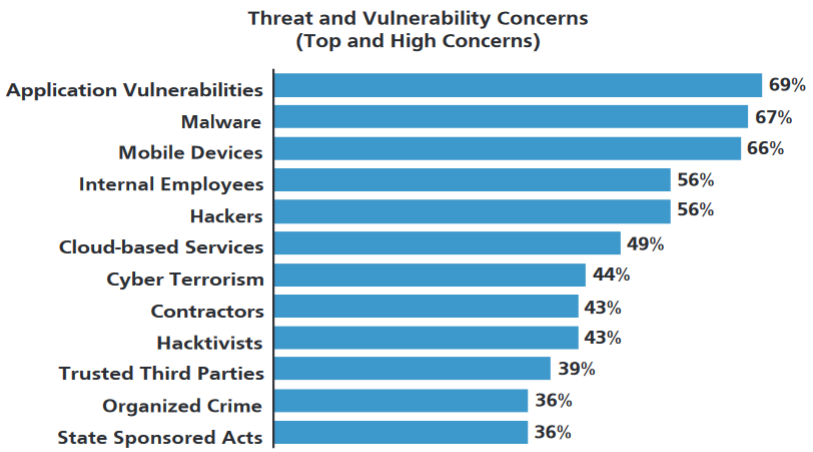
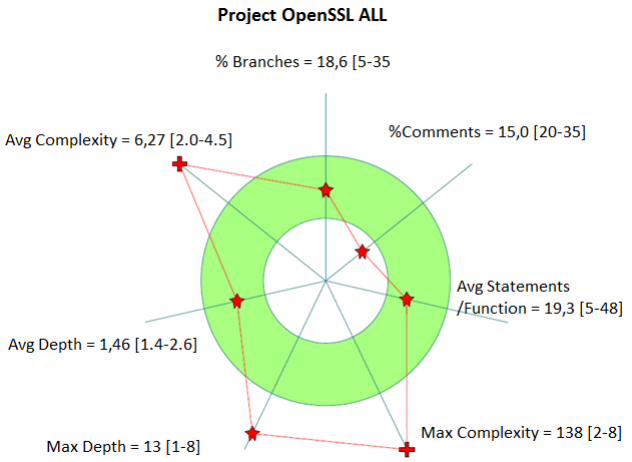


Abbildung 1: Gemäß einer Untersuchung von Frost & Sullivan, $(ISC)^2$ und Booz, Allen, Hamilton geht von Sicherheitslücken in Anwendungssoftware die stärkste Bedrohung aus (Quelle: [FIB13])

Software Quality in example OpenSSL – messured 2015



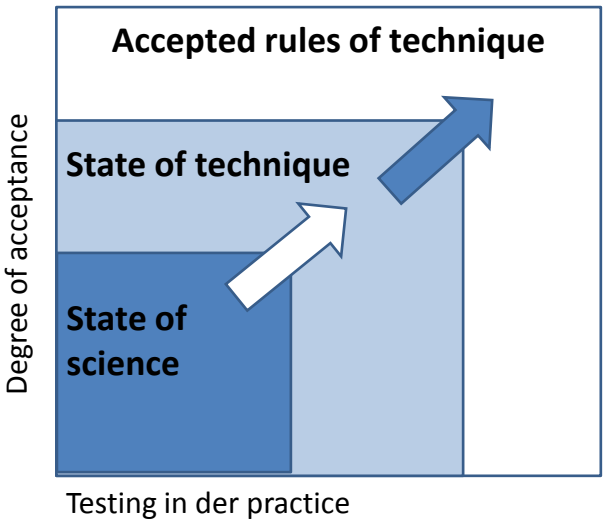
Aktuelle Software Qualität z.B. OpenSSL – (Heartbleed 2014) gemessen 2015

