Begleiten von Großprojekten aka „Architecture Governance“

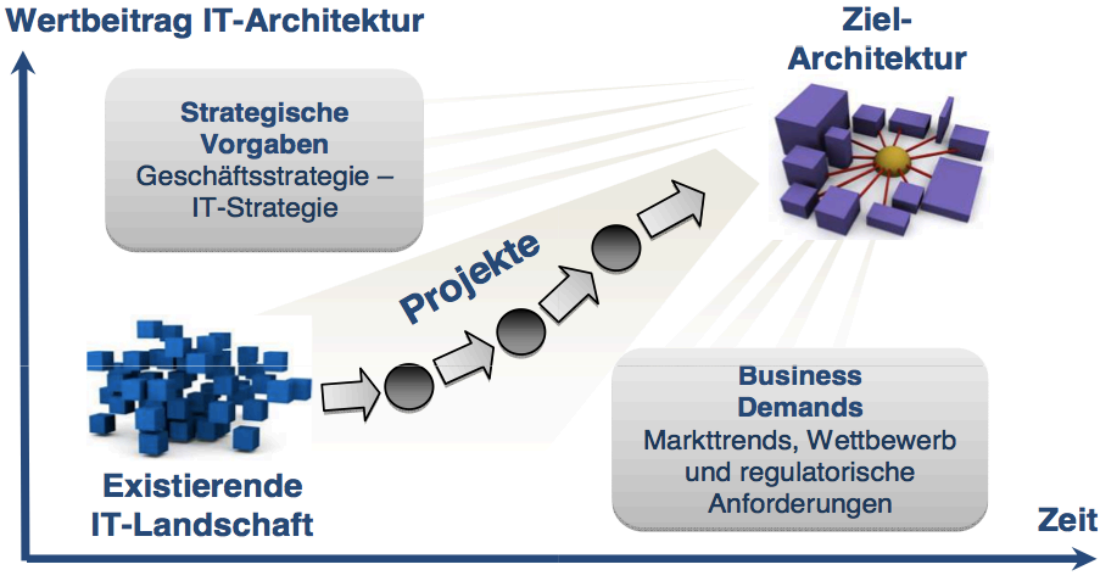
Architekturmanagement, Projektmanagement, Umgang mit großen Projekten

VL 09; Donnerstag 20. Mai 2010; Raum HPI B-E.2

Fachgebiet Software-Architekturen, Prof. Dr. Robert Hirschfeld
Dipl.-Inform. (univ.) Wolfgang Keller,
wolfgang.keller@objectarchitects.de

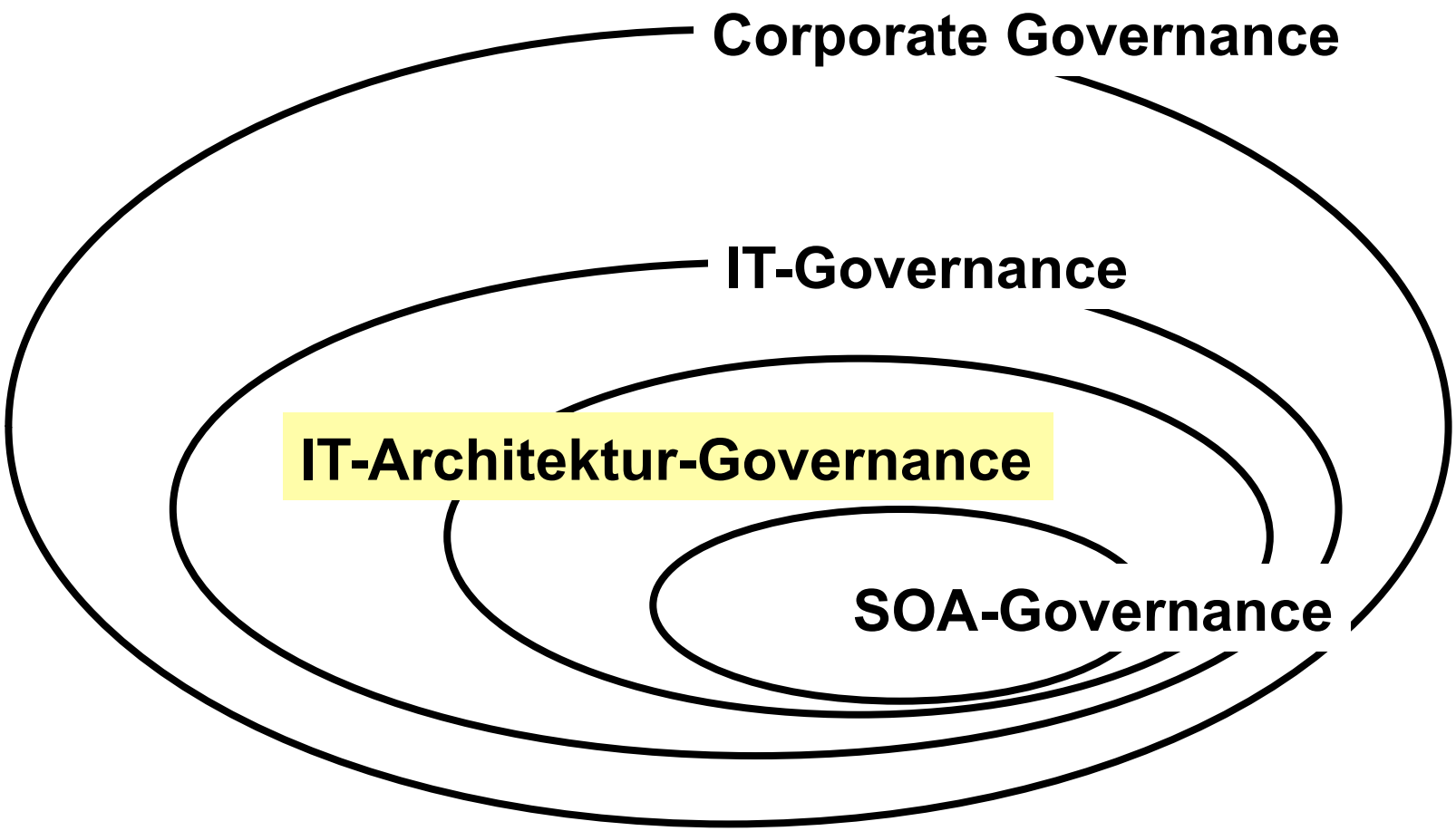
Die Frage zur Architektur-Governance

- Sie haben also einen Plan, um von einem Ist zu einem Soll zu kommen:



- Frage ist dann: Wie sorgen Sie dafür, dass das auch wirklich passiert?

Neudeutsch heißt das, worüber wir reden auch Architecture Governance ...



Das Gesamtsystem ..

Was müssen Sie alles regeln, um IT Governance zu „betreiben“

Hasso Plattner Institut

Zeit

Sie brauchen eine IT-Strategie

Sie erstellen einen strategischen Bebauungsplan und einen strategischen IT-Plan (Vorausschau 3-5 Jahre)

Sie erstellen eine rolliende Projekt-Jahresplanung (Projektportfolio-Management)

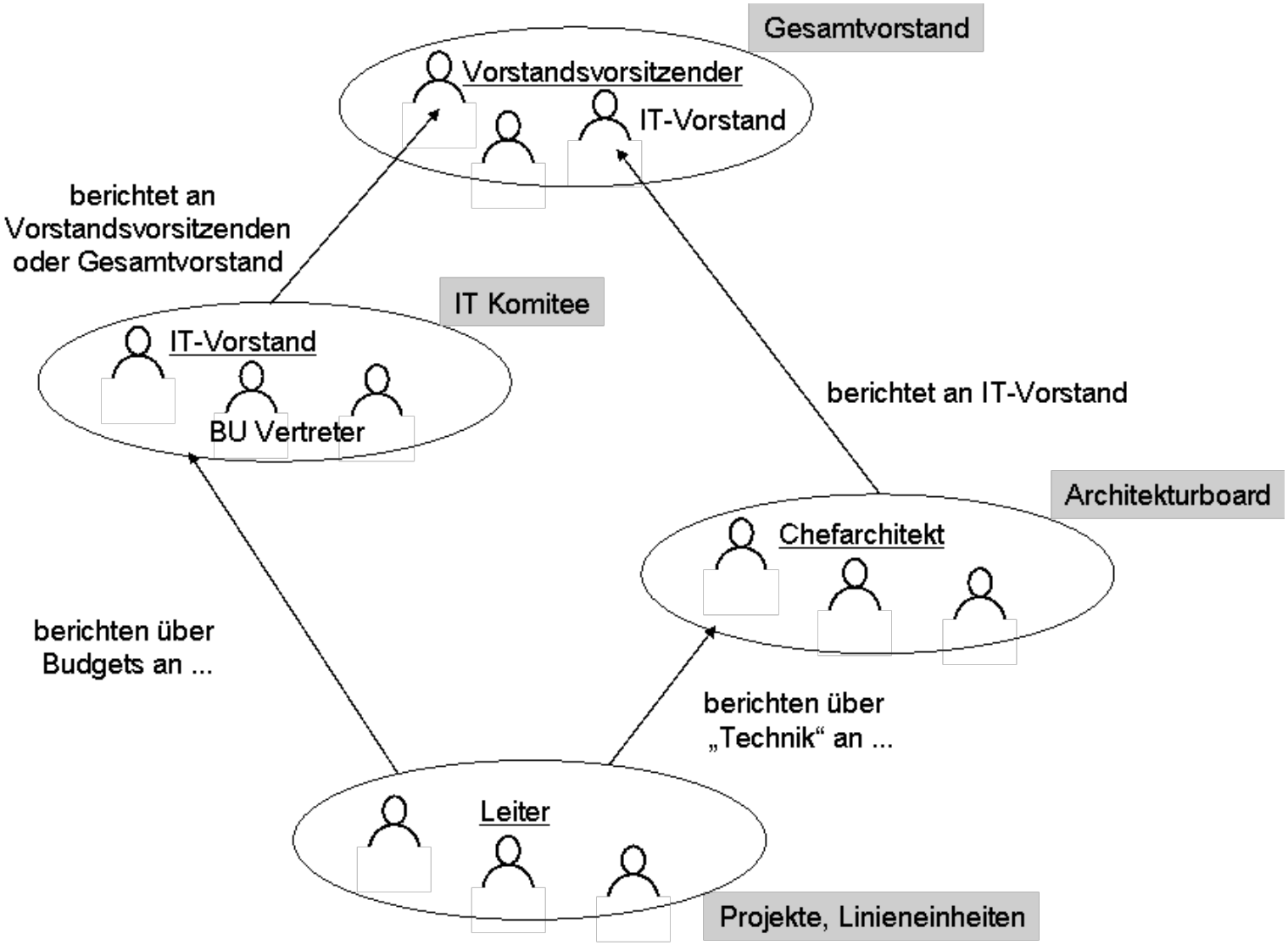
Sie tracken die Umsetzung der Projekte (Project Governance)

Sie setzen Ihre Planung in den Projekten durch (Architecture Governance)

