Software Architecture Analysis Method (SAAM)

- Motivation
- Repräsentation von Qualitätsattributen: Szenarien
- SAAM-Überblick und SAAM-Definition
- Architekturvergleich mit SAAM
- Bewertung einer einzelnen Architektur mit SAAM
- Beispiele
  - WRCS
  - EPOC (ESAPS-Projekt)
  - Internet Information Systems (IIS)
- Zusammenfassung
- Literatur

- SAAM-Checkliste

Motivation

Bei der Entwicklung eines Systems:

- Architekturen existieren früh im Entwicklungsprozess
- Architektur-Alternativen können früh verglichen werden
- Realität (nicht nur laut Kazman):
  Analyse wird während (irgendeiner) Schadensbegrenzung/-behebung spät im Projekt durchgeführt

Beim Kauf eines Systems:

- Wenn das System vermutlich lang leben wird → Investitionssicherheit
- Eine Analyse kann Aufschluss über die Entwicklungsfähigkeit des Systems geben
- Vergleiche verschiedener alternativer (Konkurrenz-)Systeme anhand ihrer Architekturen sind möglich

Probleme bei der Bewertung von Software

…bzgl. Qualitätsbewertung:

- Definitionen für Qualitätsattribute werden oft nicht einheitlich verwendet
- Es gibt kein objektives Maß für Qualitätsattribute
- Qualitätsattribute beziehen sich oft auf zukünftige Anforderungen

…bzgl. Architekturdarstellung:

- Es gibt keine einheitliche Notation für Architekturbeschreibungen
- Es ist nicht automatisch sicher, dass eine Architekturbeschreibung mit der Realität übereinstimmt

Qualitätsattribute

Funktionale Qualitätsattribute
sind zum Beispiel

- Funktionalität
- Rechengenauigkeit
- Art und Weise der Bedienung durch den Benutzer

- Erfasst bei der Anforderungsanalyse
- (In-)Direkt messbar / prüfbar

Nicht-funktionale Qualitätsattribute
erscheinen zunehmend wichtiger wegen

- zunehmender Lebensdauer von Softwaresystemen
- Bewusstsein für kommende Änderungen
- Einführung neuer SE-Ansätze z.B. zwecks Rationalisierung
- etc.

Nicht-funktionale Qualität: genaue "Definition"

Wie kann man anhand seiner Architektur bestimmen, ob ein System gut bzw. einfach

- erweiterbar,
- modifizierbar,
- portierbar,
- performant etc.

ist?

Nicht-funktionale Qualität: Beispiel für eine ungenaue "Definition"

Gegeben sei eine Software für die Verwaltung von Kundenadressen mit folgenden Eigenschaften:

- Einstellungen für Vorder- und Hintergrundfarben können in Ressource-Dateien vorgenommen werden
- Die Schriftgröße kann über einen Menüeintrag geändert werden
- Für den Import externer Adressformate müssen voraussichtlich fünf Komponenten angepasst werden

Frage: Ist dieses System "leicht modifizierbar"? Ja oder Nein?

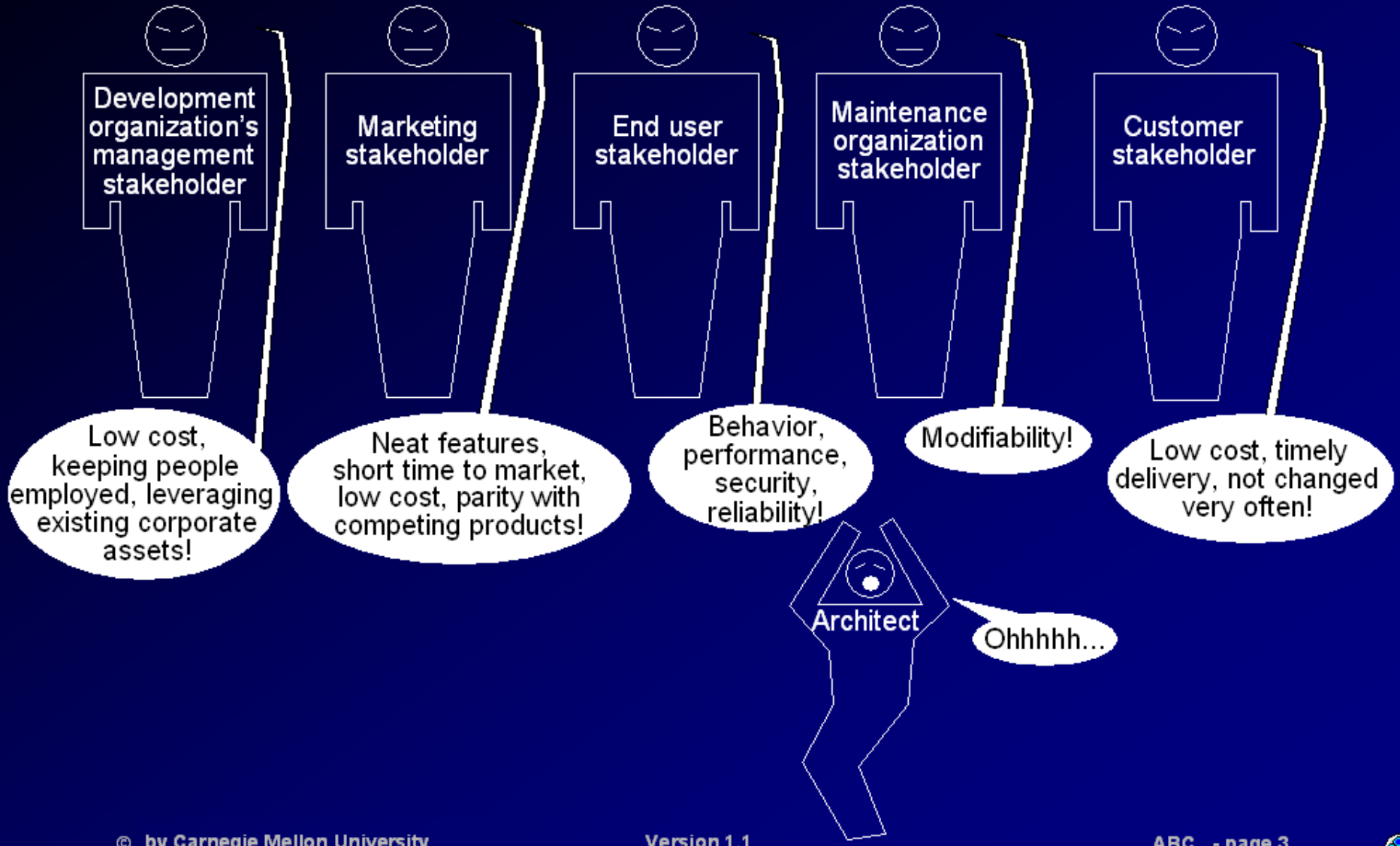
Definition "Szenario"

