Design im Domain- und Application Engineering

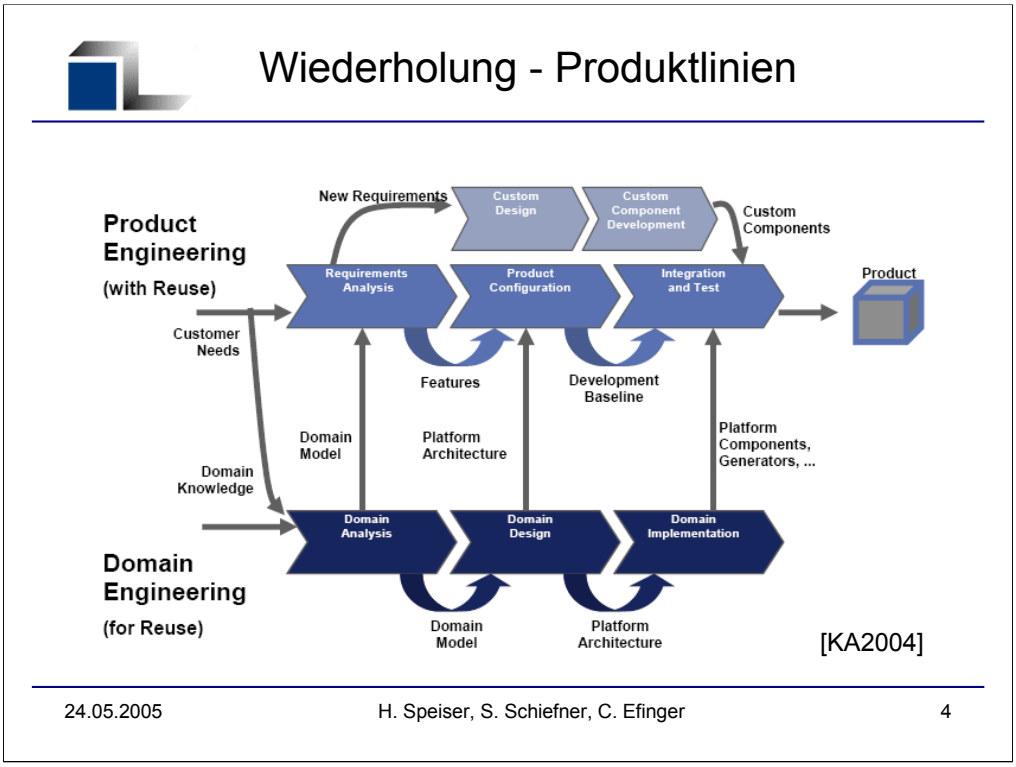
Produktlinien – Architekturentwicklung

- Die Architektur ist Kernelement der Produktlinie.
- Die Architektur entscheidet über den Erfolg oder Misserfolg einer Produktlinie.

[BKPS 2004]

Übersicht

- Wiederholung
  - Produktlinien
  - Requirement Engineering
- Architektur-Entwicklung
  - Motivation
  - AE im Domain- und Application-Engineering
  - Komponenten-Kauf
  - Komponenten-Wiederverwendung
- Fazit / Diskussion



Einführung in die Produktlinien

Die Produkte der Software-Entwicklung sind meistens keine Neuentwicklung, sondern bestehen meistens aus vorhandenen Software-Systemen die erweitert oder modifiziert werden.

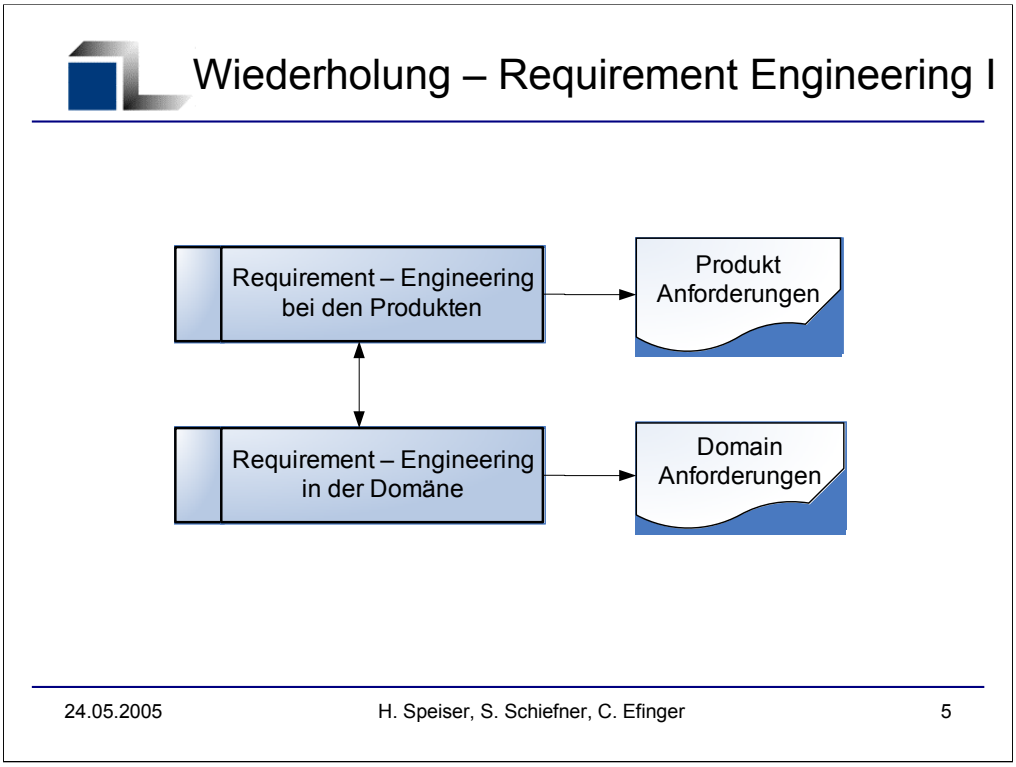
•Vorteile: Widerverwendbare Komponenten, Komponentenbasierte Entwicklung, sauberer Architekturstil, hoher ROI, niedrige time-to-market, höhere Qualität, größeres Produktfolio

•Nachteil: Sehr aufwendig, dokumentengetrieben, lohnt sich erst ab der 3. Wiederverwendung, Auch organisatorische Trennung von Application und Domain-Engineering

Requirement Engineering:

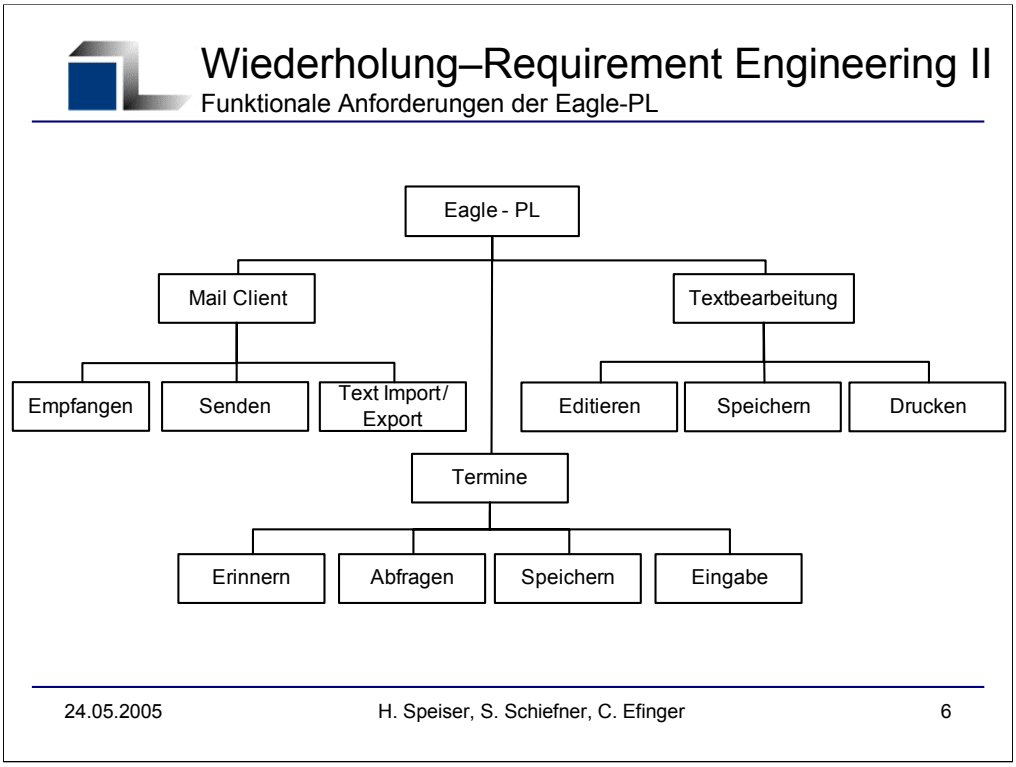
•Durch verschiedene Analyseverfahren findet in der Requirement-Engineering Phase die Definition der Anforderungen statt.

•Output dieser Phase ist eine Sammlung von funktionalen bzw. nicht-funktionalen Anforderungen.



•Bei den Produktlinien wird folgendermaßen vorgegangen

- •Zuerst Requirement – Engineering bei den Produkten
- •Danach herausfinden, welche Gemeinsamkeiten in die Domäne mitgenommen werden können
- •Resultat ist bei beiden Aktivitäten jeweils die Anforderungen, wobei zu beachten ist, dass es funktionale bzw. nicht funktionale Anforderungen gibt.



Produktlinie Eagle

•Eagle soll es in verschiedenen Versionen geben, eine kostenlose Edition, eine lowcost-Edition und eine professional-Edition

•Die Unterscheidung der Produkte findet sich im Funktionsumfang wieder

•Diese Folie enthält nur die Basisanforderungen. Erweiterte Anforderungen finden sich später im Vortrag.

Wiederholung–Requirement Engineering II

Nicht-Funktionale Anforderungen der Eagle-PL

- Preiswert
- Plattformunabhängig
- Integrität
- Leicht konfigurierbar
- Stabilität
- Performance
- Ausfallsicher
- Datensicherheit

•Neben funktionalen Anforderungen gibt es natürlich auch nicht funktionale Anforderungen, die sich nur schwer zu einer Komponente zuzuordnen sind, aber für späteren abgeleiteten Programme erfüllt sein müssen.

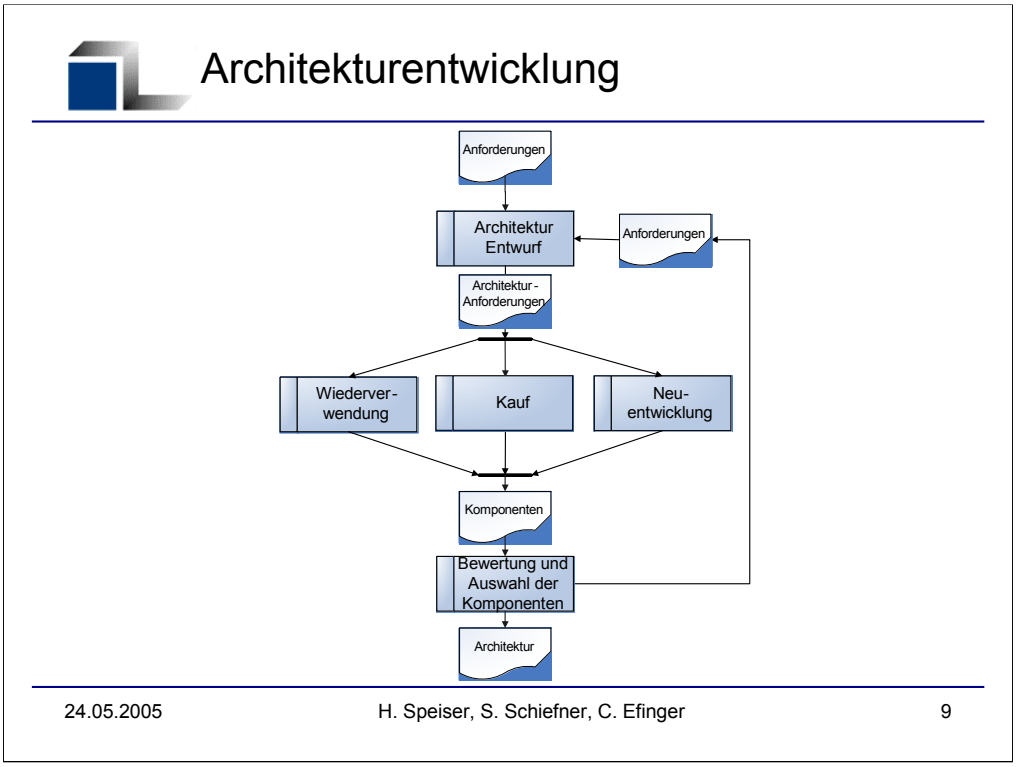
Architektur - Motivation

- Kernteil der Software Produktlinie
- Dokumentiert die wichtigsten Entwurfskomponenten und Variationspunkte
- Referenzarchitektur der Produktlinie
- Referenzarchitektur setzt die wichtigsten Anforderungen der PL um.
- Entwurfsfehler führt zu enormen Zusatzkosten
- Architektur muss überprüft ständig evaluiert werden
- Architektur wird im Application Engineering konkretisiert

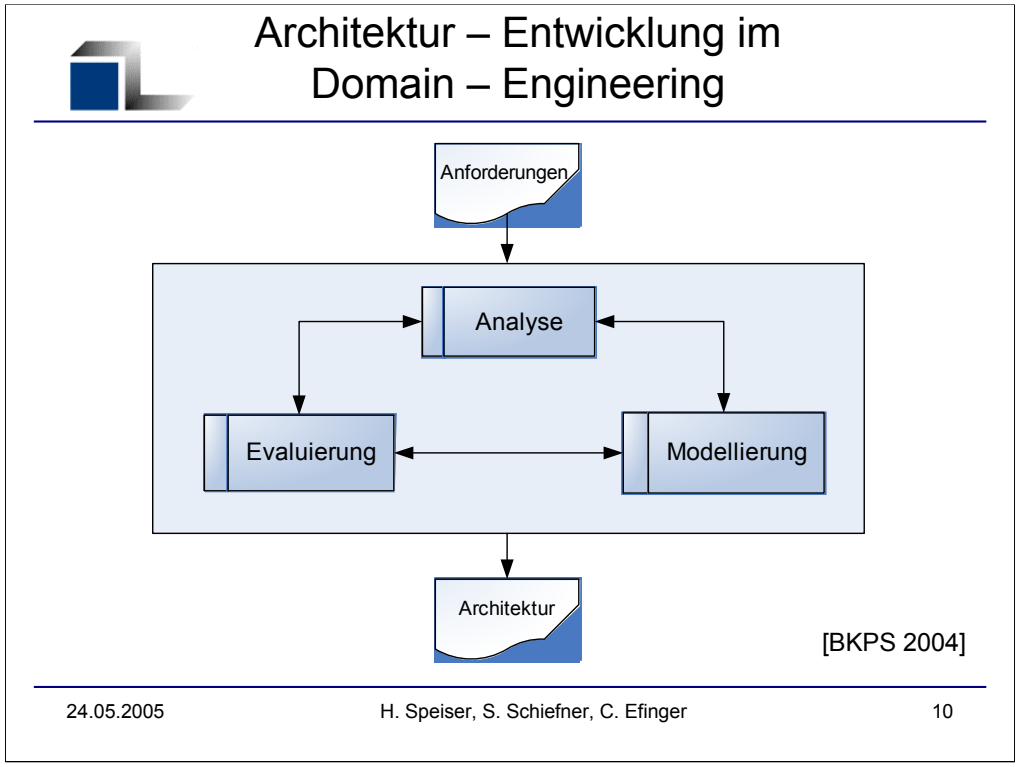
Notation zum Begriff Referenzarchitektur:

•Es sollte beim Entwurf der Referenzarchitektur darauf geachtet werden dass es auch bei dieser verschieden Sichtweisen gibt

•Logische Sicht, Entwickler Sicht, Prozess Sicht und die Physikalische Sicht



Auch Eingehen in die Make Buy Mine Entscheidung



5min, Literatur

Einführung ins Domain-Engineering

Was bekommen wir von der Anforderungsanalyse, Scoping teilweises herausfinden von Architekturtreibern, geht unmittelbar in den Entwurf, dient aber immer als Vorschlag ist nicht zwingend erforderlich, keine feste Vorlage

Beschreibung der einzelnen Phasen:

Analyse

•Analysephase – Bestimmung der Anforderung

•Identifikation der Architekturtreiber und Mechanismen

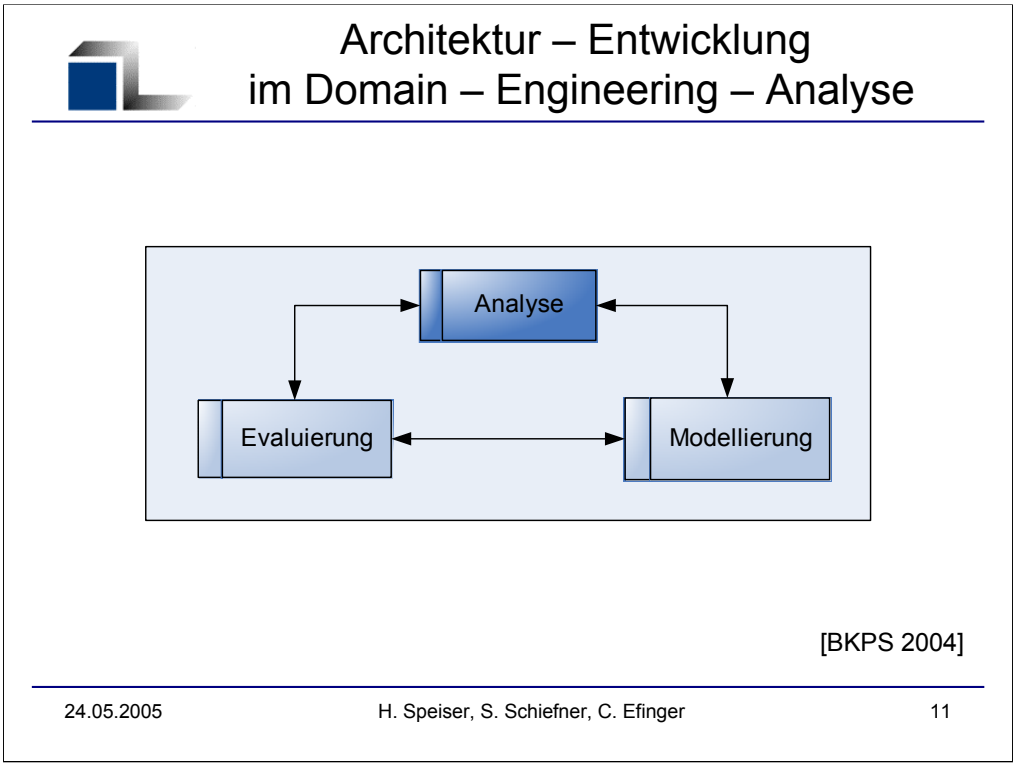
Evaluierung

•Erkennen von Risiken in den Entwurfsentscheidungen

Modellierung

•Architekturdokumentation mit Variationspunkten

•Quadrad ist ein iteratives inkrementelles Vorgehen eine qualitativ hochwertige Architektur zu schaffen



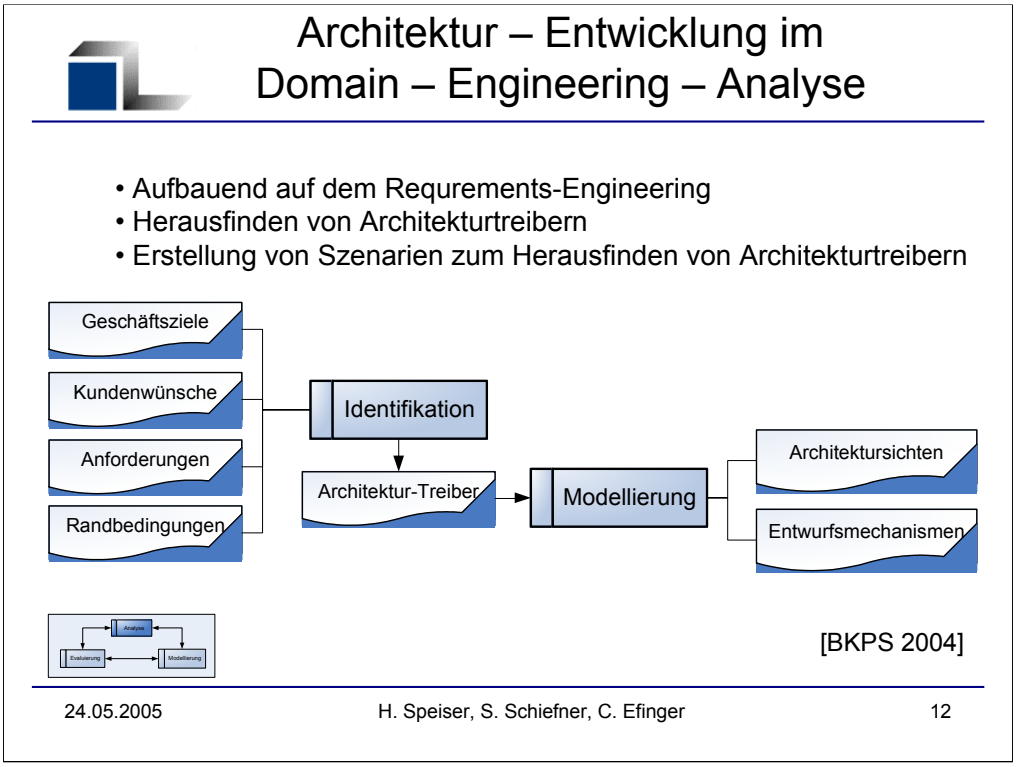
3min

Baut auf dem Requirement-Engineering auf

Aktivitäten zur Identifikation von architektur-relevanten Anforderungen

Erarbeitung von Entwurfsmechanismen für die Umsetzung von Architektur-Treibern

Einführung in die Analyse-Phase, also Architekturtreiber herausfinden



•Architekturtreiber: Identifikation architekturrelevanter Anforderungen, bewerten bzw. priorisieren von Anforderungen

•Eingehen auf verschiedene Szenarien um Architektur-Treiber herauszufinden. Szenarios können auch verwendet werden um die Architektur zu testen.

•Also ist eine der Aktivität Aufstellung von Szenarien zur Identifikation von Architektur-Anforderungen.

•Die Liste der Architektur-Treiber setzt sich üblicherweise aus konkreten Qualitätsanforderungen zusammen: Sicherheit, Zuverlässigkeit, Verfügbarkeit, Konfigurierbarkeit.

•Entwurfsmechanismen zur Lösung der Anforderungen. Vergleichen und Bewertung von verschiedenen Entwürfen. Je nach dem welches Entwurfsmuster passend für die Anforderungen ist.

•Architektursichten: Logische Sicht, Entwickler Sicht, Prozess Sicht und die Physikalische Sicht

•Für die Darstellung der Elemente einer Sicht kann eine Modellierungssprache, z.B. die UML verwendet werden. Sie kann durch formale Architekturbeschreibungssprachen ergänzt bzw. verfeinert werden.

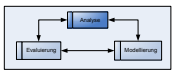
Architektur – Entwicklung im Domain – Engineering – Analyse

Beispiel

Anforderung: Einfache Konfigurierbarkeit der Komponenten der Eagle PL in Eagle Produkte.

Szenario: Komponenten der PL können in die Eagle Produkte durch An- bzw. Abmelden integriert werden.

Entwurfsmechanismus: Durch ein Observer, der in den Produkten integriert ist, können Komponenten an- bzw. abgemeldet werden.



•Architekturtreiber: Identifikation architekturrelevanter Anforderungen, bewerten bzw. priorisieren von Anforderungen

•Eingehen auf verschiedene Szenarien um Architektur-Treiber herauszufinden. Szenarios können auch verwendet werden um die Architektur zu testen.

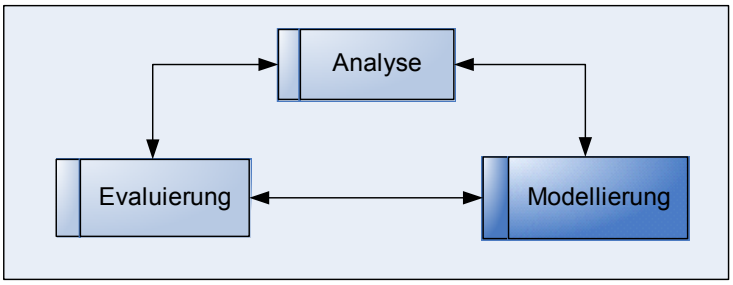
•Also ist eine der Aktivität Aufstellung von Szenarien zur Identifikation von Architektur-Anforderungen.

•Die Liste der Architektur-Treiber setzt sich üblicherweise aus konkreten Qualitätsanforderungen zusammen: Sicherheit, Zuverlässigkeit, Verfügbarkeit, Konfigurierbarkeit.

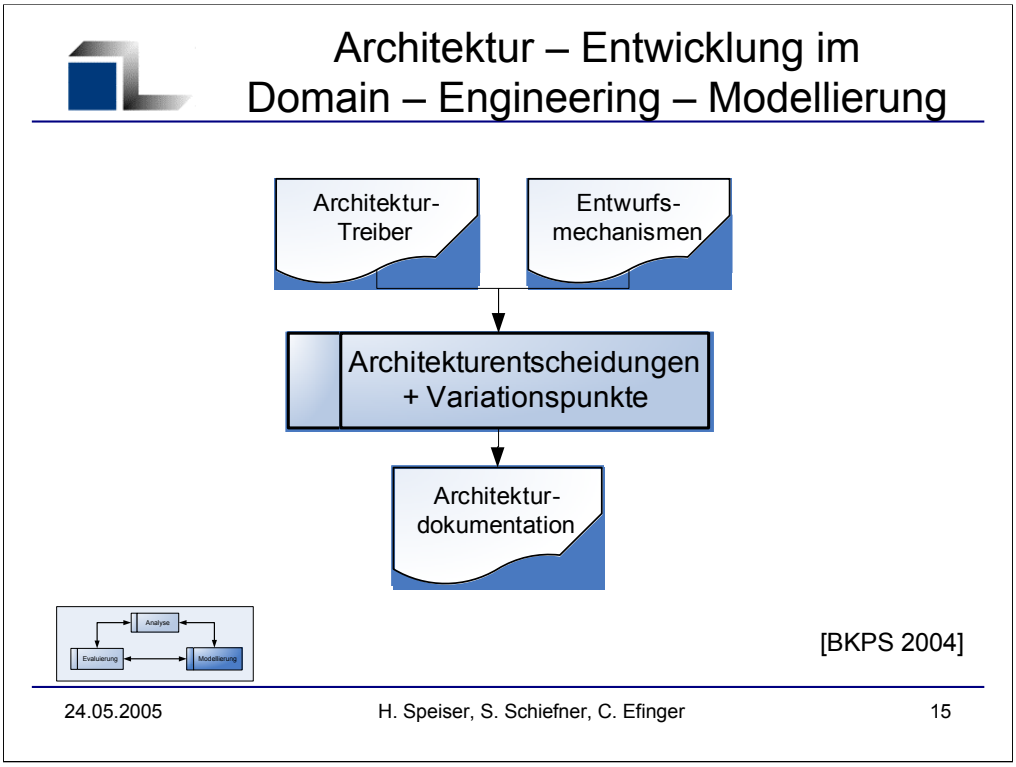
•Entwurfsmechanismen zur Lösung der Anforderungen. Vergleichen und Bewertung von verschiedenen Entwürfen. Je nach dem welches Entwurfsmuster passend für die Anforderungen ist.

•Architektursichten: Logische Sicht, Entwickler Sicht, Prozess Sicht und die Physikalische Sicht

Architektur – Entwicklung im Domain – Engineering – Modellierung



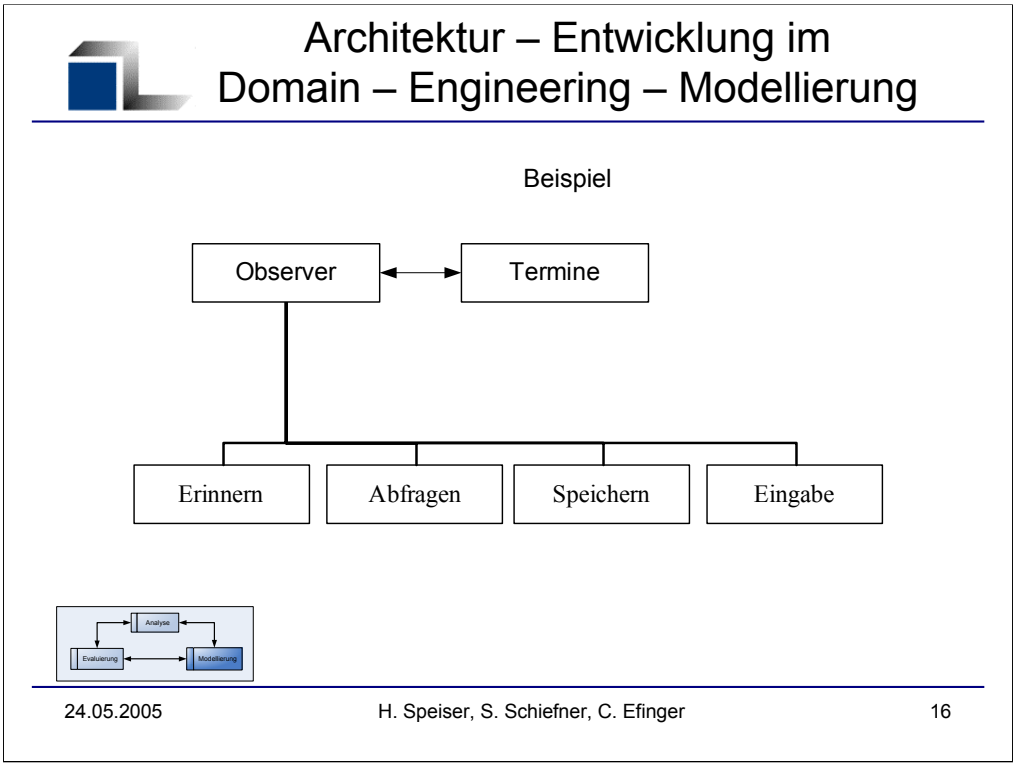
[BKPS 2004]



Architekturentwurf setzt häufig auf bereits existierende Architekturen auf.