OpenSSL C
- 422827 LoC
- 181926 Statements
- 7020 Functions
- 21 Branches
- 138 max. Complexity

t1_lib_new.c
- 4527 LoC
- 2361 Statements
- 7 Functions
- 36 Branches
- 31 max. Complexity

d1_both_new.c
- 1529 LoC
- 719 Statements
- 29 Functions
- 21 Branches
- 43 max. Complexity (o.k.=10)

State of the art



Digitalization

- **Big Data:** Volume, Velocity, Value, Veracity, Variety
- **Industry 4.0:** Individualization of products in flexibility production
- **Digitalization of business processes**
- **Internet of things**
- **Der transparent human**
- **Das intelligent home**
- **Energy management**

The other side

Goal of quality signet

- Focus is Software Quality (ISO 25.000 ff)
- Target group are small and middle enterprises
- Focus points of quality signet are:
  - sustainability
    - Maintainability and reusability of Software
  - Transparence
    - Make quality visible in product und process
  - Security
    - Confidence for Software
- Mature / responsible user
  - Free Software will be payed expensive

Goals of project

Discussion points		Fachthemen
Definition of Quality metrics	Make Quality visible	Software Quality: Standards and metrics
Requirements of Stakeholder	Technical debt	Methods of Quality messurement
Recommendation of producer	Environment	Security as quality dimension
		Methods for visibility of Quality
		Project management for the signet
		Sustainability of Validation of Quality
		Classification of Software
		Automation tests

17

Signet for tested security in Software

- „secure Software degree the trust of user. Efficient using of resources and avoid disasters in critical infrastructures."

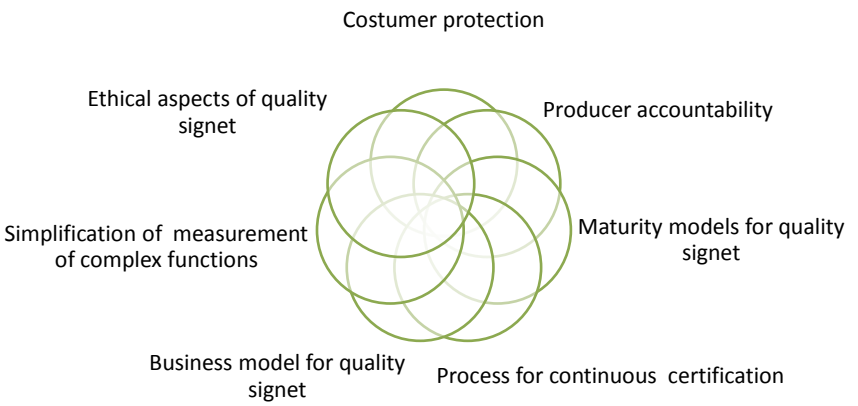
- **„Visibility of Software Quality**
- **Visibility of Software Security**
- **Make visible the observance of Software development rules**

What happen now?

Actual events connected to software Quality

- Bundesministerium develop 2016 a Cyber security strategy for Germany
- Bundestag require a IT security quality signet
- Quality sign SecurIT from TeleTrusT – Bundesverband IT Sicherheit e.V.
- TÜV Süd checking signet: Software functionality, ergonomics und Data security
- BSI certificate

Offene Themen



Partners

- BITKOM
- BMWI
- BSI
- Bundesdruckerei
- CAST e.V.
- Fraunhofer SIT
- IT Cluster Mitteldeutschland
- Procilon
- Telekom / T-Systems / MMS Dresden
- Teletrust
- TÜV Süd

Accuracy by development

conclusion

The one ring…

The one cant win, if the community lose!