Ein **Szenario** ist eine kurze konkrete Beschreibung einer einzelnen Interaktion eines Stakeholders mit dem System.
Es repräsentiert meist eine Klasse ähnlicher Szenarien.

[Kazman]

- Szenarien werden von den *Stakeholdern* für das System entwickelt
- Interaktion $\supset$ Bedienung!

Stakeholders of a System



Beispiel-System mit Szenarien
(Studentenprojekt vom SS‘07)

OMIA System

- Ziel ist es, ein **Online System** zu entwickeln, das erlaubt **Meta Informationen** von digitalen Bildern zu **verwalten**.
- **Online System:**
  Das System soll auf einem HTTP-Server betrieben werden und über einen Web-Browser bedienbar sein. Daher muss das System Multi-User fähig sein.
- **Meta Information:**
  Informationen, die in digitale Bilder eingebettet sind, wie z.B. Exif, IPTC
- **Verwalten:**
  - Verwalten beinhaltet lesen, schreiben, suchen von Meta Informationen
  - Zusätzlich sollen Funktionen realisiert werden, die einen reibungslosen Betrieb des Systems ermöglichen wie z.B. Import von Bilddateien oder anlegen von Benutzern.

Stakeholder: *Domänenexperte*

- Zu einem späteren Zeitpunkt müssen z.B. Bilder von fremden Agenturen bzw. Fotografen oder andere Metadaten verarbeitet werden können.

- Kann das System diesbezüglich schnell und kostengünstig erweitert werden?
- Ist die Architektur investitionssicher?

Stakeholder: *Benutzer*

- Die Mitarbeiter im Unternehmen werden das System zusätzlich zum Tagesgeschäft zur Organisation unserer Bilderarchive nutzen und haben keine IT-Affinität. Dies erfordert, dass die Applikation einfach, intuitiv und schnell zu benutzen ist.

- Kann die Bedienung des Systems im Nachhinein geändert werden, falls sich in der Praxis Defizite zeigen?
- Wie hoch ist der Aufwand?

Stakeholder: *Kunde*

- Das System muss zu einem fixen Termin fertig sein.

- Bei fehlender Termintreue: Kann die Applikation mit einem eingeschränkten Funktionsumfang eingeführt werden?
- Welche geforderten Funktionen können nachträglich hinzugefügt werden?

Stakeholder: *Tester*

- Ist die Analyse (das Parsen) von HTTP-Requests getrennt von den daraus resultierenden Methodenaufrufen?
- Kann die Datenbank durch ein Mock-Objekt ersetzt werden?
- Kann man (zu Testzwecken) die Authentifizierung umgehen (durch ein Mock-Objekt ersetzen)?
- Kann man dynamisch Logging-Module ankoppeln (während Stresstests)?
- Sind globale Variablen erlaubt?

Stakeholder: *Integrator / Sysadmin*

- Können bei einer parallelen Suche in diesem und einem anderen System doppelte Treffer vermieden werden?
- Kann man (nur) Metadaten zu Bildern an ähnliche Dienste übermitteln?
  Können diese bei Änderung in einem Dienst aktualisiert werden?
- Kann man dienstübergreifend arbeiten (z.B. "Fotoalbum")?
- Voraussetzung:
  Der andere Dienst bietet eine entsprechende Schnittstelle!

Stakeholder und ihre Interessen 1/3

- Kunde
  - Zeitplan und Budget
  - Nutzen des Systems
- Benutzer
  - Funktionalität
  - Benutzbarkeit (*usability*)
- Entwickler
  - Klarheit und Vollständigkeit der Architektur, des Designs
  - Hohe Kohäsion und geringe Kopplung der Komponenten
  - Klare Interaktionsmechanismen
- Wartungsentwickler
  - Wartbarkeit
  - Einfaches Lokalisieren von Stellen für Änderungen
  - Kapselung: Auswirkungen von Änderungen sind lokal begrenzt

Stakeholder und ihre Interessen 2/3

- System-Administrator
  - Einfaches Lokalisieren von möglichen Störungen im Betrieb
- Netzwerk-Administrator
  - Netzwerk-Performance
  - Vorhersagbarkeit von Netzlast
- Integrator
  - Klarheit und Vollständigkeit der Architektur
  - Hohe Kohäsion und geringe Kopplung der Komponenten
  - Klare Interaktionsmechanismen
- Tester
  - Konsistente Fehlerbehandlung
  - Geringe Kopplung der Komponenten
  - Hohe Komponenten-Kohäsion
  - Einheitlichkeit der verwendeten Konzepte (→ *texture*)