Es werden Architekturen entworfen, die die Eigenschaften von Komponenten bestmöglich umsetzen, vergleiche SWA-Vorlesung zur Auswahl geeigneter Architekturen mit Hilfe von Szenarien.

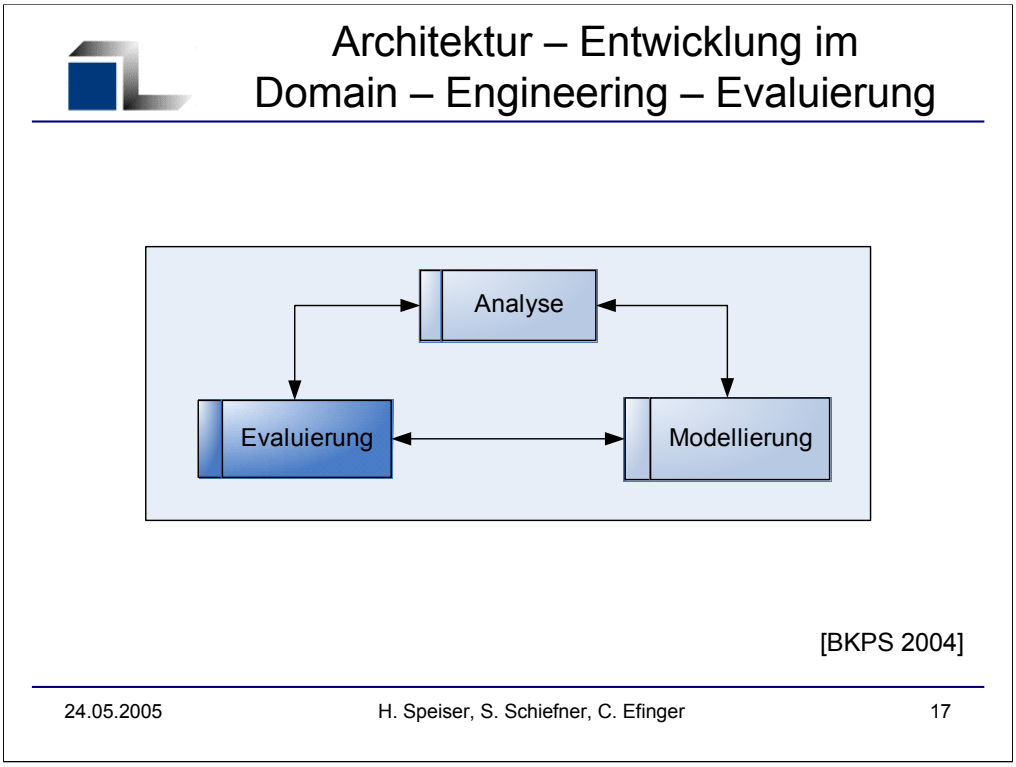
Vorstellung wie in der Architekturphase die Variabilität ausgedrückt werden kann.



Architekturentwurf setzt häufig auf bereits existierende Architekturen auf.

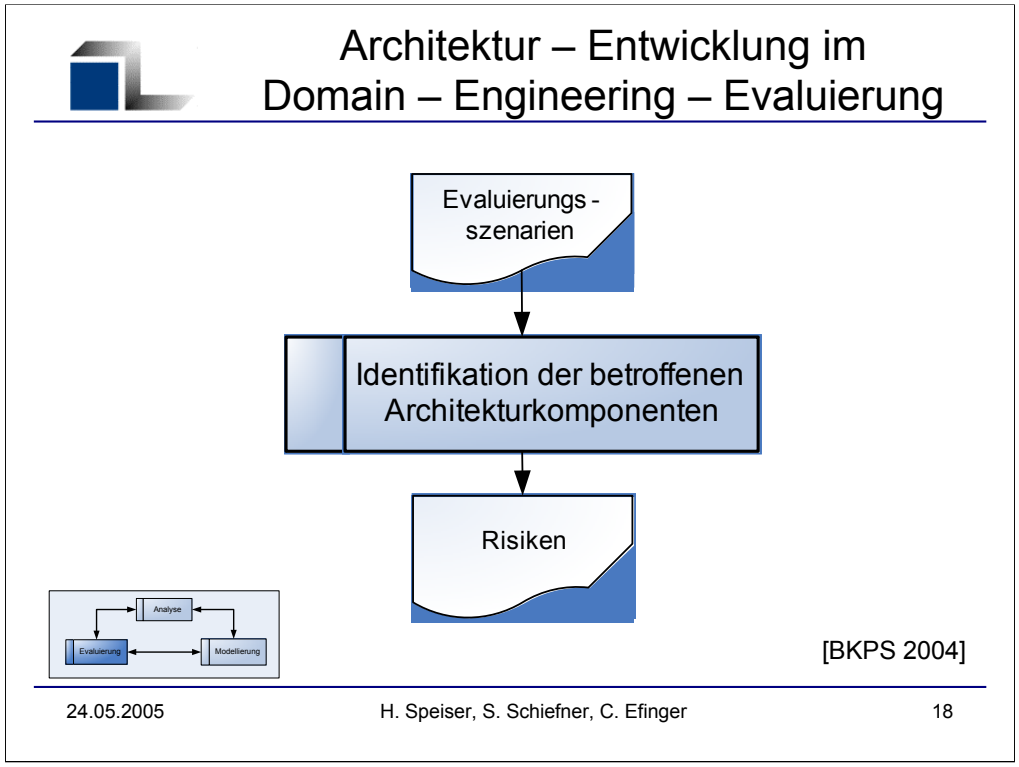
Es werden Architekturen entwurfen, die die Eigenschaften von Komponenten bestmöglich umsetzen, vergleiche SWA-Vorlesung zur Auswahl geeigneter Architekturen mit Hilfe von Szenarien.

Vorstellung wie in der Architekturphase die Variabilität ausgedrückt werden kann.



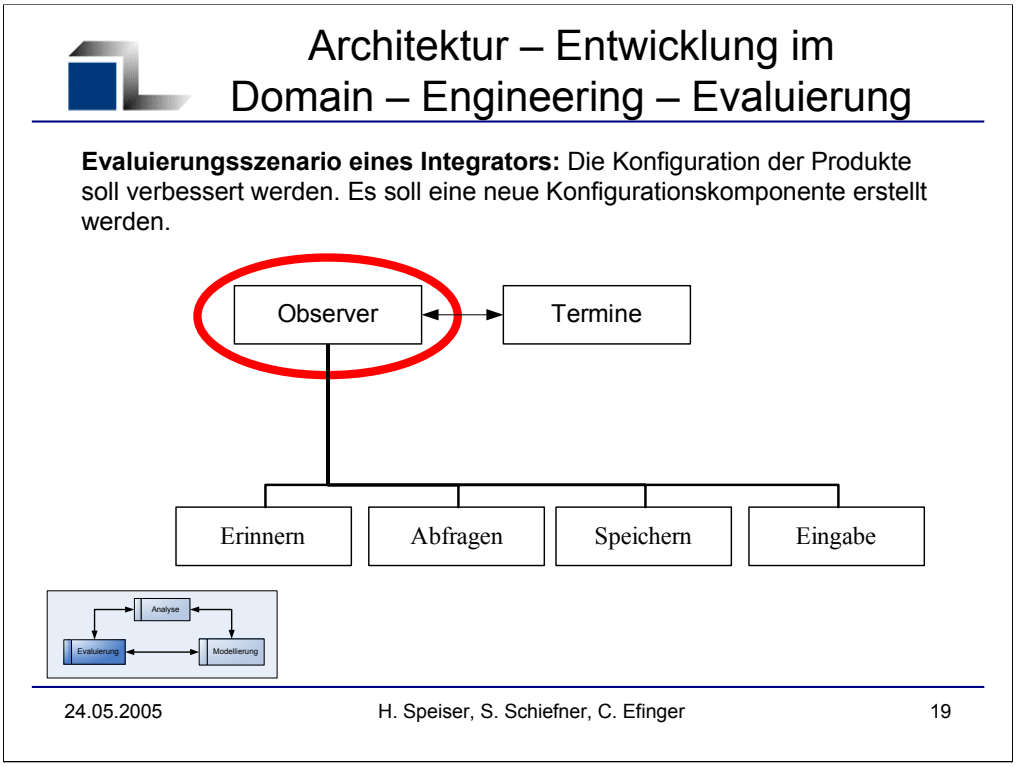
Einführung in die Evaluierung, also Herausfinden von Risiken,

Risikenanalyse bei der Bewertung der Szenarios bei den jeweilgen Architekturen
Bewertung der Architektur mit SAAM



Evaluierung der Architektur, Identifikation der betroffenen Architekturkomponenten. Identifikation der Architekturentscheidung. Daraus resultiert die Risiko – Bewertung.

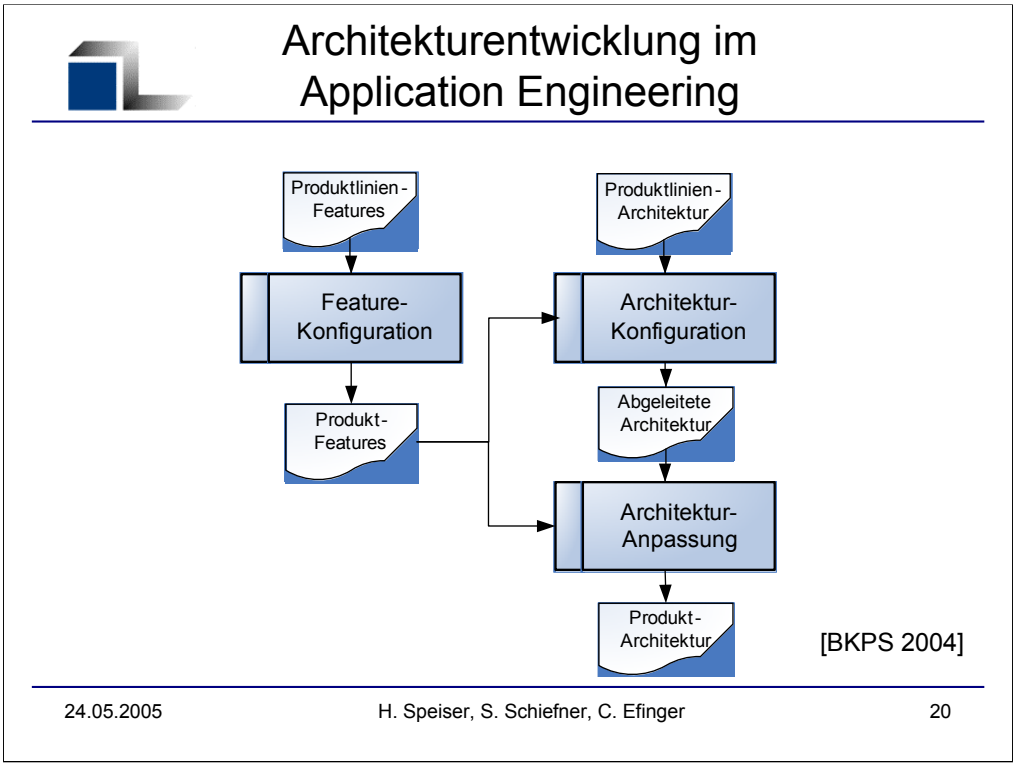
Die Risiko-Bewertung zeigt, welche Komponenten in der Architektur verbessert werden müssen. Gegebenfalls auch Anzeigen einer Neusturkturierung der Architekturdokumentation, falls unzureichend modelliert.



Evaluierung der Architektur, Identifikation der betroffenen Architekturkomponenten. Identifikation der Architekturentscheidung. Daraus resultiert die Risiko – Bewertung.

Die Risiko-Bewertung zeigt, welche Komponenten in der Architektur verbessert werden müssen. Gegebenfalls auch Anzeigen einer Neusturkturierung der Architekturdokumentation, falls unzureichend modelliert.

Daneben sollten Experten gefragt werden wie hoch der Aufwand der Änderungen ist.



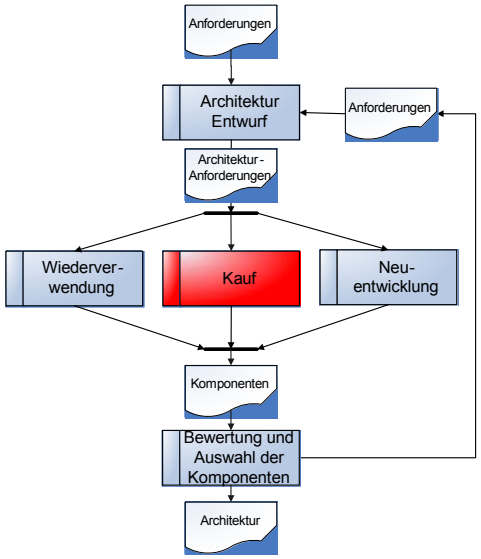
•Konkretisieren der Referenzarchitektur

•Was werden an Komponenten gebraucht, die noch nicht in der Referenzarchitektur beschrieben sind.

