Fraunhofer Institut
Experimentelles
Software Engineering

Gezielte Wiederverwendung durch Software-Produktfamilien
Vorträge, Diskussionen, Erfahrungsaustausch

Editors:
Peter Knauber
Klaus Pohl

Die Veranstaltung wurde gefördert im Rahmen des internationalen ESAPS-Projektes, das als Projekt 99005 im Eureka Σ! 2023 Programm innerhalb der ITEA Initiative durchgeführt wird

IESE-Bericht Nr. 090.00/D
Version 1.1
Dezember 2000

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer Gesellschaft.
Das Institut überträgt innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach
Sauerwiesen 6
67661 Kaiserslautern

Inhaltsverzeichnis

Kurzübersicht der Präsentationen

1. Planung und Realisierung von Produktfamilien mit PuLSE

Referent:

Dr. Peter Knauber, Fraunhofer Institut für Experimentelles Software Engineering (IESE)

Abstract:

Software-Produktfamilien (oder Produktlinien) werden allgemein als vielversprechender Ansatz gesehen, um bei der Software-Entwicklung von Wiederverwendung zu profitieren. Das Konzept ist aus traditionellen Industriebereichen, wie z.B. der Automobilindustrie oder der Unterhaltungselektronik, abgeleitet, wo Produktfamilien-Entwicklung seit langem sehr erfolgreich eingesetzt wird.

Bei der Produktfamilien-Entwicklung wird ein zweiphasiger Entwicklungszyklus angewendet, indem zunächst *für* und dann *mit* Wiederverwendung produziert wird. In der ersten Phase werden zukünftige Mitglieder der Produktfamilie geplant, darauf basierend die gemeinsame Referenzarchitektur für die Produktfamilie entworfen und entsprechende Standardkomponenten entwickelt. In der zweiten Phase werden die verschiedenen Produkte aus den Standardkomponenten zusammengesetzt und anschließend an individuelle Kundenwünsche angepaßt. Durch den hohen Anteil an wiederverwendeten Komponenten können Vorteile wie geringere Entwicklungs- und Wartungskosten, kürzere time-to-market und höhere Qualität der Produkte erreicht werden.

PuLSE (für Product Line Software Engineering) wurde am Fraunhofer IESE als Entwicklungsmethode für Produktfamilien entworfen. Der Ansatz zeichnet sich durch Modularität, zielgerichtete Planung und Unterstützung des kompletten Software-Lebenszyklus aus.

2. Bedeutung der Familienplanung für Time-to-Market von Börsensoftware

Referent:

Dr. Martin Verlage, Market Maker Software AG

Abstract:

Der Markt für Internet-basierte Börseninformationssysteme ist turbulent. Durchdachte Produktentwicklungen sind notwendig, um trotz heterogener und sich ändernder Anforderungen schnell Systeme einsatzfähig zu bekommen. Der Vortrag berichtet von der Anwendung des PuLSE-Ansatzes bei der Erstellung der Systemfamilie MERGER, die als Web-Plattform, Arbeitsplatzlösung und Spezialdienste zum Einsatz kommt.

3. Anforderungsmanagement und Produktfamilien

Referent:
Mark Strembeck, AG Software Systems Engineering, Universität Essen

Abstract:
Der Einsatz von Produktfamilien in der Softwareentwicklung führt zu einer Kostenreduktion bei der Entwicklung von kundenspezifischer Anwendungssoftware, stellt jedoch gleichzeitig neue Herhausforderungen an die Erhebung und das Management von Anforderungen an solche Systeme.
Der Vortrag gibt einen Überblick über diese Herausforderungen, skizziert beispielhaft einen Use-Case basierte Lösungsansatz und berichtet über die bisher in der industriellen Erprobung gewonnenen Erfahrungen.

4. Testen und Produktfamilien

Referent:
Dr. Josef Weingärtner, Siemens Health Services GmbH&Co KG

Abstract:
Der Einsatz von Produktfamilien in der Software-Entwicklung führt an vielen Stellen zu neuen Herausforderungen. Bereits in der ´traditionellen´ Software-Entwicklung war das Ableiten geeigneter TestCases zu möglichst frühen Zeitpunkten ein unbedingt zu beachtendes Qualitätsziel.
Der Vortrag beschreibt einen Ansatz beim Übergang zur Software-Entwicklung für Produktfamilien.

5. Ein Change Management Prozeß für ein Reuse Object Repository

Referent:
Dr. Annette Schreiber, Siemens AG

Abstract:
In der Präsentation wird ein Prozess für das Change Management eines Reuse Object-Repositories vorgestellt. Reuse Objects sind Software-Assets, z.B. Software-Module oder Code-Fragmente, und die zu den Software-Assets gehörende Umgebung bzw. Infrastruktur, z.B. die Requirements- und Design-Spezifikationen, Testmodule und Anwendungsbeschreibung des SW-Assets.
Unsere Präsentation beinhaltet die Beschreibung eines Reuse Objects, die erforderlichen Rollen mit ihren Aufgaben für einen Change Management Prozess sowie die Vorstellung der Aktivitäten des Prozesses. Der vorgestellte Prozess wird in einem Siemens-Bereich eingeführt.

6. Eine Fallstudie für den Produktlinien-Ansatz bei Bosch: Fahrzeug-Umfeldsensorik

Referent:
Dr. Stefan Ferber, Robert Bosch GmbH

Abstract:
In diesem Vortrag wird der Produktlinien-Ansatz von Bosch am Beispiel der Fahrzeug-Umfeldsensorik vorgestellt.
Fahrzeug-Umfeldsensoriksysteme umfassen eine Familie von sensorbasierten Funktionen, welche die Umgebung des Fahrzeuges überwachen, um die Sicherheit oder den Fahrkomfort zu erhöhen (z.B. Einparkhilfen mit Ultraschallsensoren, Tote-Winkel-Detektion, Pre-Crash-Detektion mit Radarsensoren). Traditionell werden im Automobilbau diese Systeme unabhängig voneinander entwickelt, produziert und installiert. Dies führt zu einer hohen Anzahl an Sensoren und Steuergeräten, die in Summe viel Energie, Gewicht und Bauraum benötigen. Durch Integration dieser Funktionen in ein System wird diesen Problemen entgegengewirkt.
Der Produktlinien-Ansatz ist ein Systementwicklungsprozeß, der systematisch die Gemeinsamkeiten und Unterschiede von Produkten einer Domäne berücksichtigt. Ziel des Ansatzes ist, möglichst viele Entwicklungsprodukte (z.B. Software- und Hardware-Komponenten, Anforderungsdokumente, Testfälle etc.) in unterschiedlichen Produktvarianten wiederzuverwenden. Eine einzelne Produktvariante wird aus der generischen Produktlinie abgeleitet und nur spezielle variable Funktionen werden zusätzlich implementiert. Dadurch werden geringere Produktentwicklungskosten, höhere Produktqualität und kürzere Entwicklungszyklen erreicht werden.
Diese Fallstudie präsentiert die ersten Ergebnisse aus der Anwendung des Produktlinien-Ansatzes im Bereich der Fahrzeug-Umfeldsensorik.