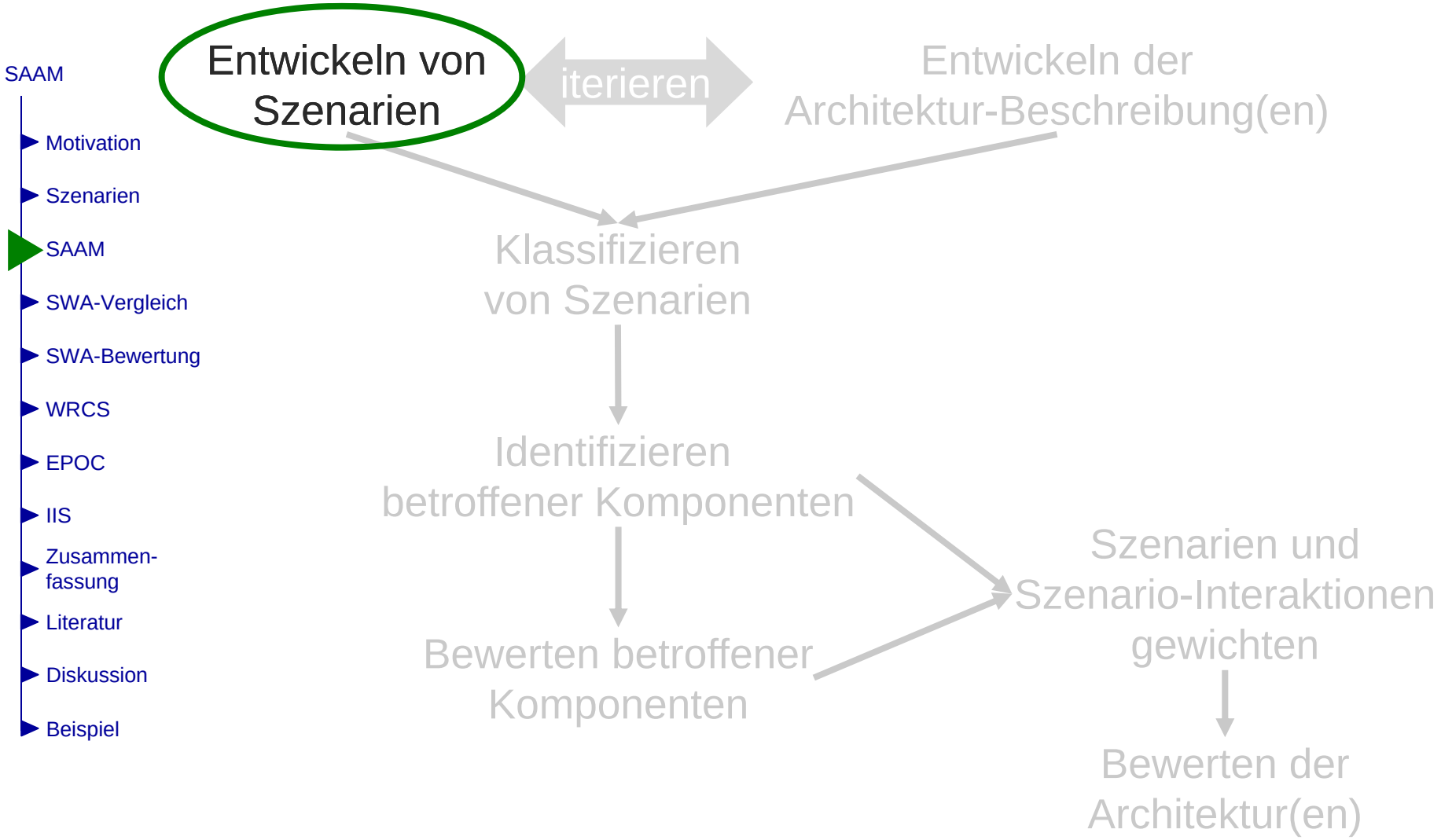
Stakeholder und ihre Interessen 3/3

- Anwendungsentwickler (falls es sich um eine Produktlinie handelt)
  - Klarheit und Vollständigkeit der Architektur
  - Klare Interaktionsmechanismen
  - Einfache Mechanismen für die Adaption (im *application engineering*)
- Repräsentant der Domäne
  - Interoperabilität mit anderen Systemen

Zusammenfassung: Entwickeln von Szenarien

- Die Stakeholder eines Systems erstellen Szenarien
- Szenarien repräsentieren typische Arten aktueller oder (potenziell) zukünftiger Nutzung des Systems bzgl.
  - Funktionalität
  - Entwicklungsaktivitäten
  - Änderungsaktivitäten
  - etc.
- Sinnvoll sind *10 – 15* Szenarien für eine Architektur-Evaluierung mit SAAM
- Die Szenarien werden priorisiert / gewichtet
  - nach ihrer Bedeutung
  - nach ihrer Eintrittswahrscheinlichkeit

SAAM - Überblick



SAAM - Definition

Die **Software Architecture Analysis Method (SAAM)** ist eine Methode für die Analyse von Software-Architekturen mittels spezieller *Szenarien*, welche abhängig vom konkreten Bedarf der *Stakeholder* eines Systems bestimmt werden um festzustellen, wie (*funktionale* und *nicht-funktionale*) *Qualitätsmerkmale* unterstützt werden.