•Feature bauen, Abdeckung der Features bzw. nicht funktionalen Anforderungen

Einsatz von COTS-Komponenten (commercial off-the-shelf)

Gesamtüberblick

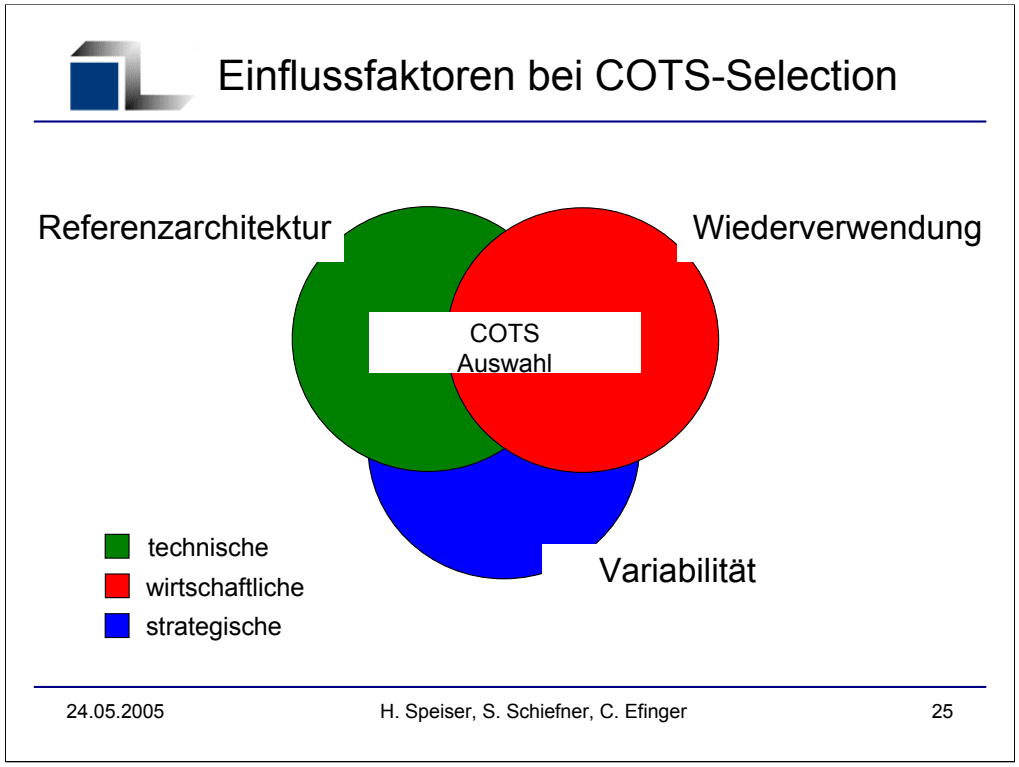


Gründe für

- sofort verfügbar
- keine kostenintensive Neuentwicklung
- Einsparungen bei der Entwicklungszeit
- wegfallen von Komponenten-Tests
- umfangreiche Funktionalität
- Updates und Support
- große Auswahl

Gründe gegen

- Verzögerung der Lizenzvergabe
- hoher Integrationsaufwand
- schlechte Dokumentation
- hohe Wartungskosten
- mangelnde Qualität
- mangelnde Variabilität
- keine Kontrolle von Updates
- Inkompatibilität zwischen Herstellern



Grundsätzliche Einflussfaktoren:

Technische –gewünschte Funktionalität mit dem geforderten Level an Zuverlässigkeit

Strategische – Gewünschter Nutzen in Bezug auf den Einsatzort, sowie im Hinblick auf die Zukünftige Entwicklung

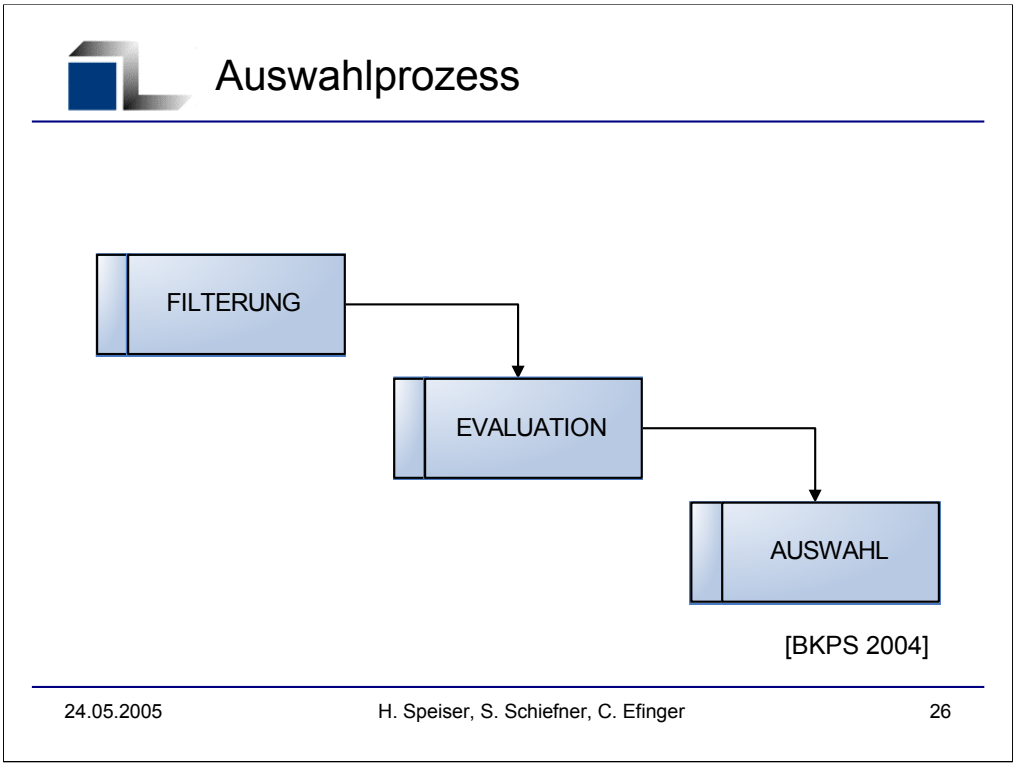
Wirtschaftliche – Integration in das neue System unter Berücksichtigung des Budgets und Zeitplans

Bei einer Produktlinie Zu überprüfen ist:

inwieweit die Komponenten die Anforderungen der Architektur erfüllen

inwieweit die Komponenten in die Referenzarchitektur integrierbar sind

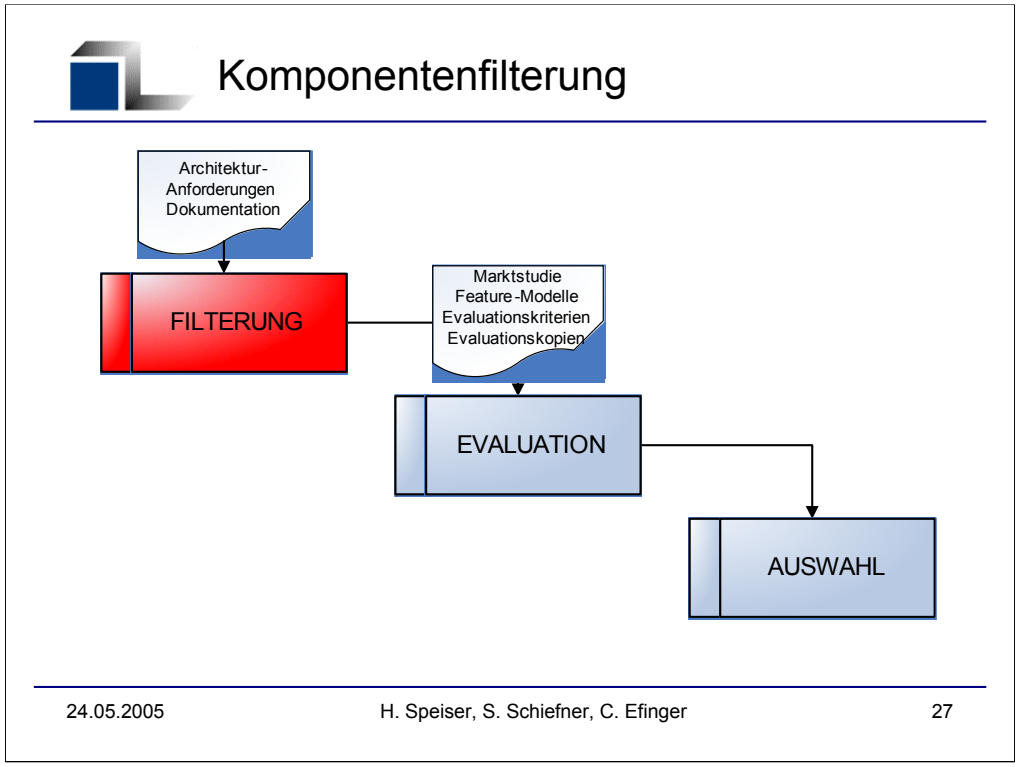
Inwieweit die Variabilität der Produktlinien durch Komponenten unterstütz wird



Filterung: Eine Vorauswahl der Komponenten mit Hilfe der Dokumentation

Evaluation: Eine genauere Untersuchung der Komponenten mit Hilfe der Evaluationonskopien

Auswahl: Auswahl der Komponenten



Marktstudie/Bewertung des Anbieters

Prüfung der Basisfunktionalität/Feature Modelle

Komponentenreduktion/Evaluationskopien

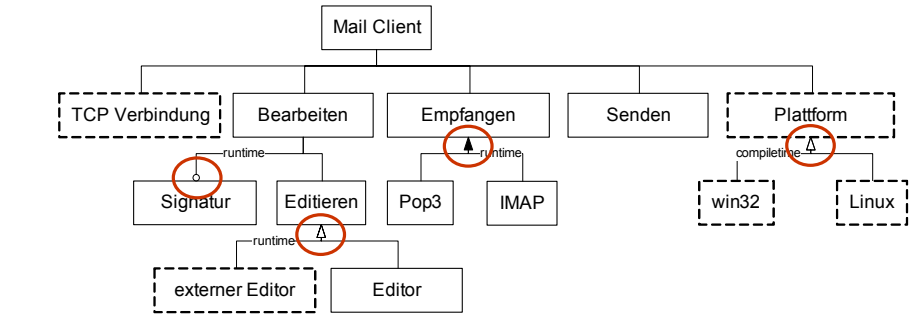
Bewertung der Stabilität des Unternehmens: Wichtig ist beispielsweise die Fragestellung nach der Zeitdauer der Marktpräsenz des Unternehmens und die Beurteilung des Risikos, dass das Unternehmen vom Markt verschwindet.

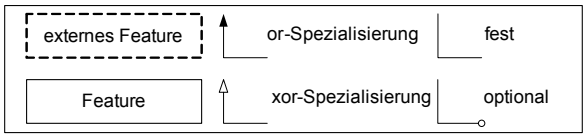
Bewertung des Entwicklungsprozesses: Vor einer Kaufentscheidung sollte der Entwicklungsprozess bewertet werden. Gesichtspunkte, wie Transparenz, Test- und Zertifizierungsaufwand spielen hierbei eine zentrale Rolle.

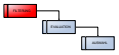
Bewertung des Wartungsvertrags: Beim Einsatz von COTS-Komponenten, übernimmt der Verkäufer die Wartung, da in den meisten Fällen der Quellcode nicht verfügbar ist. Daher ist es wichtig, Wartungsmodalitäten von vorneherein zu klären.

Bewertung des Kundenservices: Umfang und Art der Kundenbetreuung müssen auch in die Bewertung einfließen.

Feature-Modell





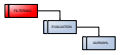


[GBS 2001]

Feature-Beschreibung

Name	*Editieren*
Beschreibung	*Ermöglicht editieren von e-Mails*
Klasse	*Basisfunktionalität*
Beziehungen	Erfordert zwingend: *Bearbeiten* Schließt aus: *keine*

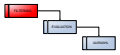
[BKPS 2004]

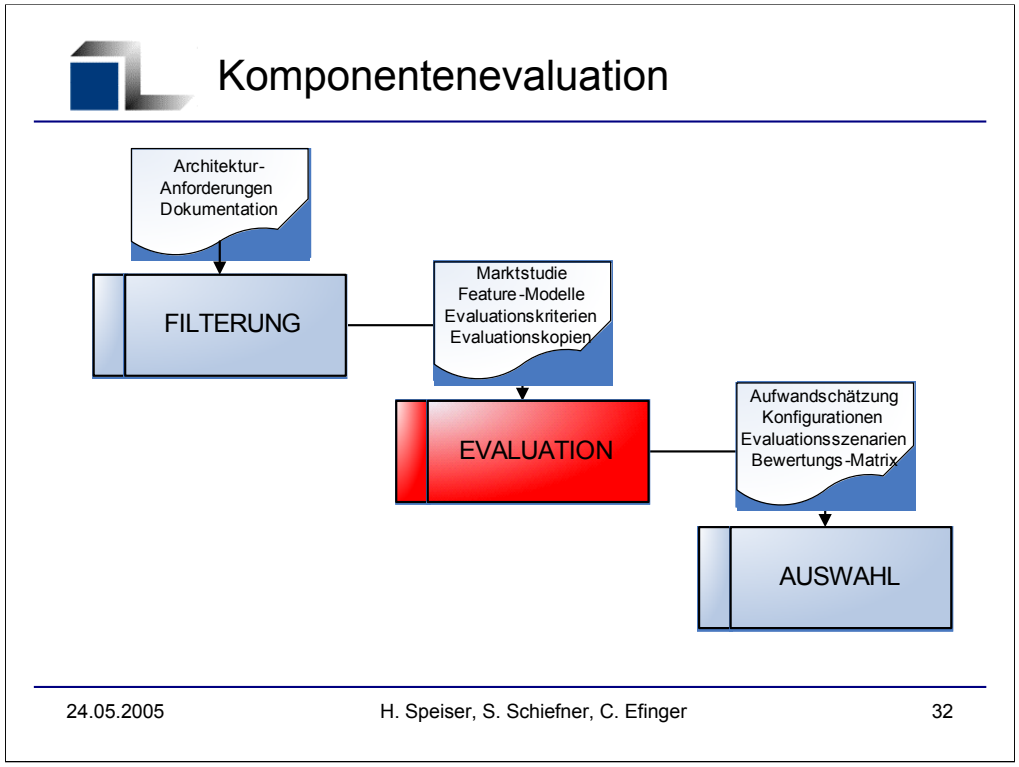


Evaluationskriterium

Nr.	*1.2*
Definition	*Stabilität*
Zweck	*Prüft die Stabilität der Komponente*
Messskala	*Rationalskala*
Einheiten der Messskala	*MTBF (Mean Time Between Failure)*
Akzeptanzschwelle	*5 Fehler innerhalb 20 Minuten*
Baseline	*Keine Fehler*
Feature Interaktionen	*Empfangen und Senden*
Priorität	*hoch*

[BKPS 2004]





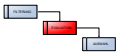
2-3 Komponenten werden genauer untersucht.

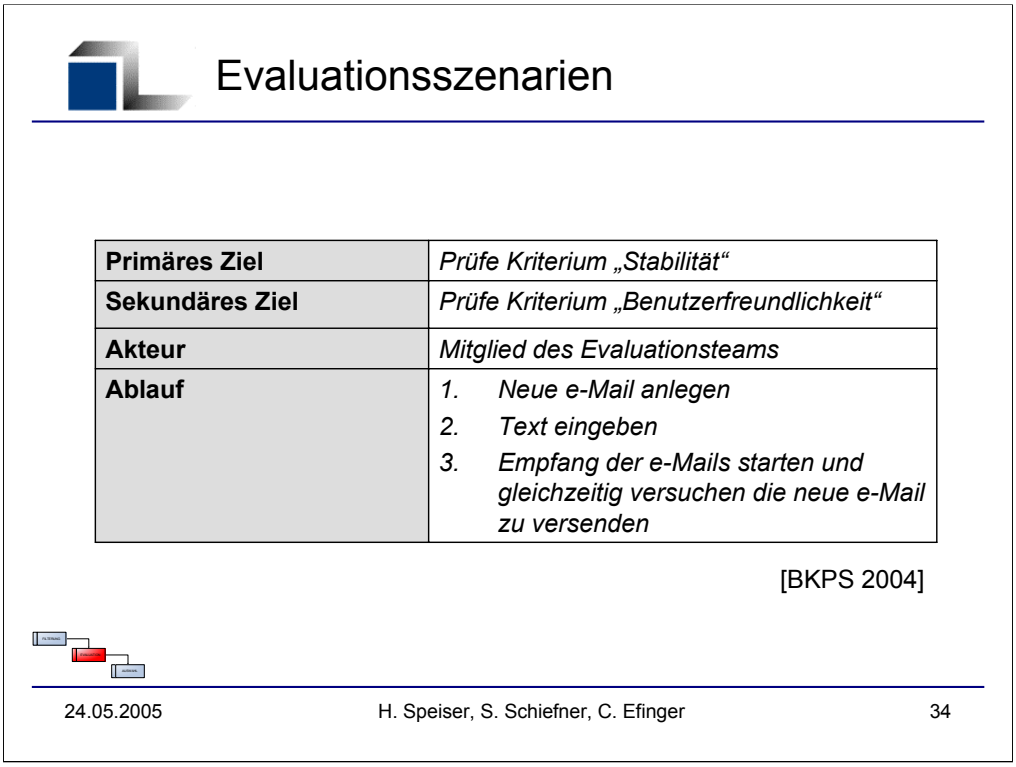
Funktionale (Feature-Modelle) sowie nichtfunktionale Anforderungen werden geprüft.