Building Blocks, ARES, ESAPS and the Future

Dr. Frank von der Linden

Software Product Families, Past, ESAPS & Future

Frank van der Linden
Philips Medical Systems

Overview

- Introduction
- Time-line & Context
- History:
  - BB
  - ARES
- Present:
  - ESAPS
- Future:
  - CAFÉ

Introduction

Personal involvement with product families

Personal Background

- Philips Research
  - Software engineering research
  - since 1984
- Philips Medical Systems
  - Product family development
  - since 1999

Time-line

- 1980's: Building Block development within Philips Telecommunication(PKI Nürnberg)

ARES

PRAISE Software Product-line

1998-2000: Thomson-CSF Robert Bosch, ESI

ESAPS

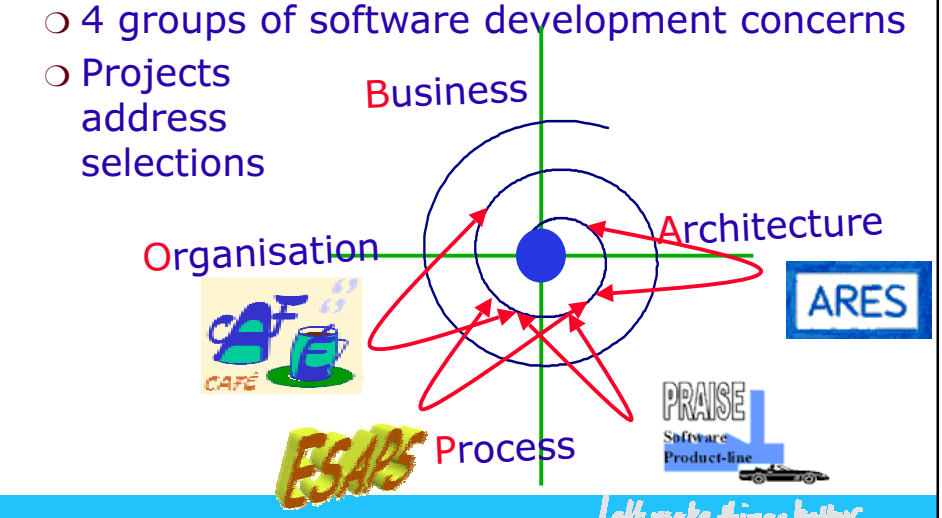
CAFÉ

Own involvement:

- 1990-1994: Consolidation of Building Block development
- 1995-1998: ARES
- 1999-2001: ESAPS
- 2001-2003: CAFÉ

Software development concerns

- 4 groups of software development concerns
- Projects address selections



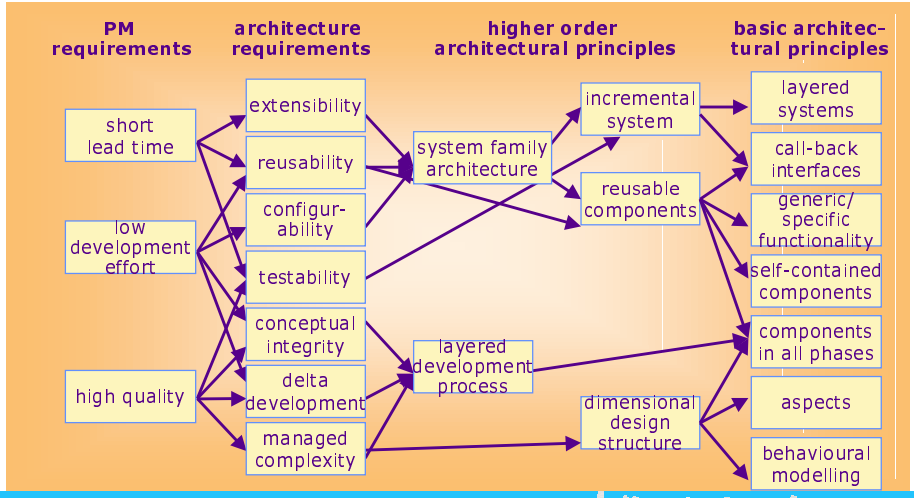
Building Block - Requirements

1980's Telecom switch family development

Basic Requirements:

- Extensibility
- Reusability
- Configurability
- Testability
- Delta development
- Managed complexity

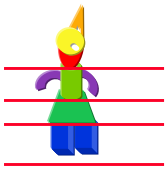
Building Block Requirements trace



Building Blocks

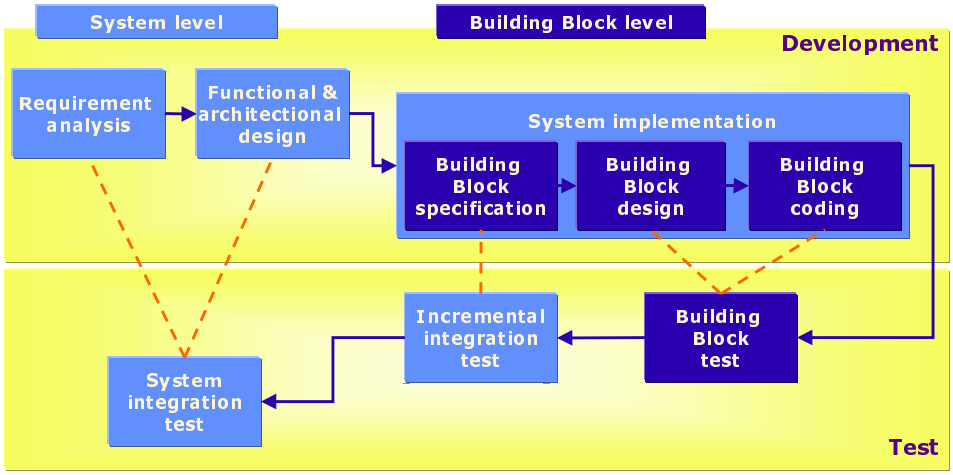
- Building Block:
  - Unit of incremental development, build, testing, …
  - Self contained:
    - Deals itself with all requirements

- Systems
  - Configurations of Building Blocks
  - Linked at System Initialisation time



- Strict layered design
  - support for incremental design, test, build, ...

Building Block Layered Development Process



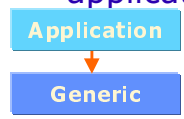
Building Block Aspects

- Aspects
  - High level functional decomposition
  - System wide responsibilities
  - Separate Global design
    - distributed over all building blocks
  - solution for (small) set of quality requirements

- Examples
  - *operational*, logging, initialisation/termination, test management, process management, resource management, …

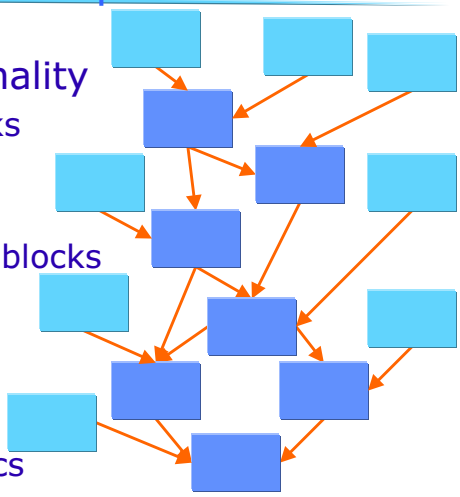
Building Block
Generic & specific

- Separate common from specific functionality
  - Generic building blocks form framework
- Specific plug-ins
  - "application" building blocks



use direction

- Software buses
  - special kind of generics



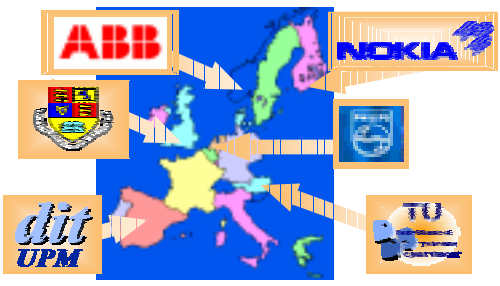
Let's make things better

PHILIPS

ARES

November 1995- November 1998

- ESPRIT project 20477
- ARES
  - Architectural Reasoning for Embedded Systems
- Three industrial and three university partners