Anwendbarkeit von SAAM

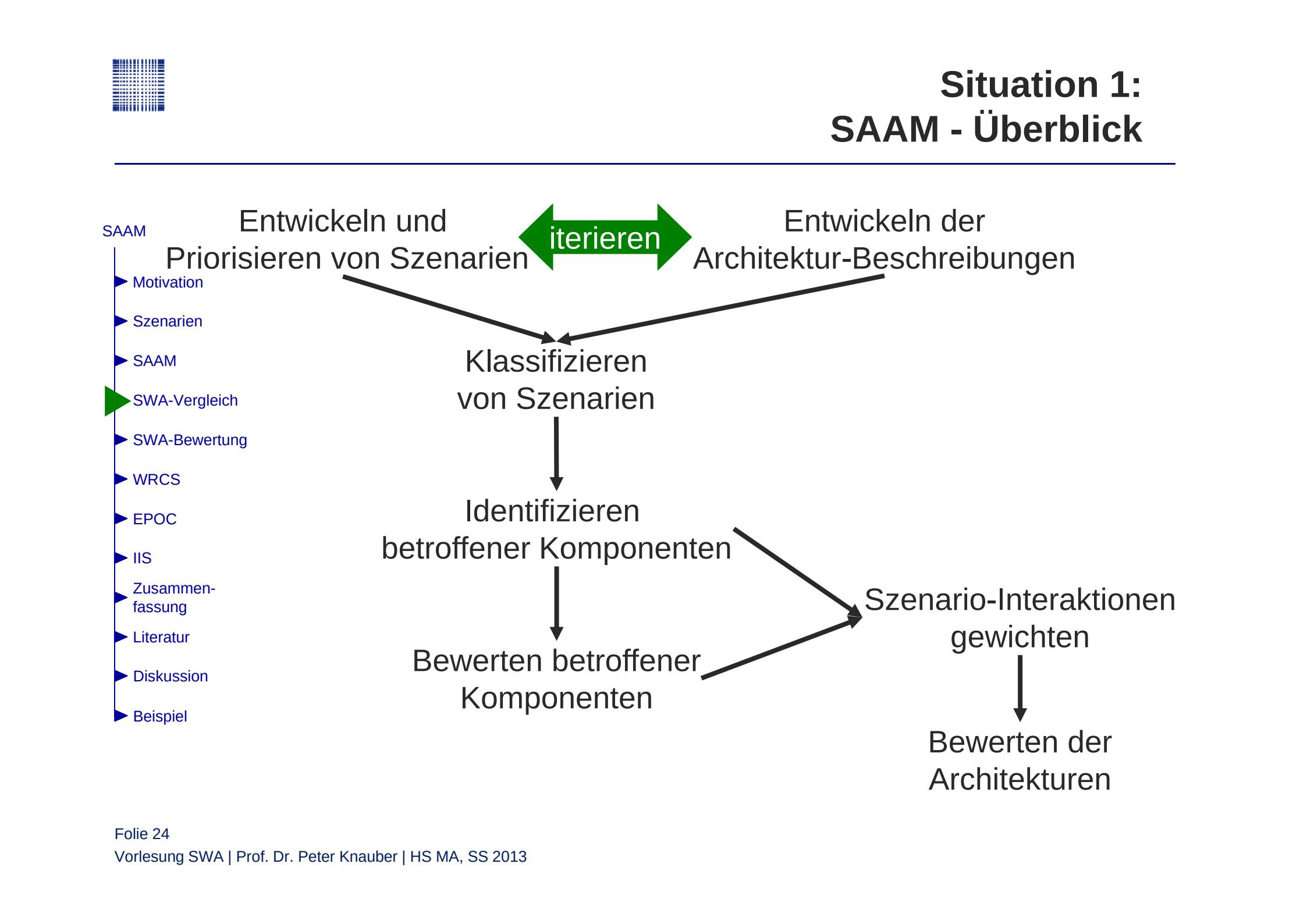
Situation 1:
Vergleich der Architekturen von zwei oder mehr Kandidaten-Systemen

- Idee:
  Bewerten, wie (gut) die gleiche Funktionalität auf unterschiedliche Architekturen abgebildet ist
- Ergebnisse von SAAM
  - Eine einheitliche Beschreibung beider System-Architekturen
  - Eine *relative* Rangfolge der Architekturen der Kandidaten-Systeme

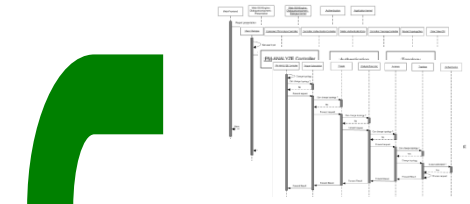
→ *Qualitative* Bewertung!

Situation 2: Bewertung einer einzelnen Architektur

- Ergebnisse von SAAM
  - Eine Notation und eine Beschreibung für die Architektur
  - Schwachstellen / Problemstellen, an denen die Architektur eventuell nicht die Anforderungen erfüllt

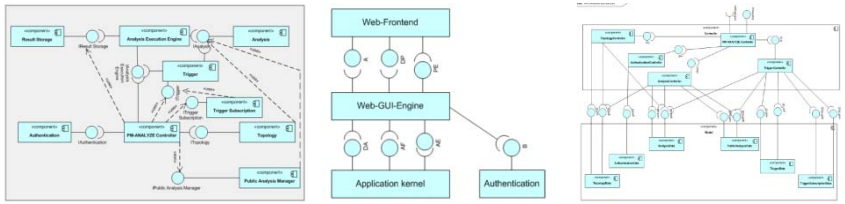

Situation 1:
SAAM - Überblick
SAAM
Motivation
Szenarien
SAAM
SWA-Vergleich
SWA-Bewertung
WRCS
EPOC
IIS
Zusammen-
fassung
Literatur
Diskussion
Beispiel
Entwickeln und
Priorisieren von Szenarien
iterieren
Entwickeln der
Architektur-Beschreibungen
Klassifizieren
von Szenarien
Identifizieren
betroffener Komponenten
Bewerten betroffener
Komponenten
Szenario-Interaktionen
gewichten
Bewerten der
Architekturen

Datenfluss-Sicht



Szenarien

Anwendung auf



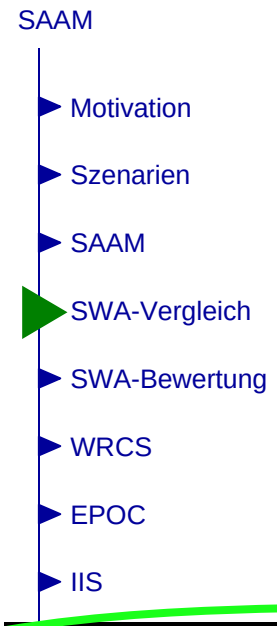
Beschreibung verschiedener Architekturen

Bewertung

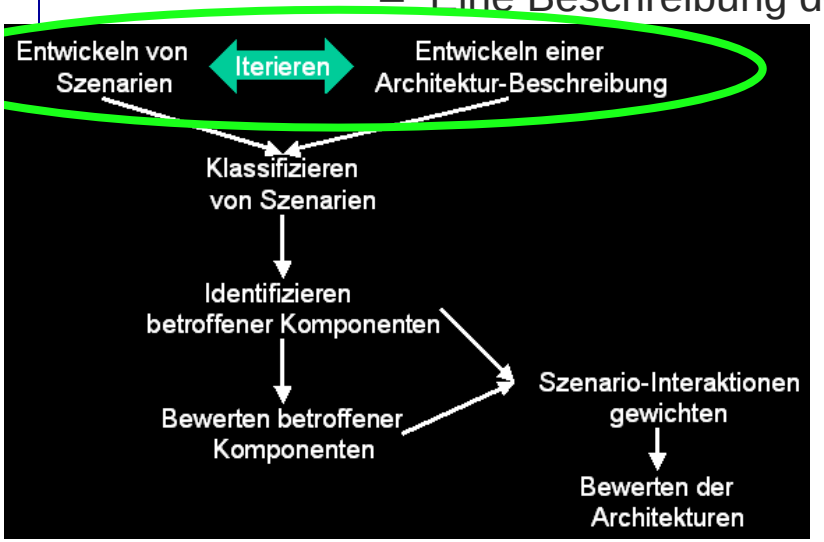
	Zu-verlässig-keit	Perfor-mance	Änder-bar-keit	Erweiter-barkeit	Testbar-keit	Wieder-verwend-barkeit	Inkre-men-telle Auslie-ferung	Ver-ständ-lichkeit direktes Szena-rio	Σ
TA	-1	1	-1	-1	0	-1	0	2	-1
QA	-1	0	0	1	0	0	0	-1	-1
LA	0	-1	1	1	1	1	2	1	6
MA	0	0	0	1	1	0	1	0	3