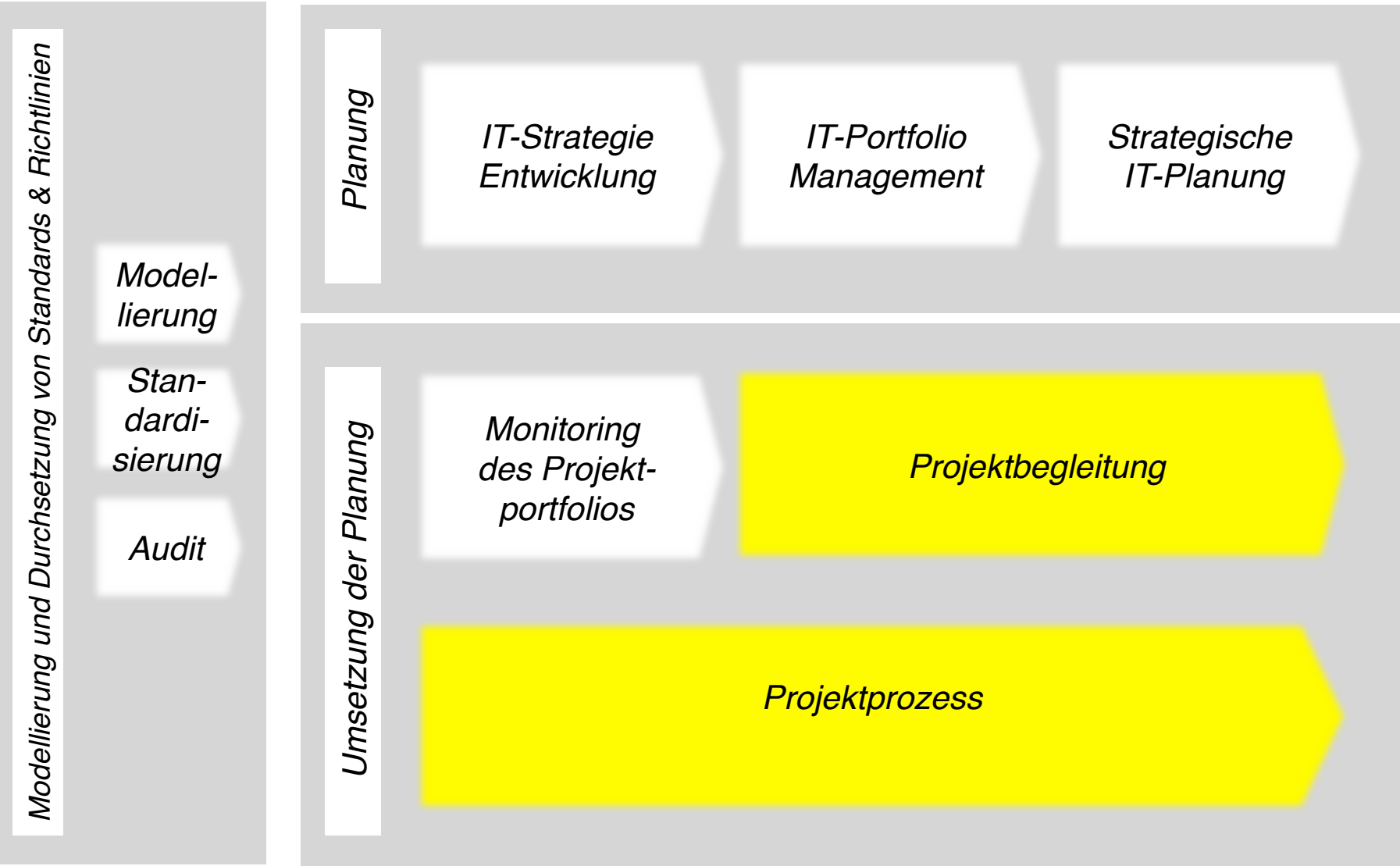
Was deckt das ab?

gebräuchliche IT Governance Modelle z.B. ITGI	IT Governance Modell Weill
IT-Strategie	IT Principles
IT-Betrieb	IT Infrastructure Strategies
IT-Architektur	IT Architecture
	Business Application Needs
IT-Programm-Management	IT Investment and Prioritization
IT-Controlling	
IT-Human-Ressources-Management	
IT-Risikomanagement	

Und welche Gremien findet man häufig?

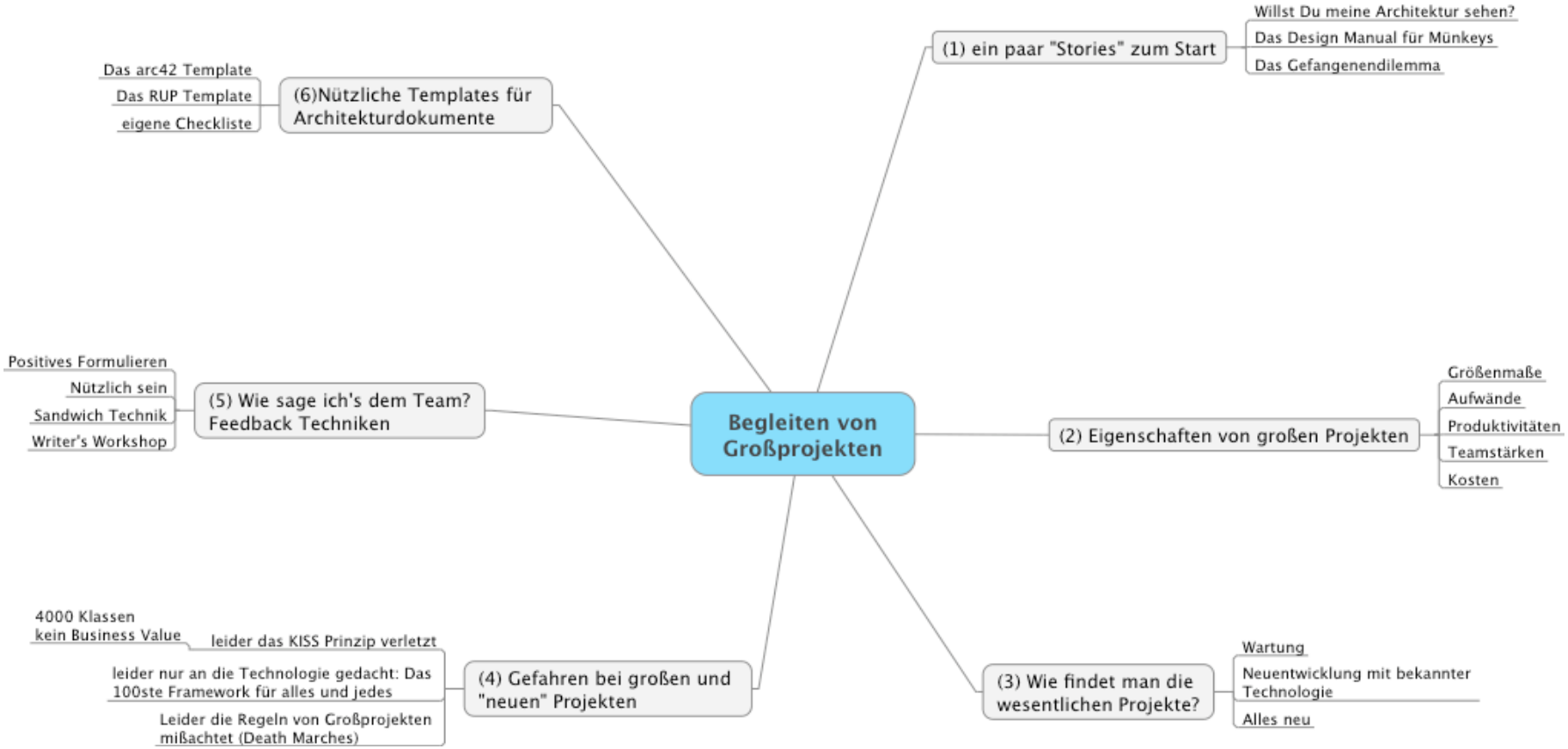


Standort in der Vorlesung



Was benötigt man für die Umsetzung ..

Überblick als Mindmap



Überblick

- Ein paar Stories zum Start ..
- Wie findet man die wesentlichen Projekte, die begleitet werden sollten?
- Eigenschaften von großen Projekten Gefahren bei großen und „neuen“ Projekten
- Wie sage ich es dem Team? Feedback Tipps
- Nützliche Templates für Architekturdokumente

Um es vorweg zu sagen ...

Unternehmensarchitekten sind bei den einzelnen Projekten nicht immer beliebt

Story 1: Regeln für eine sinnvolle Aufgabenteilung Enterprise Architecture / Project Architecture

- Nicht ganz unrealistische Situation vor der Auslieferung: Enterprise Architect (EA) besucht Project Architect (PA)

- EA: Darf ich mir bitte mal Ihre Architekturdokumentation ansehen? Ich würde gerne mal sehen, ob die zu den unternehmensübergreifenden BluePrints passt.

- PA: Sie wollen meine Architektur sehen? Die steht im Code! Außerdem - ich habe jetzt keine Zeit - in 8 Wochen ist Auslieferung und ich muss jetzt in ein Projektmeeting gehen. In 6 Wochen ist Code Freeze.

Story 1 (contd) Und nach der Auslieferung Projekt hatte ein paar Probleme :-)

- EA: Und wie war das doch jetzt gleich mit der Architektur, die im Code steht?

- PA: Kein Wunder, dass das Projekt Probleme hat. Uns haben immer klare Direktiven von der Unternehmensarchitektur gefehlt. Sie haben und ja nicht klar gesagt, was wir machen sollen

(Story 1: Resumée) Ein Projektarchitekt ist voll verantwortlich für die technische Integrität seines Projektes. Nur weil es eine Unternehmensarchitektur gibt, hat er nicht weniger Verantwortung

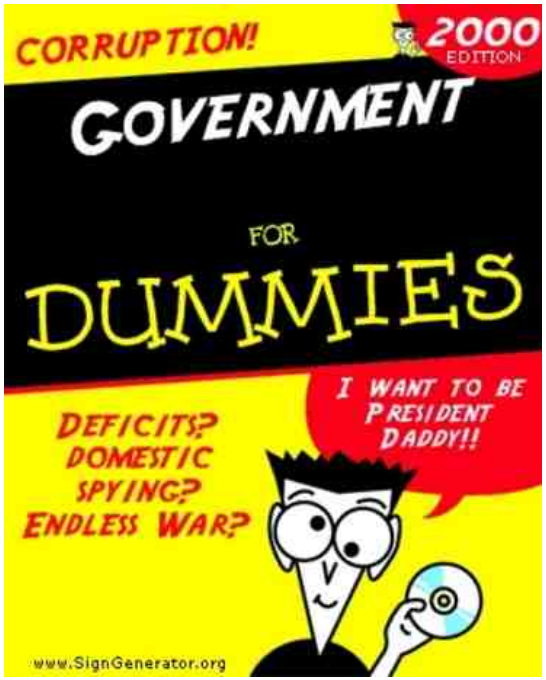
Unternehmensarchitekten ..

- sind für das Anwendungsportfolio der Gesamtfirma verantwortlich
- Kommunizieren mit den Projektarchitekten
  - ob es im Portfolio des Unternehmens Dinge gibt, die man besser wiederverwendet
  - wann fehlende Komponenten von wem zugeliefert werden
  - Ob das Projekt zu den unternehmensweiten Konstruktionsprinzipien passt
- Haben eine QS-Funktion

Projektarchitekten ..

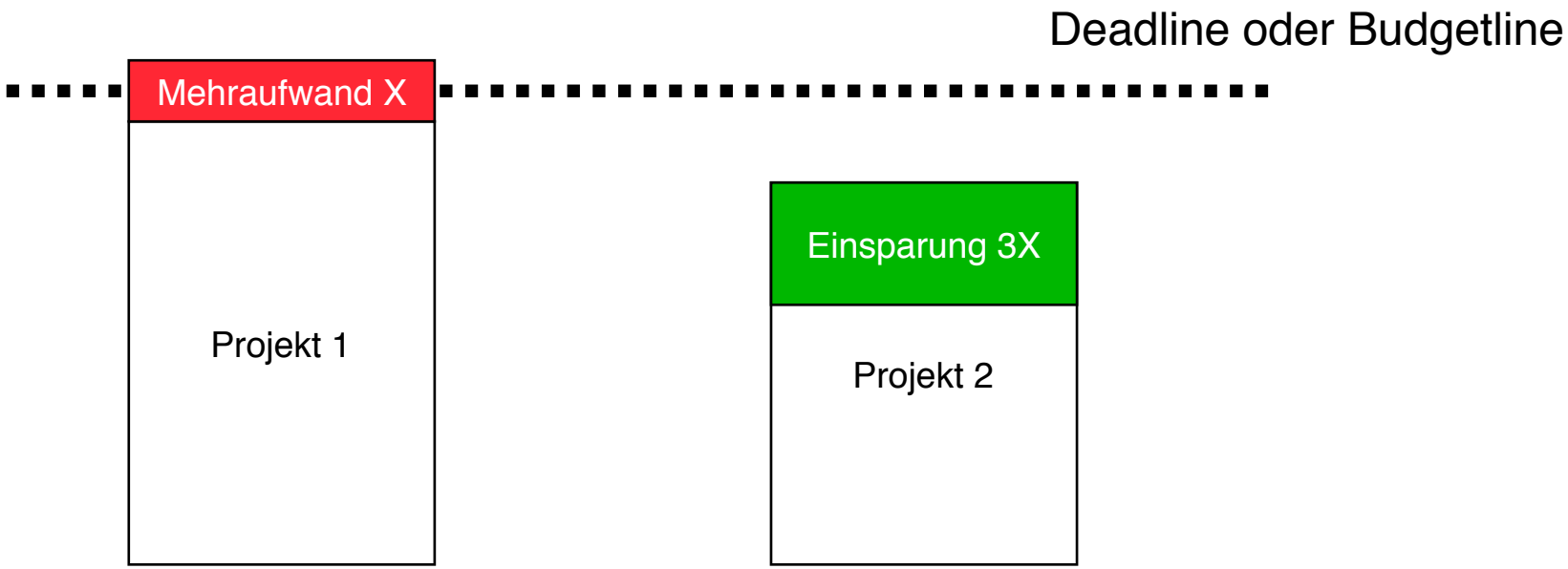
- sind voll dafür verantwortlich, dass ihr Projekt am Tag der Auslieferung funktioniert
- Benutzen dafür Komponenten aus dem Portfolio ihres Unternehmens
- Entwickeln neue Komponenten, wenn dies mit der Unternehmensarchitektur abgestimmt ist
- Können jede Situation eskalieren, wenn sie Ausnahmeregelungen benötigen oder sich sonst wie durch Regeln oder Termine beeinträchtigt fühlen und das ihr Projekt gefährden würde

Fortsetzung => Story 2: Glauben Sie, dass ein solches Buch etwas helfen würde :-)



- Projektmitarbeiter: Ich brauche detaillierte Vorgaben für meine Architektur
- Unternehmensarchitekt: Ich kann leider nicht für alle die Arbeit machen - dann werde ich nicht mehr fertig und bin völlig ineffektiv

Story 3: Kosten sparen durch Architektur Gefangenendilemma



- wie erklären Sie als Unternehmensarchitekt dem Projektleiter von Projekt 1, dass er X ausgeben soll, um der Firma insgesamt 2X einzusparen, wenn er von seinem Chef genau an der Linie gemessen wird?