Diese werden in der "detailed component evaluation" einem detaillierten Bewertungsprozess unterzogen, in dem u.a. auch ggfs. notwendige Anpassungen der Anforderungen, Architektur und/oder die Variabilität der Produktfamilie untersucht werden.

Aufwandschätzung

- **Tailoring**
  Konfigurieren der Komponente für den geplanten Einsatz
- **Volatility**
  Aufwand durch neue Produktversionen der Komponenten
- **Glue Code**
  Erstellung von neuem Programmcode um
  - Datenaustausch zu ermöglichen
  - die Komponente um fehlende Funktionen bzw. fehlende Variabilität zu erweitern
  - die Komponente in das System einzubinden

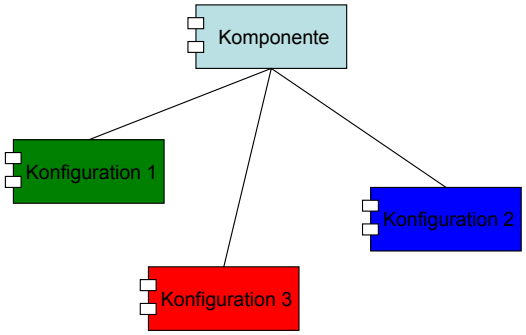




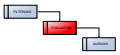
Zur Evaluierung verschiedener Konfigurationen der zur Auswahl stehenden Komponenten werden Szenarien entwickelt. Anschließend werden die zu evaluierenden Konfigurationen der Komponenten bestimmt und erzeugt.

Weiterhin wird festgelegt, welches Szenario auf welchen Konfigurationen der Komponenten ausgeführt werden soll. Jedes Szenario muss auf zumindest einer Konfiguration jeder Komponente getestet sein. Diese Szenarien werden dann (wie zuvor definiert) auf den Konfigurationen ausgeführt und die Ergebnisse in Form einer Matrix dokumentiert.

Konfiguration einer Komponente

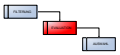


Produkt A
Produkt B
Produkt C



Konfiguration einer Komponente

- Gebotene Variabilität ermöglicht unterschiedliche Konfigurationen d.h. es gibt mehrere ausführbare Versionen der Komponente
- eine Konfiguration kann die Funktionalität und die Qualität der Komponente beeinflussen
- Überprüfung aller Konfigurationen ist nicht immer möglich



Bewertungsmatrix

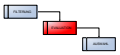
Erfüllbarkeit eines Kriteriums Skala von 1 bis 9

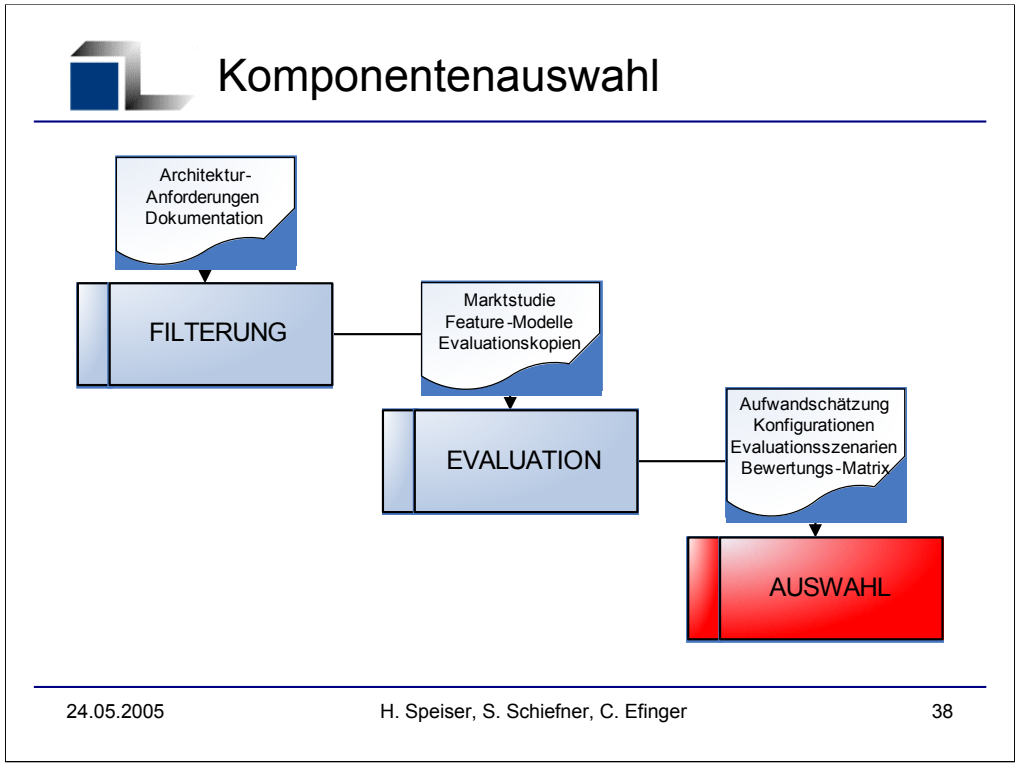
Gewichtung eines Kriteriums Skala von 1 bis 5

Kriterium	Stabilität	Benutzer-freundlichkeit	Variabilität	Punkte
Komponente A	5	7	8	69
Komponente B	9	3	5	58
Komponente C	4	8	9	73
Gewichtung	3	2	5	

z.B. Komponente A 5x3 + 7x2 + 8x5 = 69 Punkte

Ein besserer Ansatz mit **Analytic Hierarchy Process (AHP)** von Thomas Saaty

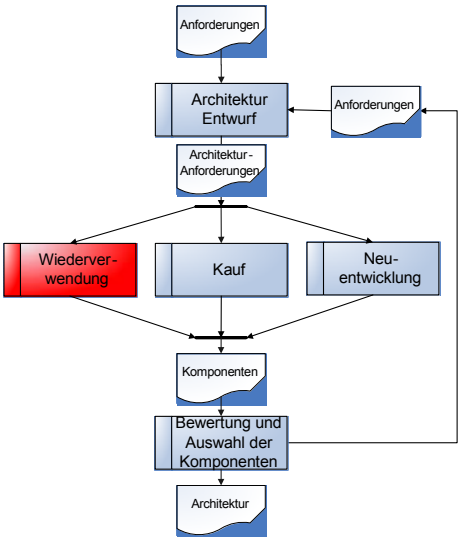




Im dritten Schritt „Auswahl" erfolgt die letztendliche Komponentenauswahl basierend auf den Ergebnissen der „Evaluation“

Komponenten – Wiederverwendung

Komponenten – Wiederverwendung



Wiederverwendung – Motivation

- In der Realität sind häufig schon Systeme vorhanden
  - Varianten vorhandener Software erstellen
  - Vorhandene Softwareprodukte zu einer Produktlinie zusammenfassen
  - Bestehende Produktlinie zu einer neuen weiter-entwickeln

Vor- und Nachteile der Wiederverwendung

+ Im Allgemeinen schon hohe Investitionen in Weiterentwicklung und Wartung vorhandener Systeme gesteckt
+ Durch Wiederverwendung von Teilen der Altsysteme können bisherige Investitionen die Kosten für eine Produktlinien-Einführung erheblich senken
- Gefahr, dass durch das Reverse-Engineering hohe Kosten entstehen

Ein Software-Projekt basiert in den wenigsten Fällen auf einer grundlegenden Neuentwicklung:

-Varianten von vorhandener Software erstellen

-Vorhandene Softwareprodukte zu einer Produktlinie zusammenfassen

-Bestehende Produktlinie zu einer neuen weiterentwickeln

Architektur-Entwurf bei vorhandenem System - Arten

- revolutionäre Einführung
  - Weiterentwicklung/Wartung der Altsysteme stoppen
  - Produktlinie neu entwickeln
- evolutionäre Einführung
  - Altsoftware Schritt für Schritt in Produktlinie überführen

-Wartung der Altsysteme kann in der Praxis nicht einfach gestoppt werden, daher wird in der Regel die Wartung der Altsysteme weiterbetrieben, bis die Produktlinie soweit entwickelt ist, dass man die Wartung der Altsysteme einstellen kann, und das Altsystem durch ein Produkt der Produktlinie ersetzen kann.

-Bei der Schritt für Schritt-Einführung wird die Altsoftware noch weiter entwickelt und parallel die Produktlinie eingeführt, das hat den Vorteil, dass die Einführung der Produktlinie jederzeit gestoppt werden kann. Entweder mit dem Ziel, sie zu einem gewissen Zeitpunkt zu stoppen oder die Produktlinie aus Kostengründen nach einer gewissen Zeit einstellen.

Reverse Engineering

Hauptziele im Produktlinien-Kontext:

- Rückdokumentation der Architektur existierender Einzelsysteme
- Verbessern des Verständnisses
- Wiederbeschaffung verlorengegangener Informationen
- Ermöglichen von Wiederverwendung
- Lokalisieren von einzelnen Features im Quellcode

Dokumentationen der Architektur sind meist veraltet oder gar nicht verfügbar. Während langer Wartungsprozesse und Weiterentwicklungen wurde oft die ursprüngliche Architektur nicht berücksichtigt. Daher bestehen folgende Ziele des Reverse Engineering:

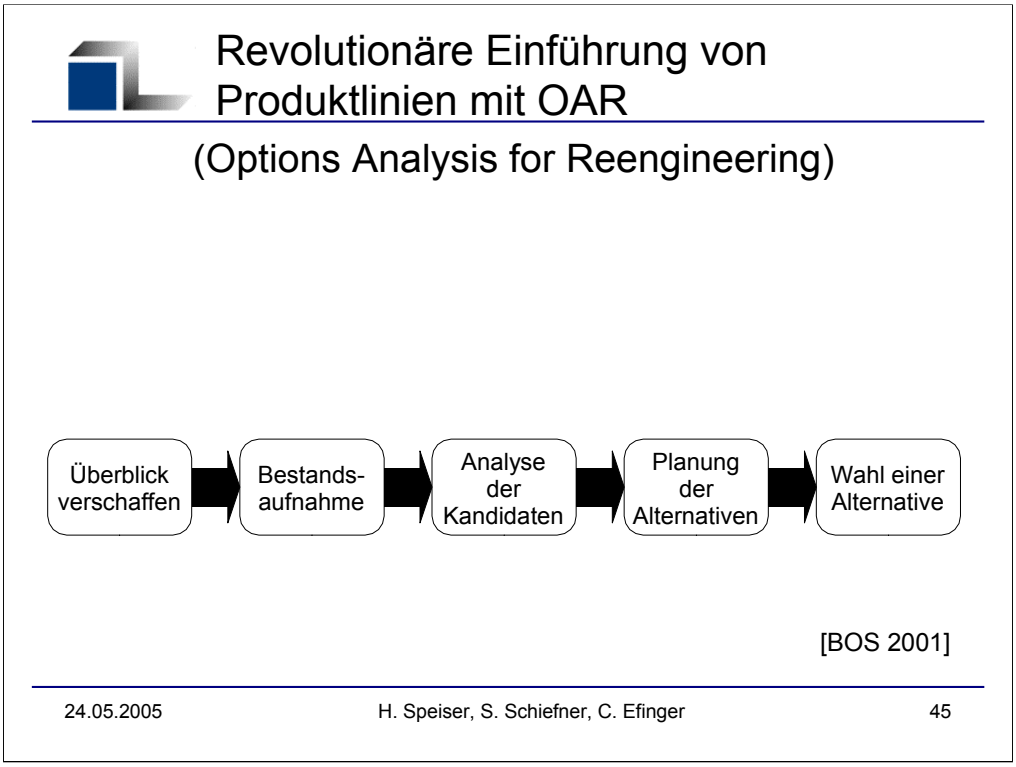
-Rückdokumentation der Architektur existierender Einzelsysteme: Die Dokumentation muss auf den aktuellen Stand gebracht werden.

-Verbessern des Verständnisses: Die vorhandenen Systeme besser verstehen, um sie besser bewerten zu können.

-Wiederbeschaffung verlorengegangener Informationen: Notwendig für die Wiederverwendung

-Ermöglichen von Wiederverwendung: Komponenten (oder ganze Teilsysteme) in die Produktlinie integrieren

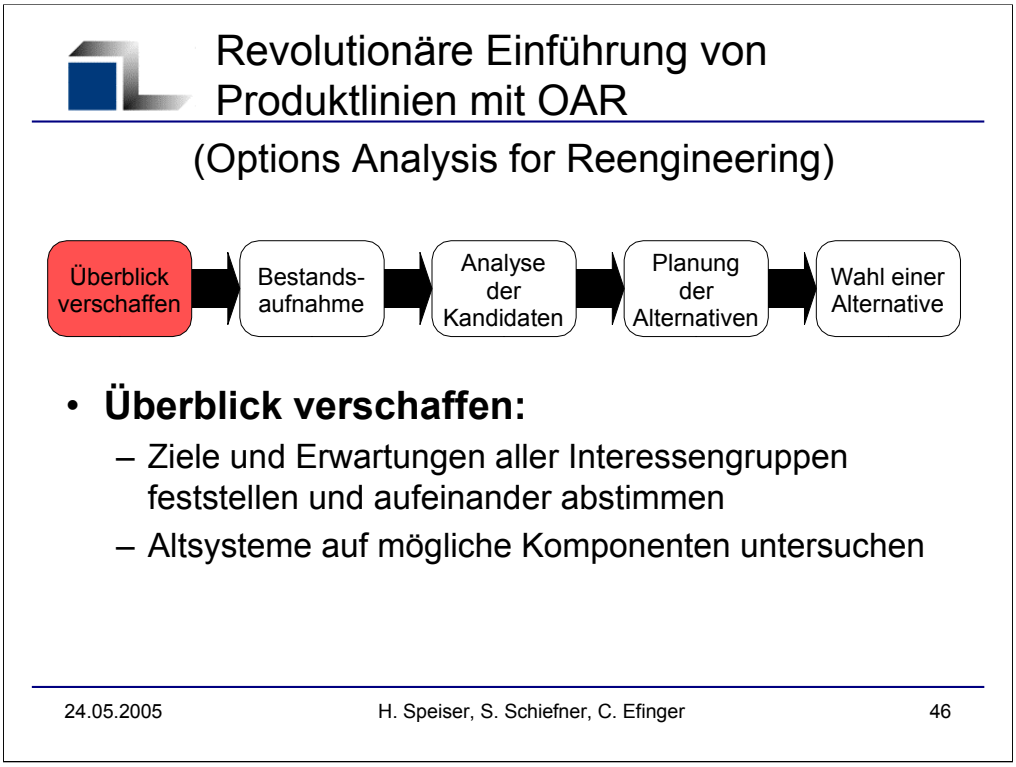
-Lokalisieren von einzelnen Features im Quellcode: Die Funktionalität kann dann in die Produktlinie übernommen werden.