Ergebnis: relative Rangfolge der Architekturen

SAAM-Durchführung 1/5: Szenarien & Architekturbeschreibung(en) entwickeln



- Diese Schritte beeinflussen einander
  - Die Szenarien bestimmen den notwendigen Detailgrad der Architekturbeschreibungen
  - Die Generierung neuer Szenarien endet, wenn neue Szenarien die Darstellungen nicht weiter beeinflussen
- Ergebnisse
  - Szenarien zur Beschreibung ...
    - → aktueller und zukünftiger Anwendungen der Systeme sowie
    - → relevanter Aufgaben der Systeme mit den jeweils involvierten Rollen
    - → mit einer Priorisierung (einige Szenarien sind wichtiger / wahrscheinlicher als andere)
  - Eine Beschreibung der Architektur jedes Kandidaten-Systems

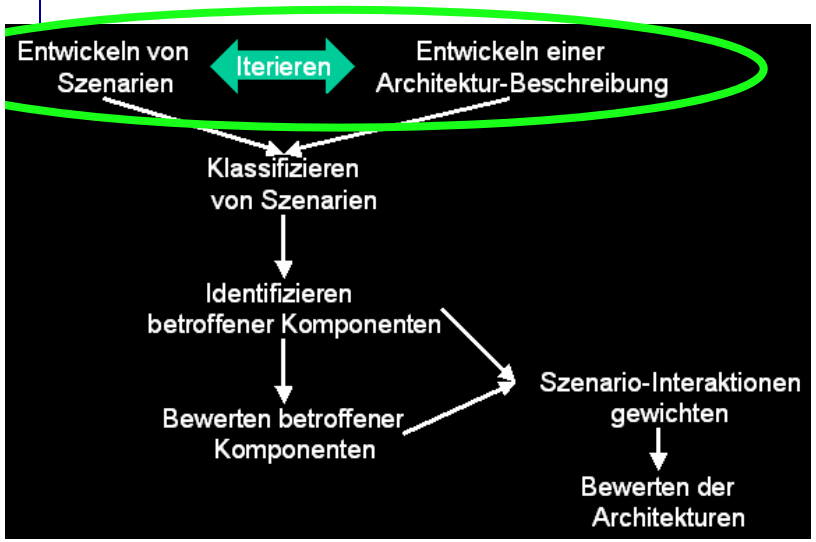


Szenarien & Architekturbeschreibung entwickeln

SAAM

- Motivation
- Szenarien
- SAAM
- SWA-Vergleich
- SWA-Bewertung
- WRCS
- EPOC
- IIS

- Ein Architekt sollte für eine SAAM-Bewertung folgende Dinge vorbereiten
  - Eine Beschreibung der logischen (und/oder *Uses*-)Sicht mit einer klaren Zuordnung von Zuständigkeiten zu Komponenten
  - Eine Beschreibung der Interaktionsmechanismen der Komponenten
  - Eine dynamische, z.B. Datenfluss-Sicht, welche die Interaktionen zur Laufzeit des Systems beschreibt
  - Eine *Uses*-Sicht, die zeigt, wie Teile des Systems entwickelt / ersetzt / erweitert werden können

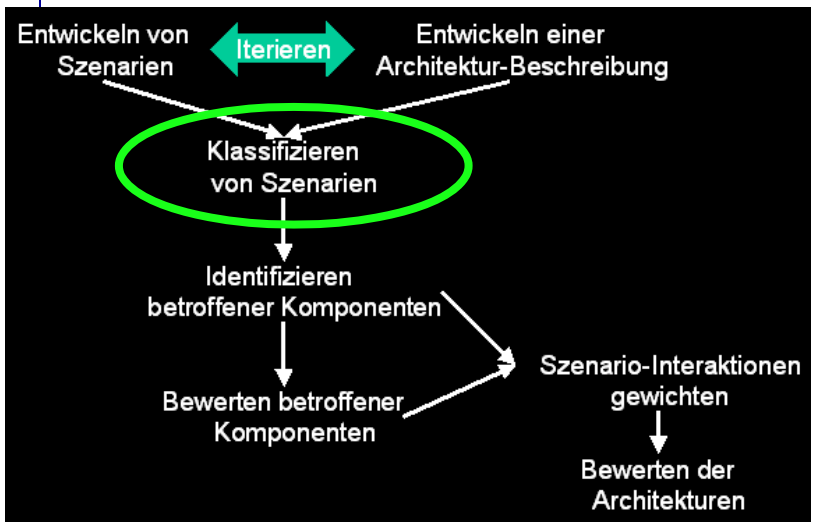


Klassifizieren von Szenarien

SAAM

- Motivation
- Szenarien
- SAAM
- SWA-Vergleich
- SWA-Bewertung
- WRCS
- EPOC
- IIS

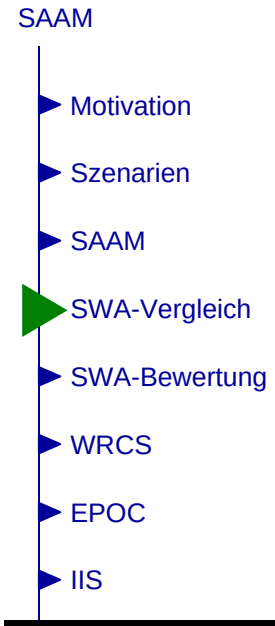
- Die zuvor entwickelten Szenarien werden klassifiziert als
  - **direkt**:
    wenn sie (von einem Kandidaten-System) *ohne Modifikation* unterstützt werden
  - **indirekt**:
    sonst
- Ergebnis
  - Eine Liste klassifizierter Szenarien