Das Thema ist auch bekannt als Chicken Race

- das bekannte Autorennen
- keiner will nachgeben
- wer zuerst „rüberzieht“ verliert
- wenn beide in der Mitte weiterfahren, haben sie leider einen Frontalunfall und beide verlieren

- Diese Situation gibt es in Unternehmen gar nicht so selten
  - zwischen Fachabteilung und IT
  - oder zwischen Projekten

Wie findet man die wesentlichen Projekte, die begleitet werden sollten?

Charakteristika eines Projektportfolios

- Ein Projektportfolio umfasst typisch eine vierstellige Anzahl Maßnahmen von wenigen PT bis zu Hunderten Personenjahren

- Um die 25% des Aufwandsvolumens eines Projektportfolios sind Wartungs- und Kleinaufträge
- der Rest sind normale bis sehr große Projekte
- die wenigsten davon sind wirklich innovativ
  - viele verwenden bewährte Technologie
  - und ändern die Architektur des Unternehmens nicht wesentlich

- Wenige Projekte ändern die Architektur
  - durch Einstieg in neue Technologien
  - durch Einstieg in neue fachliche Bereiche

- Und dann gibt es noch die Leute, die schlicht gegen Architekturprinzipien verstoßen
  - aus Bequemlichkeit
  - oder Unwissenheit

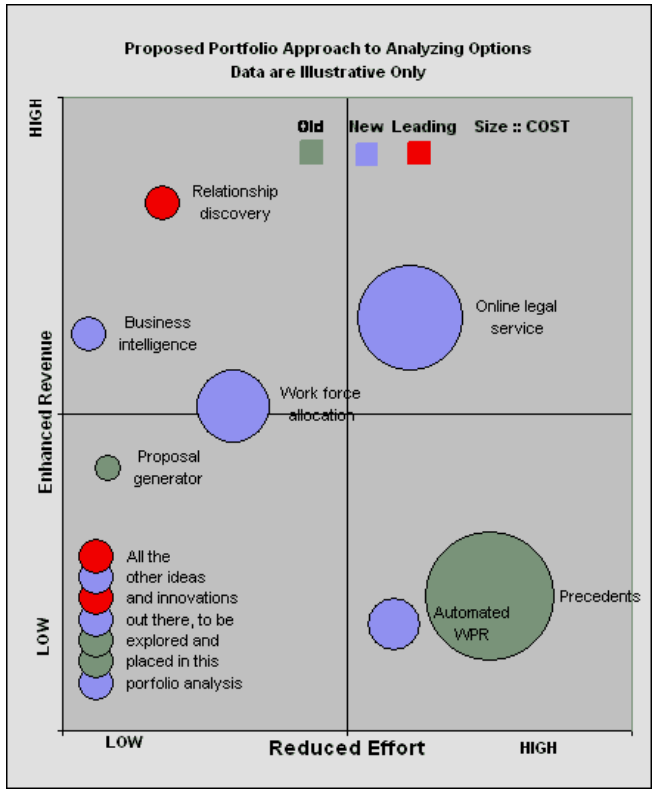
Was suchen Sie ....

- Sie suchen alle Projekte,
  - in denen Dinge im größeren Umfang fachlich oder technisch neu gemacht werden und die damit die Gesamtlandschaft verändern
  - die die Heterogenität des Portfolios steigern - weil dadurch die Kosten steigen würden
  - Die aufgrund ihrer Größe oder Neuigkeit für Ihr Unternehmen gefährlich werden könnten
  - In denen aufgrund von Architekturfehlern Probleme für Ihr Unternehmen entstehen ..

Die Probleme die Sie suchen, können fachlich oder technisch sein

- Bespiel für technisches Problem:
  - Jemand implementiert Bäume in einer relationalen Datenbank und wundert sich, dass die Performance schlecht ist
  - Findet man mit Architektur-review und/oder Kollegenreview
- Beispiel für fachliches Problem
  - Jemand implementiert Partnerfunktionalität in einem System zum Beispiel für Verträge, wo sie nichts zu suchen hat
  - Findet man mit Architektur-review und/oder Kollegenreview

Und wie finden Sie die? Ausschlussprinzip



- sehen Sie sich die Projektliste periodisch an
- sortieren Sie die harmlosen Sachen aus
- und schauen Sie sich das an, was neu rein gekommen ist und „verdächtig“ aussieht, weil
  - groß
  - neu
  - schlecht beschrieben
  - unklar
  - nicht einzuordnen
  - ....

Eigenschaften von großen Projekten

Das sind die, die Sie nicht suchen müssen,
sondern die zu Ihnen gebracht werden

Überblick
Große Softwaresysteme

- Was ist groß?
  - Function Points
- Ein großes Projekt – Fallstudie Phoenix
- Einzelprojekte und Anwendungsportfolio
- Projektlaufzeiten früher und heute
- Death March Projekte

Großes Softwaresystem
Wie kann man das messen

- Lines of Code
  - Problem: Assembler != C++ != Smalltalk
- Aufwand für die Erstellung
  - Problem: Produktivität von Projekten sehr unterschiedlich
- Anzahl der Masken, Eingabefelder, Datenbanktabellen, Ausdrucke ..
  - nicht immer akkurat, aber besser, als LoC

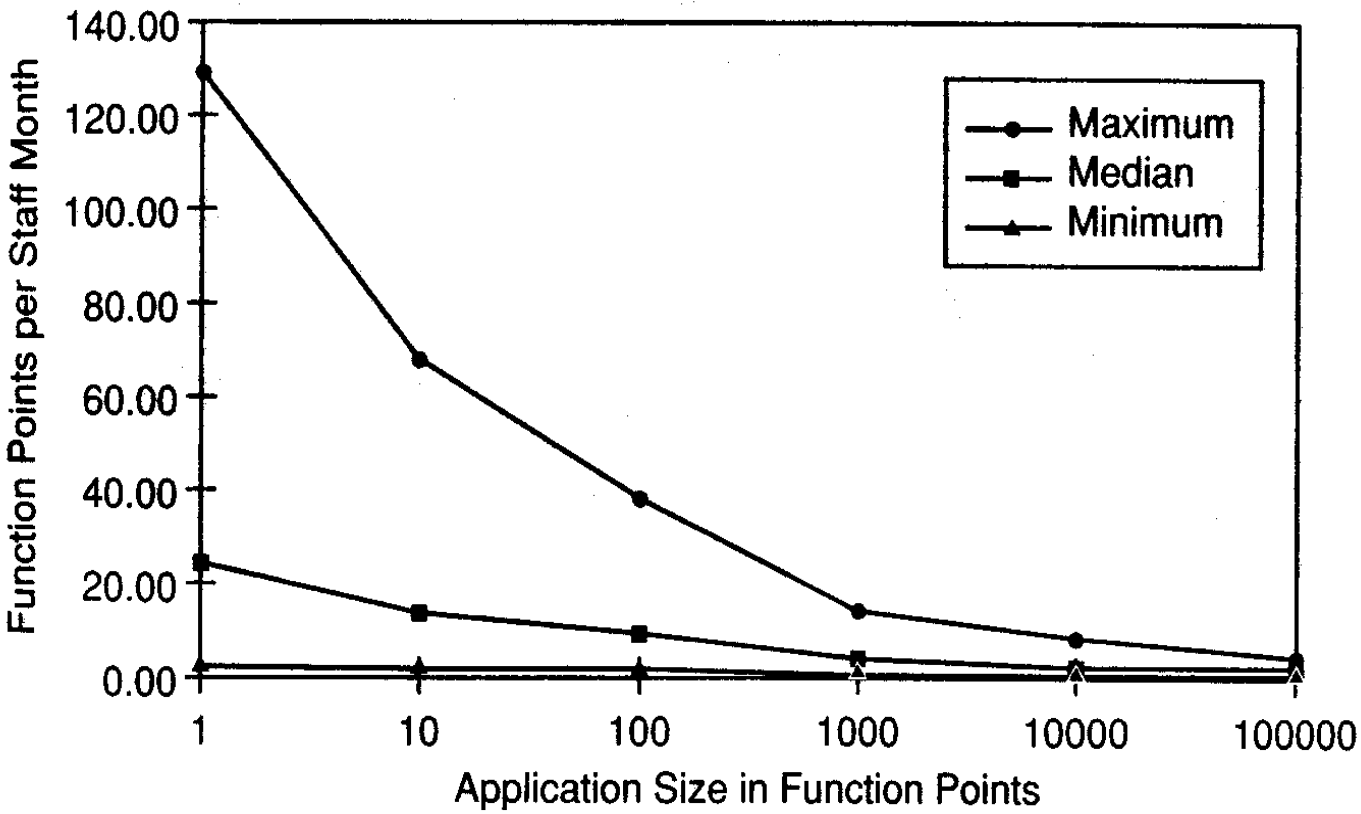
LoC bei gleicher „Programmgröße“

Table 1: Function Point and Source Code Sizes for 10 Versions of the Same Project (A PBX Switching System of 1,500 Function Points in Size)

Language	Size in Func. Pt.	Lang. Level	LOC per Func. PT.	Size in LOC
Assembly	1,500	1	250	375,000
C	1,500	3	127	190,500
CHILL	1,500	3	105	157,500
PASCAL	1,500	4	91	136,500
PL/I	1,500	4	80	120,000
Ada83	1,500	5	71	106,500
C++	1,500	6	55	82,500
Ada95	1,500	7	49	73,500
Objective C	1,500	11	29	43,500
Smalltalk	1,500	15	21	31,500
Average	1,500	6	88	131,700