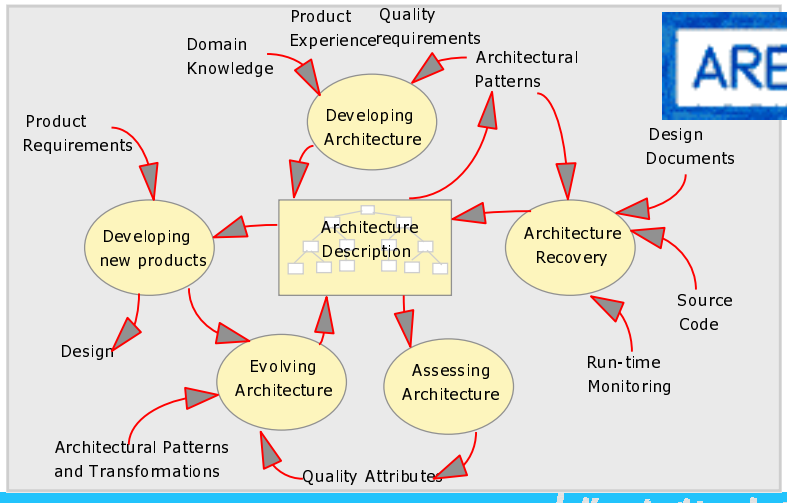
Let's make things better

PHILIPS

ARES motivation

- Develop software variants as a family
- Improve degree of reuse
  - OOD and OOP is not enough
- Predict/control performance
  - distributed real-time system
- Localisation of change
  - Communication overhead & modularity
- Conformance
  - high-level design and implementation

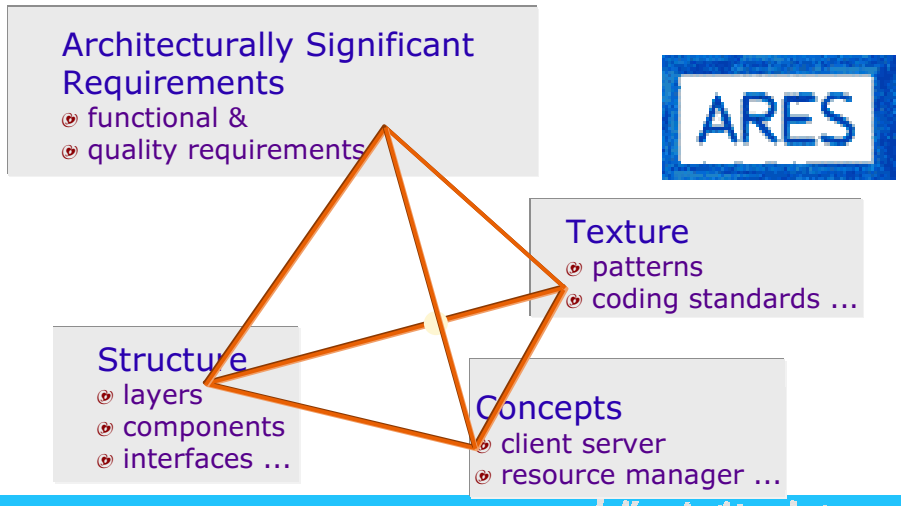
ARES subjects



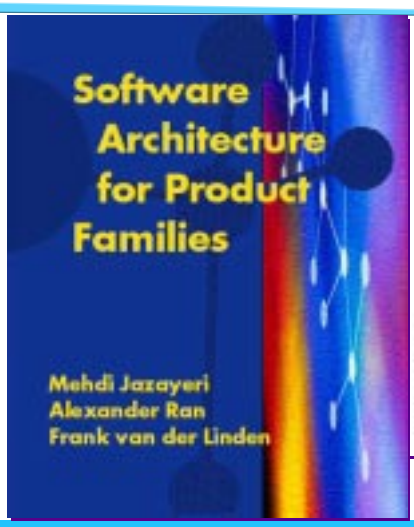
ARES findings

- Investigation in techniques for Product-family architecture
- Research ➪ Industrial practice
  - industry-as-laboratory
- Separate views
  - Kruchten, Soni, Building Block design dimensions, ...
- Description
  - Simple language works best
  - UML too complex
- Recovery in the development life-cycle

ARES model of software architecture



ARES Book



ARES

ADDISON-WESLEY
An Imprint of Addison Wesley Longman, Inc.
Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Let's make things better
PHILIPS

ESAPS

1999-2001

- Engineering Software Architectures, Processes and Platforms for System-Families

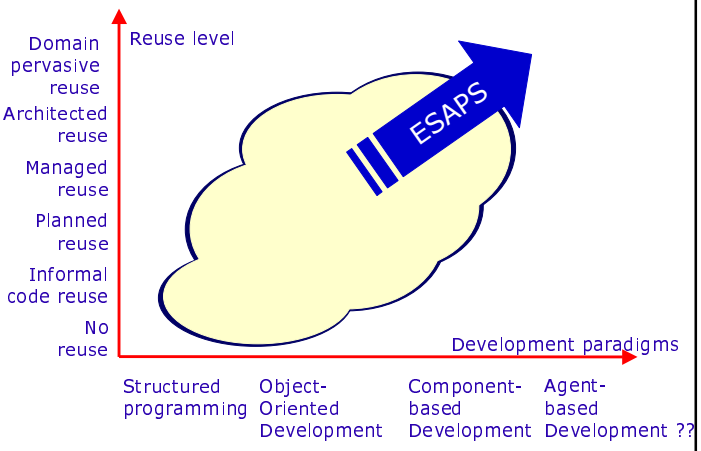
ESAPS is an ITEA-labelled project (99005)
Eureka Σ! 2023 Programme

ITEA

Let's make things better
PHILIPS

ESAPS Goals

Software for
single
systems
≻
system-
families

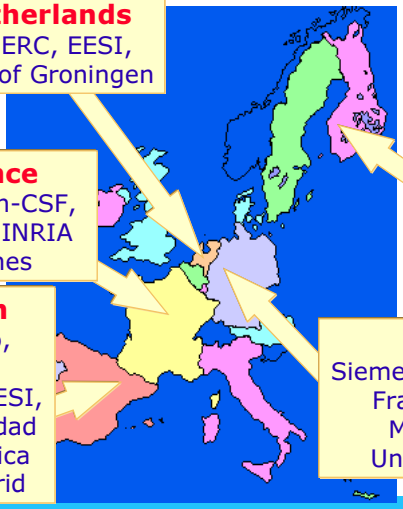


21 ESAPS Partners

the Netherlands
Philips, SERC, EESI, University of Groningen

France
Thomson-CSF, Alcatel, INRIA Rennes

Spain
Sainco, Unión Fenosa, ESI, Universidad Politecnica de Madrid



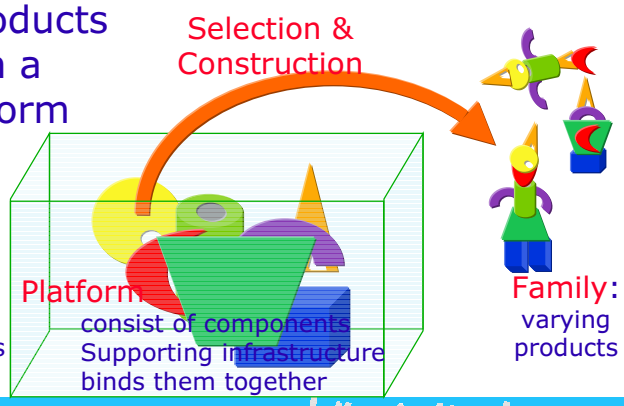
Finland/ Sweden
Nokia, Helsinki University of Technology University Karlskrona/ Ronneby Combitech

Germany
Siemens, Robert Bosch, Fraunhofer IESE, Market Maker, Universität Essen

ESAPS common basis

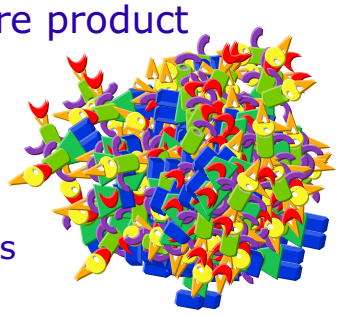
Component based software family architecture