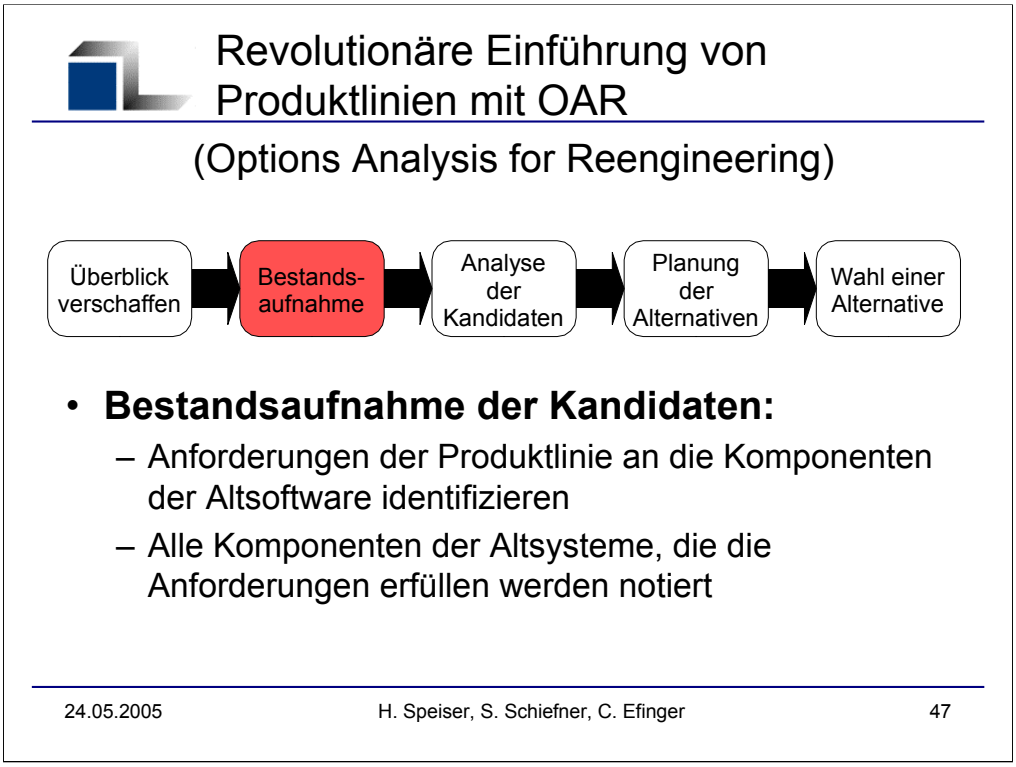
-Mit OAR kann entschieden werden ob und welche Komponenten in einer neuen Produktlinie wieder verwendet werden können,

-Neben dem Quellcode werden auch die anderen vorhandenen Dokumente der Altsystem für die Wiederverwendung in der Produktlinie betrachtet.

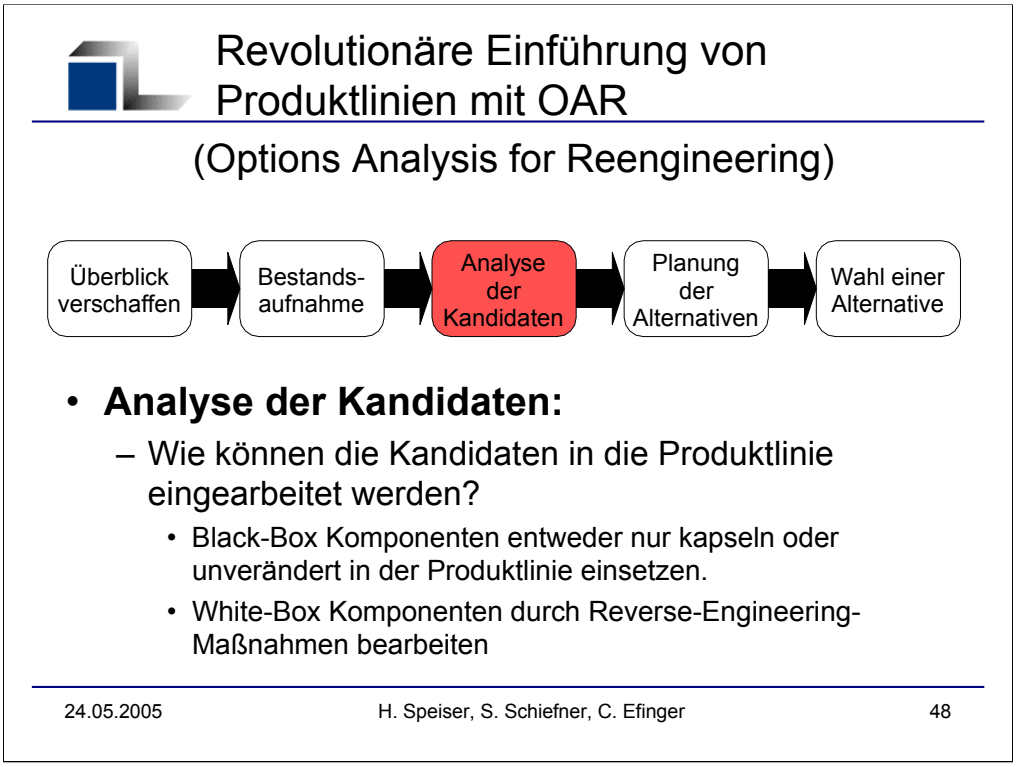
-Bei OAR werden alle möglichen Interessensgruppen z.B. Entwickler, Designer, Manager und deren Interessen berücksichtigt.



-Ziele und Erwartungen der einzelnen Interessensgruppen werden durch Interviews ermittelt und dann aufeinander abgestimmt.



-Resultat dieser Aktivität: Eine Menge von potentiellen Komponentenkandidaten

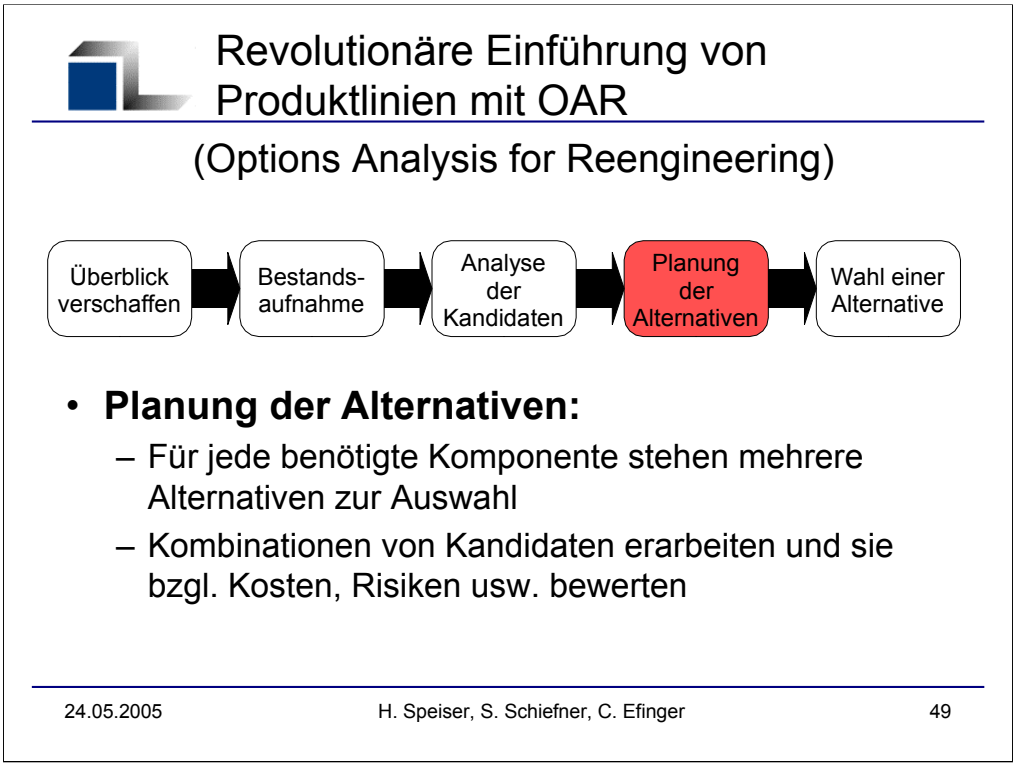


-Die Komponentenkandidaten werden in zwei Kategorien eingeteilt:

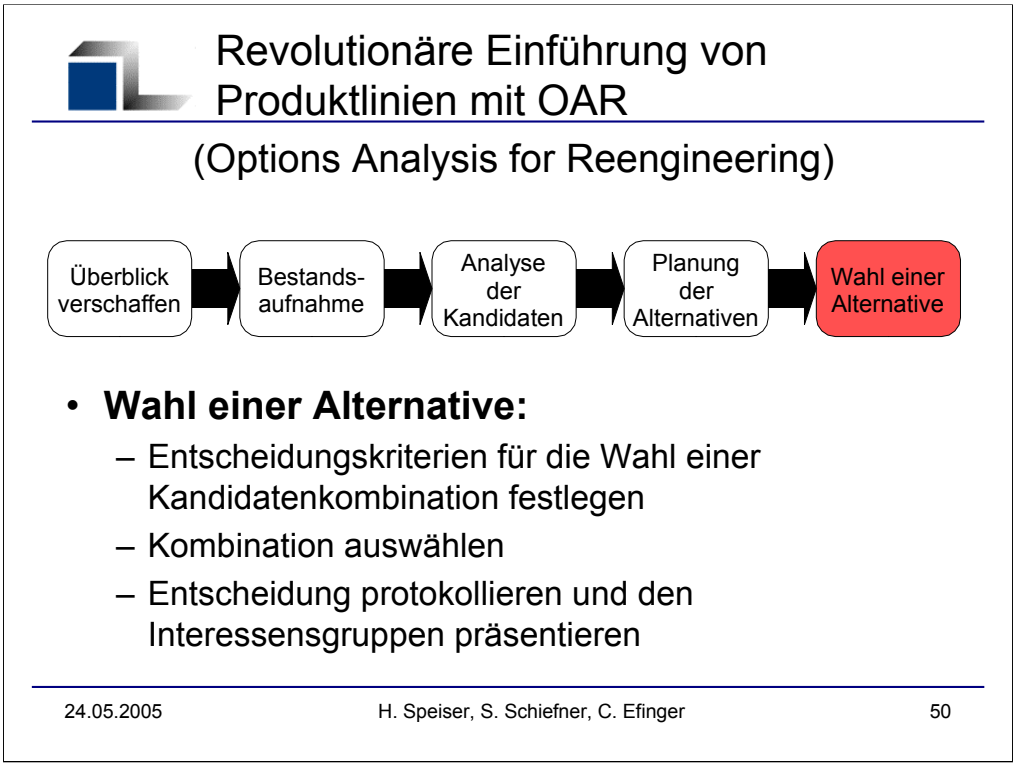
-Black-Box Komponenten: hierzu können auch Testfälle, Teile der Spezifikation, Teile des Entwurfs und Werkzeuge gehören.

-White-Box Komponenten: müssen durch Reverse-Engineering-Maßnahmen bearbeitet werden um in die Produktlinie aufgenommen werden zu können

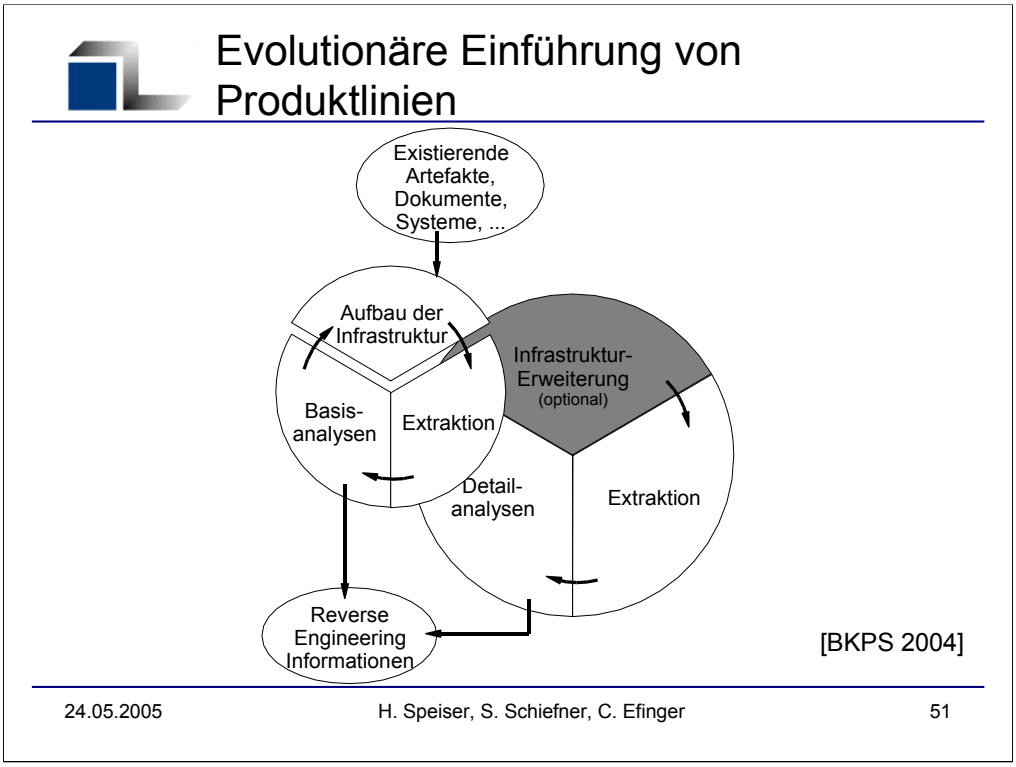
-Zusätzlich werden in diesem Schritt die Kosten, Risiken und Aufwand der Bearbeitung der Komponentenkandidaten abgeschätzt, sowie die Kosten einer Neuentwicklung ermittelt.



