sieber & partners

Return on Software-Investments

Wie Sie den ROI Ihrer Entwicklungsprojekte erhöhen

«Was man nicht messen kann, kann man nicht lenken.»

(Peter Drucker, Pionier der modernen Managementlehre)

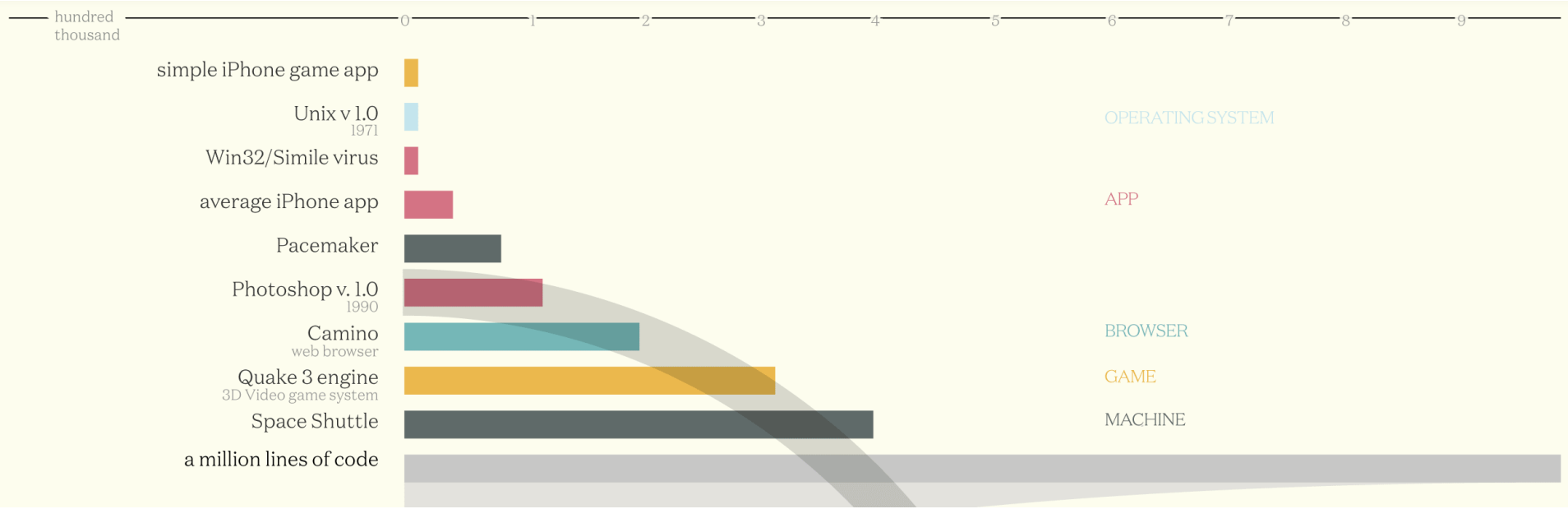
Agenda

1.	Software is eating the world	(Kay Grosskop)
2.	Herausforderungen	(all)
3.	Software Qualitätsmodell	(Marc André Hahn)
4.	Produktivitätsmessung	(Kay Grosskop)
5.	Beschaffung und Lieferantenführung	(Marc André Hahn)
6.	Use Cases (Beschaffung / Betrieb)	(Marc André Hahn)
7.	Customer Cases	(Kay Grosskop)
8.	Anwendung und Diskussion	(all)
9.	The human factor: K@R Erfolgsformel	(Vladimir Riecicky)