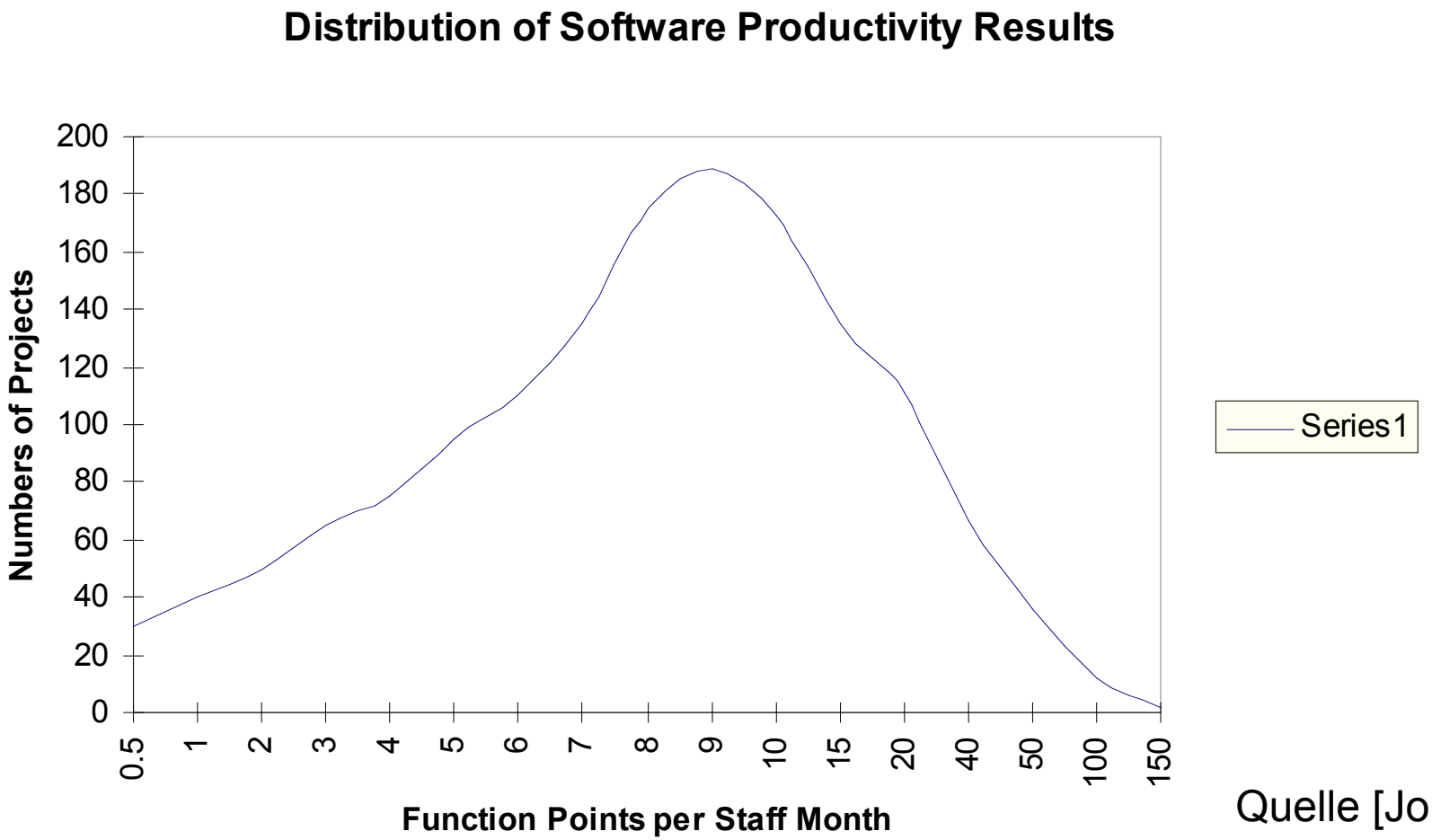
Quelle [Jones1997]

ein Einflussfaktor ist die Projektgröße



Quelle [Jones1996]

Gemessene Produktivitäten unterscheiden sich stark



Quelle [Jones1998]

Arbeit in verschiedenen Programmiersprachen ist nicht gleich produktiv

Table 2: Staff Months of Effort for 10 Versions of the Same Software Project (A PBX Switching System of 1,500 Function Points in Size)

Language	Req. (Months)	Design (Months)	Code (Months)	Test (Months)	Doc. (Months)	Mgt. (Months)	TOTAL (Months)
Assembly	13.64	60.00	300.00	277.78	40.54	89.95	781.91
C	13.64	60.00	152.40	141.11	40.54	53.00	460.69
CHILL	13.64	60.00	116.67	116.67	40.54	45.18	392.69
PASCAL	13.64	60.00	101.11	101.11	40.54	41.13	357.53
PL/I	13.64	60.00	88.89	88.89	40.54	37.95	329.91
Ada83	13.64	60.00	76.07	78.89	40.54	34.99	304.13
C++	13.64	68.18	66.00	71.74	40.54	33.81	293.91
Ada95	13.64	68.18	52.50	63.91	40.54	31.04	269.81
Objective C	13.64	68.18	31.07	37.83	40.54	24.86	216.12
Smalltalk	13.64	68.18	22.50	27.39	40.54	22.39	194.64
Average	13.64	63.27	100.72	100.53	40.54	41.43	360.13

Quelle [Jones1997]

Function Points

- sind ein Maß für die „Größe“ und Komplexität von Software, das von der Programmiersprache unabhängig ist

Function Points
Grundformel

- Summe(Eingabefunktionen (EF) * Gewichte)
- Summe(Zahl der Berechnungsfunktionen (BF) * Gewichte)
- Summe(Zahl der Abfragefunktionen (AF) * Gewichte)
- Summe(Zahl der gespeicherten Dateneinheiten (DE) * Gewichte)
- Summe(Zahl der ext. Schnittstellen (SES) * Gewichte)

- ergeben Unadjusted Function Points (UFP)

Function Points
Gewichte

		einfach	mittel	komplex	Werte
Eingabefunktionen	**EF**	3	4	5	
Ausgabefunktionen	**AF**	4	5	7	
Abfragefunktionen	**AF**	3	4	6	
Dateneinheiten	**DE**	7	10	15	
Schnittstellen	**SES**	5	7	10	
				Summe	

Function Points
Komplexitätsfaktor

- es gibt einen Katalog mit Zu- und Abschlägen, die einen Systemkomplexitätsfaktor ergeben (SCF)
- FP = UFP * SCF

Wie „zählt“ man Function Points? ex ante

- benötigt wird eine Spezifikation
  - mit Bildschirmmasken
  - Datenbankentwürfen
  - Funktionenmodell
  - Schnittstellen
  - Druckoutput-Entwürfen
- oder etwas, was einer solchen Spezifikation möglichst nahe kommt…

Wie „zählt“ man Function Points? ex post

- durch Messen der Lines of Code
- und Division durch den entsprechenden Konversionsfaktor der jeweiligen Programmiersprache

Function Points
Vorteile

- Programmgröße ist messbar und damit vergleichbar unabhängig von Programmiersprache
- Methode kann man ausreichend schnell erlernen (1-2 Tage)
- Mit FP wird auch Produktivität ex post beurteilbar (FP / Personenmonat)
- Aus FPs kann man viele weitere Erfahrungsgrößen berechnen - zum Beispiel erwartete Fehler

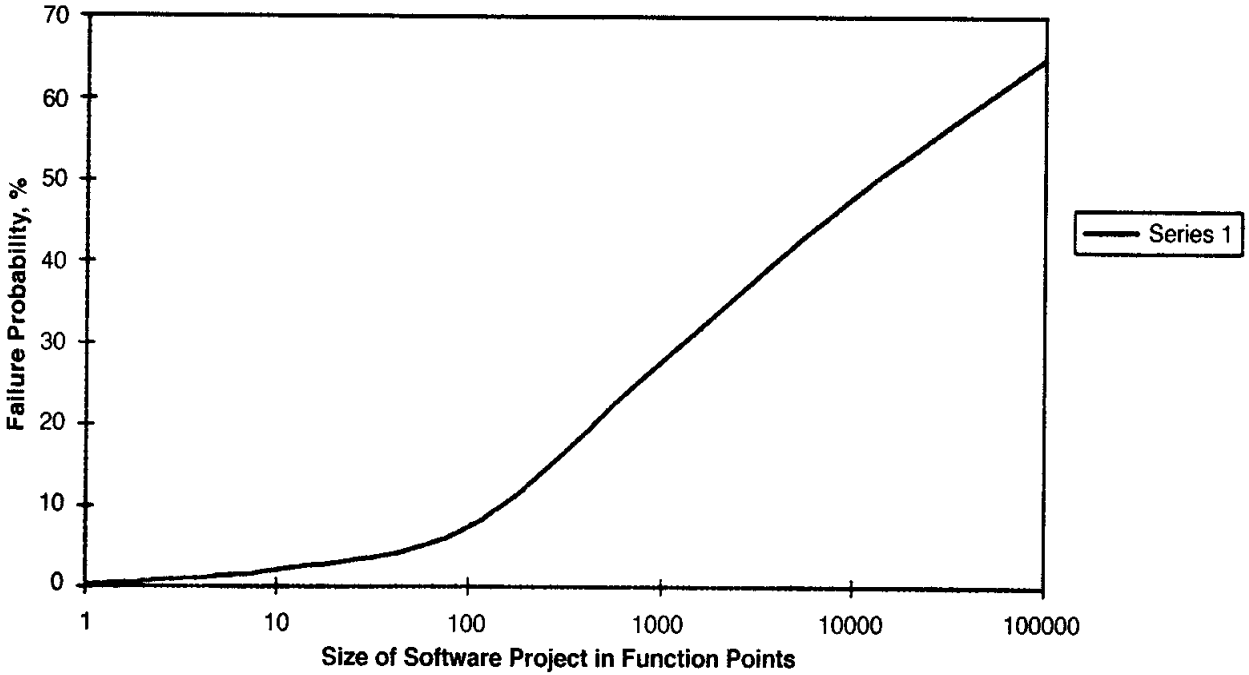
Function Points
Weitere Eigenschaften

- Man benötigt eine einigermaßen ergiebige Spezifikation - die ist oft nicht vorhanden
- Wiederverwendung ist problematisch einzubeziehen (wie viele FPs entspricht die Verwendung eines Moduls mit 2.500 FP)
- Ausgelegt auf funktionale Technologie - für OO, Web, .. gibt es Erweiterungen
- Man muss die Methode seriös erlernen, um sie zu verwenden (Buch reicht allerdings)

Führt uns zurück zur Frage .. Was ist groß?

- Durchschnittliches Projekt in der Finanzindustrie
  - 2.000 FP
  - 1-2 Jahre Durchlaufzeit
  - ca. 10 MitarbeiterInnen
- ab solchen Größen lohnen sich die Methoden, die in dieser Vorlesung beschrieben werden ...

Projektrisiko als Funktion der Projektgröße

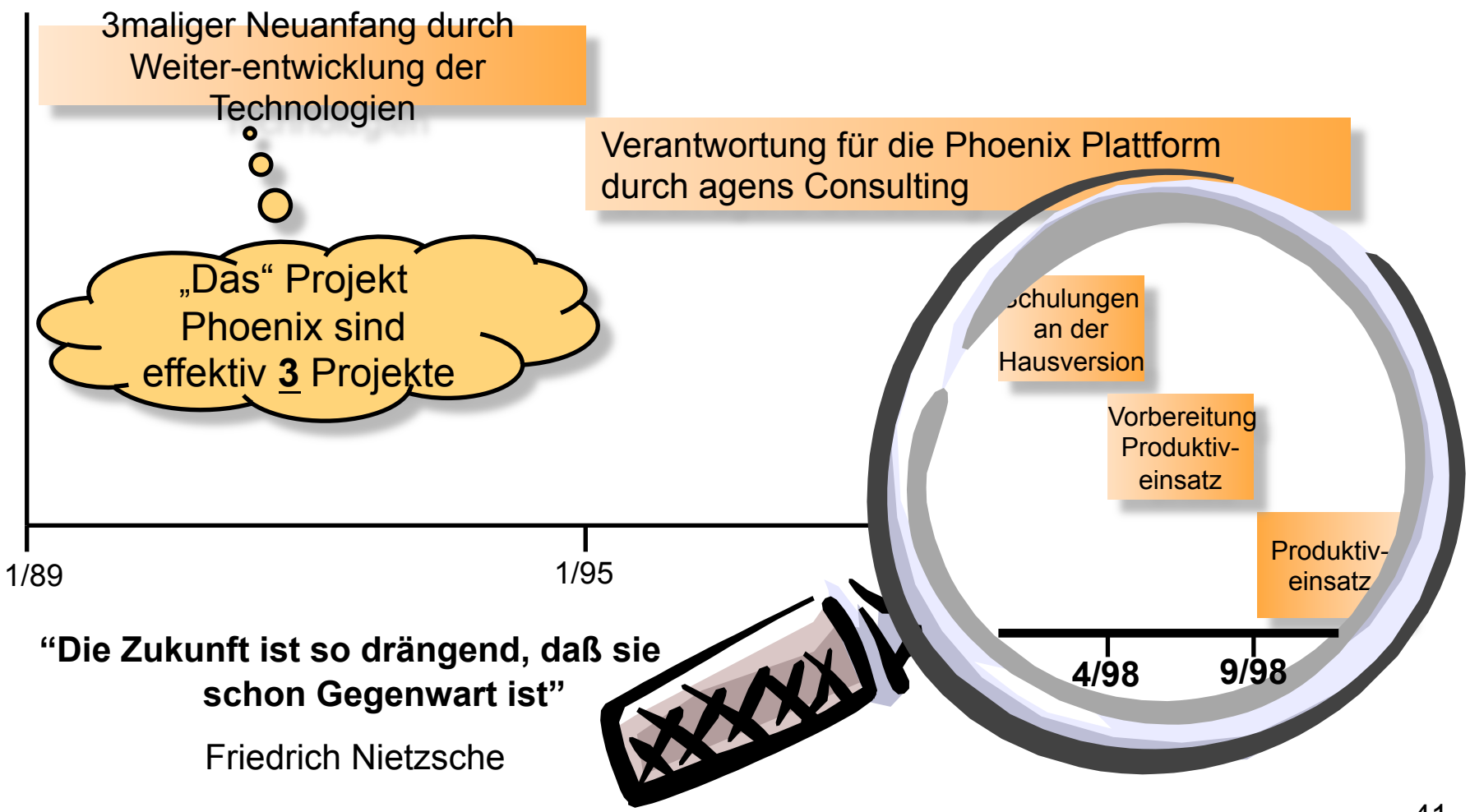


Quelle [Jones1996]

Ein großes Projekt
Fallstudie Phoenix

Zeitlicher Abriß

Durch welche markanten Eckwerte kann das Projekt beschrieben werden?