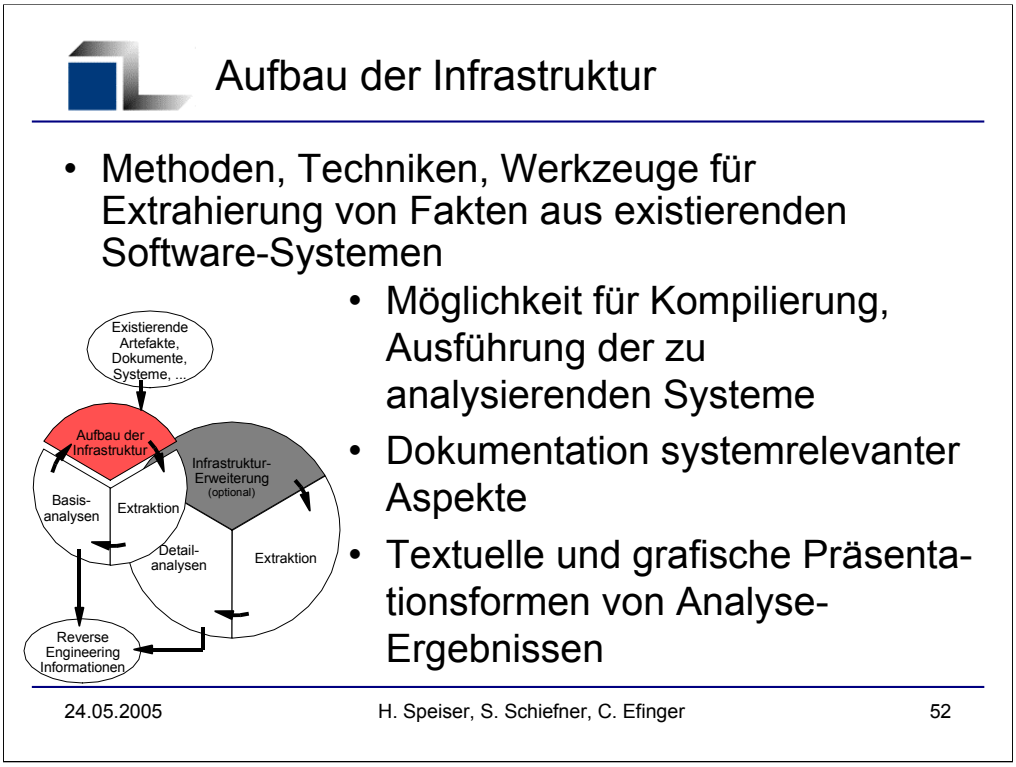
Kombination von Kandidaten erarbeiten: Hier wird davon ausgegangen, dass mehrere Altsystem vorliegen, und man aus diesen Systemen neue Komponenten kombinieren kann.



Zusätzlich werden in jedem Schritt die Entscheidungen überprüft und gegebenenfalls die Zeitplanung der Produktlinien-Entwicklung aktualisiert.

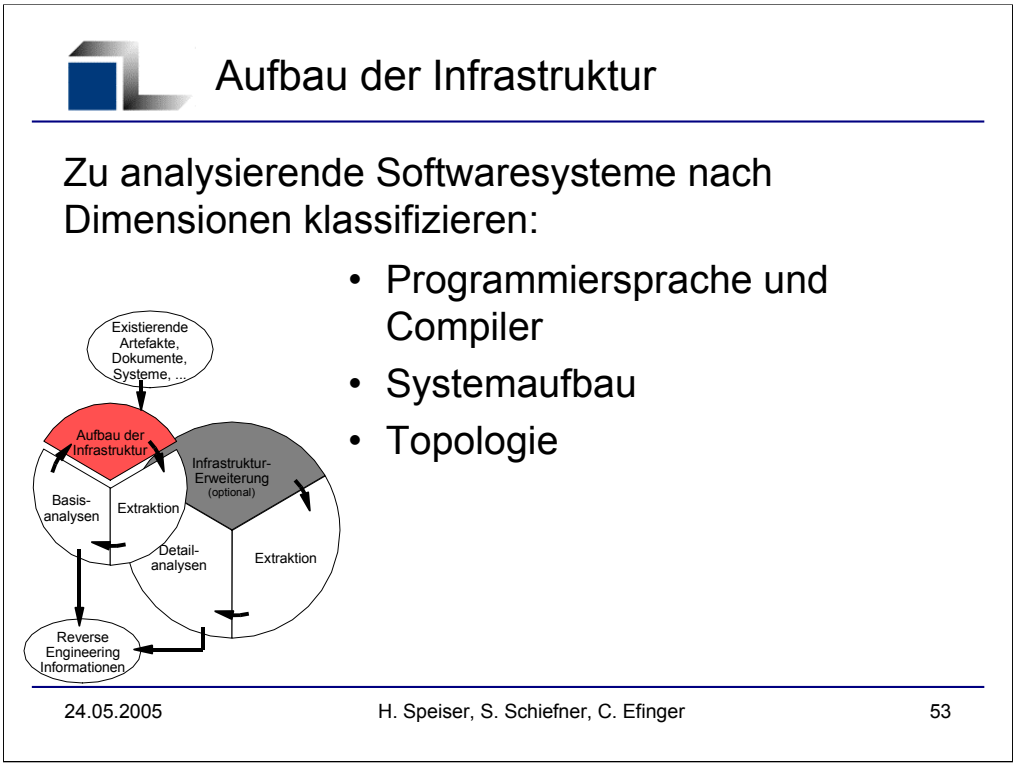


Wichtigster Unterschied zum revolutionären Ansatz (und zu OAR): Es ist eine Wiederholung des Reverse-Engineering-Prozesses vorgesehen. Wichtigste Eigenschaft, die alle evolutionären Ansätze gemeinsam haben.



Infrastruktur = Infrastruktur für die Grundsätzliche Anwendung von Reverse-Engineering-Techniken.

In dieser Phase wird zusätzlich eine Charakterisierung des vorhandenen Systems vorgenommen.

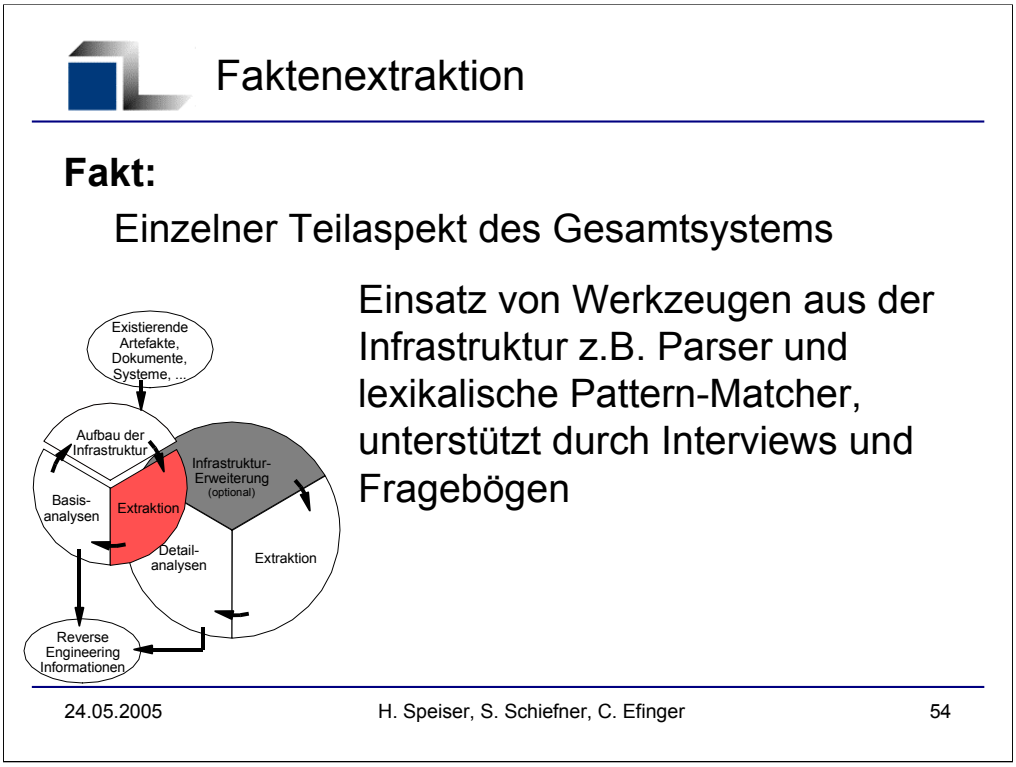


Um Infrastruktur aufbauen zu können müssen die Systeme zunächst nach folgenden Dimensionen klassifiziert werden:

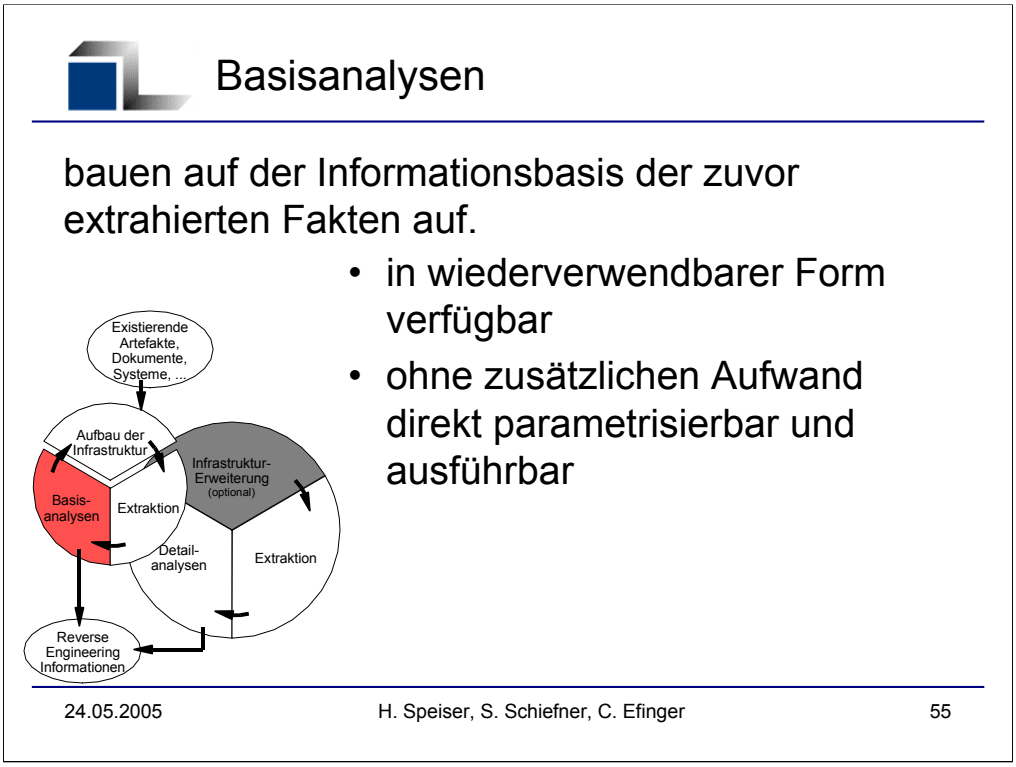
-Programmiersprache und Compiler: Die in den Systemen verwendete Programmiersprache hat durch ihre Eigenschaften (prozedural, objektorientiert, funktional) Einfluss auf die Erstellung der Infrastruktur und das Vorgehen im Reverse-Engeineering, so können z.B. Vererbungen nur bei objektorientierten Sprachen analysiert werden. Bei Software-Systemen, die in mehreren Programmiersprachen implementiert wurden muss jede einzelne von diesen berücksichtigt werden.

-Systemaufbau: Der Systemaufbau weist auf Informationen hin, die zusätzlich zu berücksichtigen sind (z.B. mehrere Produktvarianten durch bedingte Kompilierung oder Konfigurationsdateien, die zu berücksichtigen sind).

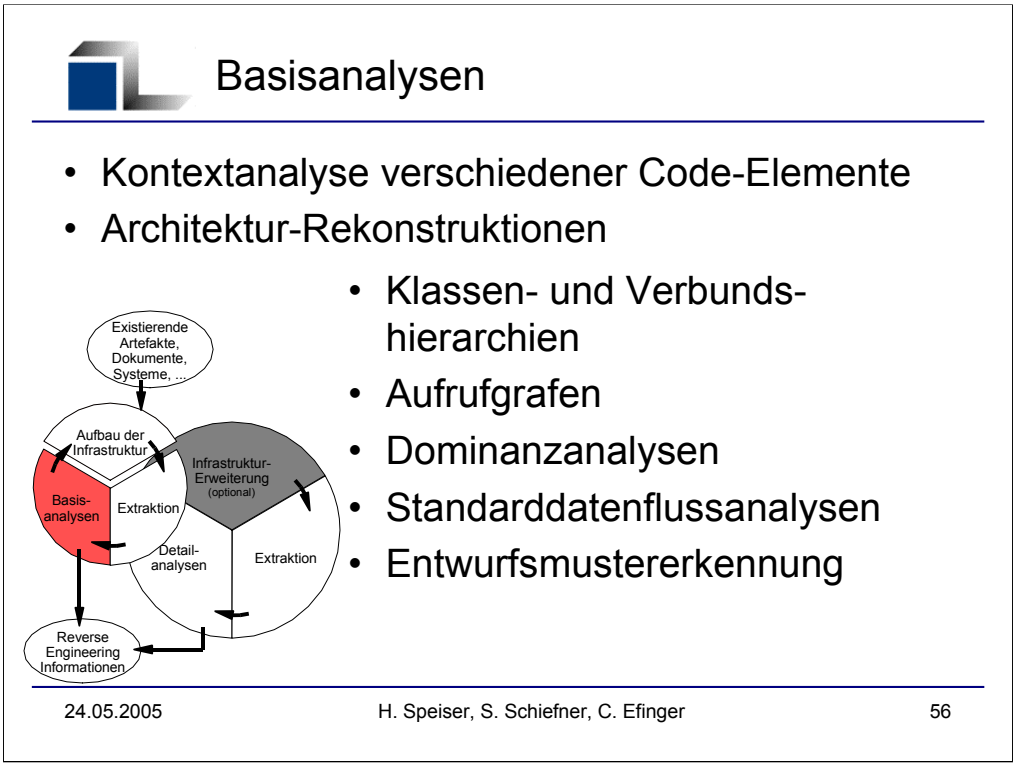
-Topologie: Ein System kann auf verschiedene Arten in die Organisationsstruktur der Kunden, oder in die Verzeichnisstruktur bei den Kunden eingebunden sein. Daraus können sich komplexe Verzeichnisstrukturen, in denen viele Codefragmente existieren ergeben, oder eine flache Verzeichnisstruktur, bei der der Quellcode in einem Verzeichnis (oder sogar nur in einer Datei) liegt.



Beispiel für einen Fakt: Eie Klasse besitzt die Methode mit dem Namen xyz.