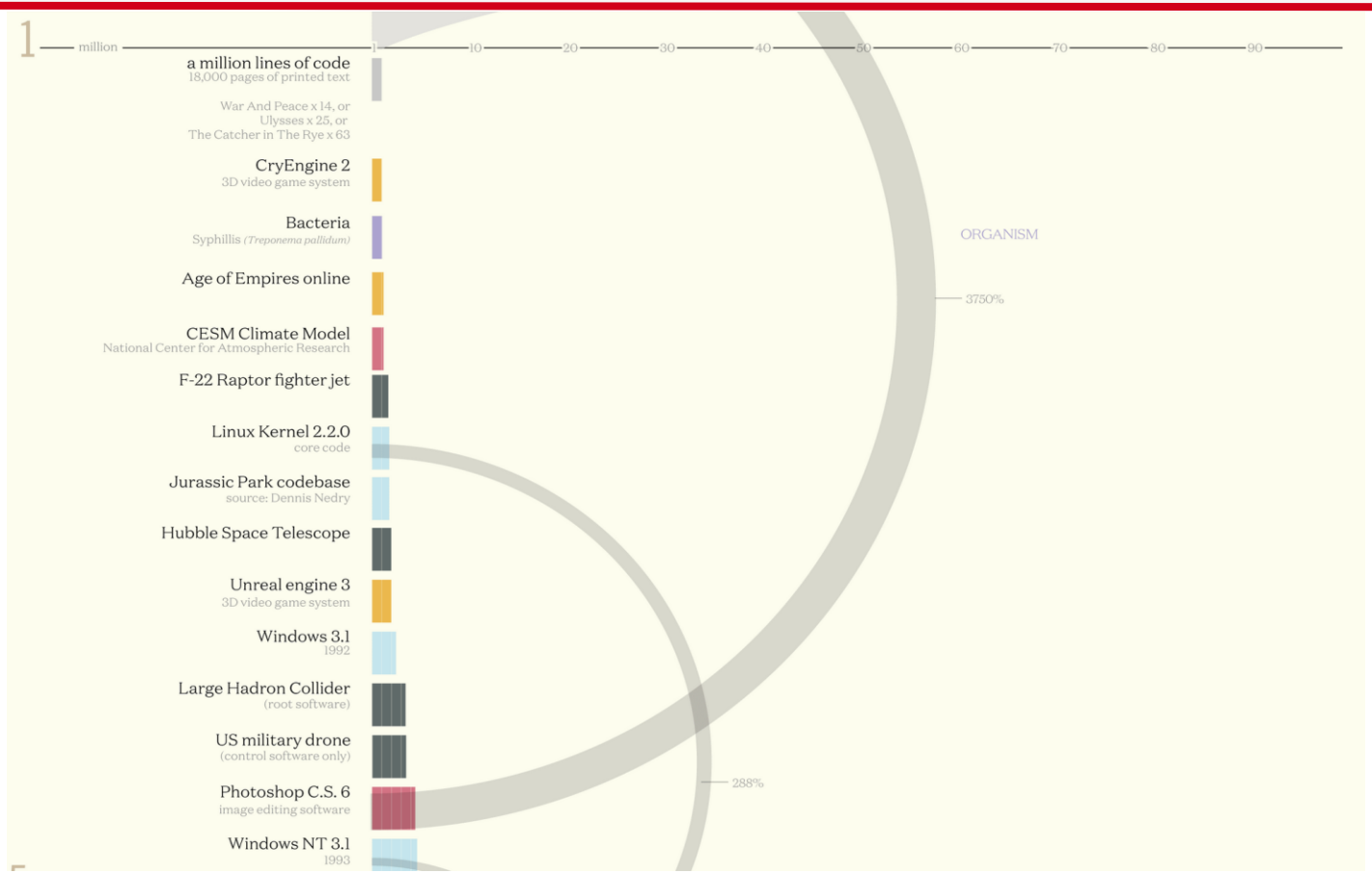
1. Software is eating the world

Software wird immer wichtiger

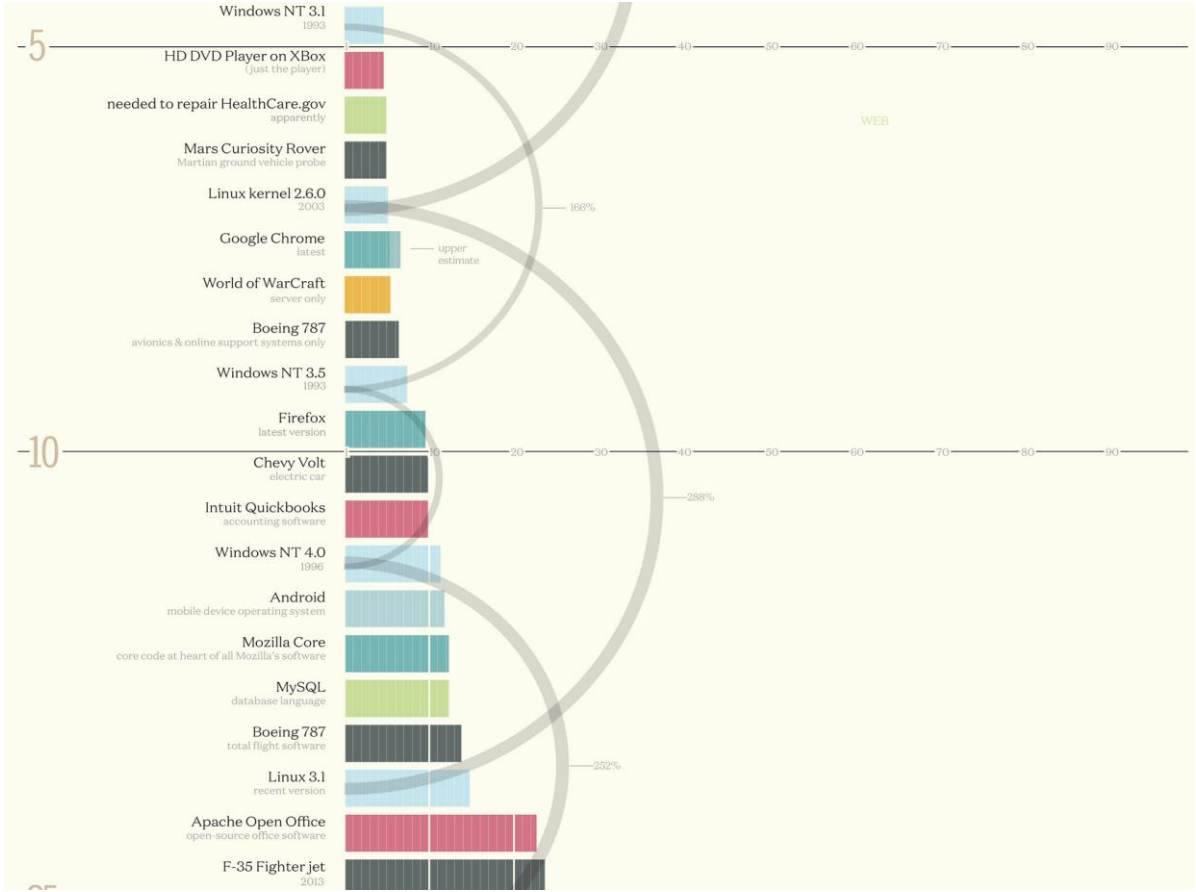
Software ist gross …



… sehr gross

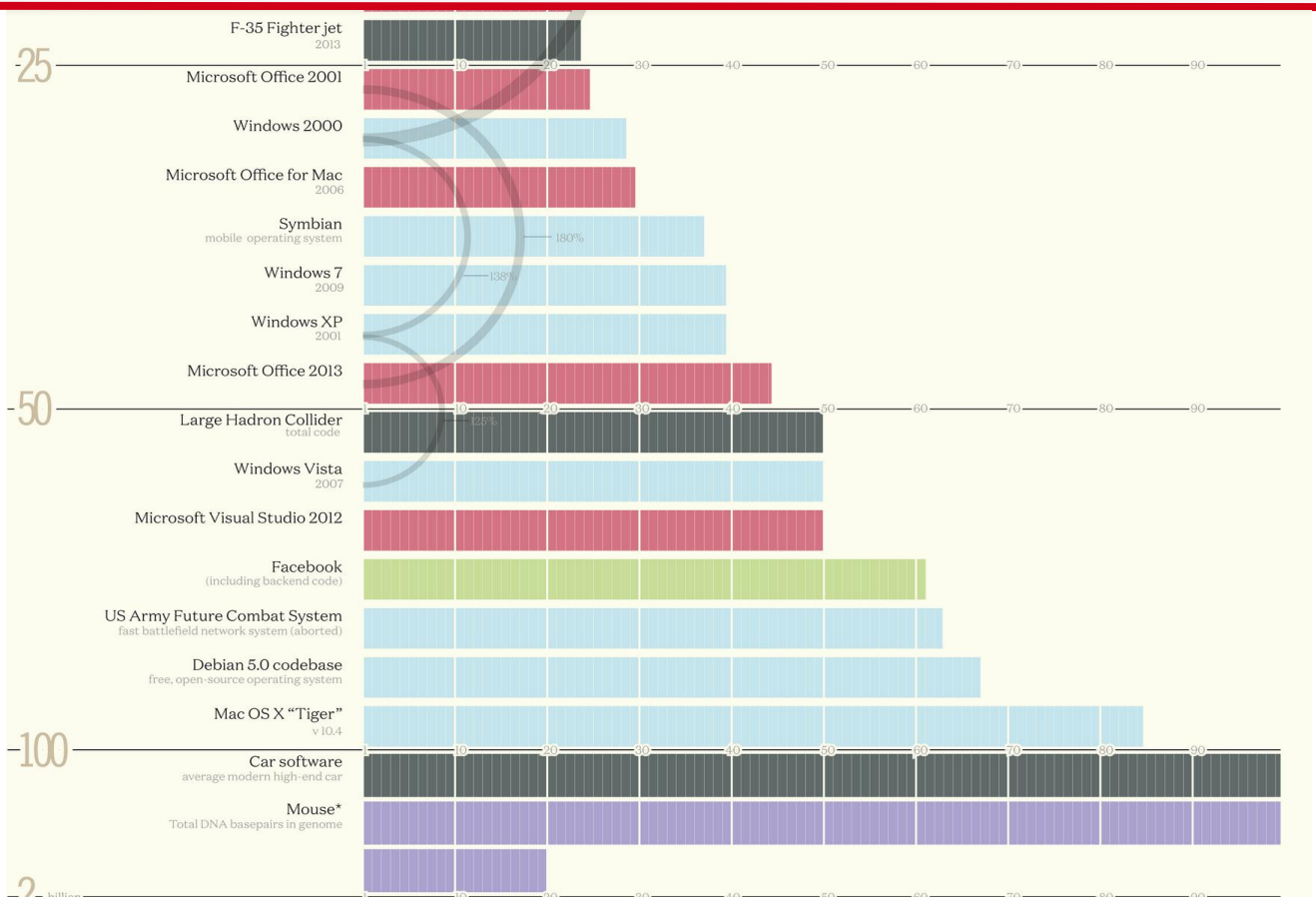

1 million
1
10
20
30
40
50
60
70
80
90
a million lines of code
18,000 pages of printed text
War And Peace x 14, or
Ulysses x 25, or
The Catcher in The Rye x 63
CryEngine 2
3D video game system
Bacteria
Syphillis (Treponema pallidum)
Age of Empires online
CESM Climate Model
National Center for Atmospheric Research
F-22 Raptor fighter jet
Linux Kernel 2.2.0
core code
Jurassic Park codebase
source: Dennis Nedry
Hubble Space Telescope
Unreal engine 3
3D video game system
Windows 3.1
1992
Large Hadron Collider
(root software)
US military drone
(control software only)
Photoshop C.S. 6
image editing software
Windows NT 3.1
1993
ORGANISM
3750%
288%

Und sie wächst …



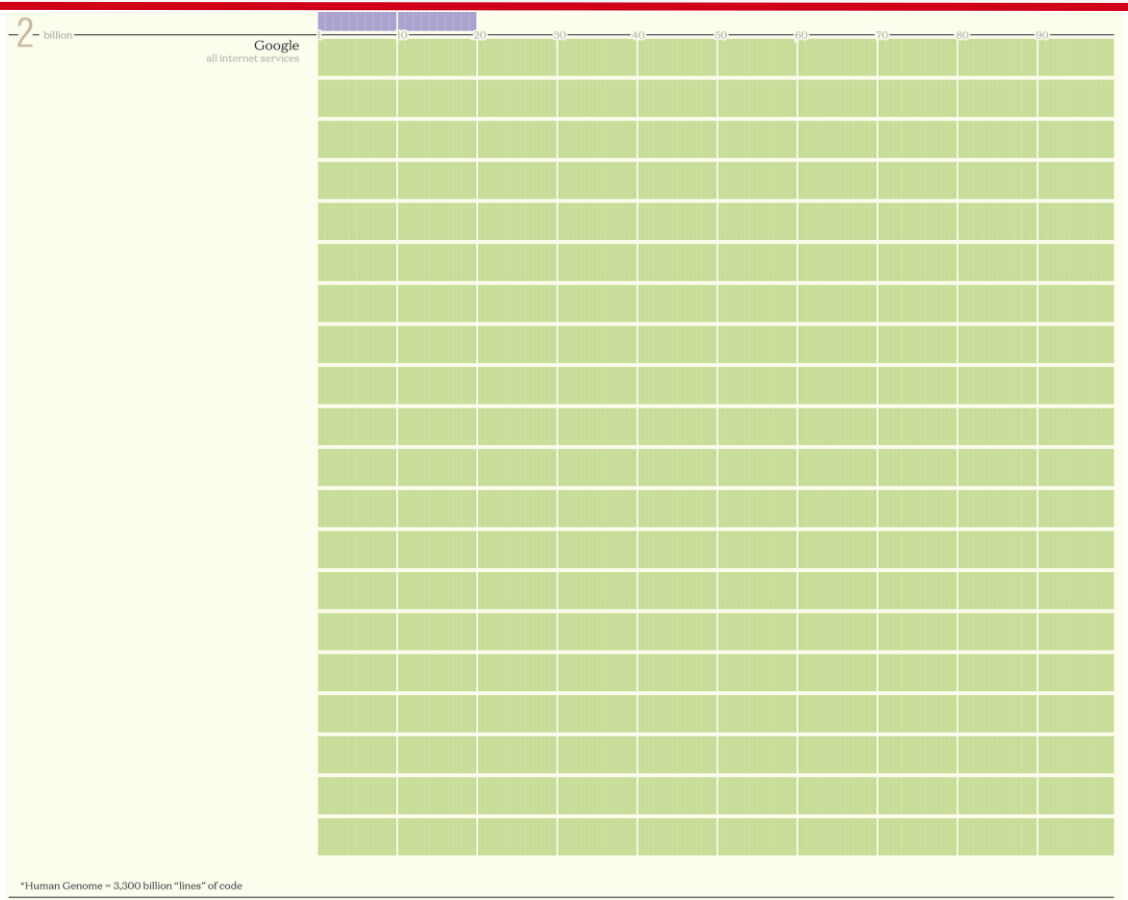
sieber & partners

Und wächst …



sieber & partners

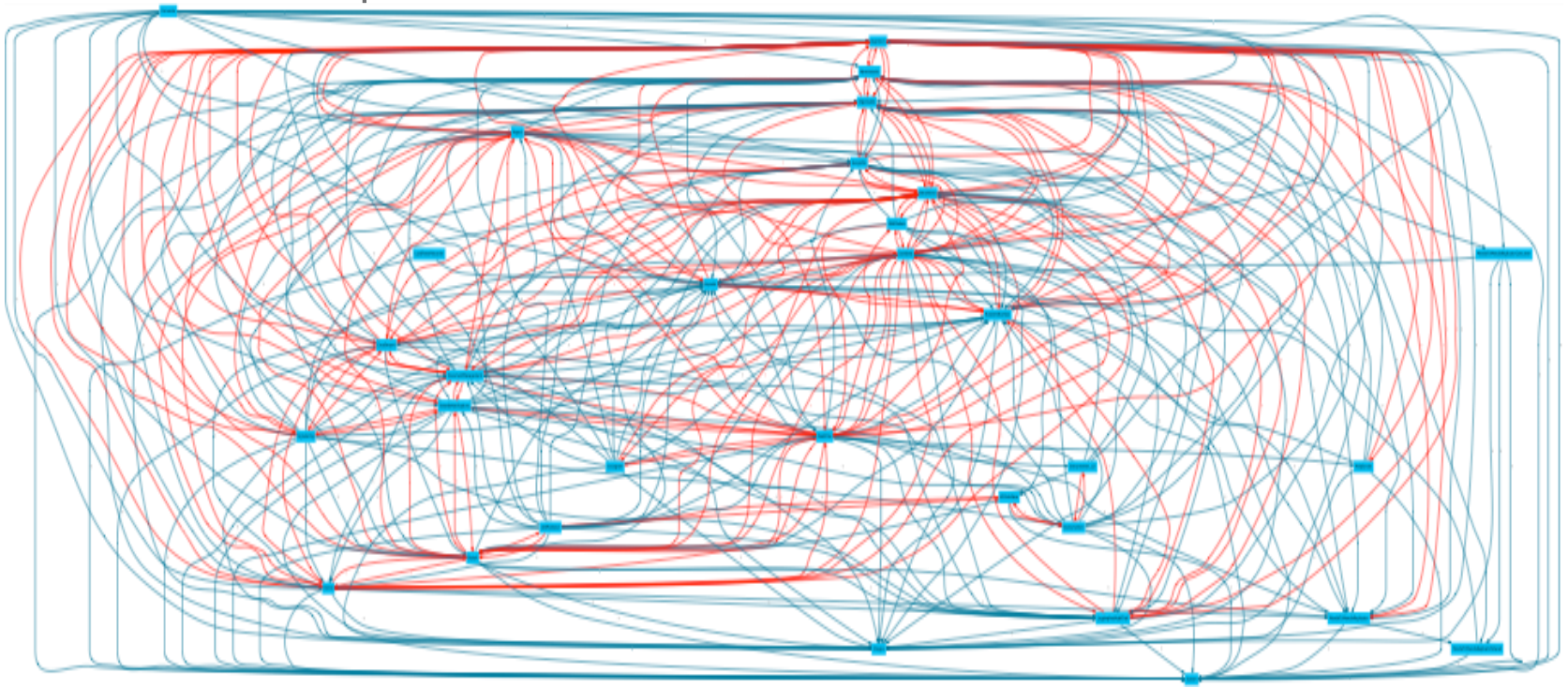
‘Software is eating the world’ (Andreessen Horowitz, 2011)



2015: informationisbeautiful.net
https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

Komplexität muss beherrscht werden

Wie konnte dies passieren?