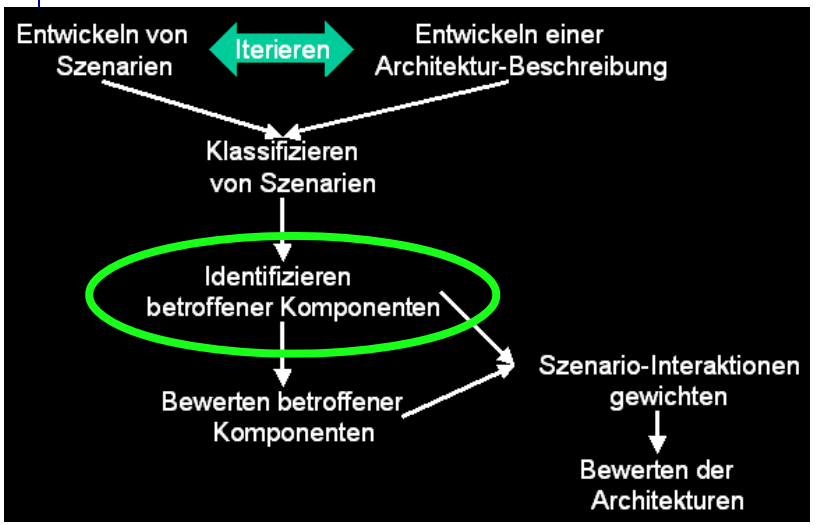


Anmerkung:
Systeme, die ein Szenario direkt unterstützen,
werden (natürlich) am besten bewertet...

Identifizieren betroffener Komponenten

- Für jedes Kandidaten-System und jedes *indirekte* Szenario
  - Notwendige Änderungen an der Architektur werden aufgelistet
  - Betroffene Komponenten werden identifiziert
- Ergebnis
  - Eine Liste von Komponenten, die von den Szenarien betroffen sind



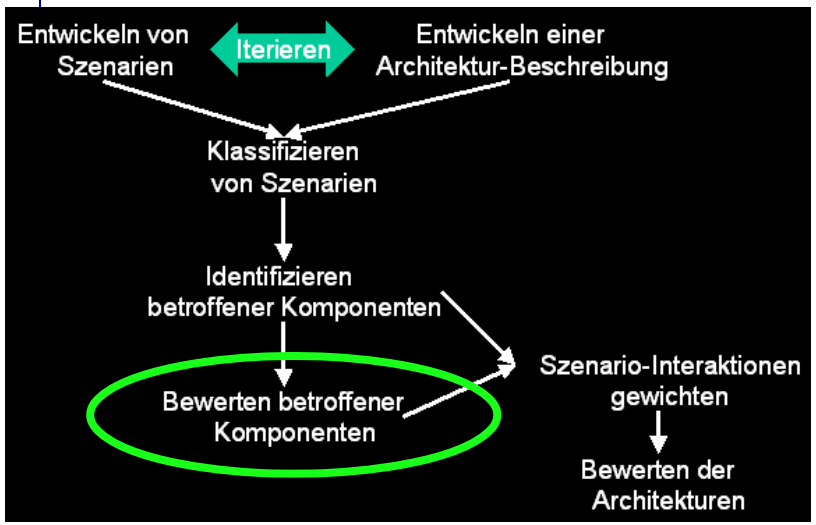


Bewerten betroffener Komponenten: Aufwandsschätzung

SAAM

- Motivation
- Szenarien
- SAAM
- SWA-Vergleich
- SWA-Bewertung
- WRCS
- EPOC
- IIS

- Für jedes Kandidaten-System und jede betroffene Komponente
  - Aufwand und/oder Kosten für die Modifikation werden geschätzt: Personen-Tage, -Wochen, -Monate, -Jahre
- Ergebnis
  - Eine Liste betroffener Komponenten mit den potenziellen Änderungskosten

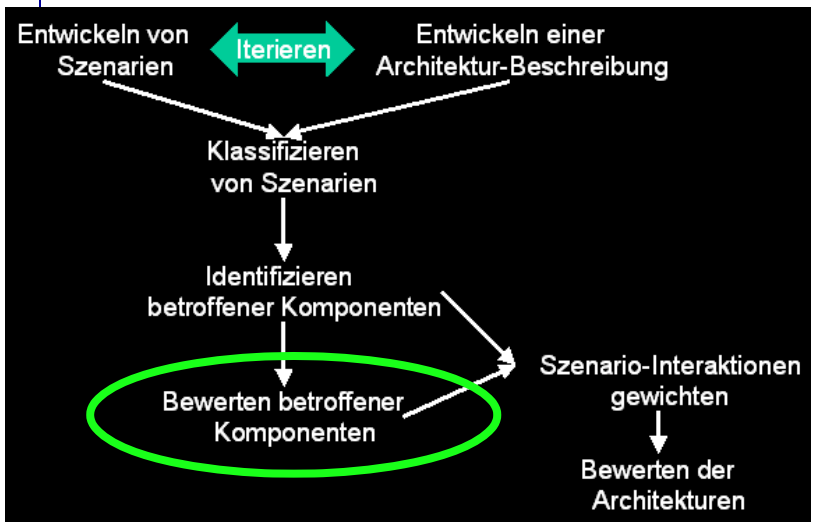


Bewerten betroffener Komponenten: Szenario-Interaktion

SAAM

- Motivation
- Szenarien
- SAAM
- SWA-Vergleich
- SWA-Bewertung
- WRCS
- EPOC
- IIS