NEMESIS – the goddess of der balancing fairness

Thanks for attention
and always
good cultivated Software

Kontakt

Profn. Dr. Sabine Radomski
Applied computer science
Hochschule für Telekommunikation
Leipzig (HfTL)
Germany
wieland@hft-leipzig.de

References

1. Hofmann, K.: IT Business, 18.3.2015: CeBIT 2015: Anwendungen mit den meisten Sicherheitslücken
2. BSI Lagebericht 2014: https://www.allianz-fuer-cybersicherheit.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2014.pdf?__blob=publicationFile
3. Balzert, H.; Lehrbuch der SW Technik – Basiskonzepte und Requirements Engineering, Spektrum Akademischer Verlag; Auflage: 3. Aufl. 2009
4. Glenford J. Myers: *Reliable Software through Composite Design*. Mason and Lipscomb Publishers, New York 1974.
5. Easter eggs: https://de.wikipedia.org/wiki/Easter_Egg
6. River Rafting Foto: Source: rafting-in-rishikesh.com
7. Barry Boehm: in Sophist White Paper 2011
8. Andreas Essigkrug, Thomas Mey: *Rational Unified Process kompakt.* Spektrum Akademischer Verlag, Heidelberg u. a. 2003, ISBN 3-8274-1440-7.
9. https://www.google.de/search?q=Essen+lecker&tbm=isch&tbo=u&source=univ&sa=X&ved=0CDwQsARqFQoTCKXGh_Of78gCFUEWDwodHcoP4w&biw=1024&bih=671
10. https://www.google.de/search?q=gourmet+essen&tbm=isch&tbo=u&source=univ&sa=X&ved=0CEAQsARqFQoTCLbA74Sg78gCFcUADwodlTMJ6w&biw=1024&bih=671
11. http://www.ferchau.de/news/details/schoene-smarte-welt-4042/
12. Von LCPL JOHN MCGARITY, USMC - http://www.defenselink.mil/multimedia/, Gemeinfrei, https://commons.wikimedia.org/w/index.php?curid=3647061
13. Bilder : fotocommunity.de;
14. http://www.fitforfun.de/beauty-wellness/haut-haare/huebsch-schoenheit-ist-messbar-_aid_8673.html
15. http://www5.in.tum.de/lehre/seminare/semsoft/unterlagen_02/ariane/website/Ariane.Htm
16. **https://de.wikipedia.org/wiki/Alter**
17. **https://de.wikipedia.org/wiki/McDonnell_Douglas_F/A-18**
18. **https://timedotcom.files.wordpress.com/2015/12/rtrtbv.jpg?quality=75&strip=color&w=1100**