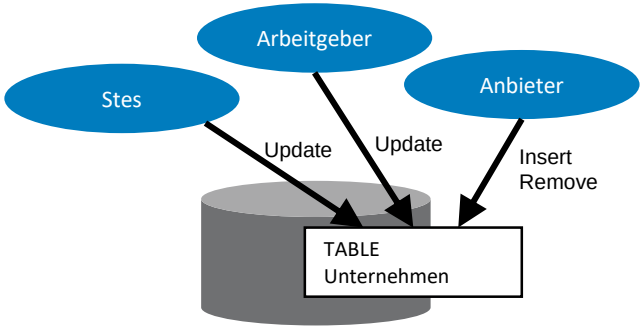
Methodenaufrufe zwischen Componenten eines Beispielsystems. rot =zyklische Aufrufe

Komplexität muss beherrscht werden

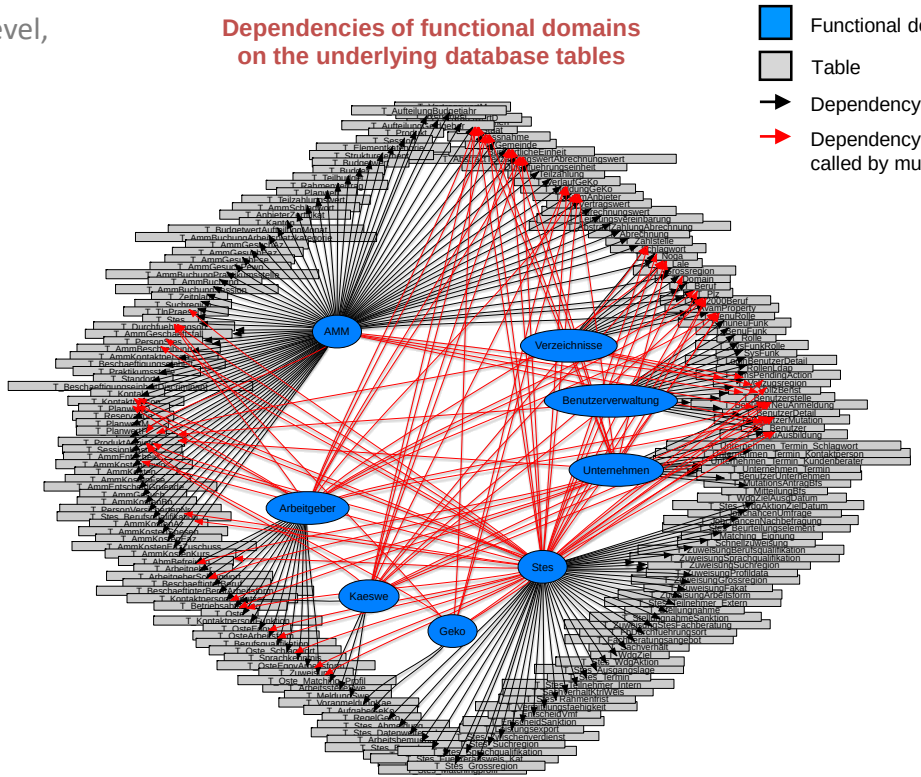
Beispiel: Kopplung der Komponenten über die Datenbank

- Functional domains are decoupled on the service level, but are coupled on the database level

Example: Domains *Stes*, *Arbeitgeber* and *Anbieter* all mutate database table *Unternehmen*



Dependencies of functional domains on the underlying database tables



Ihre Herausforderungen bei Softwareprojekten

- Projektmanagement
- Lieferanten
- Qualität
- Funktionalität
- Kosten
- ...

3. Software Qualitätsmodel

Wartbarkeit als Katalysator der Softwareentwicklung

Softwarequalität ist messbar.

- Der Qualitätsstandard ISO 25010 definiert acht Eigenschaften

Wartbarkeit ist die Schlüsseleigenschaft

- Einfluss auf **Effektivität** und **Effizienz** von Änderungen am System
- Einfluss auf **Behebung** von **Fehlern**
- Einfluss auf **funktionale Erweiterungen**
- Einfluss auf die **Veränderbarkeit** der anderen **Qualitätseigenschaften**
- Wartbarkeit ist der **Kostentreiber**

Objektivität durch Standards

- **Standard**: **SIG/TÜViT** Bewertungsverfahren basierend auf dem **ISO/IEC 25010** Standard für Software Qualität.

- **Zertifizierung**: «Maintainability Trusted Product»

- **Labor**: nach ISO/IEC 17025 vom TÜViT **zertifiziertes Labor** für die Messung von Software

Zertifikat für Produktwartbarkeit.
Gemessen nach Standards der TÜViT

Wie die Wartbarkeit faktenbasiert gemessen wird

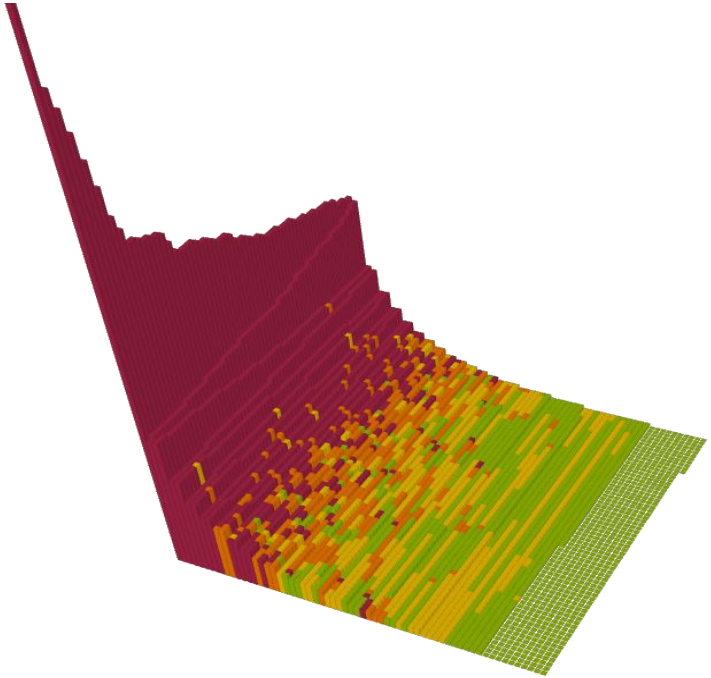
	Volume (Volumen)	Duplication (Duplikation)	Unit Size (Grösse d. Unit)	Unit Complexity (Komplexität d. Unit)	Unit Interfacing (Schnittstelle d. Unit)	Module Coupling (Kopplung d. Moduls)	Component Balance (Balance d. Komponenten)	Component Independence (Unabhängigkeit d. Komponenten)
Analysierbarkeit	●	●	●				●	
Anpassbarkeit		●		●		●		
Testbarkeit	●			●				●
Modularität						●	●	●
Wiederverwendbarkeit			●		●			

- Die Wartbarkeit nach ISO 25010 und ihre untergeordneten Eigenschaften verknüpft mit dem Bewertungsmodel.
- Das Model ist unabhängig von der Programmiersprache und der Businessdomäne.

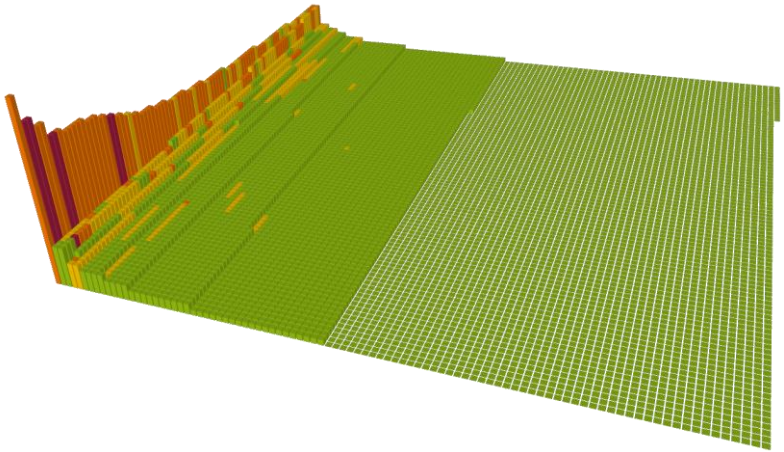
sieber & partners

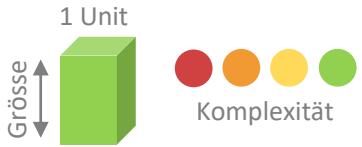
Keine Codebasis ist perfekt: Es kommt auf die Verteilung an

Nicht nachhaltig



Nachhaltig





Beispiel für eine Eigenschaft: *Unit complexity*

Einfache Units sind einfach zu **modifizieren** und einfach zu **testen**

Die *unit complexity* wird mit der McCabe* Metrik bestimmt. Die Unit wird dann einer Risikostufe zugeordnet:

Beispiel:
eine unit mit
McCabe: 4

```
void removePackage(String name, RuleBase rb) {
    Package[] ps = rb.getPackages();
    if (ps == null) return;
    for (int i = 0; i < ps.length; i++) {
        Package p = ps[i];
        if (p.getName().equals(name)) {
            rb.removePackage(name);
            return;
        }
    }
}
```

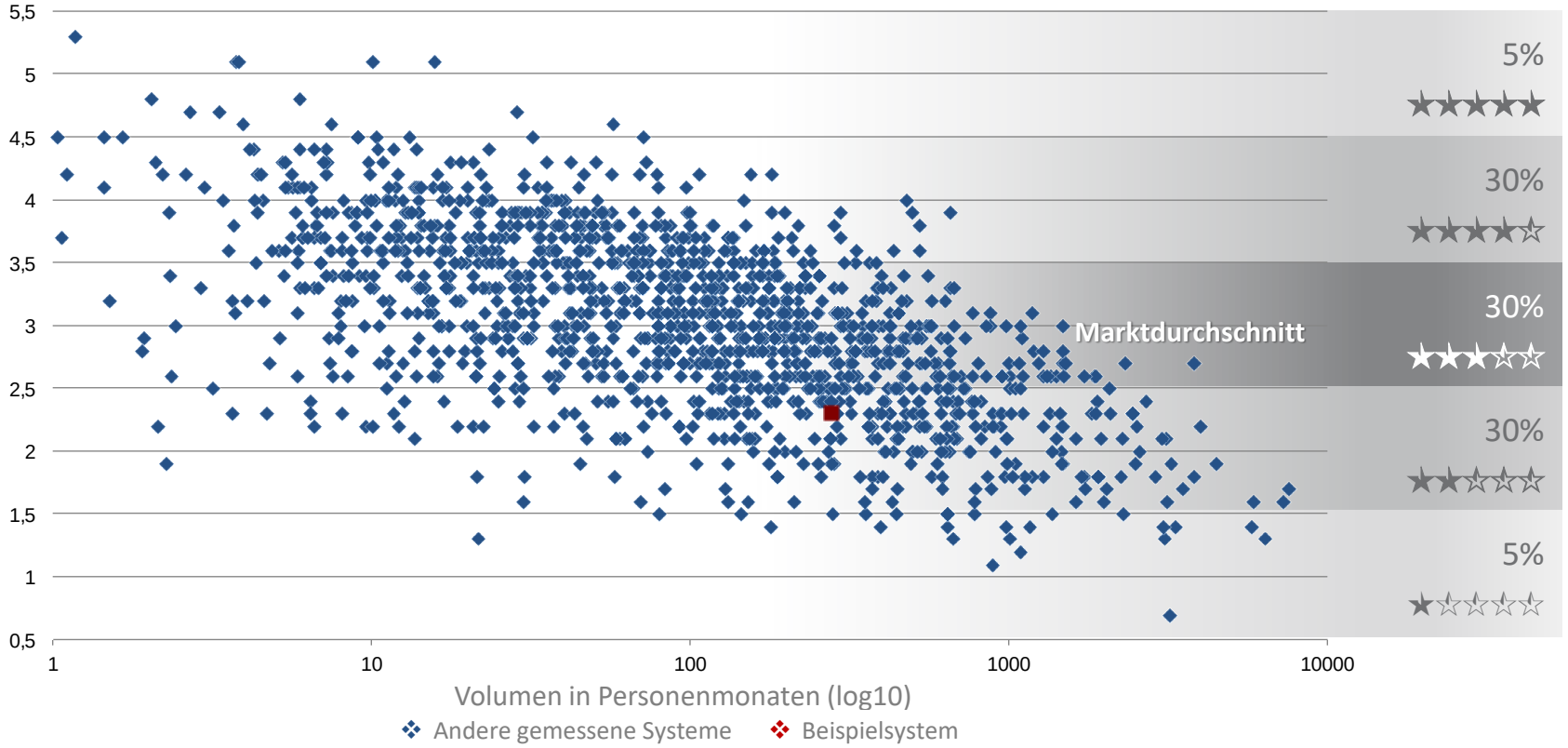
Cyclomatic complexity	Risk estimation
1-5	Clear code, low risk
6-10	Complex, moderate risk
11-25	Very complex, high risk
> 25	Not understandable, untestable, very high risk