- Varying products based upon a single platform

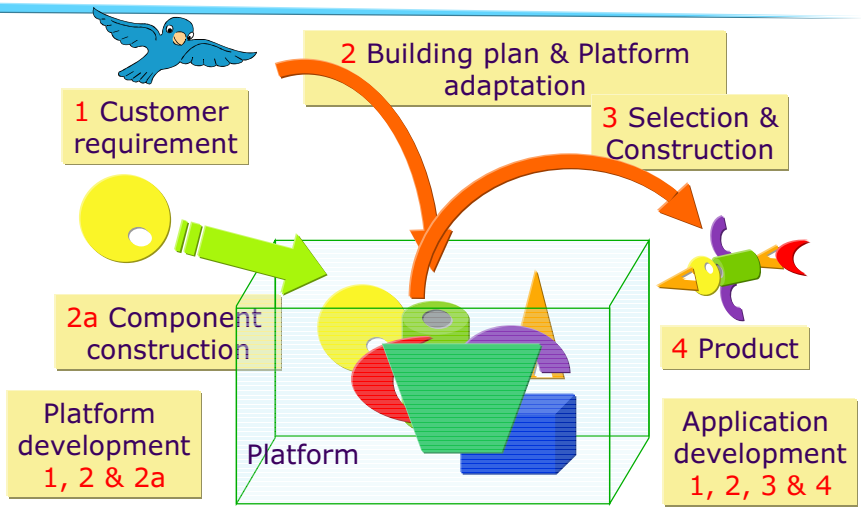


ESAPS results

- Application in real software product development
  - Millions of variants
    - many components & possible combinations
  - Thousands of required variants
    - regional & cultural differences
    - local interchange standards
  - Hundreds of software developers
    - lot of design, coding and support needed

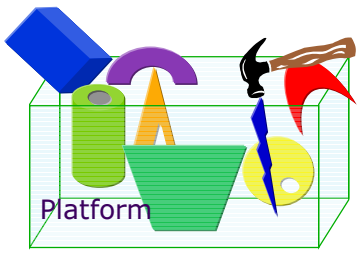


ESAPS development process



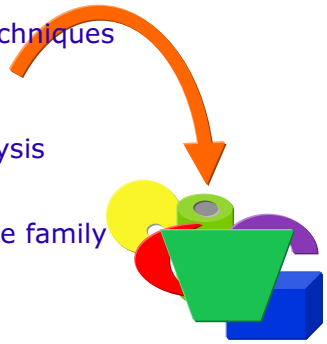
ESAPS technical content – 1.1

- Architectural analysis and modelling
  - is the architecture good enough for its purpose?
  - Assessment
    - Architecture Assessment Techniques
    - Architecture Mismatch Analysis
    - Joint architectural assessment of the EPOC operating system
  - Validation
    - Architecture Metrics
    - Architecture Verification
  - Recovery
    - Reverse Architecting



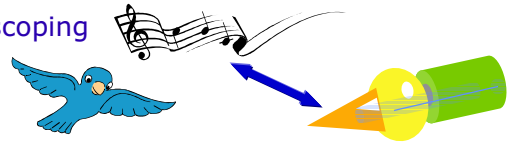
ESAPS technical content – 1.2

- Domain analysis and modelling
  - Important concepts in the domain of the family
  - Domain analysis
    - Overview of Domain Analysis techniques
    - Conceptual Analysis
    - Feature Analysis
    - Unified process for domain analysis
  - Scoping
    - How to decide what is part of the family
    - Economic modelling



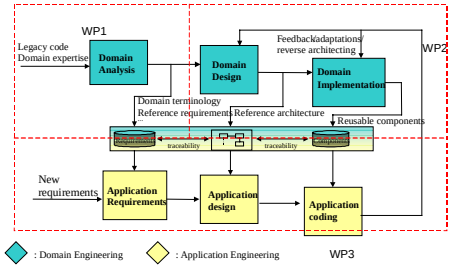
ESAPS technical content – 1.3

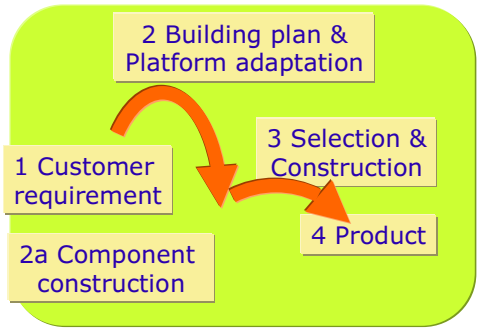
- Aspect analysis and modelling
  - dealing with system qualities
  - Aspects, Views
    - Conceptual model
    - Development of Aspect Models
    - Unified process for aspect engineering
  - Aspect-Based Techniques
    - Evaluate aspects
    - Using aspects for scoping
    - Aspect-guided functional decomposition



ESAPS technical content – 2.1

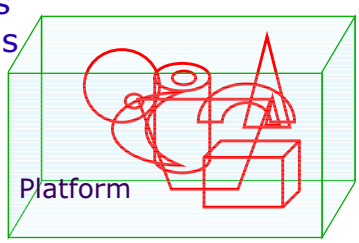
- Definition of system family processes
  - who, what, when
  - System family process framework
    - Template & Comparison





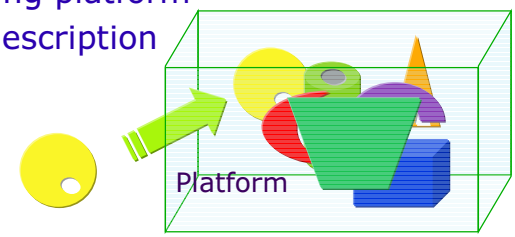
ESAPS technical content – 2.2

- Definition of system family assets: reference architecture
  - common structure & rules
  - System family software architecture glossary
    - Common vocabulary
  - System family requirements classification and formalisms
  - Style, structures and views for handling commonalties and variabilities
    - use of UML
    - variation points



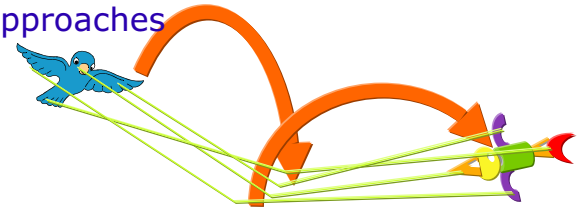
ESAPS technical content – 2.3

- Platform & components
  - build it according to the rules
  - Components: interfaces and realisation techniques
  - Domain engineering platform
  - Target platform description and configuration



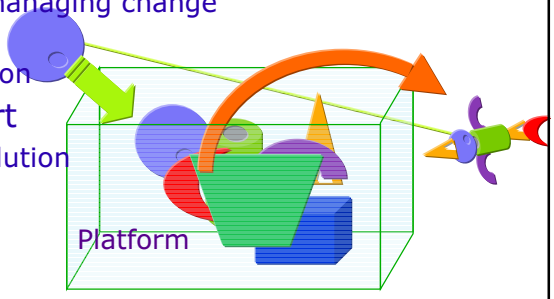
ESAPS technical content – 3.1

- Requirements modelling and traceability
  - links between requirements, intermediate products and products
  - Guidelines for requirements traceability
  - Requirements traceability and evolution
    - Model-based Requirements Engineering
  - Traceability Approaches
    - Tool
    - Methods



ESAPS technical content – 3.2

- Change impact and change impact propagation
  - predict costs and effort for a change
- Change management
  - The Process of managing change requests
  - Change integration
- Evolution support
  - Architecture evolution