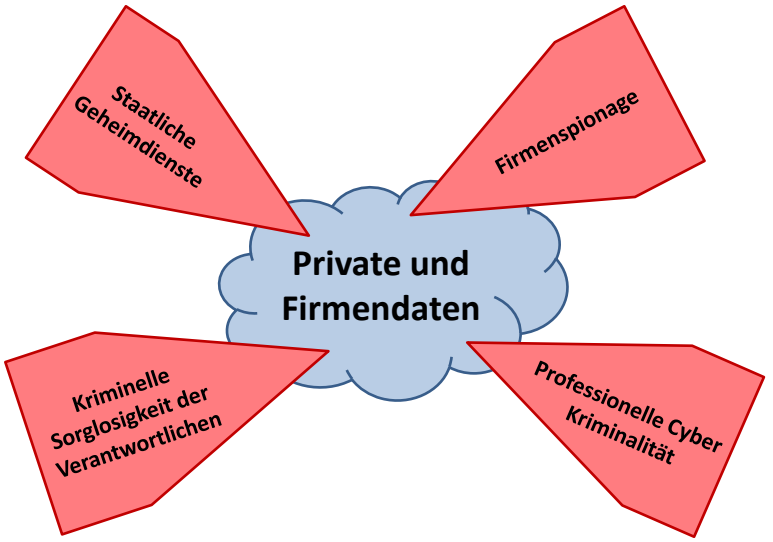
References

19. https://de.wikipedia.org/wiki/McDonnell_F-4
20. For11b] Forrester Research: Software Integrity Risk Report — The Critical Link Between Business Risk And Development Risk . http://www.coverity.com
21. [FIB13] Frost & Sullivan; (ISC) 2 ; Booz Allen Hamilton: The 2013 (ISC) 2 Global In- formation Security Workforce Study .
22. https://www.isc2.org/workforcestudy/Default.aspx , 2013
23. VK11] Vorgang, Blair R.; Karry, Alec: Addressing Software Security in the Federal
24. Acquisition Process . Cigital White Paper, https://www.cigital.com , 2011
27. Content disclaimer: Standard content warnings on xkcd comic pages
28. Programmieren: Reuters
29. broken computer: http://www.loj.name/comp_svcs.htm
30. desktop glass broken: http://www.tophdwallpaersland.com/pc-desktop-backgrounds.html/desktop-broken-glass-windows-computer-wallpaper-backgrounds
31. bad code bart simpson: http://www.codeproject.com/Tips/1029496/How-to-Avoid-Low-Quality-Development-With-Third
32. 2cb71: https://www.linkedin.com/pulse/20141207144637-10429763-good-and-bad-software-engineering-manager

References

33. Bad language: https://pixabay.com/de/wort-zorn-software-ingenieur-904454/
34. Baby at Computer: https://hecticparents.com/author/hecticparents/page/3/
35. Baby Computer: https://www.tnooz.com/article/open-apis-in-travel-need-to-grow-up/
36. Windows Fehler: http://www.pcwelt.de/ratgeber/Fun-Special-witzige-Windows-Fehlermeldungen-240507.html
37. https://de.wikipedia.org/wiki/Akne#/media/File:AcneVulgarisUSM1L.jpg
38. http://bestcarmag.com/gallery/1964-ferrari-275/page/2
39. https://www.it-agile.de/wissen/methoden/kanban/
40. https://familiefamilienrecht.wordpress.com/2015/01/01/kleiner-bub-spricht-mit-mr-barack-obama/
41. Spiegel; TV5 Monde
42. BA Marco Bauer, HfTL, 2015
43. https://www.lorent-online.com/softwareentwicklung/

Der Schatten



Firmenspionage

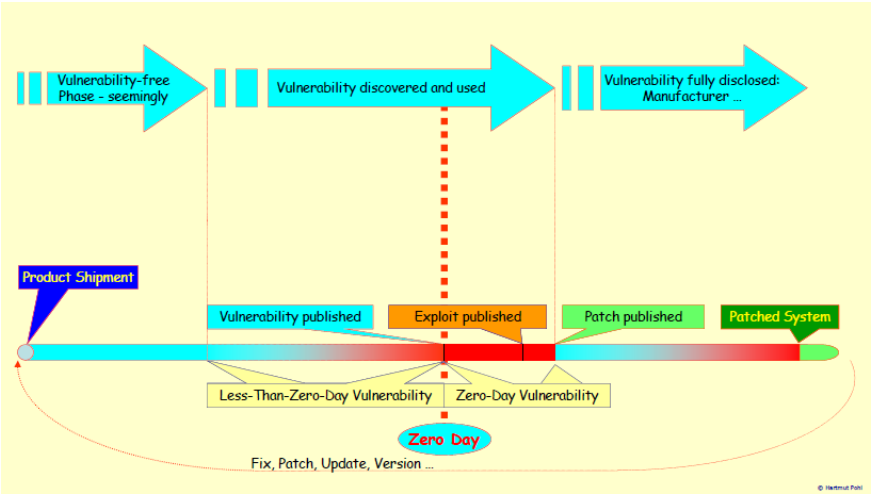
Milliardenschäden in der deutschen Wirtschaft durch Internetkriminalität.
Im Freistaat Sachsen wurden in einem Jahr 11663 Straftaten registriert, bei denen das Internet genutzt wurde.
2300 Straftaten mit Cyber Kriminalität (Daten ausspähen, Rechner kompromitieren, etc.)