** Cyclomatic complexity, after: McCabe, IEEE Transactions on Software Engineering, 1976*

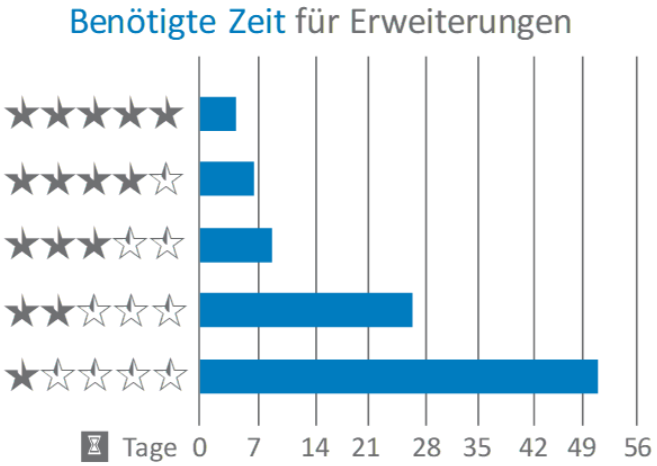
Von der Messung zum Star-Rating

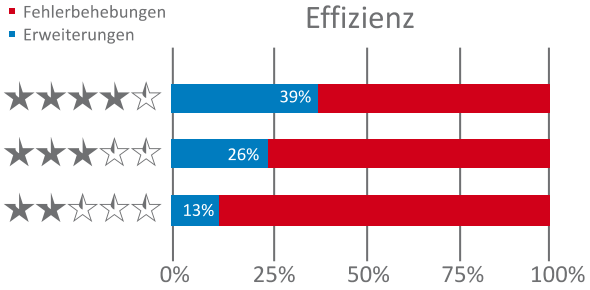
- Einzelne Kennzahlen werden zu einem Rating verdichtet
- Die Bewertung erfolgt anhand eines Benchmarks

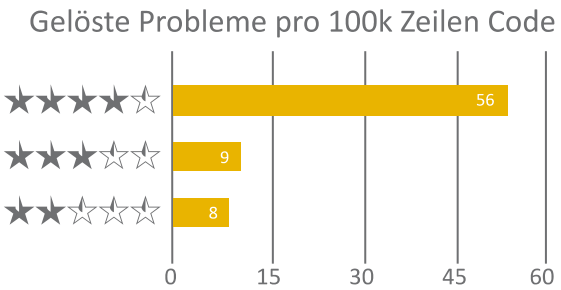
5 Milliarden Zeilen Code im Benchmark

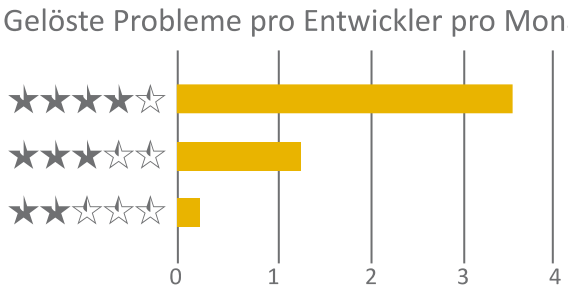


Auswirkung von Wartbarkeit









4. Produktivitätsmessung

Produktivität messen

$$\text{Produktivität} = \frac{Umfang * Qualität * Relevanz}{Aufwand}$$

Option 1: Funktionalität (externe Sicht)

Traditionelle art den Umfang zu bestimmen: Function Points (FP)

Nachteile

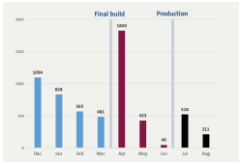
- **Customer valued pricing**: Der Aufwand des Lieferanten muss nicht dem Funktionsumfang entsprechen.
- **Externe Perspektive. Keine Sicht auf *interne* Qualität**: Risiken für Flexibilität und Wartungskosten.
- **Aufwändig** zu messen, da nicht automatisierbar.

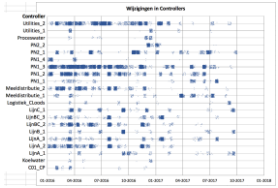
Option 2: Automatisierte Softwareanalyse (interne Sicht)

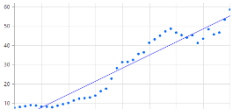
Qualität

Quality measurement

Aggregation & reporting



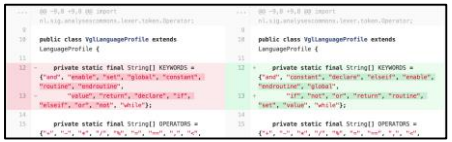




Version Control System

Reconstruction of coding activity

Churn calculation

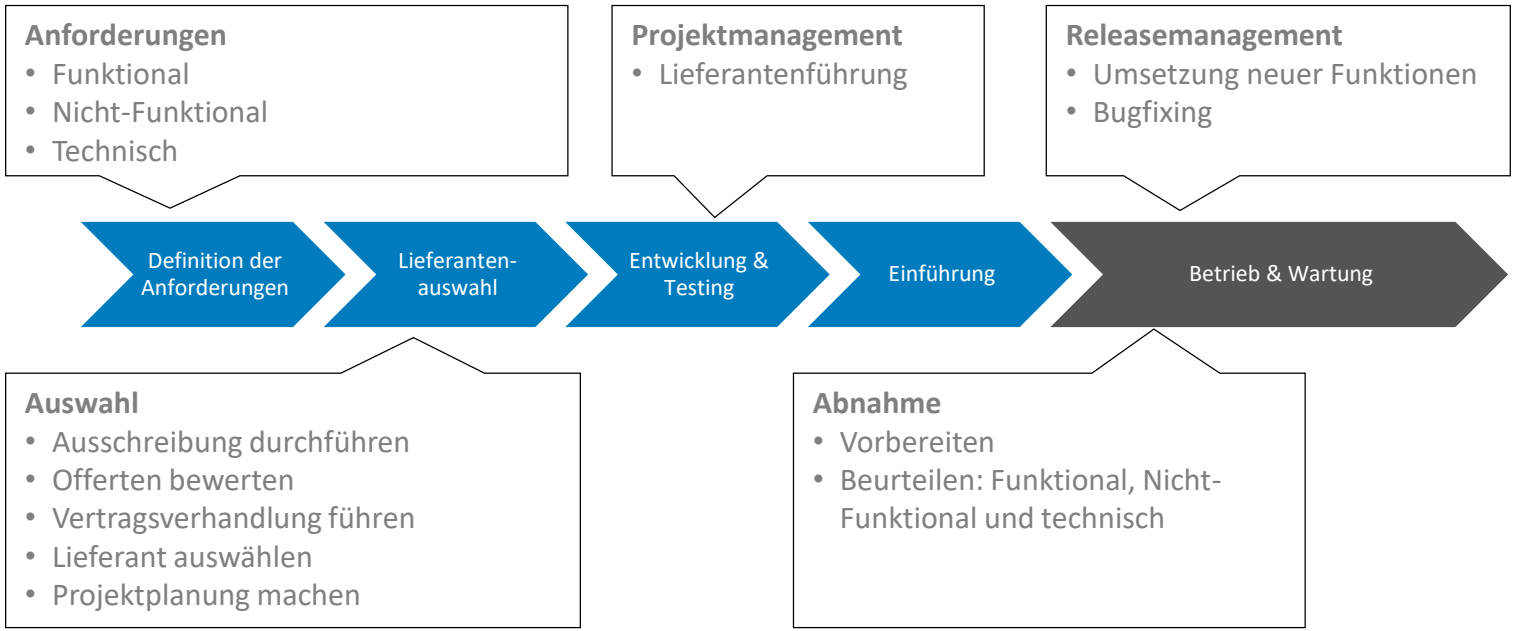


Advanced filtering

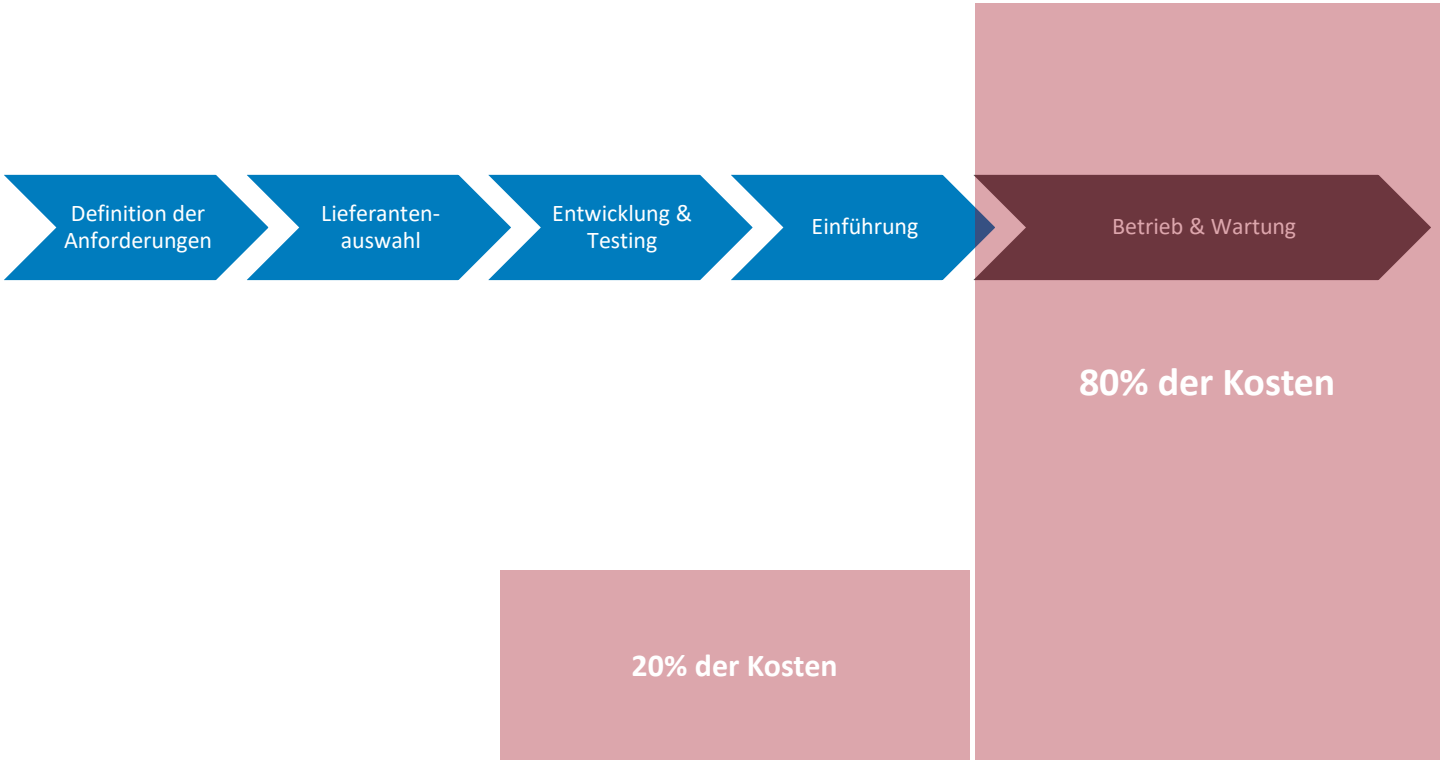
Quantität

5. Beschaffung und Lieferantenführung

Beschaffungsprozess



Beschaffungsprozess



Auswahlkriterien in der Beschaffung

Im Beschaffungsprozess kommen in der Regel nur Kriterien zum Einsatz, die nicht auf das Endprodukt abzielen!