ESAPS technical content – 3.3

- System family variant configuration and derivation
  - how to select configuration components
  - configuring the product
- Asset management
  - Asset Characterisation
  - Visualisation, Retrieval, and Analysis
  - Configuration Management
- Derivation of variants
  - Variant Configuration
- Platform and support mechanisms

CAFÉ

- ESAPS follow-up project
  - July 2001- June 2003
- From Concept to Application in System-Family Engineering
  - 290.1 my
  - 39.94 M□
- ESAPS +/−
  - 22 Partners
  - 8 countries

CAFÉ - 22 Partners

Norway
ICT-Norway, SINTEF, DNV Software, EDB 4tel

Sweden
University Karlskrona/ Ronneby
Combitech

the Netherlands
Philips, ~~SERC, EESI~~
University of Groningen

Finland
Nokia, University of Helsinki, ~~Helsinki University of Technology~~

France
Thomson-CSF, Alcatel, Ivorium, ~~INRIA Rennes~~

Germany
Siemens, Robert Bosch, Fraunhofer IESE, Market Maker, Universität Essen

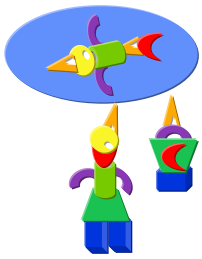
Spain
Sainco, ~~Telvent, Unión Fenosa~~, ESI, Universidad Politecnica de Madrid

Italy
Istituto di Elaborazione della Informazione

Austria
JKU Linz, TU Vienna

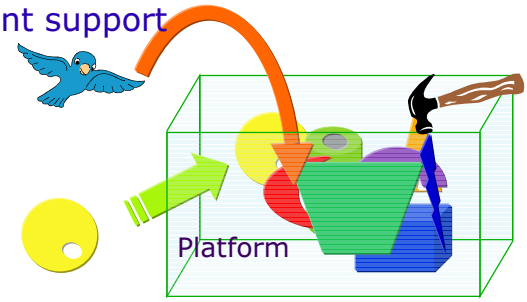
CAFÉ - work package 1

- System family adoption
  - Business and Market analysis
  - Product line scoping
  - Product-line transition & adoption



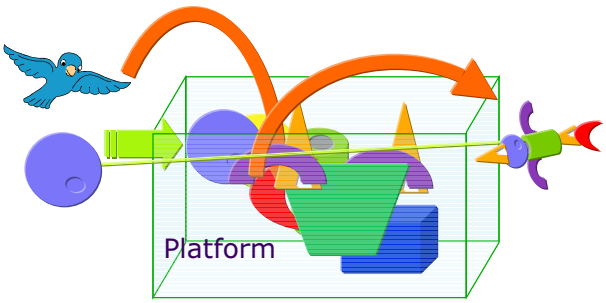
CAFÉ - work package 2

- Asset building
  - Requirements engineering
  - Platforms & components
  - Design for quality
  - Asset development support



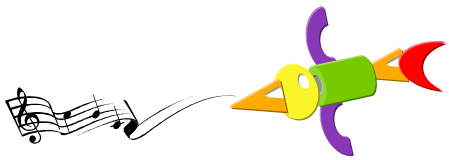
CAFÉ - work package 3

- Asset usage
  - Traceability & change management
  - Configuration & version management
  - Product derivation & Family evolution



CAFÉ - work package 4

- Validation and testing
  - Testing
  - Validation



Summary

- The European way
- Sequence of co-operation projects
  - central theme: Software family development
- Focus shifts:
  - Architecture - Process - Organisation - Business
- Main reasons for co-operation:
  - "don't reinvent the wheel"
  - independence of USA
- Other initiatives:
  - SEI Product-line initiative
    - USA defence industry
  - smaller scale initiatives

The end

Thank you

Planung und Realisierung von Produktfamilien mit PuLSE

Dr. Peter Knauber

Planung und Realisierung von Produktfamilien mit PuLSE™

Kaiserslautern
9. November 2000

Fraunhofer IESE
Institut
Experimentelles
Software Engineering

Dr. Peter Knauber

Sauerwiesen 6
D-67661 Kaiserslautern
Germany

Inhalt

- Motivation
  - Wiederverwendung
  - Domänenspezifische Wiederverwendung
- Die Entwicklung von und mit Produktfamilien
  - Scoping
  - Modellieren
  - Architektur entwickeln
  - Produkte ableiten
  - Weiterentwickeln
- Anwendung in Projekten

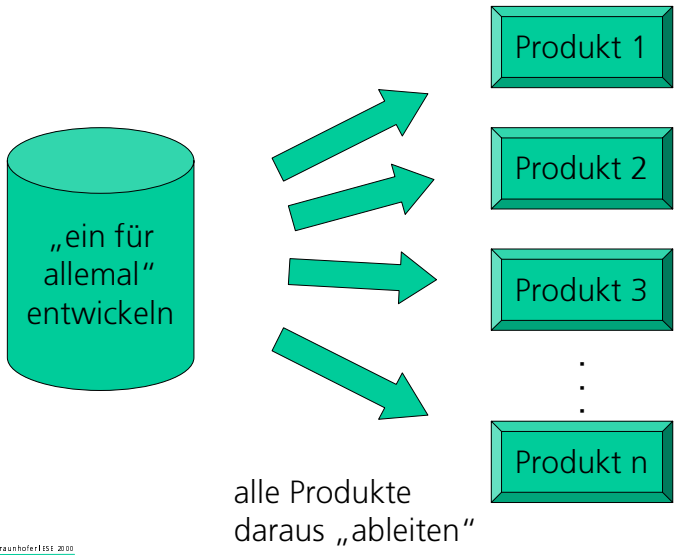
Motivation: Wiederverwendung

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung

- Wiederverwendung von Software ist eine Lösung für viele typische Probleme von Software-Entwicklungsorganisationen
- Erwartete Vorteile:
  - Kürzere time-to-market
  - Verbesserte Qualität
  - Weniger Aufwand für Entwicklung und Wartung
  - Reduktion der Kosten
- Das Konzept ist intuitiv verständlich
  - Analogie: Hardware (Computer, Unterhaltungselektronik, Automobilbau etc.)

Zentrale Produktarchitektur

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



Probleme mit der Wiederverwendung

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung

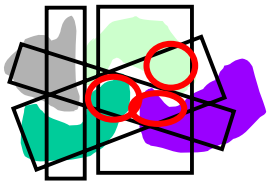
Probleme bei der Wiederverwendung:

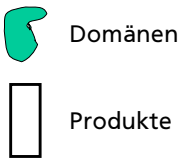
- Projektgrenzen
  - Assets sind nicht über die Projektgrenzen hinaus bekannt
  - Suche nach Assets ist oft nicht systematisch möglich/unterstützt
  - Assets sind nur bedingt projektunabhängig
  - Asset-Anpassung ist teuer und fehleranfällig
- Kein Bewertungsmaßstab für die Effizienz bei der Wiederverwendung

Domänen und Produkte

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung

Lösungskonzept:
- Zielgerichtete Wiederverwendung konzentriert auf bestimmte Domänen
- Entwicklung von / mit Produktfamilien

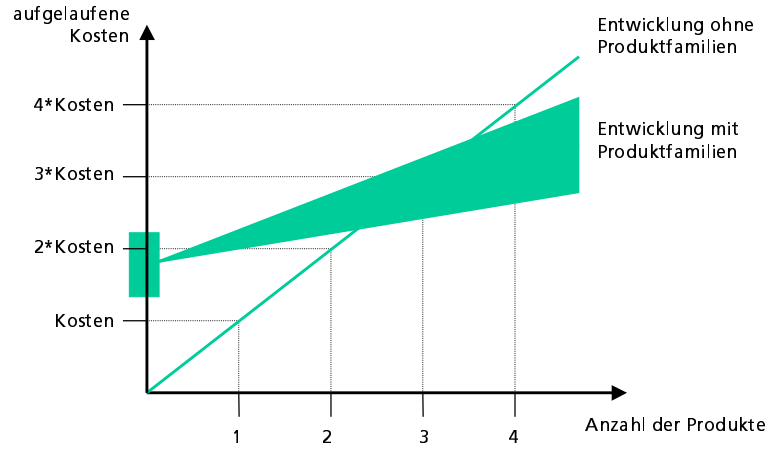




Entwicklung von Produktfamilien am IESE:
PuLSE™ – Product Line Software Engineering