Die meisten Fälle werden nicht gemeldet, um Image Verlust zu vermeiden.
Firmen melden lieber annonym an die Allianz für Cyber-Sicherheit.

Dark Data:

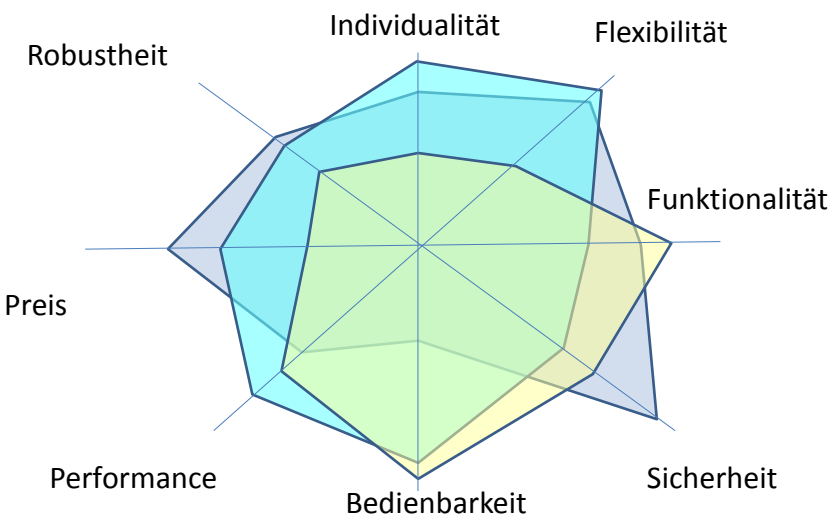
Ausspähen von Firmengeheimnissen durch Nutzung von Personenprofilen und allgemein zugänglichen Informationen zur Ermittlung von Firmengeheimnissen: Hotelempfehlung, Kontakte, bereits bekannte Technologien, Teilnahme an Konferenzen, persönliche Wertung wie erfolgreiche Woche etc.

Bedrohungen



Prof. Dr. Hartmut Pohl

Qualitätsmerkmale



Sorgfalt bei der Herstellung

Gütesiegel „geprüfte Sicherheit“ für Software

- Entsprechend den von der GI formulierten „Grand Challenges der Informatik“soll erreicht werden: Nachhaltigkeit, Transparenz und Sicherheit.

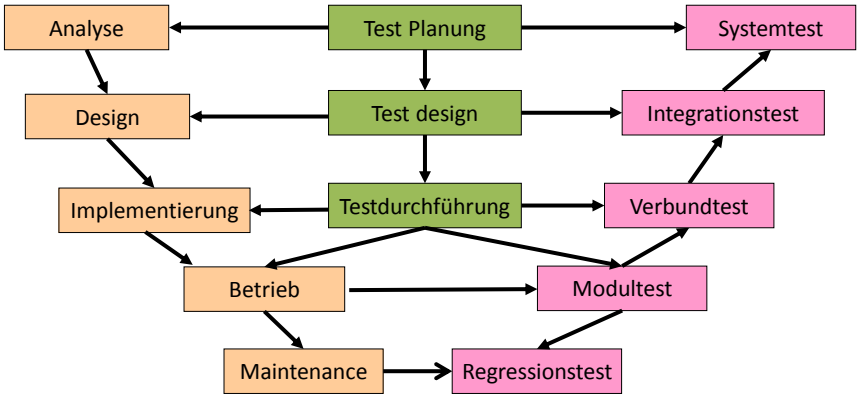
- „Software ist heute allgegenwärtig und kommt im täglichen Leben überall zum Einsatz: in der Kommunikation, der Unterhaltungselektronik, im Haushalt, in der Medizintechnik und in sicherheitskritischen Bereichen.”