Kriterien – und warum sie nicht ausreichen

Entwicklungs-kosten

- Berücksichtigen nicht zukünftige Wartungskosten
- Machen 20% der Kosten im Lebenszyklus aus
- Man erkauft sich tiefe Entwicklungskosten mit hohen Wartungskosten
- Bem: Bei öffentlichen Ausschreibungen sehr kleiner Spielraum

Entwicklungs-prozess

- Standards wie CMMI werden beachtet
- Sind nur auf die Organisation bezogen, nicht auf das Produkt
- Können Hinweis auf schlechte Qualität, aber nicht auf gute Qualität geben
- Garantieren keine gute Qualität
- Sagt nichts über die Effizienz aus

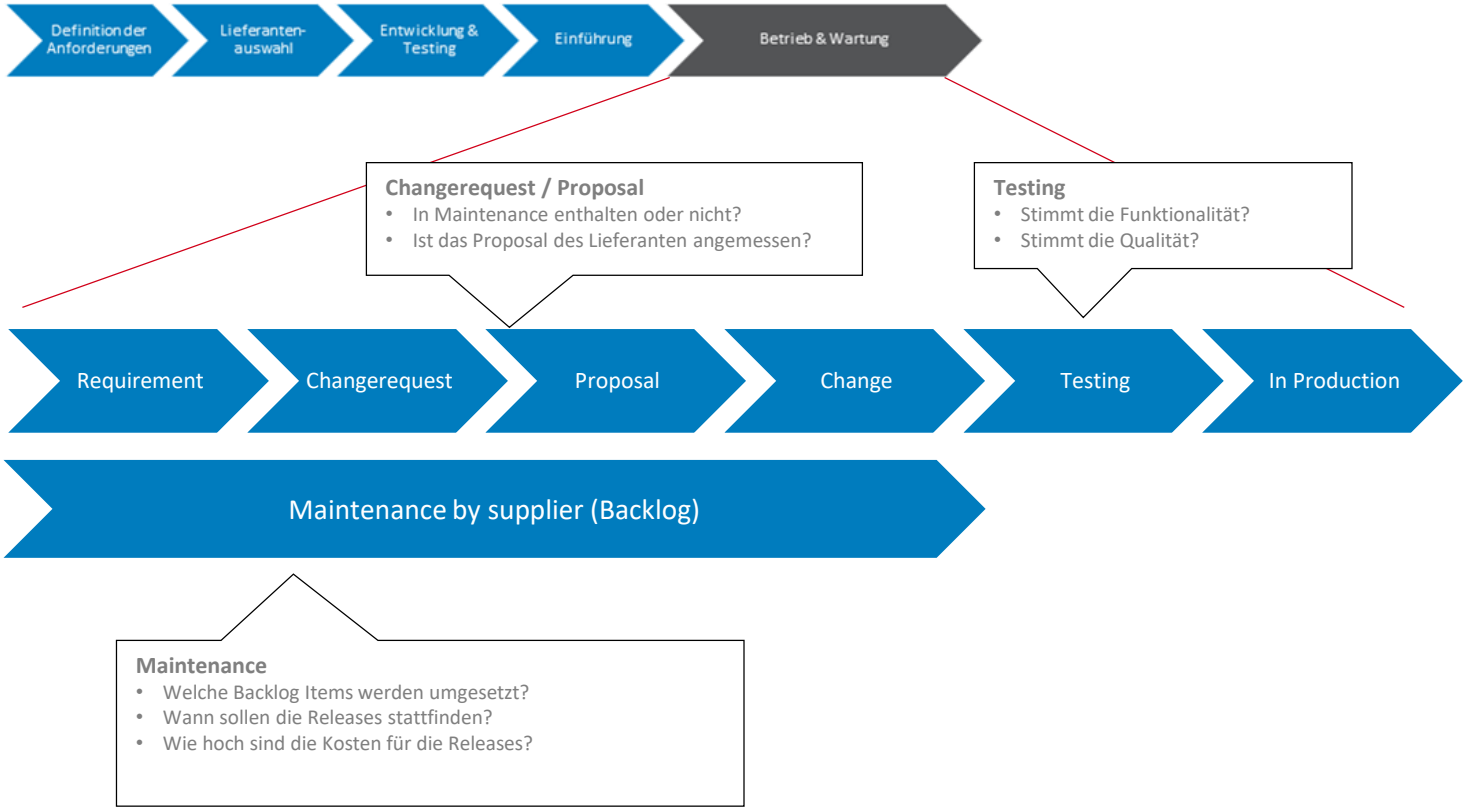
Wartungs-kosten

- Machen den grössten Teil der Kosten eines System im gesamten Lebenszyklus aus (bis zu 80%)
- In der Regel sind nur Kosten und nicht Leistungen im Blick
- Sagen nichts über die Angemessenheit aus

Referenzen

- Geben Kundenzufriedenheit wieder
- Zeigen Kompetenzen und Erfahrungen auf
- Sagen nichts über Qualität und Angemessenheit des Preises aus

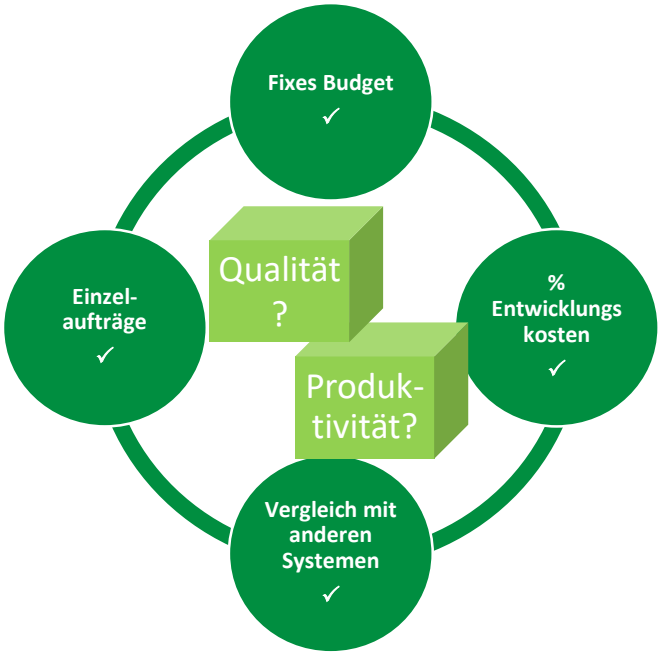
Betriebsphase



Steuerungsgrössen in der Betriebsphase

Beschaffung

Betrieb



Kriterien – und warum sie nicht ausreichen

Einzel-aufträge

- Es sind nur Kosten im Blick
- Es werden meist nur funktionale Anforderungen spezifiert
- Vergleichbarkeit fehlt
- Overhead muss mitfinanziert werden

Fixes Budget

- Regelt nur die Kosten, nicht die Leistung
- Prüft nicht, ob die Leistungen abnehmen
- Sagt nichts über eingesetzte Ressourcen
- Gegenleistung ist nicht messbar

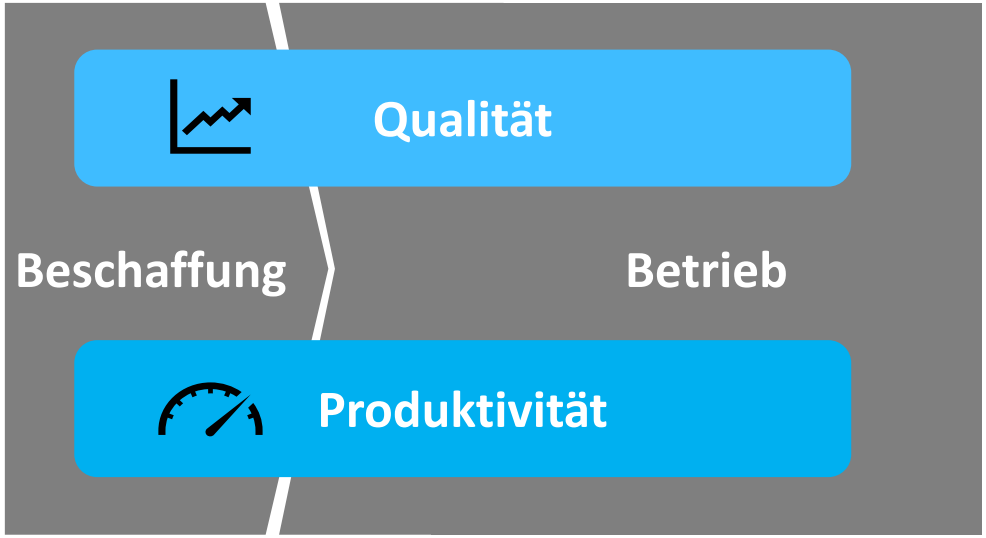
% Entwicklungs kosten

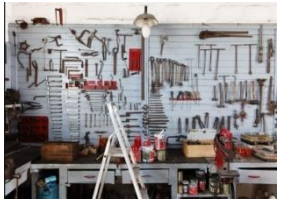
- Regeln nur Kosten, nicht die Leistungen
- Berücksichtigen nicht die Wartbarkeit
- Sagen nichts über Anzahl neuer Funktionen
- Sagen nichts über eingesetzte Ressourcen

Vergleich mit anderen Systemen

- Sagt nichts über die Angemessenheit aus
- Berücksichtigt Qualität (Wartbarkeit nicht) der verschiedenen Systeme

Qualität und Produktivität sind messbar!





Typische Fragen von Entscheidern

Wie kann ich meine Wartungskosten senken?

Wie produktiv ist das Entwicklerteam?

Ist der Zustand meiner Software ein Risiko für die Unternehmensentwicklung?

Wie kann ich den Fortschritt meines Softwareprojektes objektiv messen?

Entspricht meine Software den Architekturvorgaben?

Wie kann ich die Abhängigkeiten von meinem Lieferanten reduzieren?

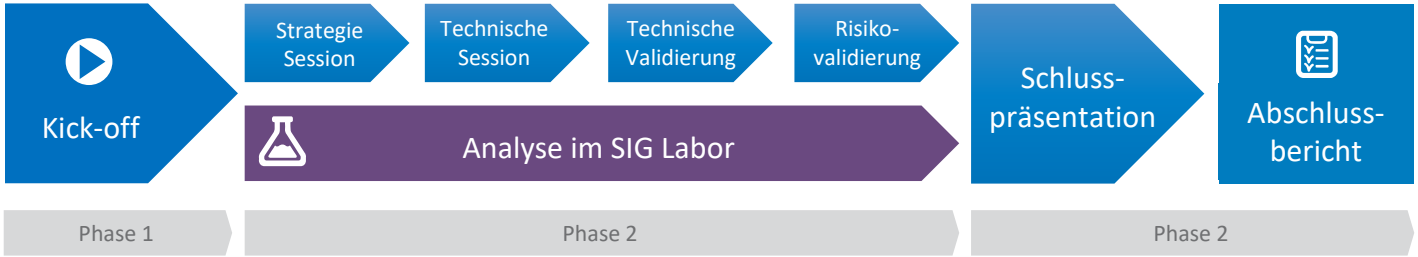
Ist mein interner Lieferant besser als mein externer?

Soll ich meine Software renovieren, neu schreiben oder lassen wie sie ist?

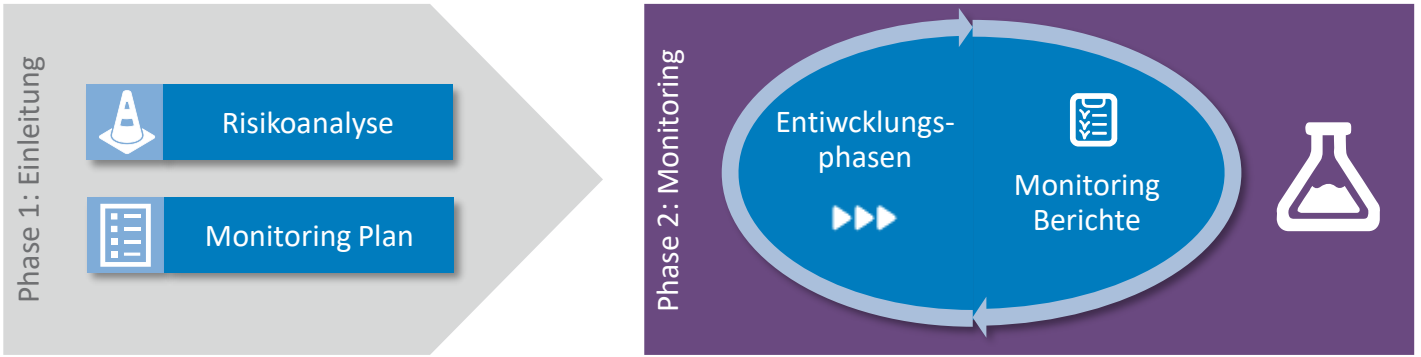
Ist die Offerte meines Lieferanten angemessen?

Bekomme ich eine angemessene Gegenwert von meinem Software-Lieferanten?