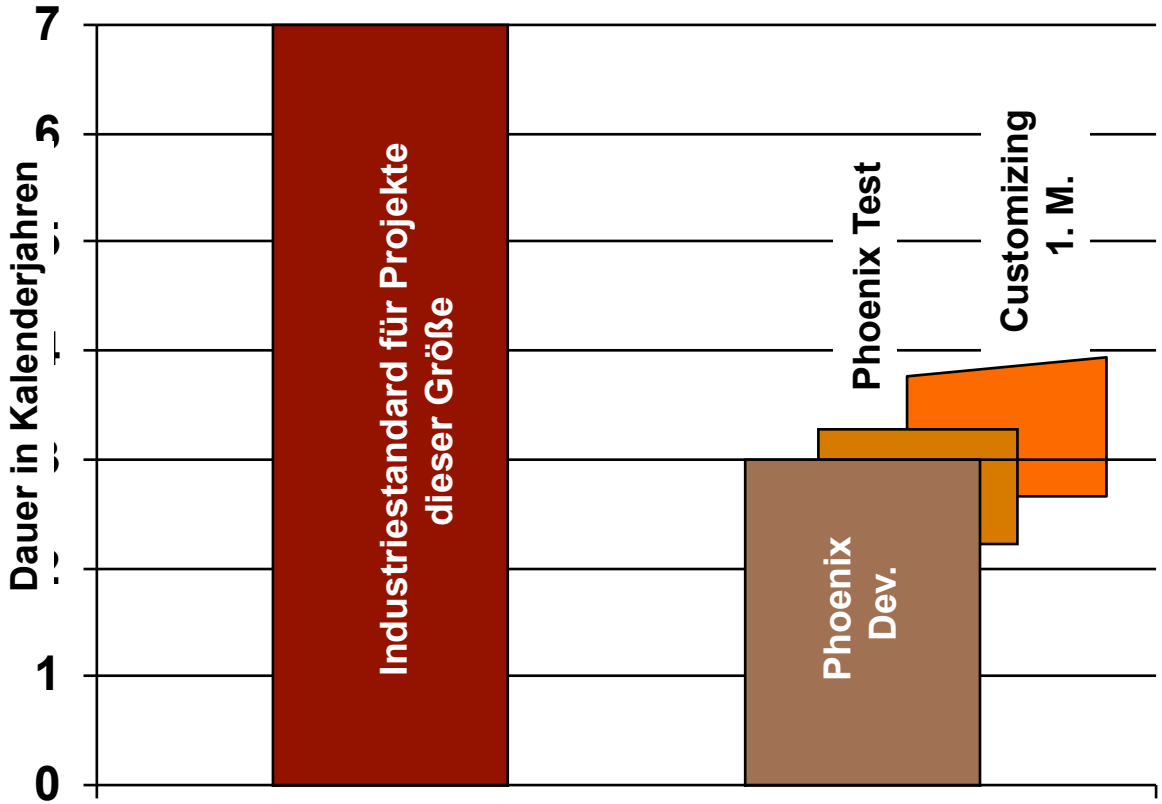


“Die Zukunft ist so drängend, daß sie schon Gegenwart ist”

Friedrich Nietzsche

Projektlaufzeit

Durch welche markanten Eckwerte kann das Projekt beschrieben werden?



Function Points

Durch welche markanten Eckwerte kann das Projekt beschrieben werden?

Hasso Plattner Institut

Größe von Phoenix: 16.500 Function Points

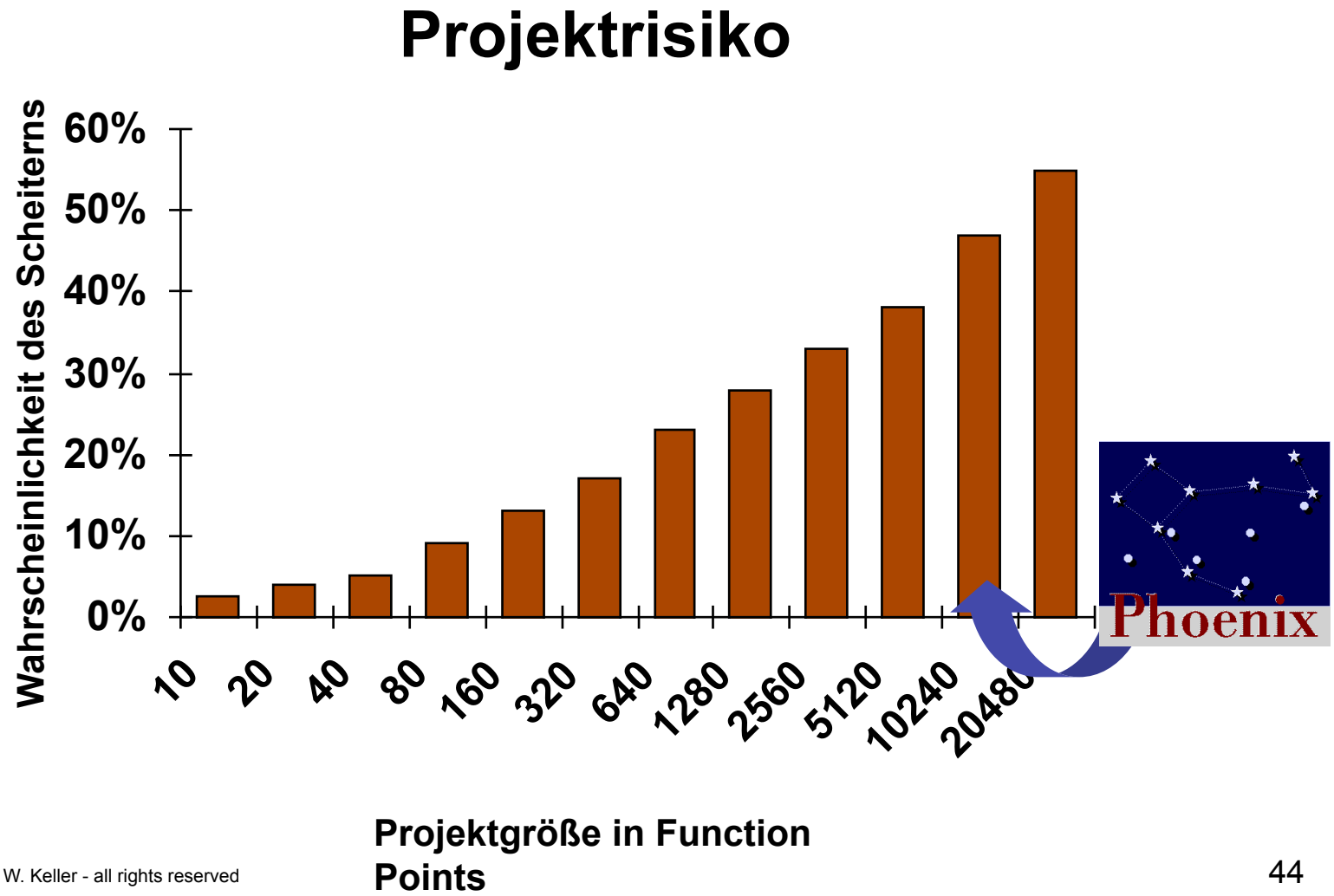
Dies entspricht ca. 450.000 Lines of Code (Smalltalk)

Die Größe eines durchschnittlichen Softwareprojekts in Checkpoint* beträgt ca. 2.000 Function Points!

Checkpoint ist ein Produkt der SPR

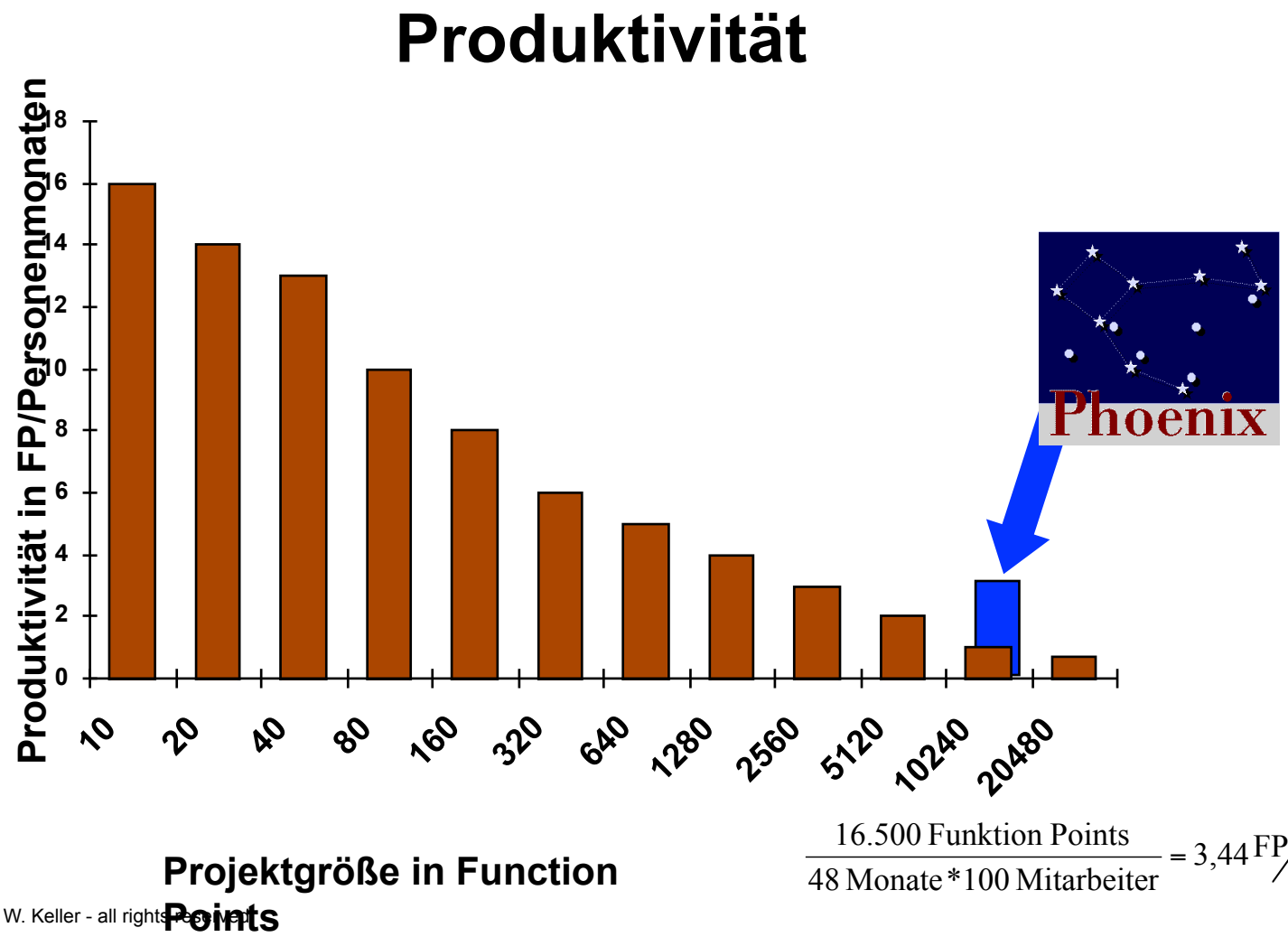
Bewertung der Produktgröße

Durch welche markanten Eckwerte kann das Projekt beschrieben werden?



Bewertung der Produktgröße

Durch welche markanten Eckwerte kann das Projekt beschrieben werden?