HfTL

Die 10 SWE Gebote

10. Du sollst nicht begehren deines Nächsten Gut.

10. Du sollst ausreichend testen. Miss es oder vergiss es!

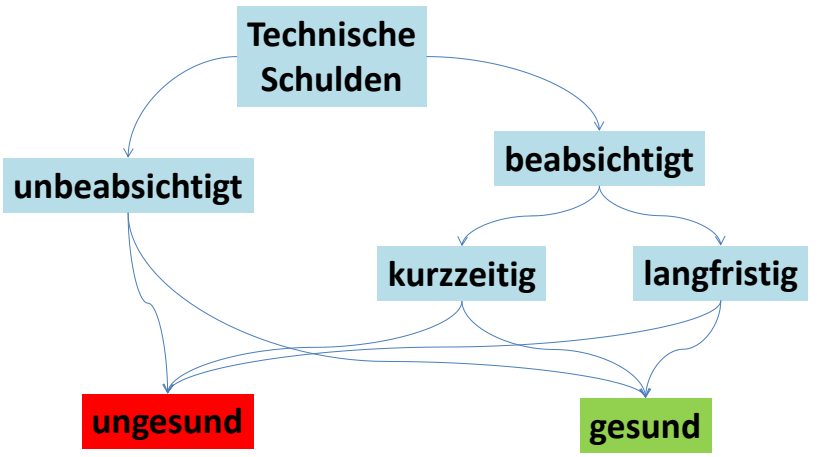


HfTL

Technische Schulden (aus BA Marco Bauer)

Technische Schulden	waghalsig	umsichtig
bewusst	Wir haben keine Zeit für Design	Der Termin steht, wir müssen mit den Konsequenzen leben
unbewusst	Was ist Modularisierung?	Jetzt wissen wir, was wir hätten tun müssen

Umgang mit technischen Schulden



Arten von technischen Schulden

Ausprägungsart	Beschreibung	gesund / ungesund
Quelltextschulden	hohe Quelltextkomplexität, Nicht-Einhaltung des festgelegten Programmierstils	ungesund
Architekturschulden	Modularisierbarer Quelltext	ungesund
Dokumentationsschulden	Unverständliche Kommentare, defizitäre Informationsstruktur	ungesund
Testschulden	Nicht getesteter Quelltext (manuelle und Regressionstests)	gesund / ungesund
Schulden durch eine technologische Lücke	Nutzung veralteter Technologien	gesund / ungesund
Sicherheitsschulden	Nicht-Einhaltung der Sicherheitsrichtlinien	ungesund

HfTL

Software Metriken

Metrikgruppe	Ziel	Beispiele
Größenmetriken	Verständnis der Größe des Programmes und Bestimmung von Verhältnissen innerhalb des Quelltexts	Lines of Code (LOC), Logical Lines of Code (LLOC), Comment Lines of Code (CLOC), Non-Comment Lines of Code (NCLOC), Redundancy Free Lines of Code (RFLOC)
Komplexitätsmetriken	Einschätzung der Wartbarkeit des Quelltexts	Zyklomatische Komplexität, Halstead-Metrik
Architekturmetriken	Fokus auf Objektorientierung, Bewertung des Modularisierungsgrads des Quelltexts	Weighed Methods per Class (WMC), Depth of Inheritance Tree (DIT), Number of Children of a Class (NOC), Coupling Between Objects (CBO)

HfTL

Ermittlung der technischen Schulden mit Metriken

Gestellte Frage
1. Aus wie vielen Zeilen Quelltext besteht das System?
2. Wie ist das Verhältnis zwischen Kommentaren und Nutzcode?
3. Wie ist das Verhältnis zwischen redundanzbehaftetem und redundanzfreiem Code?
4. Wie gut werden die festgelegten Programmierrichtlinien eingehalten?
5. Ist die manuelle Testabdeckung zum Veröffentlichungszeitpunkt ermittelbar? Wenn ja, wie hoch ist sie?
6. Ist die Regressionstestabdeckung ermittelbar? Wenn ja, wie hoch ist sie?
7. Wie komplex sind einzelne Programmbestandteile aufgebaut?
8. Wie hoch ist der Refaktorierungsaufwand für die technischen Schulden, die in der Analyse ermittelt wurden?