Projektvorgehen



Einmalige Analyse



Monitoring

6. Use Cases

Typische Anwendungsfälle

1. Beschaffung von Software (Individualentwicklung)
2. Gezielte Führung eines Lieferanten im Betrieb

Case 1: Beschaffungsprozess: Messbarkeit

- Definition von Qualitätsstandards als Vertragskriterium
- Transparente Arbeitsweise des Lieferanten einfordern

Objektive Informationen für das Management
- Fortlaufendes Monitoring der Qualität
- Überwachung des Projektfortschritts

- Einforderung von Arbeitsproben in Form von Prototyping oder bestehender Software.
- Beurteilung des Entwicklungsprozesses

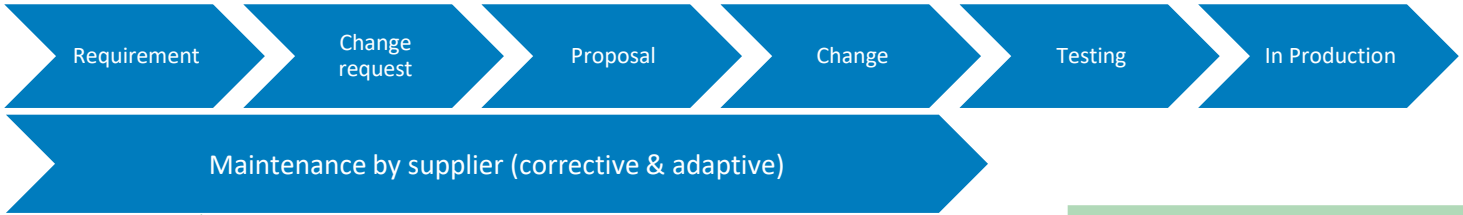
- Qualität als Abnahmekriterium
- Zertifizierung des Endproduktes

Case 2: Wartung - Führung eines Lieferanten

Prüfung u. Plausibilisierung der Offerte:
- Was soll gemacht werden?
- Welche Komponenten sind betroffen?
- Ist der Aufwand angemessen?
- → später Überprüfung der Aufwände

Betrieb & Wartung

Kontrolle:
- Testabdeckung für neuentwickelten Code
- Produktivität des Entwicklerteams



Reduktion von Wartungskosten durch
- Identifikation von Kostentreibern
- Empfehlungen für gerichtete Verbesserungen

Überwachung von
- Qualitätsvorgaben
- Testabdeckung
- Unterstützung einer Kulturumschwungs in der Entwicklung

Overall Controlling
- Wie viele Codeänderungen habe ich für die Wartungsaufwände bekommen?
- Wie stehen Produktivität und Qualität im Vergleich zu anderen Lieferanten?
- Will ich den Lieferanten für gute Qualität honorieren?

7. Customer Cases

6. Customer Cases

Case 1: Evaluation eines ERP Systems

Produkt- und Lieferantenauswahl

Customer case 1: Evaluation eines ERP Systems

Situation

- Manufactoring company wants to replace their outdated ERP System with a new standard software
- 25 production sites in Europe
- Project size about 10 Mio CHF in three years
- Only three industry specific solutions on the market

Action

- Quality Analysis of all three codebases.
- Review of the suppliers improvement plans.

Questions

- Which ERP System is the best to quickliy implement future requirements?
- With which supplier do we have the lowest risk to implement our software in time?
- Which supplier has the right ressources to maintain the software?
- Which ERP System has the best quality to get most value for maintenance fees?

Result

- Strong impact in negotiation phase because of knowledge about the quality -> price reduction of more than **3 Mio**.
- Maintenance fees are coupled to the improvement of the software.
- Customer has the best solution to implement new requirements quickly.

Customer case 1: Evaluation eines ERP Systems

★★☆☆☆ 1.9 (1.6)*

System properties

Volume	★☆☆☆☆	0.5
Duplication	★★☆☆☆	1.9
Unit size	★☆☆☆☆	1.0
Unit complexity	★☆☆☆☆	1.1
Unit interfacing	★★☆☆☆	1.5
Module coupling	★★☆☆☆	1.8
Component balance	★★☆☆☆	2.4
Component independence	★★★★★	5.(

★☆☆☆☆ 1.3

System properties

Volume	★☆☆☆☆	1.2
Duplication	★★☆☆☆	2.0
Unit size	★☆☆☆☆	0.5
Unit complexity	★☆☆☆☆	0.6
Unit interfacing	★☆☆☆☆	1.0
Module coupling	★★☆☆☆	1.7
Component balance	★☆☆☆☆	0.6
Component independence	★★★☆☆	2.8

★★☆☆☆ 1.9

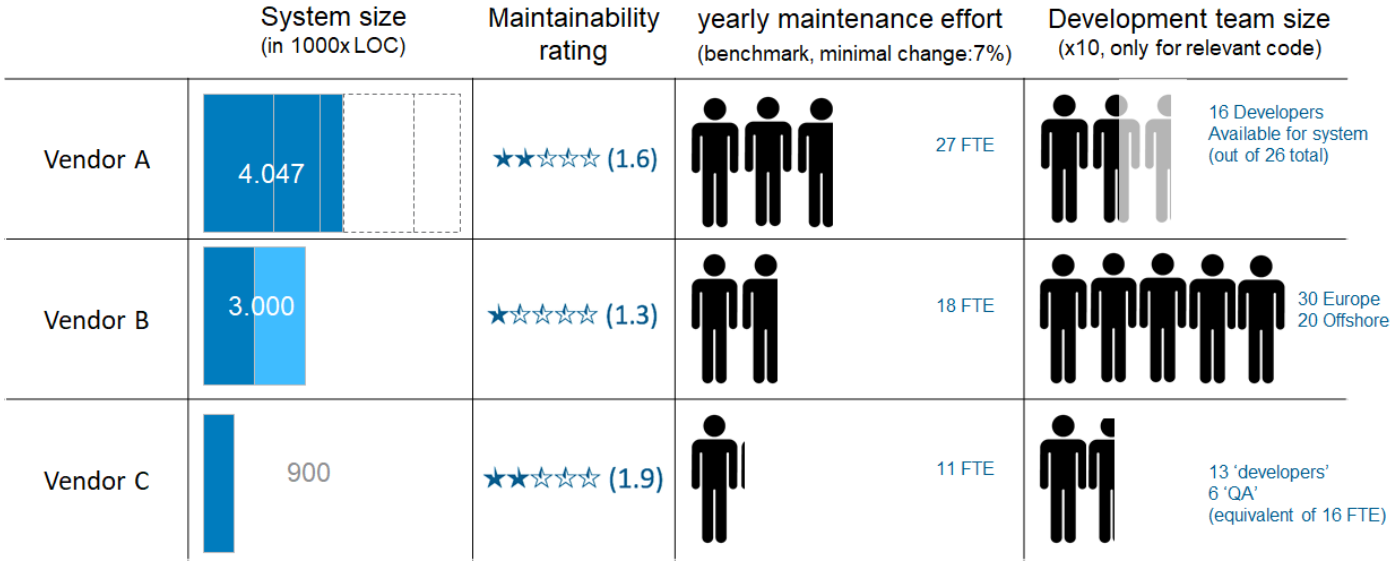
System properties

Volume	★☆☆☆☆	1.0
Duplication	★★☆☆☆	1.7
Unit size	★★☆☆☆	1.9
Unit complexity	★★☆☆☆	2.1
Unit interfacing	★☆☆☆☆	1.3
Module coupling	★★★☆☆	2.9
Component balance	★☆☆☆☆	1.0
Component independence	★★★☆☆	3.3

Customer case 1: Evaluation eines ERP Systems

Berieselung durch Anbieter → **Sourcecode analyse** → Lastenheft → Offert-präsentation → Verhandlung & Lieferanten-auswahl → Vertrags-gestaltung → Entwicklung & Testing → Einführung → Betrieb & Wartung

System size vs Development capacity



6. Customer Cases

Case 2: Kosten für Historisierungskonzept

Quote challenge

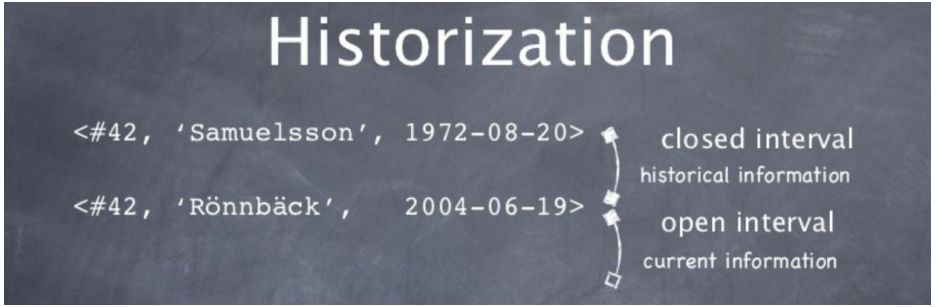
Customer case 2: Kosten für Historisierungskonzept

Situation

- Public sector organization
- Requests supplier to add historization concept in the software
- The customer is surprised by the height of the quote (100.000 CHF)

Questions

- Is the solution the vendor's team proposes architecturally sound?
- What is the expected effort to implement it?
- Provide technical arguments for the negotiation with the vendor.



Customer case 2: Kosten für Historisierungskonzept

Action

- Review of the proposed plans of the vendor
- Interview with the vendor's team
- Calculation of effort based on planned code changes.

Result

- Advice : the change has a standard proven solution. The proposed solution was insufficient in some points.
- The expected effort was 1/4 of the original quote
- Change was executed as part of the regular maintenance contract.

- 100.000 CHF scheint mir nicht gerechtfertigt. Ich würde eine Grössenordung von 1-2 Monaten erwarten (ca 20.000 CHF). Und da denke ich zu vertreten ist, dass das Aufsetzen der Historisierung schon zum ursprünglichen Vertrag gehöhrte, schätze ich den Mehraufwand für [redacted] auf maximal 1-2 Wochen. Also praktisch noch eine Kulanzleistung). Die Bemerkung zur

6. Customer Cases

Case 3: Überwachung der Qualität in einem Neubauprojekt

Software monitoring

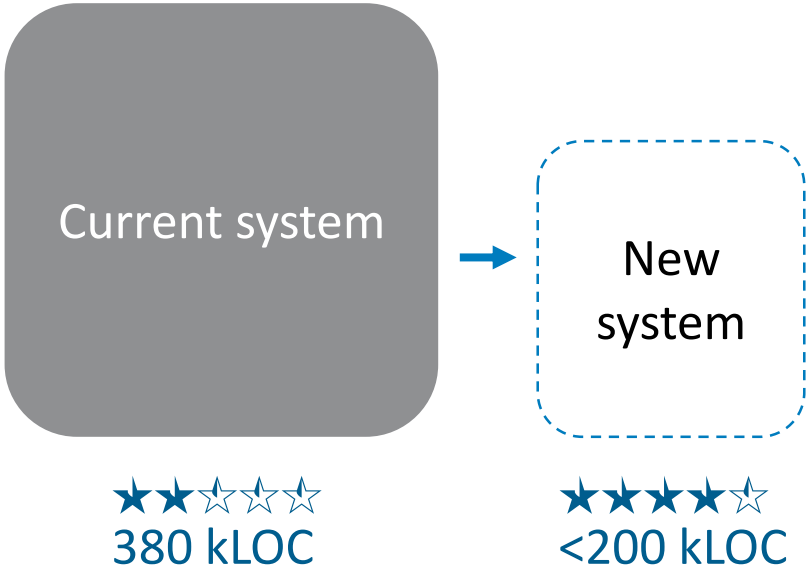
Customer case 3: Überwachung der Qualität in einem Neubauprojekt

Situation

- Public sector organization embarks in rebuild of an existing application with new external supplier.
- Project size: 2 years, about 4 million CHF

Questions

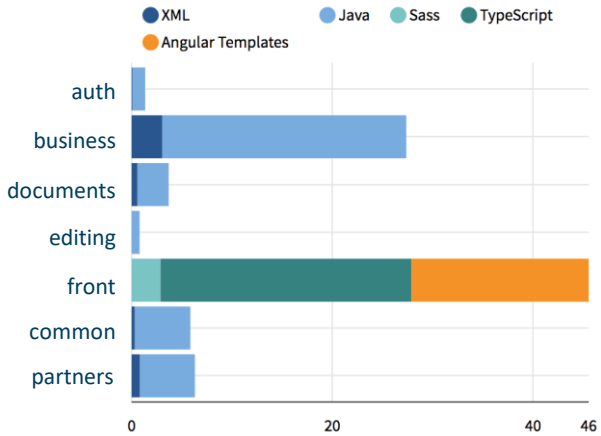
- Is the technical quality of the new application good?
- Does the development team work according to best practices?
- What are the risks associated with the technical quality?
- Which actions need to be taken to reduce the risks?