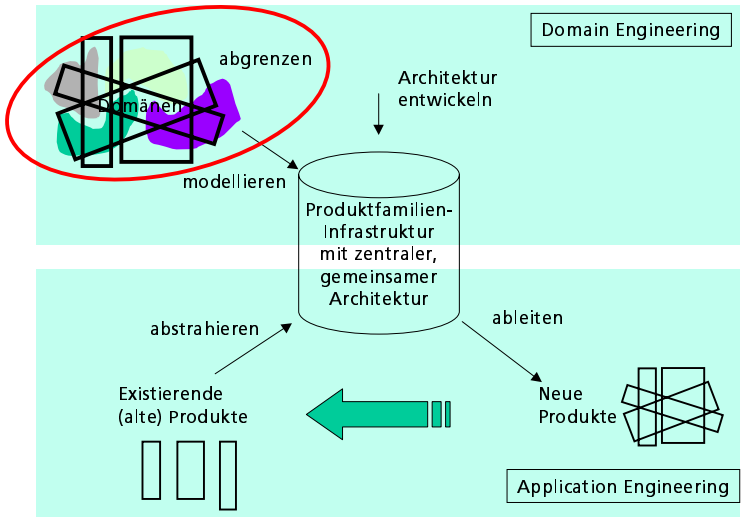
Erwarteter Nutzen

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



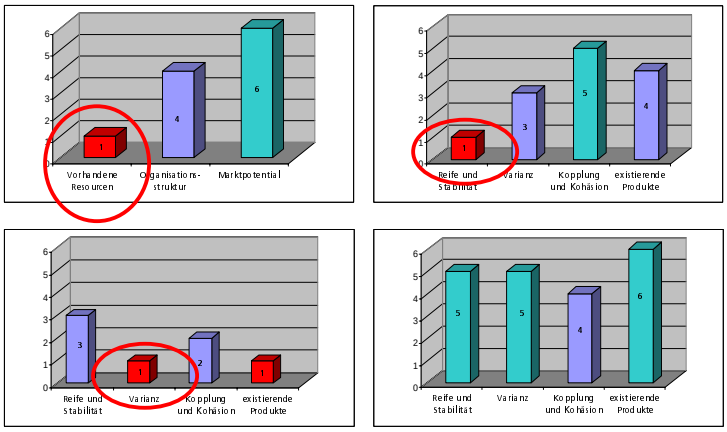
Phasen der Entwicklung (1/5): Scoping

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



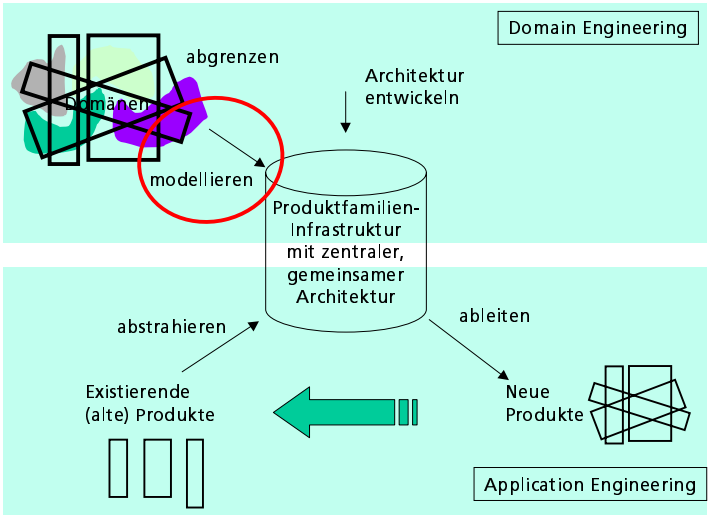
Auswahl geeigneter Domänen

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



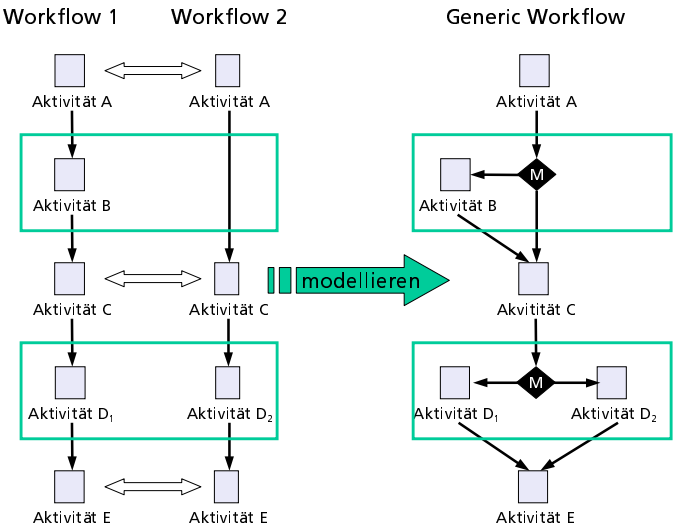
Phasen (2/5): Modellieren

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



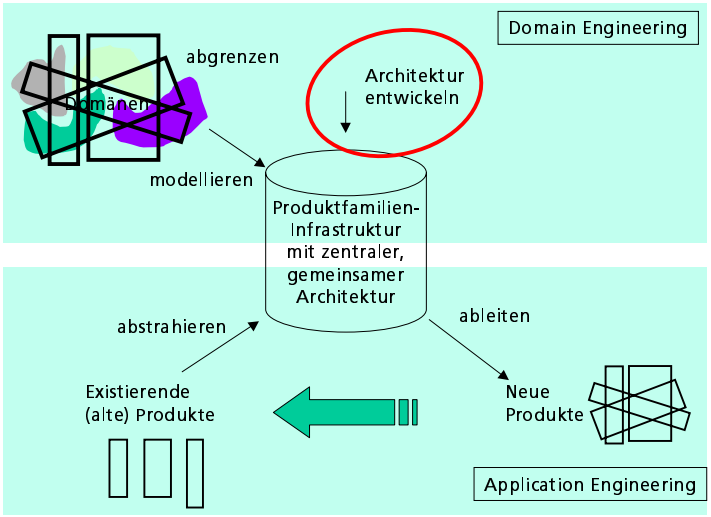
Beispiel

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



Phasen (3/5): Architektur entwickeln

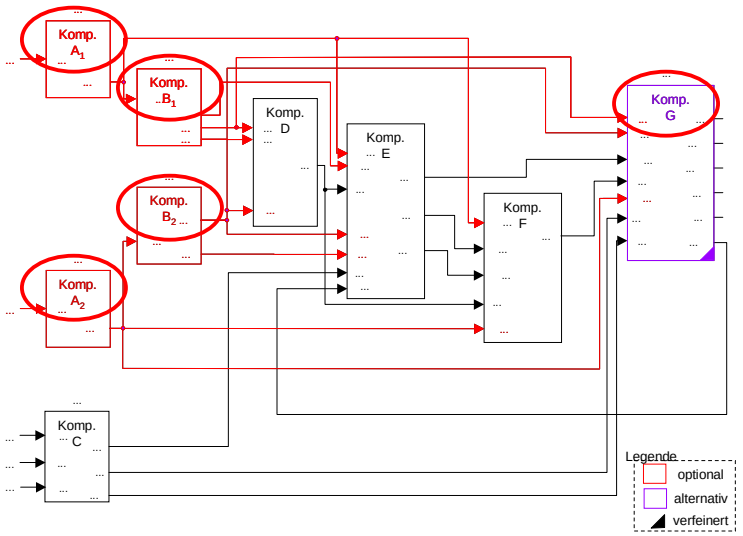
Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