Beispielprojekt

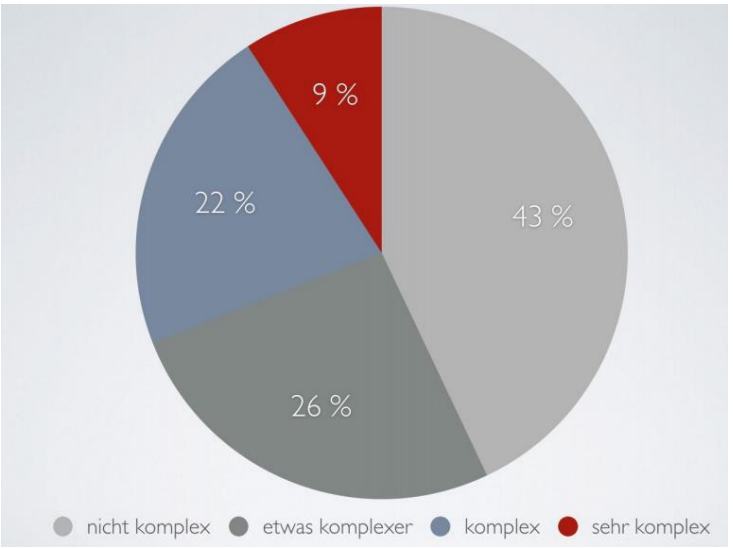
Anzahl der Klassen	LLOC	Anzahl der Statements	Anteil Kommentare	Anteil Quelltext
142	19.467	12.511	11,6 %	88,4 %

Kritikalitätsbewertung	Kritikalitätseinschätzung
0	kein Risiko
1	kritisch
2	sehr kritisch

Zyklomatische Komplexität

Zyklomatische Komplexität	Risikoeinschätzung
1-10	einfaches Programm, geringes Risiko
11-20	etwas komplexer, moderates Risiko
21-50	komplex, Programm mit hohem Risiko
>50	untestbares Programm (sehr hohes Risiko)

Komplexität im Quellcode



Auswirkungen von technischen Schulden

Auswirkung	Beschreibung
Geringe Wartbarkeit Geringe Verständlichkeit	Die Klasse ist nur sehr schwierig wartbar. Ein mit dem System nicht vertrauter Entwickler hat Schwierigkeiten den Aufbau und die Verwendung der Klasse nachzuvollziehen.
Geringe Testbarkeit	Die Komplexität der Klasse lässt die Formulierung von sinnvollen Testfällen nicht zu.
Hohe Anzahl notwendiger Testfälle	Um alle möglichen Pfade der Klasse zu testen müssen mindestens 206 Regressionstests geschrieben werden.