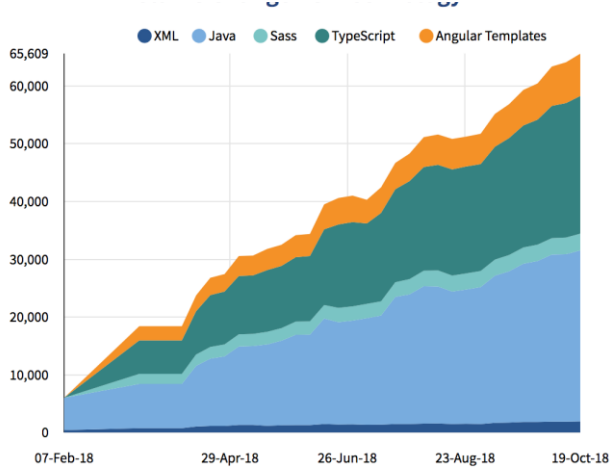
Customer case 3: Überwachung der Qualität in einem Neubauprojekt

Some overviews for the system under development

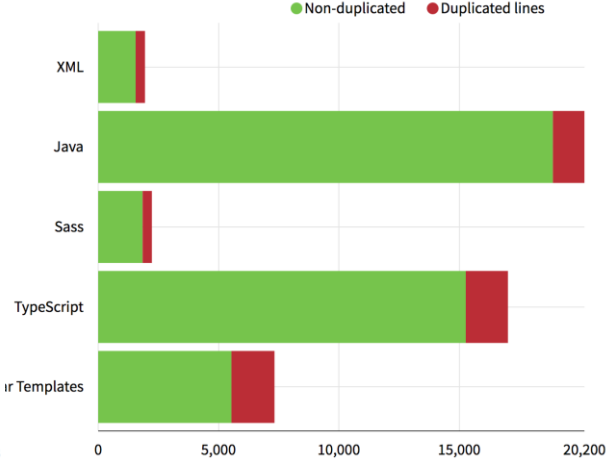
Volume per component (in man month)



Volume change per technology



Duplication per technology



Customer case 3: Überwachung der Qualität in einem Neubauprojekt

Proprietary libraries and **copyright** from the supplier are found in the code

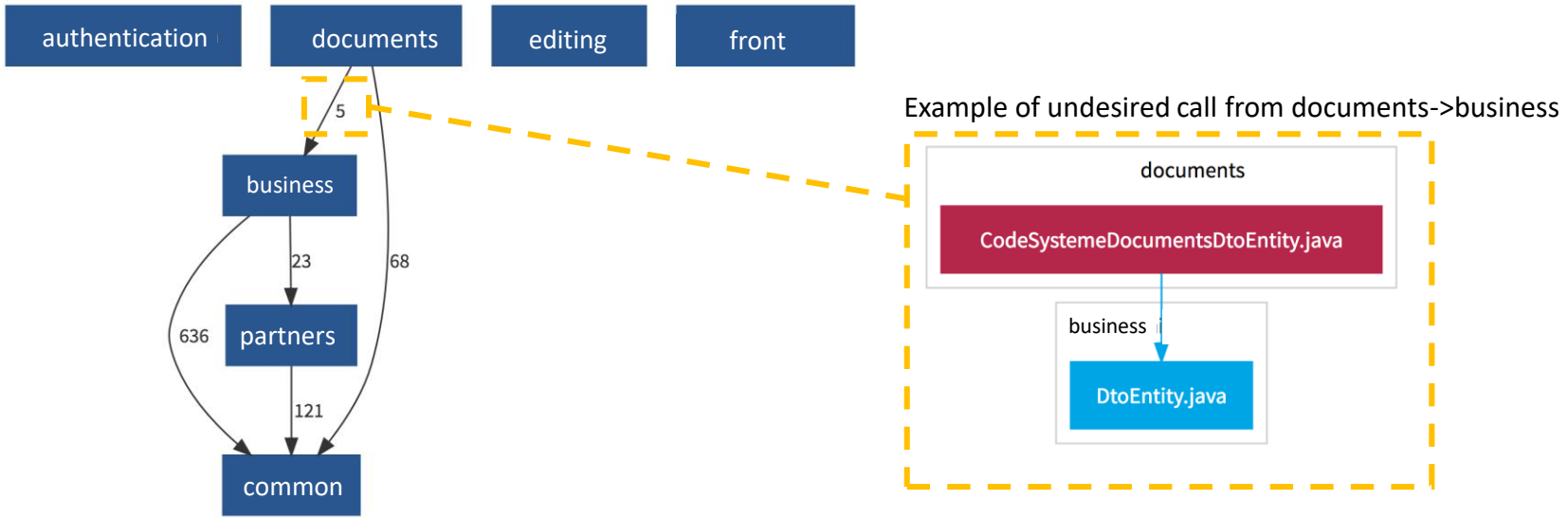
- The codebase does not contain the code for the proprietary [redacted] library
- Who owns the library code?
- Who has the IP? Why does the supplier name appear in package naming?

```
1 /* © 2017 - [redacted] - http://www[redacted].ch */
2
3 package ch.[redacted].back.infrastructure.search.elasti
4
5 import org.springframework.stereotype.Repository;
6 import ch.[redacted].back.domain.model.dossierAssure.Do
7 import ch.[redacted].back.domain.model.dossierAssure.Do
8 import ch.[redacted].socle.infrastructure.search.elasti
```

Customer case 3: Überwachung der Qualität in einem Neubauprojekt

Some dependencies violate architecture guidelines.

- E.g. between components ***documents*** and ***business***
- This proves the development team does not have sufficient controls in place



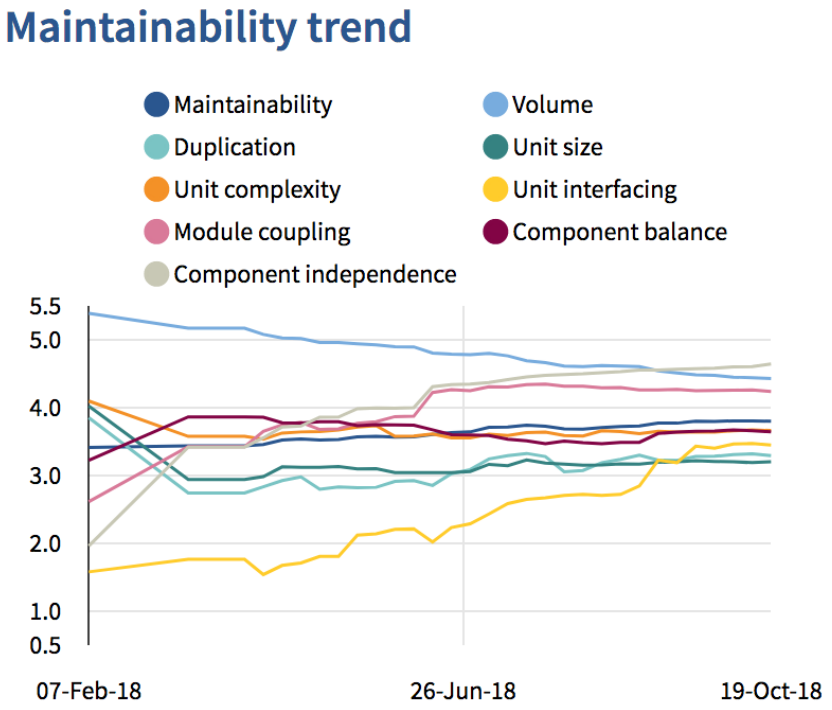
Customer case 3: Überwachung der Qualität in einem Neubauprojekt

Action

- Monitoring of code quality. Continuous automated and manual measurements.
- Monitoring of productivity
- Coaching the supplier's team on establishing a qualtiy development process according to best practices.
- Architecture review

Result

- Supplier was confronted with lack of quality in the code which would result in high maintenance costs..
- The supplier has switched to a proactive mode concerning code quality
- Current maintainability of the software is 4-star
- The supplier is professionalizing its development process



6. Customer Cases

Case 4: Scenario evaluation: Neubau oder Renovierung?

Software Risk Assessment

Customer case 4: Scenario evaluation: Neubau oder Renovierung?

Situation

- Public sector organization suffers from unmaintaintainable system
- Needs insight for decision making
- System rebuild value 5-10 Mio CHF

Action

- Software Analysis of the current application
- Architecture analysis
- Cost estimation
- Scenario evaluation

Questions

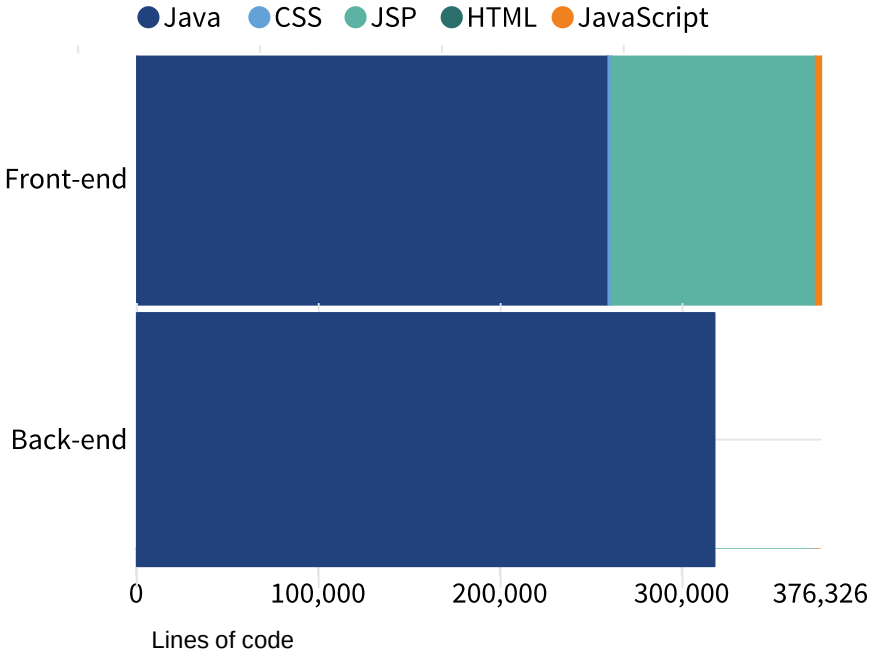
- What are the main technical drivers for poor maintainability of the system?
- Is the current system architecture sound?
- Can we renovate or is a rebuild necessary?
- Need all parts to be rebuild?
- Where is renovation needed?

Result

- Customer decided to rebuild the frontend and later renovate the backend based on the analysis
- The analysis report was part of the request for proposal to suppliers.
- Cost estimations were used in the contract negotiations with the suppliers.

Customer case 4: Scenario evaluation: Neubau oder Renovierung?

XXX is a large system with a rebuild value of 89 man years



Frontend

Technologies	Lines of code	Rebuild value (MM)
Java	259,215	355
JSP	112,191	271
JavaScript	3,086	3
CSS	1,834	2
Total Frontend	**376,326 LOC**	**631 MM (= 53 MY)**

Backend

Technologies	Lines of code	Rebuild value (MM)
Java	317,632	434
Total Backend	**317,632 LOC**	**434 MM (= 36 MY)**

Total AVAM	693.958 LOC	89 MY

Customer case 4: Scenario evaluation: Neubau oder Renovierung?