- Drei mögliche Gründe, wenn eine Komponente von mehreren Szenarien betroffen ist:
  - Die Szenarien gehören eigentlich zur selben Klasse: *high cohesion:* das ist *gut*!
  - Die Szenarien gehören zu unterschiedlichen Klassen, die betroffenen Komponenten könnten aber weiter unterteilt (dargestellt) werden: → Die Architekturbeschreibung ist weiter zu *verfeinern*
  - Die Szenarien gehören zu unterschiedlichen Klassen, die betroffenen Komponenten könnten nicht weiter unterteilt (dargestellt) werden: Keine gute/angemessene Komponentenaufteilung (*separation of concerns*)

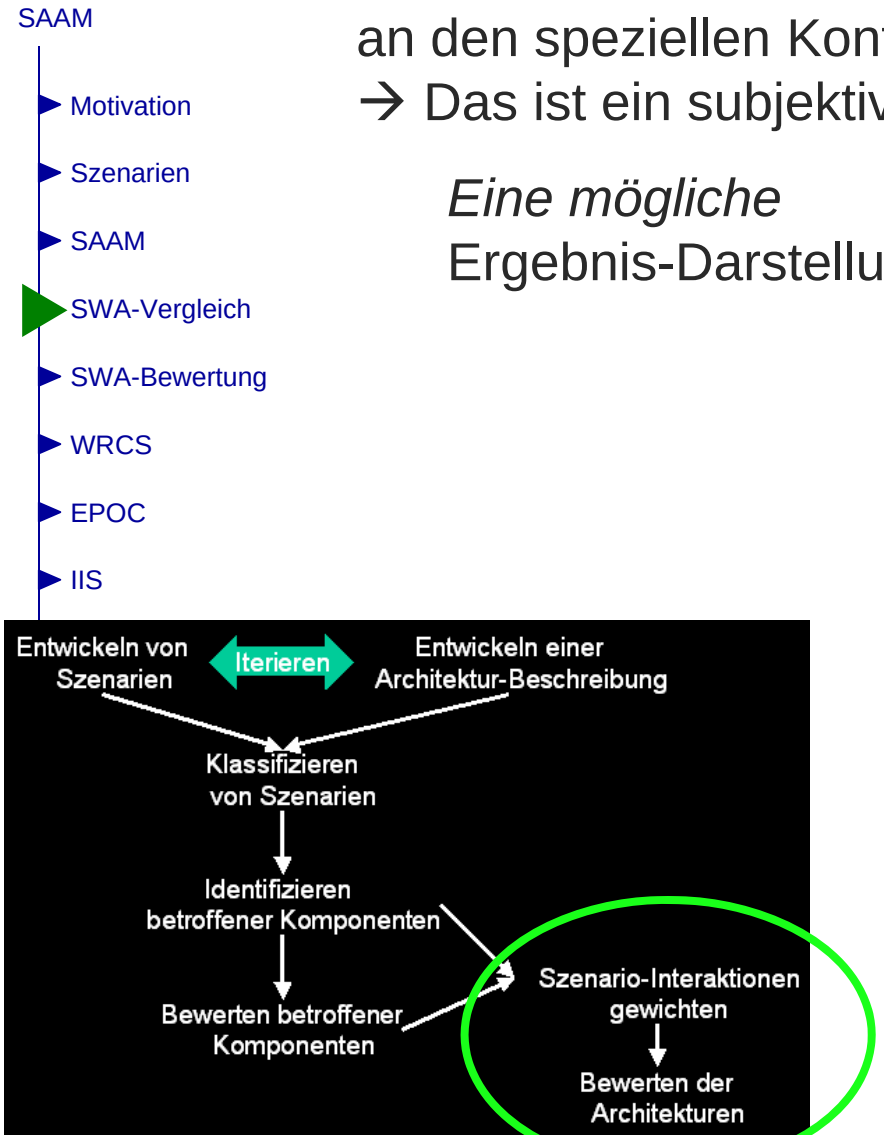


Ergebnis:
potenzielle Schwachstellen
bzw. Problemstellen
in der Architektur

Architekturen bewerten

- Die Methode zur Bestimmung einer Rangfolge muss sorgfältig an den speziellen Kontext angepasst werden
  → Das ist ein subjektiver Prozess!

  Eine mögliche
  Ergebnis-Darstellung:



	Gewicht	Aufwand (PT)		
		Architektur-Kandidat 1	Architektur-Kandidat 2	Architektur-Kandidat 3
Szenario 1	5	1	2	2
Szenario 2	4	1	3	4
Szenario 3	3	0	6	0
Szenario 4	2	8	5	2
...				
Szenario 12	2	7	8	4
Interakt. 2, 3	8	-	x	-
Interakt. 4, 8	3	x	x	-
Ranking		42	77	38

Situation 2: SAAM - Überblick

Entwickeln und Priorisieren von Szenarien

iterieren

Entwickeln *einer* Architektur-Beschreibung

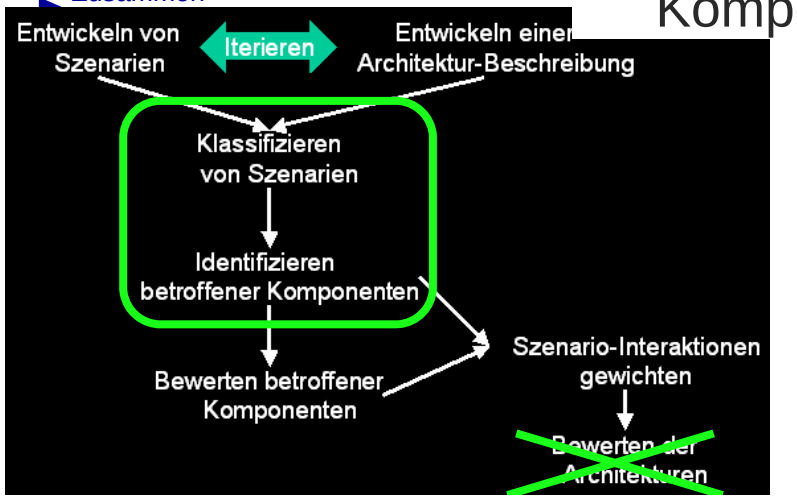
SAAM

- Motivation
- Szenarien
- SAAM
- SWA-Vergleich
- SWA-Bewertung
- WRCS
- EPOC
- IIS
- Zusammen-