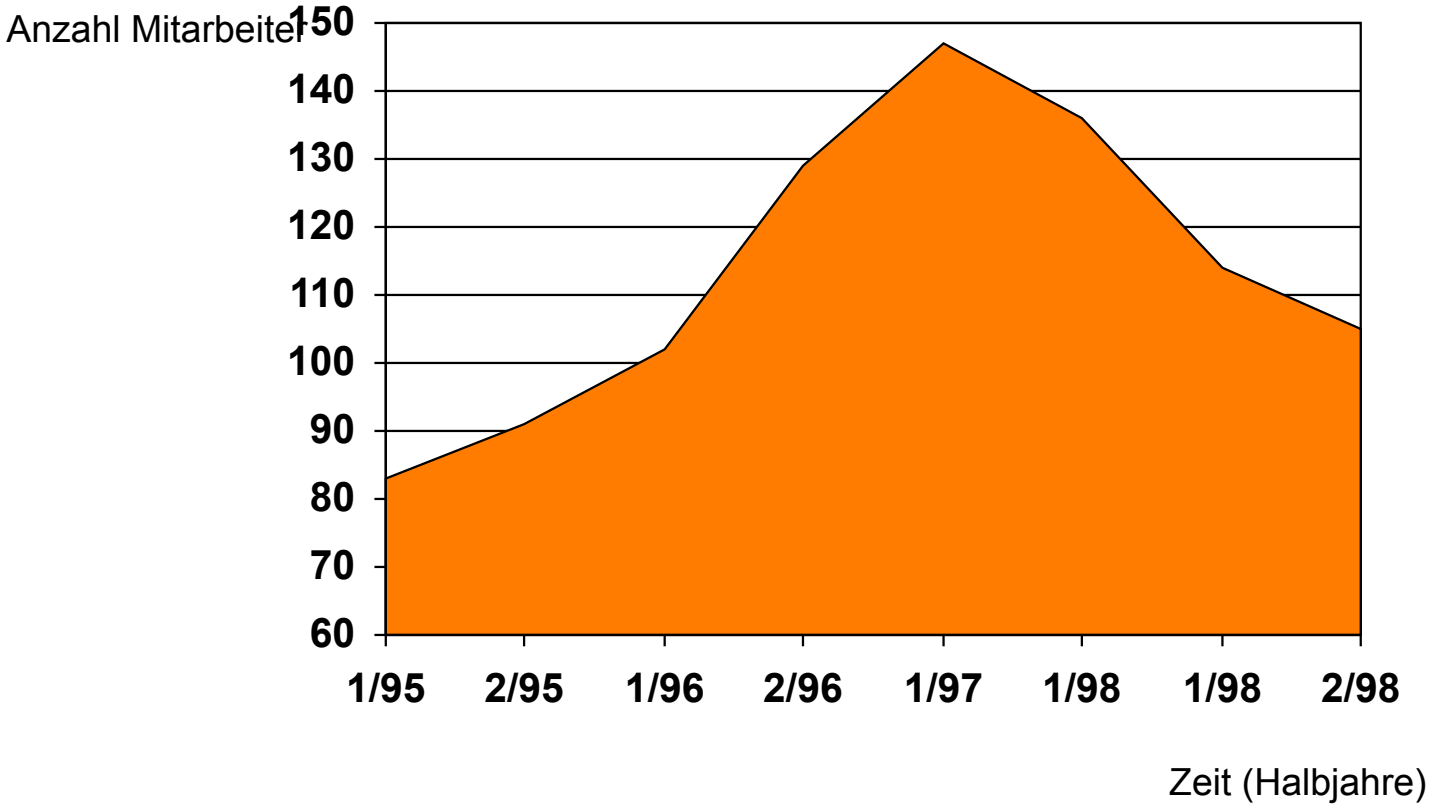


$$\frac{16.500 \text{ Funktion Points}}{48 \text{ Monate} * 100 \text{ Mitarbeiter}} = 3{,}44 \, \text{FP}/\text{PersMonate}$$

Bewertung der Produktgröße

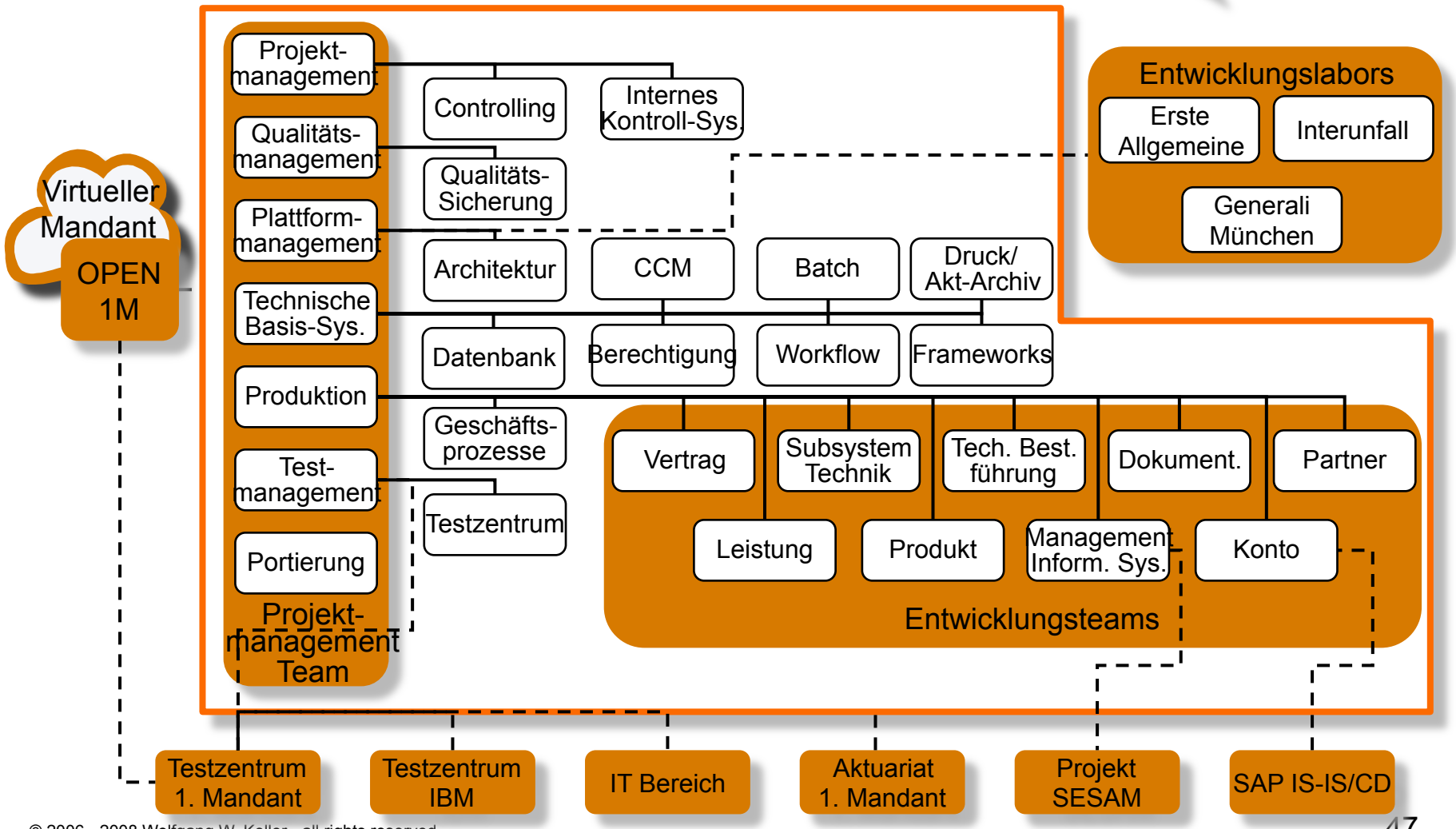
Durch welche markanten Eckwerte kann das Projekt beschrieben werden?

Entwicklung der Mitarbeiteranzahl (nach Köpfen)



Projektorganisation

Mit welcher Projekt-organisation kann man solche Großprojekte managen?



Death March Projekte

Death March Projekte
Definition nach Yourdon

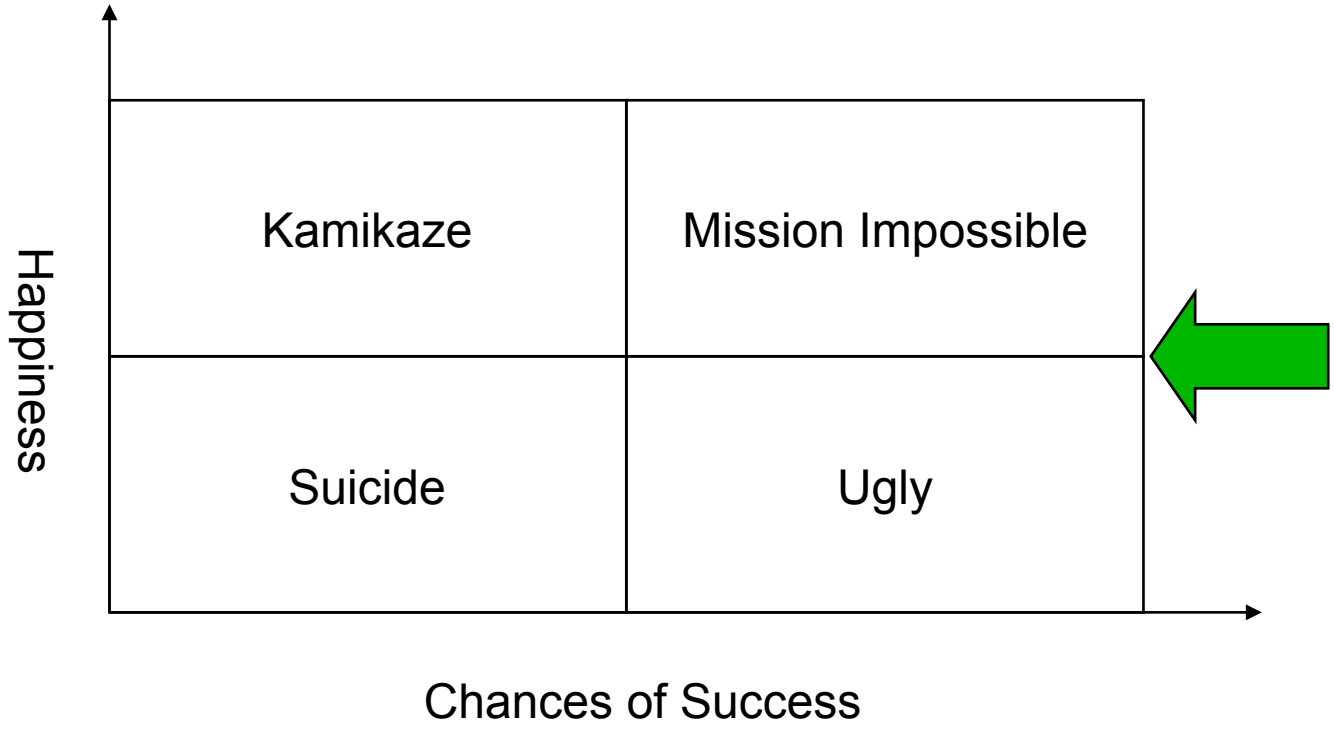
- sind Projekte, die um den Faktor 50% von der Norm abweichen, zum Beispiel
  - Wurde die Projektzeit um 50% gekürzt, gegenüber dem normal errechneten Wert
  - oder die Mannschaft halbiert, gegenüber normaler Teamstärke
  - oder aber das Budget halbiert (entsprich halber Mannschaft)
  - andere Faktoren liegen um den Faktor 2 neben den normalen Werten

[Yourdon1997]

Death Marches nach Größe ..

- Klein: <= 10 Teammitglieder mit einer Durchlaufzeit von ca. 3-6 Monaten
- Mittel: 20-30 Bearbeiter mit einer Durchlaufzeit von 1-2 Jahren
- Groß: 100 - 300 Mitarbeiter, 3-5 Jahre Durchlaufzeit
- Gigantisch: 1000+ Mitarbeiter, 7-10 Jahre Durchlaufzeit

Death March Arten



[Yourdon2002]

Sie sagen, so etwas tut doch heute niemand mehr ...

- In den 1990er Jahren war häufige Ursahe von Death Marches, dass man glaubte, große Teile von Anwendungslandschaften auf einmal neu bauen zu können

- Heute sind oft gesetztliche Anforderungen Ursachen für Death March Projekte
  - Beispiel: Einführung neues Versiherungsvertragsgesetz zum 1.1.2007
  - Selbst im Oktober davor waren wichtige Details zur Ausführung des Gesetzes noch nicht bekannt
  - Aber es war klar, dass es ab 1.1.2008 „scharf umgesetzt wird“
- Leider kein Einzelfall!

Weiterer Treiber für Death March Projekte Time to Market

- 90er Jahre
  - Projektlaufzeiten von 3 Jahren waren normal
  - erste Software nach 1,5 - 2 Jahren

- Heute
  - am liebsten Software nach 3-6 Monaten
    - Pay Off nach 2 Jahren
    - Erste Software nach 3 Monaten
  - Beispie 1: Web Banken < 1 Jahr war schon 2001 üblich – Software sieht handwerklich teilweise schlimm aus
  - Beispiel 2: Heute zum Beispiel normal:
    - Produktidee für Finanzprodukt im Februar, Verkaufsstart August, nur Teil der Geschäftsvorfälle vorhanden

Im Normalfall kann man Death Marches u.a. wie folgt vermeiden ...