Rebuild versus Renovate

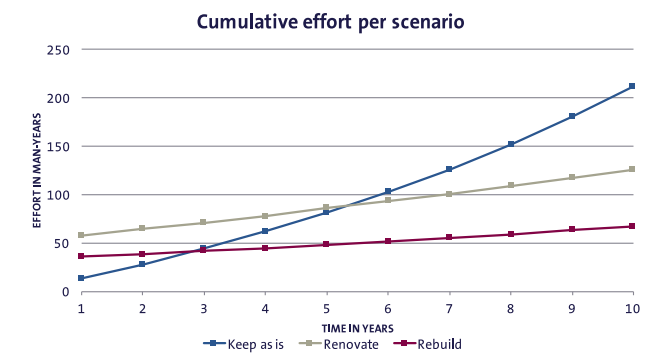
Rebuilding the *frontend* is economically sound

- Given the current quality and size of the frontend it is cheaper and quicker to rebuild than to renovate.
- The investment of a rebuild will break-even with the ‘keep as is’ scenario after 2 - 3 years

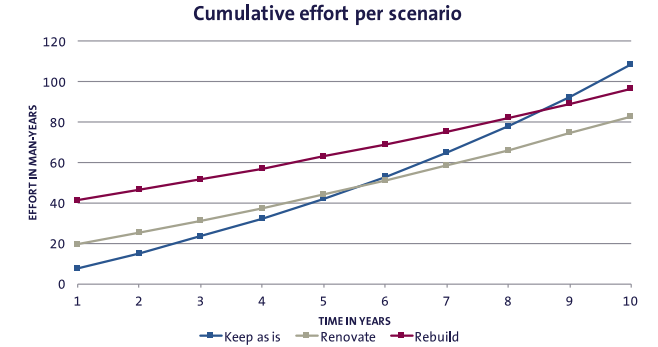
Renovation of the *backend* is economically sound

- After 5 - 6 years the investments will break-even with the ‘keep as is’ scenario.

Frontend scenarios



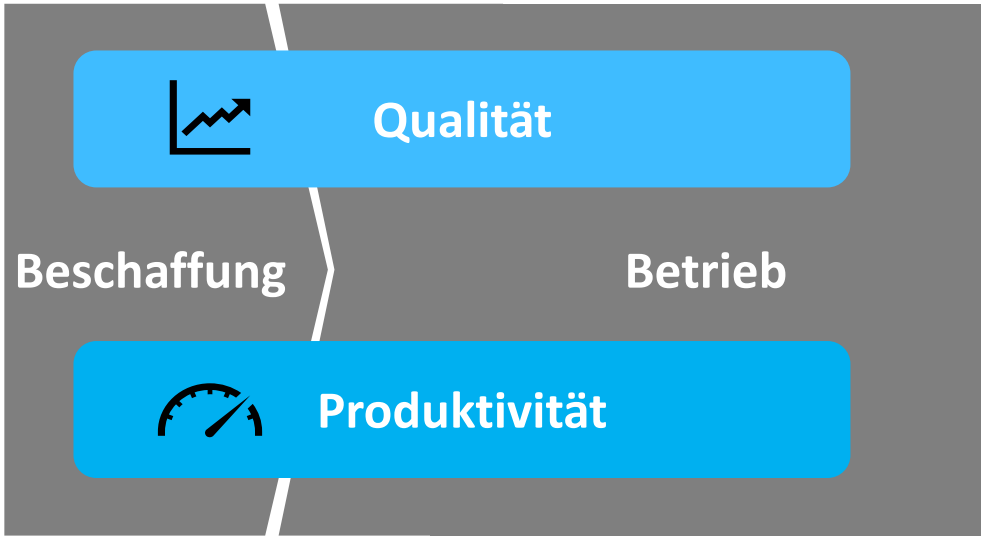
Backend scenarios



8. Anwendung und Diskussion

Ideenentwicklung (Gruppenarbeit 10min)

- Wie würden Sie mit dem neuen Wissen den Herausforderungen begegnen?
- Welche Chancen/Ideen sehen Sie für die zukünftige Projekte?



Zusammenfassung

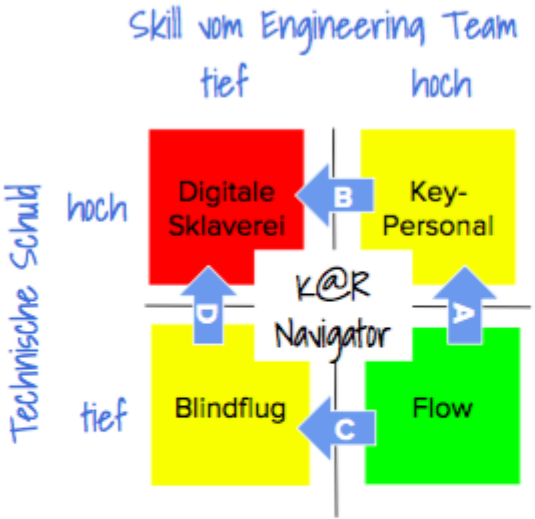
Qualität und Leistung in der Softwareentwicklung sind messbar!

Werden Qualität und Leistung gemessen führt dies zu:

- Tiefere Projektkosten
- Tiefere Wartungskosten
- Grössere Flexibilität
- Kürzere Time to Market
- Bessere Entscheidungsgrundlagen
- Nachhaltigere Software
- Nachhaltigere Partnerschaften

9. The human factor: K@R Erfolgsformel

Software-Änderungs-Kosten/-Risiko ist der vitale Indikator eines jeden digitalen Unternehmens. Werden Software-Releases immer teurer und/oder riskanter, so wird die (Legacy-) Software anstelle eines digitalen Enabler zu einem Fessel am Fuss (Digitale Sklaverei).



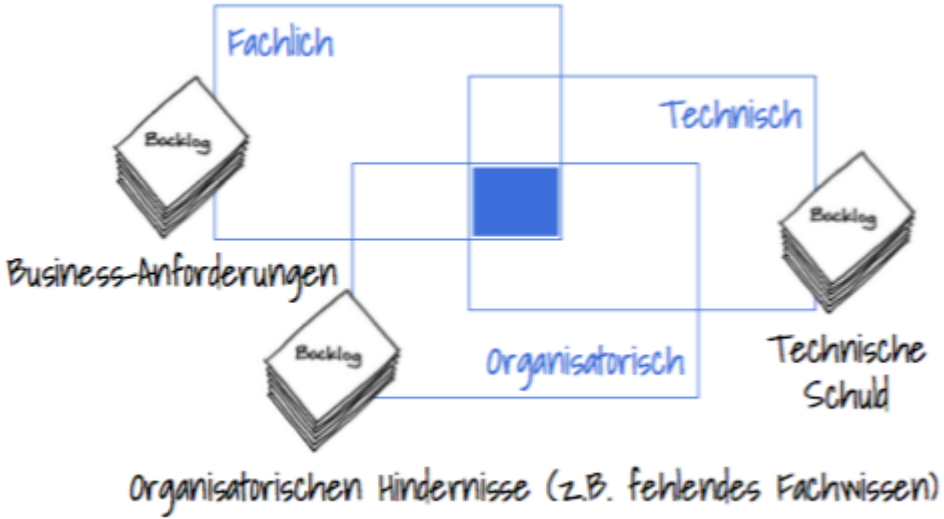
A: Ein zu starker Business-Delivery-Druck auf das Engineering-Team führt zu einem Anstieg der technischen Schuld.

Als technisch verschuldet bezeichnet man eine schlecht geschriebene resp. dokumentierte "Spaghetti"-Software, welche nur das Key-Personal beherrscht.

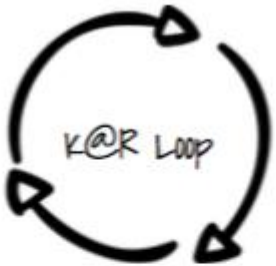
B: Engineers resignieren und verlassen das Team, wenn sie ihren Job wiederholt unter Druck nicht sauber ausführen dürfen.

C: Engineers verlassen das Team, wenn sie nicht adäquat gefordert werden.

D: Fehlender Skill im Team führt zu einer versteckten Inflation von technischer Schuld.


Fachlich
Technisch
Organisatorisch
Backlog
Backlog
Backlog
Business-Anforderungen
Technische Schuld
Organisatorischen Hindernisse (z.B. fehlendes Fachwissen)

1. Wende den K@R Navigator an.

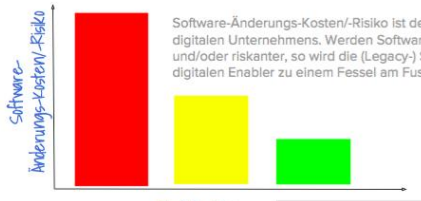


2. Schneide Arbeitspakete nach der K@R Balanced Scorecard zu.

3. Erledige die Arbeitspakete.

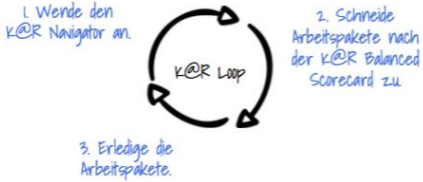
Schneide die Arbeitspakete (Wer macht was) so zu, dass sie in der Überlappung der drei Backlogs liegen, um deine Digitale Fitness hoch zu halten.

Knowledge@Risk
Erfolgsformel (no agile inside)

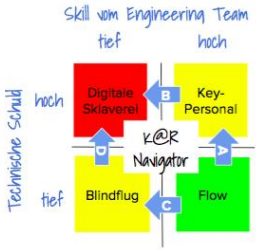


Software-Änderungs-Kosten/-Risiko ist der vitale Indikator eines jeden digitalen Unternehmens. Werden Software-Releases immer teurer und/oder riskanter, so wird die (Legacy-) Software anstelle eines digitalen Enablers zu einem Fessel am Fuss (Digitale Sklaverei).

... unterliegt einer Dynamik



Schneide die Arbeitspakete (Wer macht was) so zu, dass sie in der Überlappung der drei Backlogs liegen, um deine Digitale Fitness hoch zu halten.



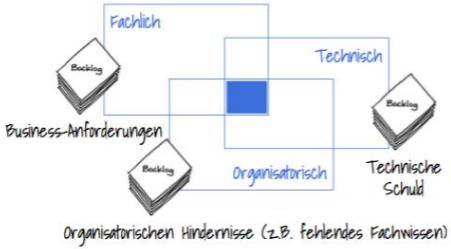
A: Ein zu starker Business-Delivery-Druck auf das Engineering-Team führt zu einem Anstieg der technischen Schuld.

Als technisch verschuldet bezeichnet man eine schlecht geschriebene resp. dokumentierte "Spaghetti"-Software, welche nur das Key-Personal beherrscht.

B: Engineers resignieren und verlassen das Team, wenn sie ihren Job wiederholt unter Druck nicht sauber ausführen dürfen.

C: Engineers verlassen das Team, wenn sie nicht adäquat gefordert werden.

D: Fehlender Skill im Team führt zu einer versteckten Inflation von technischer Schuld.



Wobei, jede Software-Änderung ist eine Chance drei Bedürfnisse zu erfüllen. Daraus ergibt sich die...

K@R Balanced Scorecard

Kunden & Kontaktangaben

Kontakt

Marc André Hahn
sieber&partners
Schwanengasse 1, Bern
Usteristrasse 19, Zürich
+41 78 686 85 16
marc.hahn@sieberpartners.com

Kay Grosskop
sieber&partners, SIG
Schwanengasse 1, Bern
Usteristrasse 19, Zürich
+41 31 566 93 00
kay.grosskop@sieberpartners.com

Kunden aus der Schweiz

IBM, logifleet, SCS super computing systems, DIE POST, swisslex, swisstopo, glaströsch, swisscom, GIDA.I L'informatique des Offices AI, MIGROS, A+W, seco, Thalia, bbv MAKING VISIONS WORK.

Kunden weltweit

ING, vodafone, POLITIE, T-Mobile, KLM, SKAT, PHILIPS, DHL, INTERAMERICAN PART OF ACHMEA, BMW, kadaster, postnl, HUAWEI, ProRail, nationale nederlanden, THALES, Rijkswaterstaat, PARTENA, ABN·AMRO, Nykredit, POLITI

digital business excellence

sieber & partners

Bern Zürich Triesen