IESE

Beispiel

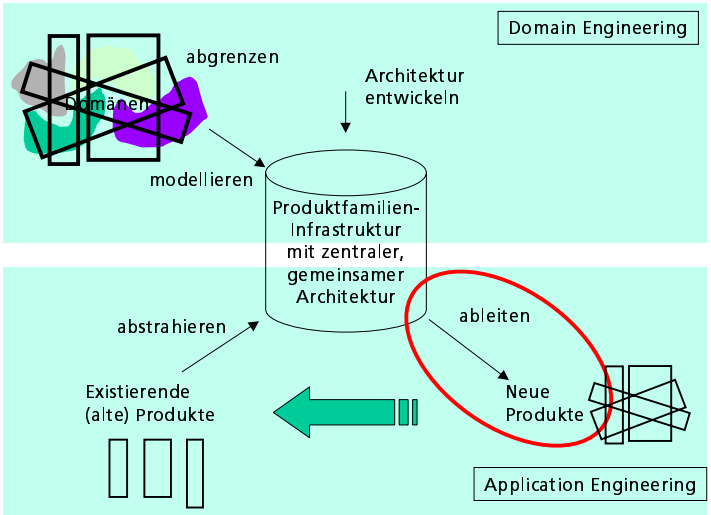
Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



IESE

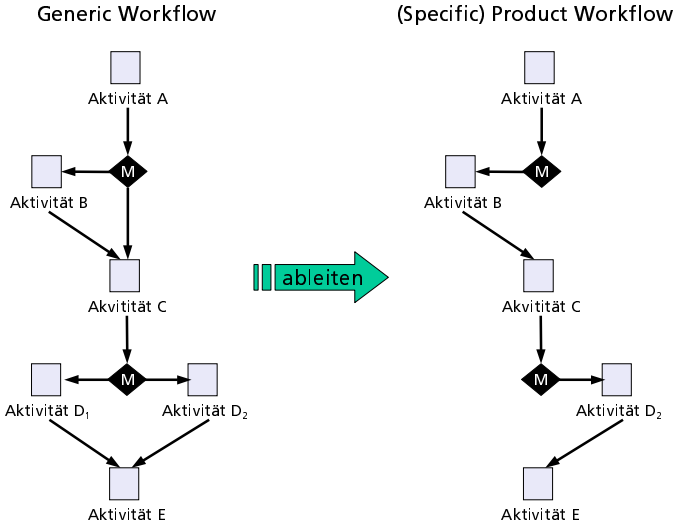
Phasen (4/5): Produkte ableiten

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



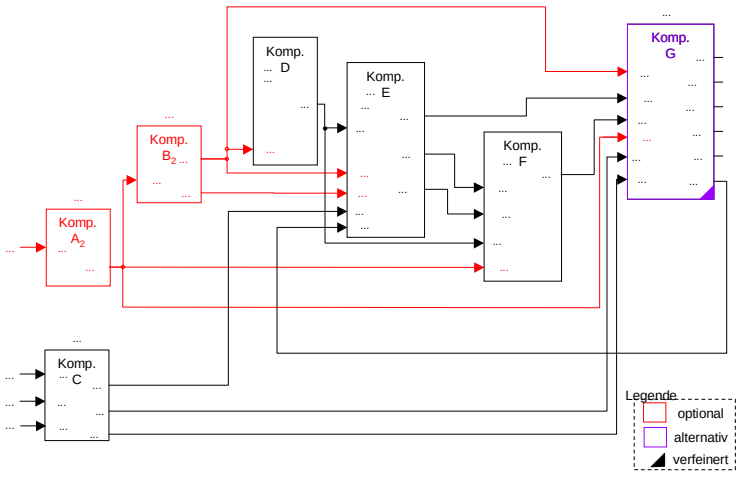
Beispiel (1/2): Modell

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



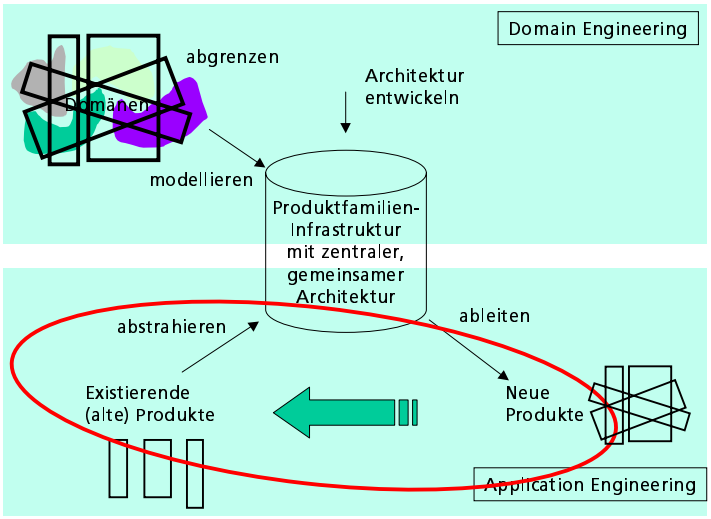
Beispiel (2/2): Architektur

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



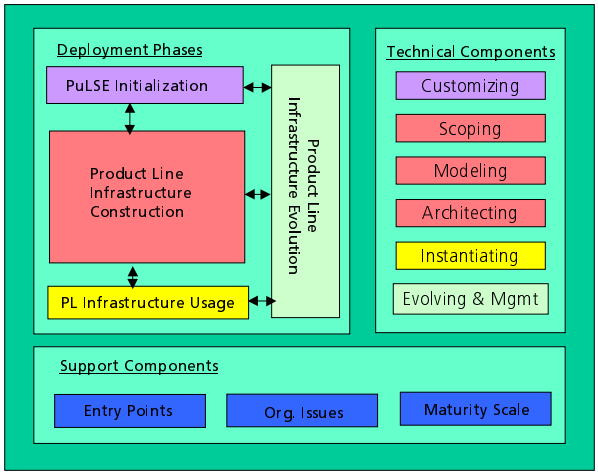
Phasen (5/5): Weiterentwickeln

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



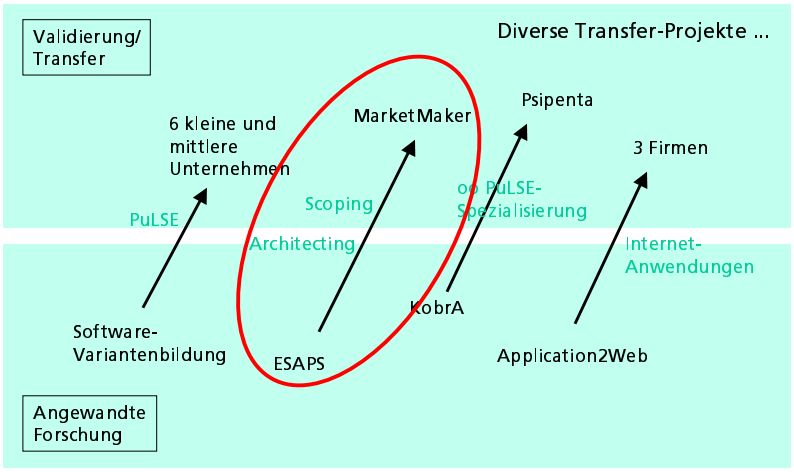
PuLSE™ – Product Line Software Engineering