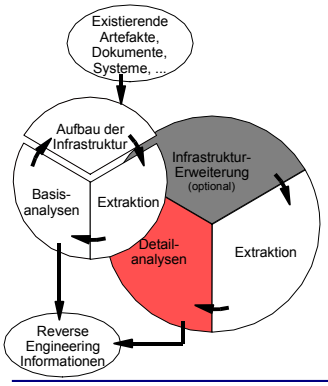
Dadurch, dass Basisanalysen ohne zusätzlichen Aufwand direkt parametrisierbar und ausführbar sind, können sie mit geringen Kosten ausgeführt werden.



Näheres vgl. Vorlesung ASE.

Detailanalysen

- gehen abhängig von den individuellen Anforderungen tiefer → aufwendiger / teurer
- erfordern zusätzlichen Extraktionsaufwand und
  - Verfügbarkeit von Experten
  - Klassifizierung nach:
    - Informationsart
    - Verfügbare Ressourcen
    - Analyseart
    - Technische Domänen



Detailanalysen anhand der folgenden Kriterien klassifizieren:

-Informationsart: Die Art der gewünschten Information ist grundlegend für die zu verwendende Analysemethode:

-Kontext einzelner Code-Elemente

-Einzelne Komponenten (oder Subsystem) identifizieren

-Diese Komponenten extrahieren und deren Wiederverwendung vorbereiten

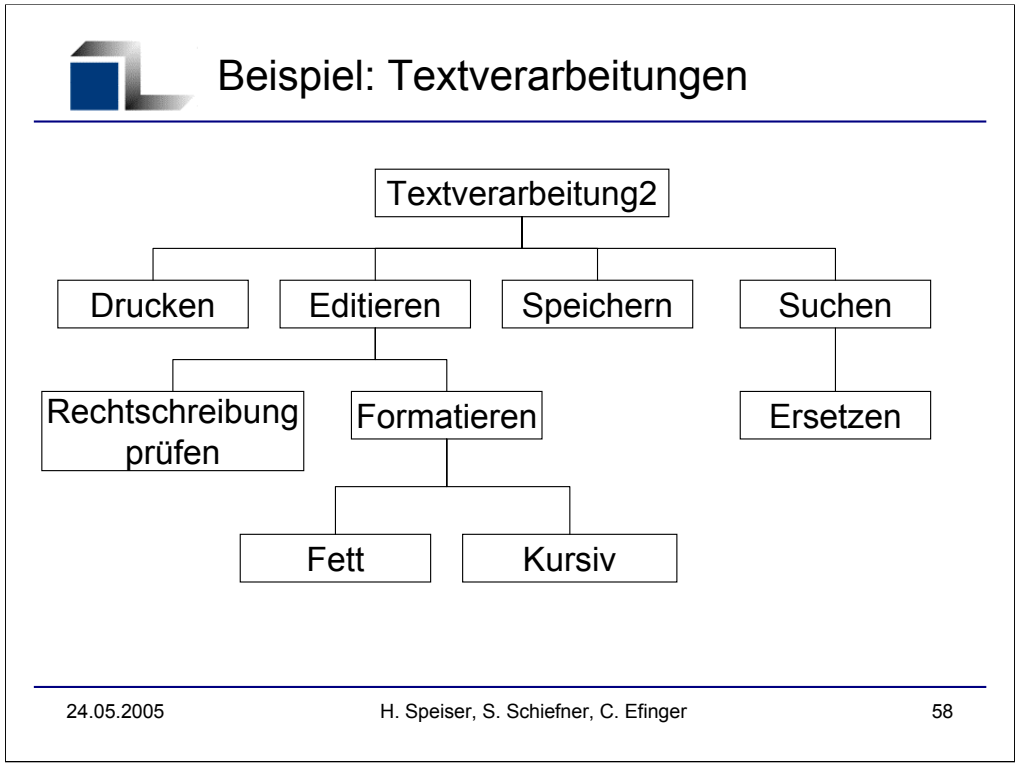
-Zugrunde liegende Architekturstile und Entwurfsmuster in bestehenden Systemarchitekturen identifizieren

-Gründe für Architekturentscheidungen ermitteln

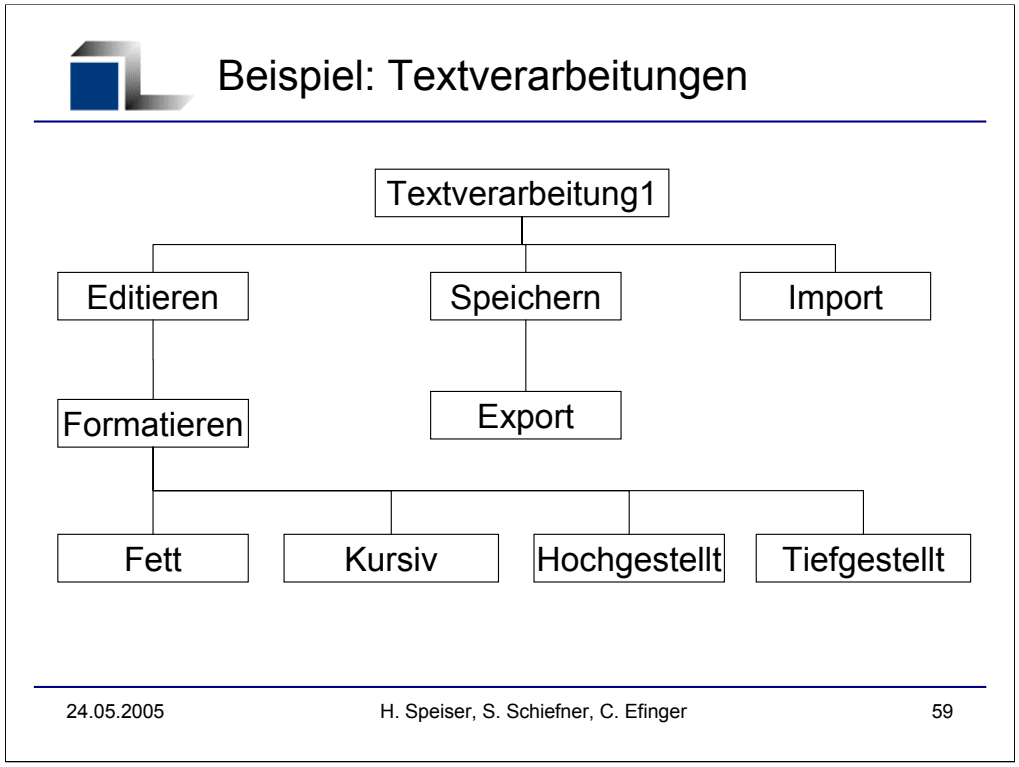
-Verfügbare Ressourcen: Auf welche Dokumente kann für die Analyse zurückgegriffen werden? Neben Quellcode können auch existierende häufig inkonsistente Dokumentationen hilfreich sein. Hierzu zählen auch Experten (sie sind für eine Detailanalyse unverzichtbar), die durch Fragebögen und Interviews eingebunden werden.

-Analyseart: Es gibt statische und dynamische Analysearten. Bei statischen Analysearten werden nur vorhandene Artefakte des Systems isoliert betrachtet. Bei dynamischen Analysearten werden Informationen, durch vorher definierte Ausführungsszenarien während der Laufzeit eines Systems, gewonnen.

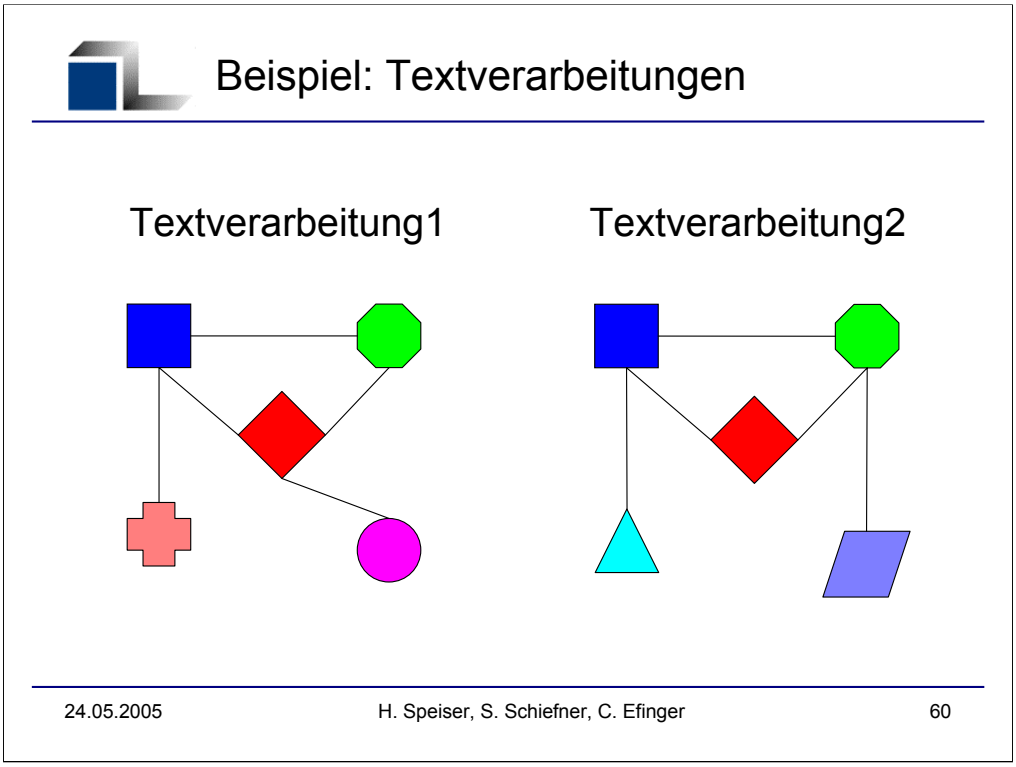
-Technische Domänen: Die technischen Domänen geben Rahmenbedingungen für Detailanalyse vor (z.B. Echtzeitsysteme, Datenbanksysteme, eingebettete Systeme). → Für Detailanalysen bei eingebetteten Systemen z.B. Speicherverbrauch und Laufzeitverhalten berücksichtigen.



In unserem Beispiel haben wir z.B. zwei Textverarbeitungen, von denen die eine den hier aufgezeigten Feature-Baum hat.

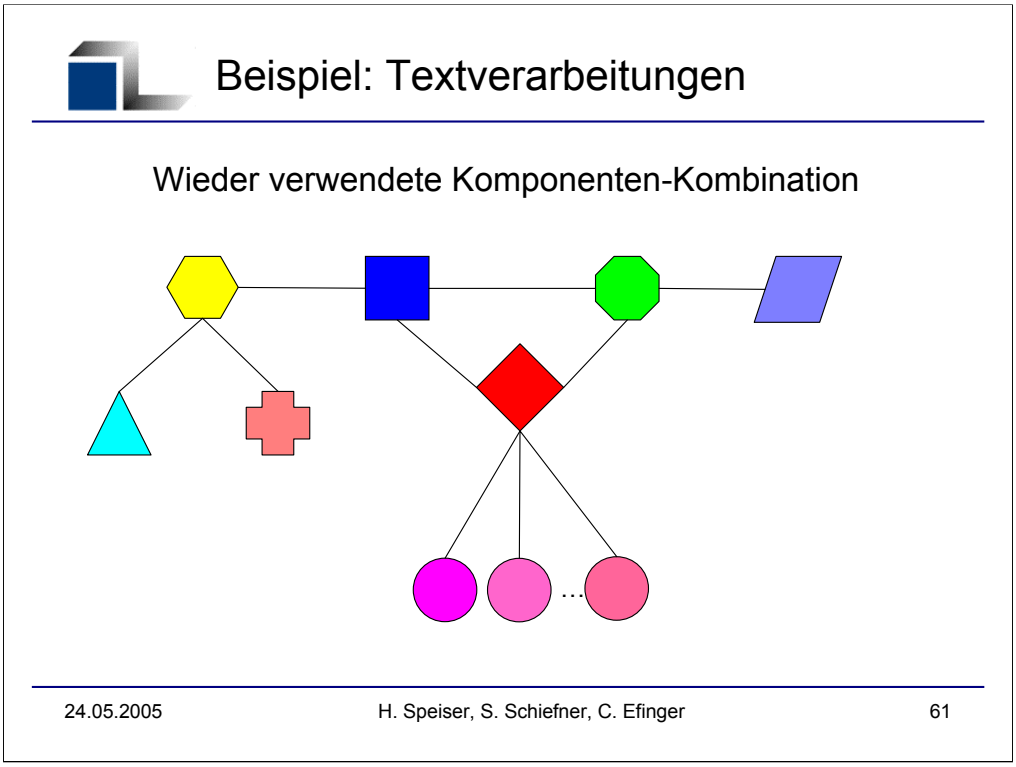


In unserem Beispiel haben wir z.B. zwei Textverarbeitungen, von denen die eine den hier aufgezeigten Feature-Baum hat.



Für dieses Beispiel werden die Präsentationsformen für die Reverse-Engineering-Ergebnisse außer acht gelassen.

In den vorhandenen Systemen sind dabei die hier gezeigten Komponenten, Module (was auch immer) (im Folgenden nur wird nur noch von Komponenten gesprochen) vorhanden und haben die dargestellten Beziehungen unter einander. Hier ist schon zu erkennen, dass drei Kernkomponenten vorhanden sind, die auf die gleiche weise miteinander Arbeiten und die die gleiche Funktionalität aufweisen (gleiche Form und Farbe = gleiche Funktionalität). Da die zwei Textverarbeitungen verschiedene Features besitzen, haben sie auch noch voneinander verschiedene Komponenten.



So könnten die Komponenten nach den Analysen im Reverse-Engineering ohne großen Aufwand zusammengefügt werden, es müsste nur noch eine Komponente ‚neu' dazukommen.

Bei einer revolutionären Einführung (wie OAR) steht dieses Ergebnis nach einmaligem Durchlauf der Phasen fest, während man bei evolutionären Einführungen zunächst evtl. sich auf die drei Kernkomponenten konzentriert. Bei einer Erweiterung der Architektur der Produktlinie wird dann der Reverse-Engineering-Prozess erneut durchlaufen, und es werden nacheinander die anderen Komponenten ‚gefunden' und in die schon vorhandene Architektur integriert.

Fazit

- Ein neues Thema mit wenig Basiswissen
- Viele theoretische und wenig praktische Ansätze in der Literatur
- Das Thema erforderte viel Einarbeitungszeit
- Schwergewichtiges Thema im PL-Kontext
- Viele Parallelitäten zu anderen Vorlesungen (SWP,SWA,KPT,…)
- Aktuelles Thema
- Grenzt sich von der „normalen“ SWE durch höheren Komplexitätsgrad ab

Literatur

- [BKPS 2004] G.Böckle, P.Knauber,K.Pohl,K-Schmidt, Software Produktlinien, 2004
- [KA 2004] Klaus Alfert,Vom Projekt zum Produkt, 2004
- [GBS 2001] Jilles van Gurp,Jan Bosch,Mikael Svahnberg, On the notion of Variability in Software Product Lines 2001
- [BOS 2001] J. Bergey, L. O'Brien D. Smith, Options Analysis for Reengineering (OAR): A Method for Mining Legacy Assets, Technical Report CMU/SEI-2001-TN-013, Juni 2001

Fragen ?