- Projekte möglichst klein schneiden (<2000 FP)
  - Begründung: VL heute
- dafür erforderlich ein Generalbebauungsplan
  - Architekturmanagement, siehe VL 03
- Projekte ordentlich schätzen
  - Begründung: VL heute
- Und im Projekt professionell arbeiten
  - Hinweise finden Sie in anderen Vorlesungen ...
  - zum Beispiel Vorlesung "Software-Engineering für große betriebliche Informationssysteme“ http://www.objectarchitects.de/lectures/se/

Warum haben Sie hier etwas über Großprojekte gehört?

- Sie werden als Unternehmensarchitekt fast „zwangsweise in solche Projekte „hereingezogen“

- Sie müssen sie erkennen und Sie müssen die Probleme solcher Projekte kennen,
  - damit Sie Ihre Kollegen besser coachen können
  - und damit Sie ihr Management beraten können, wie man mit solchen Projekten umgeht

Gefahren bei großen und „neuen“ Projekten

Gefahren bei „großen“ und „neuen“ Projekten

Bei kleinen Projekten unkritisch - bei großen sehr auffällig

- Wenn die Fachlichkeit und Technik nicht verstanden wurde, neigen Entwickler oft dazu, das Projekt zu optimistisch zu sehen
  - Aufwand wird zu niedrig geschätzt
  - => Death March
- Wenn Projekte unterschätzt wurden, leidet am Ende oft die Qualität, wenn Zeit und Kosten fixe Größen sind
- Unter Zeitdruck vergisst man dann gerne „good pracitices“, wie sie hinten in dem Architekturdokument beschrieben sind
  - Wenn das spät auffällt, bekommt man Probleme kurz vor Projekteinführung

Gefahren bei „großen“ und „neuen“ Projekten

Bei kleinen Projekten unkritisch - bei großen sehr auffällig

- Große und neue Projekte sind anfällig für Death March Kulturen

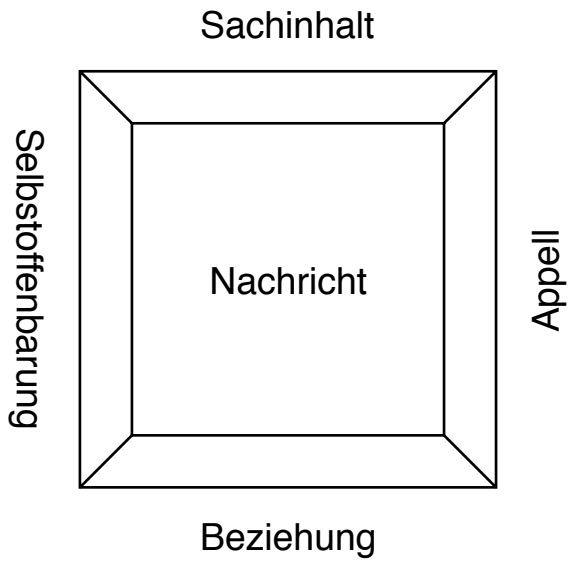
- Manager vergessen gerne die Regeln großer Projekte
  - das Chinesenprinzip funktioniert nicht
  - Produktivität sinkt drastisch, wenn das Projektteam vergrößert wird

„adding manpower to a late project makes it later“

Wie sage ich es dem Team?
Feedback Tipps

Software-Entwicklung und -Architektur haben viel mit Kommunikation zu tun ..

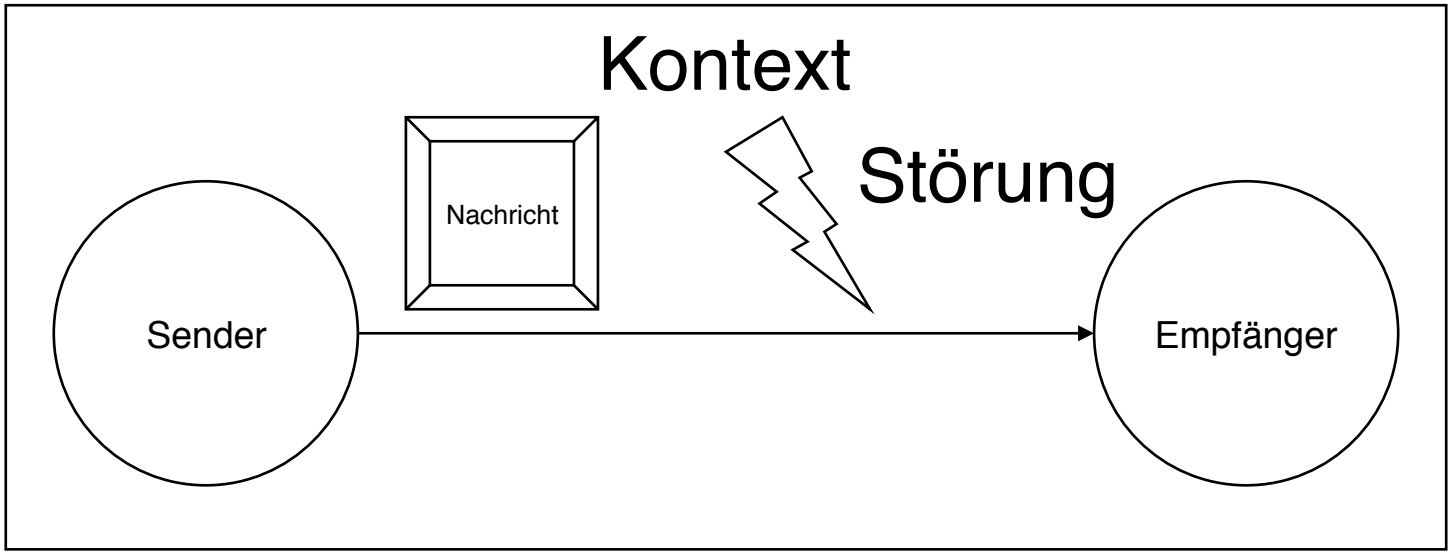
Kennen Sie zum Beispiel dieses Modelle?



siehe z.B. Schulz von Thun: Miteinander Reden 1

Software-Entwicklung und -Architektur haben viel mit Kommunikation zu tun ..

oder dieses?



Soft-Skills sind wichtig, wenn man als Architekt Botschaften transportieren muss

siehe z.B. Schulz von Thun: Miteinander Reden 1

Wissen über Kommunikation hilft Störungen zu verstehen und oft auch zu vermeiden bevor sie Schaden anrichten ...

- was man sagt, muss nicht das sein was ankommt
  - unterschiedliche Landkarten der Kommunikationsteilnehmer führen zu unterschiedlichen Interpretationen von Nachrichten
- auch wenn jemand etwas hört, heißt das nicht, dass er es so versteht, wie wir glauben es gesagt zu haben
- Und selbst wenn er es verstanden hat, heißt das noch lange nicht, dass er entsprechend handelt

- Darüber hinaus kann es zu diversen Störungen zum Beispiel über die Beziehungsebene kommen

wer diese Modelle kennt, kommuniziert schneller und sicherer und mit weniger unerwünschten Nebenwirkungen

Einige einfach Kommunikationstricks
Ich-Botschaften und Du-Botschaften

Hasso Plattner Institut

Du-Botschaft	Ich-Botschaft
• Du bist laut!	• Ich fühle mich gestört.
• Sei ruhig	• Könntest bitte etwas leiser sein

Welche Botschaft wird als aggressiv empfunden?
Welche wird eher angenommen?

Einige einfach Kommunikationstricks
Negative versus positive Botschaften

Hasso Plattner Institut

Negative Botschaft	Positive Aussage
• Das Design ist schlecht!	• Du kannst das Design besser machen, wenn Du <das> machst

Welche Botschaft ist hilfreicher?

Einige einfach Kommunikationstricks Sandwich Technik

- Schlechtes Gefühl vs. Sandwich

- fangen wir mal an .. das ist Mist
- na ja - das geht
- aber was ich noch sagen wollte, das ist auch Mist

- start with good news
- dazwischen Punkte, was besser gemacht werden kann - positiv formuliert
- end with good news

Aus welchem Meeting geht der Gereviewte mit besserer Laune heraus. Welcher Reviewer ist der erfolgreichere

Zu den Zielen von Reviews, Test, Management: Der beste Tester, Manager, Reviewer ist der, ..

- der die meisten Probleme aufdeckt?
- oder der, der die meisten abgestellt und verbessert bekommt?

- alles was man negativ formulieren kann
- kann man auch positiv umformulieren

Deutsche „Techies“ formulieren gerne negativ.
Über positives Formulieren kommuniziert man effizienter!

Nützliche Templates für Architekturdokumente

oder: wenn jeder Architekt immer solche Templates verwenden würde, würden viele Probleme nicht auftreten, weil weniger „vergessen“ würde

Zur Kommunikation von Projektarchitekturen gibt es nützliche Templates. U.a.

- das arc42 Template
  - wird hier heute vorgestellt
  - Download unter: http://www.arc42.de/LinkedDocuments/arc42_template_V1.5.zip
  - aufgerufen 27.4.2007

- das Template aus dem RUP (Rational Unified Process)
  - nicht völlig unähnlich, aber älter
  - http://atlas-connect-forum.web.cern.ch/Atlas-connect-forum/documents/Software%20Architecture%20Document.doc
  - aufgerufen 27.4.2007

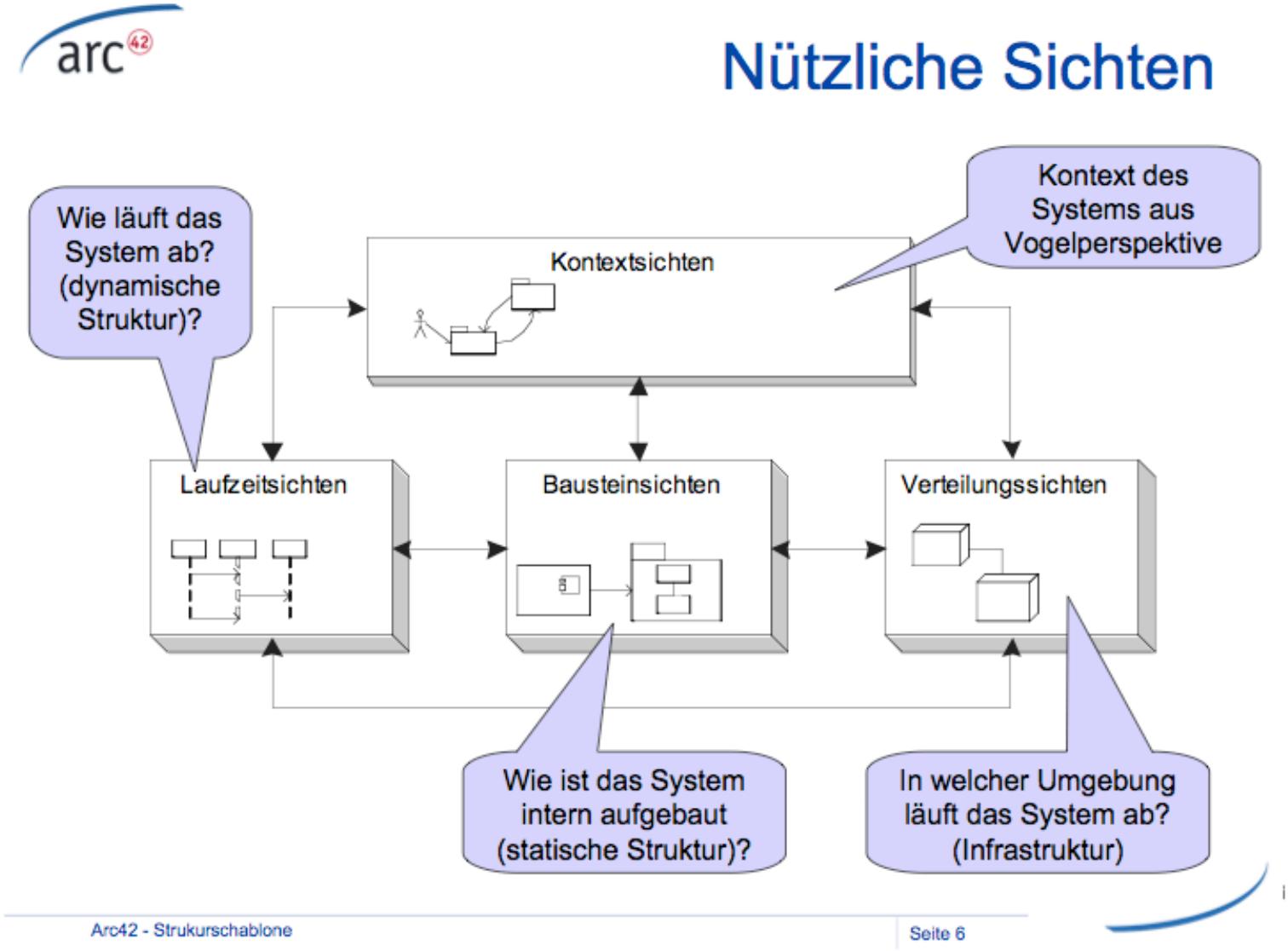
- Dazu können Sie noch die Checkliste aus dem Buch „IT-Unternehmensarchitektur“ verwenden
  - http://www.amazon.de/gp/product/3898644197/
  - aufgerufen 27.4.2007

Copyright Notice: Einige der folgenden Folien entstammen dem Vortrag ...