Klassifizieren von Szenarien
inklusive
Identifizieren
betroffener Komponenten

Bewerten betroffener Komponenten

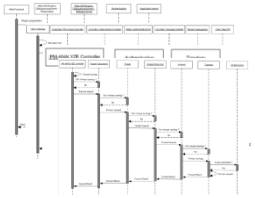
Szenario-Interaktionen gewichten



Datenfluss-Sicht

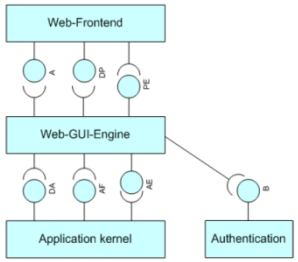
SAAM

- Motivation
- Szenarien
- SAAM
- SWA-Vergleich
- SWA-Bewertung
- WRCS
- EPOC
- IIS
- Zusammen-fassung
- Literatur
- Diskussion
- Beispiel



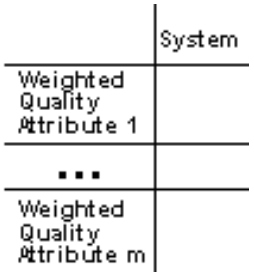
Szenarien

Anwendung auf



Beschreibung der Architektur

Zusammenfassung (Bewertung)



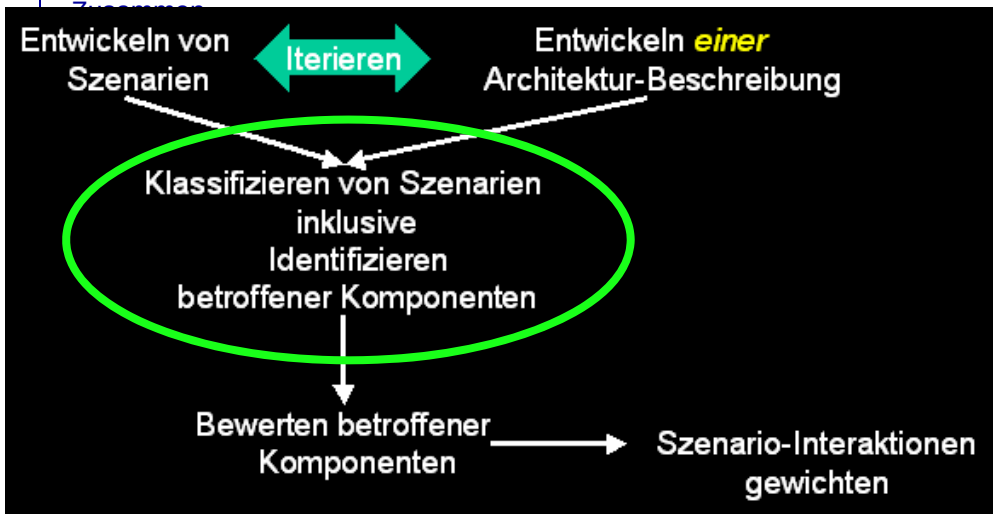
Ergebnis: potenzielle Schwachstellen der Architektur

Abweichung 1: Klassifizieren von Szenarien

SAAM

- Motivation
- Szenarien
- SAAM
- SWA-Vergleich
- SWA-Bewertung
- WRCS
- EPOC
- IIS

- Die zuvor entwickelten Szenarien werden klassifiziert als
  - direkt, wenn sie ohne Änderung von dem betrachteten System unterstützt werden
  - indirekt, sonst
    - → Notwendige Änderungen an der Architektur werden aufgelistet
    - → Betroffene Komponenten werden identifiziert
- Ergebnis
  - Eine Liste klassifizierter Szenarien
  - Eine Liste von Komponenten, die von den Szenarien betroffen sind

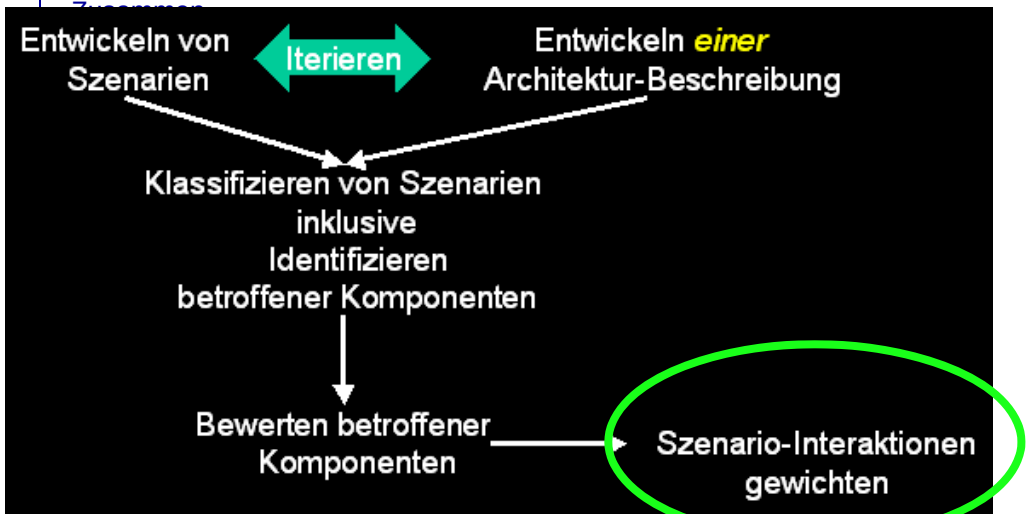


Abweichung 2: Szenarien gewichten

- Einige Szenarien / Szenario-Interaktionen werden als wichtiger / wahrscheinlicher angesehen als andere

SAAM

- Motivation
- Szenarien
- SAAM
- SWA-Vergleich
- SWA-Bewertung
- WRCS
- EPOC
- IIS



Mögliche Ergebnis-Darstellung:

	Gewicht	Aufwand (PT) Architektur-Kandidat
Szenario 1	5	1
Szenario 2	4	1
Szenario 3	3	0
Szenario 4	2	8
...		
Szenario 12	2	7
Interakt. 1, 2, 4	5	
Interakt. 6, 8	2	
Interakt. 7, 12	2	
Ranking		39