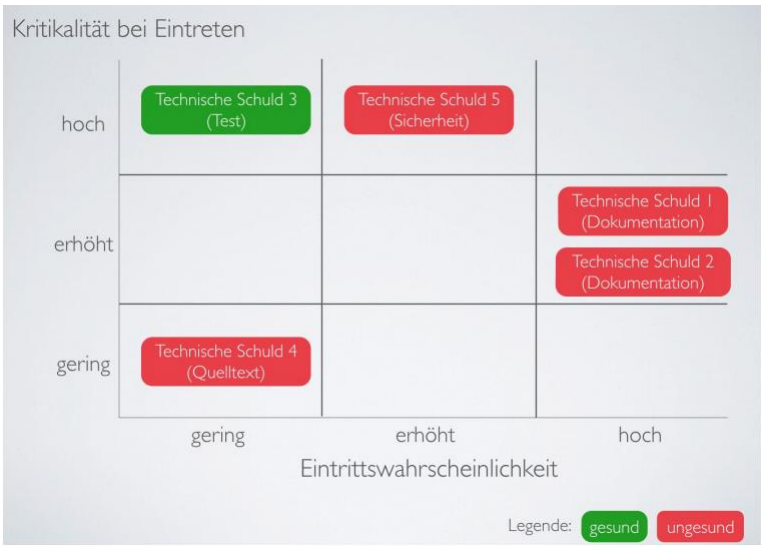
Erkannte technische Schulden

Programmierrichtlinienverstöße in Abrechnung.vb
17 auskommentierte Quelltextzeilen befinden sich noch im Quelltext
15 öffentliche Methoden sind nicht dokumentiert
12 Variablen sind nicht in CamelCase geschrieben
8 Quelltextzeilen überschreiten die Länge von 150 Zeichen
3 Methoden sind nicht in der jeweils kleinst möglichen Zugriffsebene definiert
1 Funktion wird nicht durch einen Regressionstest abgedeckt

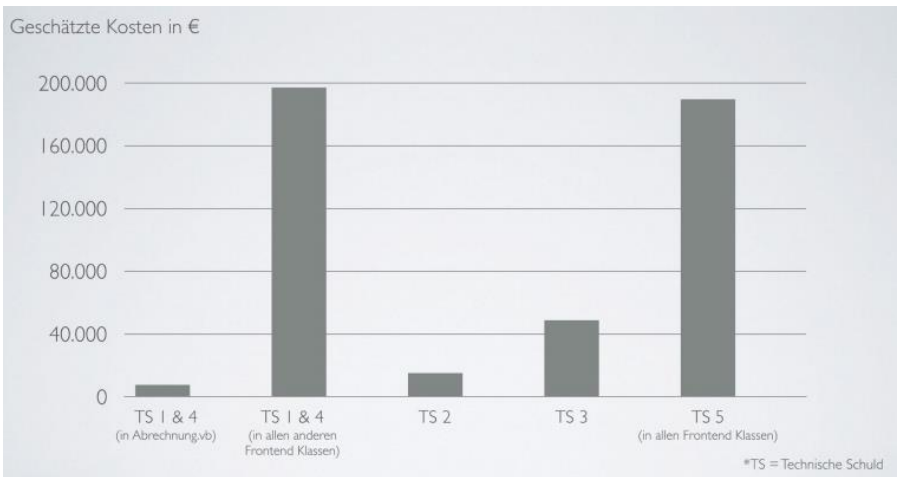
Erkannte technische Schulden im Projekt

- T1: defizitäre Kommentarqualität
- T2: kein persistentes Logging im Frontend
- T3: Frontendfunktion ohne Regressionstests
- T4: Nicht-Einhaltung von Programmierrichtlinien in Abrechnung.vb
- T5: Unsichere Zugriffsebenen in Abrechnung.vb

Kritikalität der erkannten technischen Schulden



Refaktorierungskosten der erkannten technischen Schulden



Erfolgreicher Einsatz von Software Metriken

Metrik(gruppe)	Sinnvolle Anwendung als Analysehilfsmittel auf technische Schulden	Verwendete Softwarewerkzeuge
Größenmetriken	nein	Source Monitor
Redundanzmetrik	ja	CodeRush
Komplexitätsmetrik	ja	CodeRush Duplicate Detection and Consolidation
Programmierrichtlinien	ja	manuelle Untersuchung, da die Richtlinien vom Entwicklungsteam selbst formuliert wurden

Notwendiger Aufwand zur Beseitigung von technischen Schulden

- **Der Refaktorierungsaufwand umfasst insgesamt 99 Story Punkte. Das entspricht ungefähr 460.000 €**
- Die analysierten Abweichungen vom gewünschten Qualitätsstandard lassen sich in Entwicklungskosten für die Behebung der technischen Schulden umrechnen.