Quelle: Gernot Starke: Praktische Architekturdokumentation, Tagung Wirtschaftsinformatik, Karlsruhe 2007

http://www.gernotstarke.de/publikationen/publikationen/vortraege_assets/2007-WI-Starke-ArcTemplates.ppt.pdf

Architekturdokumentation dokumentiert mindestens die Sichten auf ein System ..

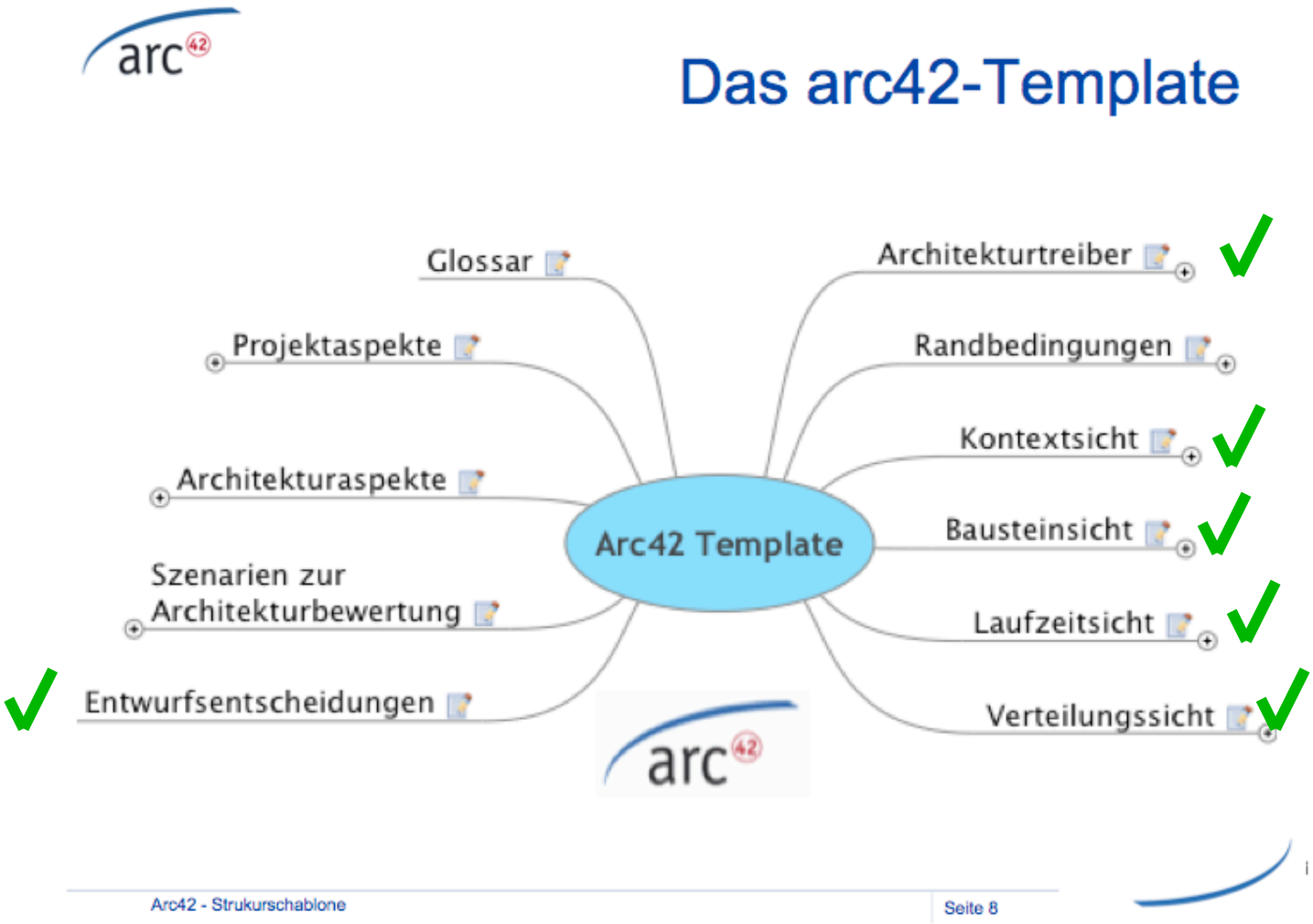


Es ist allerdings nicht hinreichend nur die Lösung zu kennen

- Alter Spruch: Now that we have a solution - what was the problem?
  - Wenn man sich das als Architekt fragen lassen muss, kann man noch besser werden
- Man sollte auch das Problem beschrieben haben
  - dazu gehört eine Zusammenfassung der funktionalen Anforderungen
  - und für Architekturen wichtiger - eine Zusammenfassung der nicht-funktionalen Anforderungen

- Ähnlich wie bei Patterns sollte man auch die Zielkonflikte (Forces) beschreiben, die dazu geführt haben, dass man eine bestimmt Lösung gewählt hat

Damit haben wir viele der Teile schon erklärt ..



Bleiben noch zum Beispiel Architekturaspekte

10 Architekturaspekte
- 10.1 Persistenz
- 10.2 Benutzungsoberfläche
- 10.3 Ergonomie
- 10.4 Ablaufsteuerung
- 10.5 Transaktionsbehandlung
- 10.6 Sessionbehandlung
- 10.7 Sicherheit
- 10.8 Kommunikation und Integration
- 10.9 Verteilung
- 10.10 Ausnahme- /Fehlerbehandlung
- 10.11 Management /Adminsistrierbarkeit
- 10.12 Logging, Protokollierung, Tracing
- 10.13 Konfigurierbarkeit
- 10.14 Parallelisierung und Threading
- 10.15 Internationalisierung
- 10.16 Migration
- 10.17 Testbarkeit
- 10.18 Plausibilisierung und Validierung
- 10.19 Build Management

- das sind typische Themen, die man für jedes (nicht nur) kommerzielle Informationssystem **bewusst** gelöst haben muss
- Man benötigt für alle diese Dinge für jedes Projekt Lösungen
- Am besten, man erbt viele davon aus einer fertigen Entwicklungsumgebung

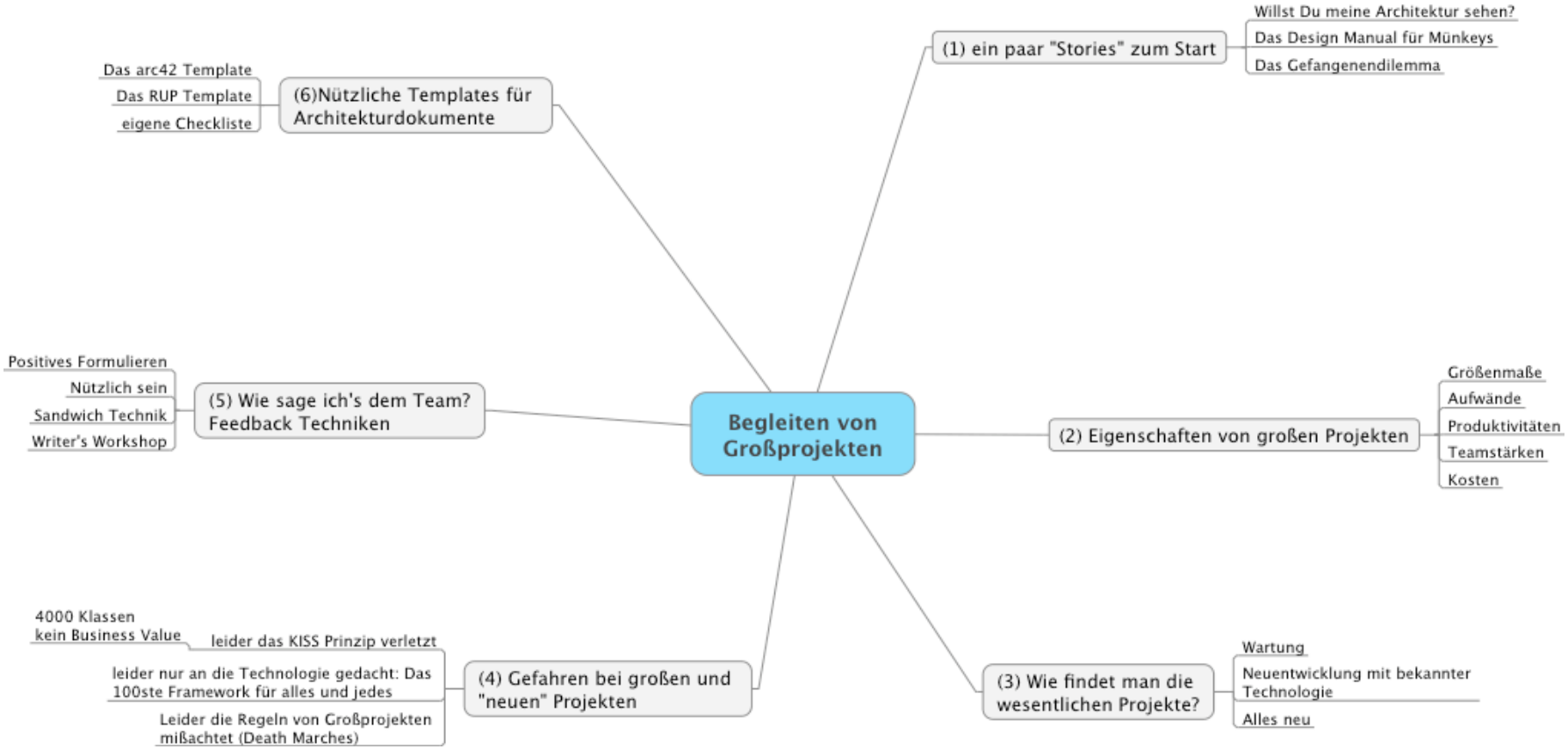
Bleiben noch zum Beispiel Randbedingungen und Projektaspekte

3 Randbedingungen
- 3.1 Technische Randbedingungen
- 3.2 Organisatorische Randbedingungen
- 3.3 Konventionen

11 Projektaspekt
- 11.1 Change Requests
- 11.2 Technische Risiken
- 11.3 Offene Punkte
- 11.4 Erwartete Änderungen
- 11.6 Auswirkungen

- sind Beschreibungen des Projektumfeldes
- Das Template ist also „nur vernünftig“ und keine Rocket Science
- Trotzdem werden Sie sich oft schwer tun, eine solche Dokumentation zu bekommen
- Wir erinnern uns an die Geschichte zum Einstieg in diese Vorlesung - Die Architektur steht zu oft im Code

Zur Zusammenfassung noch einmal das Mindmap



Fragen?

und wenn Ihnen später noch Fragen einfallen ....

Wolfgang Keller
BusinessGlue GmbH
Liebigstr. 3
82166 Lochham
wolfgang.keller@businessglue.de

Literatur unter anderem ...

- [Garmus+2000] David Garmus, David Herron: Fuction Point Analysis, Addison-Wesley 2000.
- [Jones1996] Capers Jones, Applied Software Measurement, 2nd Edition, McGraw Hill 1996
- [Jones1997] Capers Jones, The Economics of Object-Oriented Software, White Paper, spr.com 1997.
- [Jones 1998] Capers Jones, Positive and Negative Factors that Influence Software Productivity, White Paper, spr.com 1998.
- [Yourdon1997] Ed Yourdon: Death March, The Complete Software Developer‘s Guide to Surviving Mission Impossible Projects, Prentice Hall 1997.
- [Yourdon2002] Ed Yourdon: Managing High Intensity Internet Projects, Prentice Hall 2002