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



Die IESE-Mission: Technologie-Transfer

Inhalt
- Motivation
- Entwickeln mit Produktfamilien
- Projekt-anwendung



Bedeutung der Planung von Produktfamilien für Time-to-Market von Börsensoftware

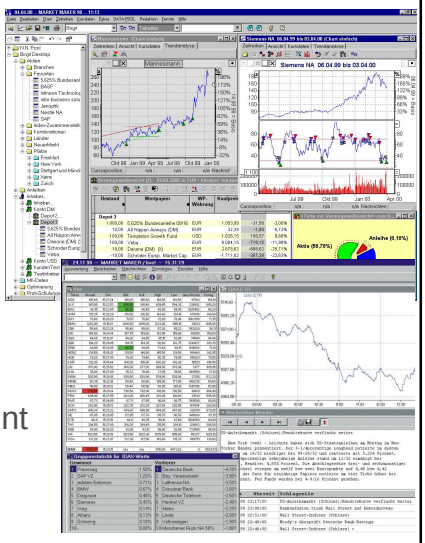
Dr. Martin Verlage

Bedeutung der Planung von Produktfamilien für Time-to-Market von Börsensoftware

Dr. Martin Verlage
Bereichsleiter Online-Produkte
MARKET MAKER Software AG
Kaiserslautern

Was macht MARKET MAKER?

- **MARKET MAKER**
  Wertpapier-Analyse und Verwaltung
- **MARKET MAKER *live!***
  Realtime-Informationssystem
- **DATA-POOL**
  Börsendatenbank
- **MARKET MAKER MERGER**
  der plattformunabhängige Client



MARKET MAKER *unsere Kundenvielfalt*

- Landesbank Sachsen, Baden-Württemberg, Rheinland-Pfalz und Schleswig Holstein
- SI-BW
- mehrere hundert Sparkassen und Volksbanken
- GIS (Genossenschaftl. Informations-Systeme)
- HypoVereinsbank
- Deutsche Bank
- Dresdner Bank
- Commerzbank
- Weberbank
- Oppenheim
- Bankhaus Metzler
- Berenbergbank
- Hauck & Aufhäuser
- Debeka
- Gothaer Versicherung
- Wüstenrot
- Hypo Fonds Marketing
- Union
- Deka
- Gebser & Partner
- Packenius, Mademann & Partner
- Familie Flick
- Familie Fissler
- Preussag
- IVG
- Mannesmann
- Hewlett-Packard
- BayWa
- sowie einige tausend weiterer privater und institutioneller Kunden
- > 130.000 WISO Börse Kunden

Partner und Kooperationen

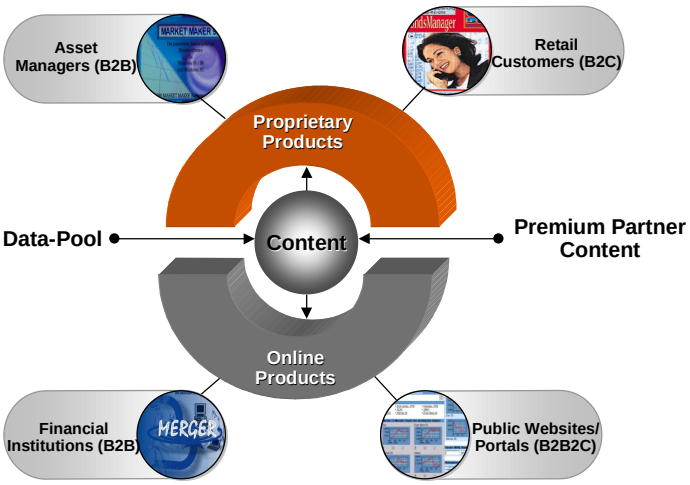
- **Buhl GmbH**

  WISO Börse
  ... die “vielleicht” erfolgreichste Börsensoftware
  - > 130.000 Verkäufe
  - kleiner Bruder von MM

  WISO FondsManager
  - aktuelles Produkt (Juli 2000)
  - auf Investmentfonds zugeschnitten

- **Telebörse TRADER**

Content / Domänenwissen als Treibstoff für Produktfamilien



Beispiele neuer Anforderungen durch Web-Produkte (Internet / Intranet)

- Fokus auf breiten Markt (Consumer)
- Benutzerschnittstelle
- Nicht-Funktionale Anforderungen
  - Performanz
  - Robustheit
- Rechenzentrumsbetrieb (24/7)
- Sicherheit (Security)
- Skalierbarkeit

Zielsetzungen

- Aufbau neuer Systeme unter Verwendung vorhandener Systeme
- Schnelles Time-to-Market
- Flexibilität wegen bewegender Märkte
- Offene Systeme wegen möglicher Kooperationen
- Geringe Wartungsaufwändungen (keine Bestandssicherung, sondern Expansion)

Resultate

- Innerhalb von 12 Monaten Produktreife
  - Web-Sites
    - www.tagesspiegel.de/Finanzen
    - www.vwd.de (Stocks, Warrants)
    - www.diba.de (Wertpapier-Info-System)
  - Arbeitsplatzlösungen bei Sparkassen
  - Requestor-IF für Web-Sites
  - RMI-Schnittstellen für Applikationen
  - Spezialanfragen für IR-Seiten
- Aufwand circa 24 Personenmonate

Schritte

- Scoping
  „Was soll alles rein?“
- Modeling
  „Produktunabhängige Entwicklung“
- Architecting
  „Dokumentation der Entscheidungen“

Scoping

- „Brainstorming“, Intensive Überlegungen
  - IESE als Moderator, kritisches Hinterfragen
- Strukturierte Interviews
  - Systematik, Vollständigkeit
- Bewertung und Dokumentation
  - IESE liefert Methode
- Technische Machbarkeitsstudien
  - nicht Teil von PuLSE
  - günstiger Zeitpunkt

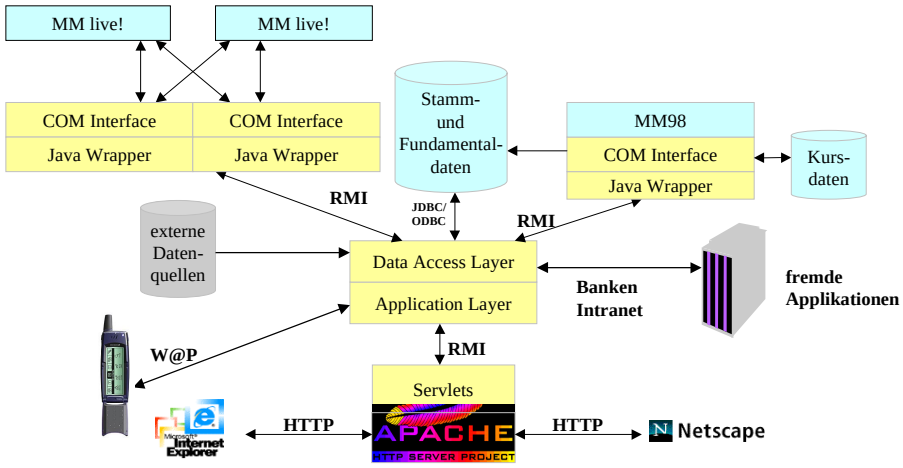
Modeling

- Domänenmodell / Fachmodell
  - Aufbau durch Standardliteratur zum Teil möglich
  - Generelle Beziehungen
- Geringe Interaktion mit IESE
- Vertrautheit

Architecting
„Aufbau generischer Systemarchitektur“

- Bilden von Komponenten
- Verbinden der Komponenten
- Definition von Variabilitäten
- Dokumentation der Konsequenzen von Entscheidungen
- Überarbeiten

MARKET MAKER MERGER



Einschätzungen

- Objektorientierung notwendig (keine Systembrüche)
- Neue Techniken bieten für Produktfamilien mehr Möglichkeiten, nicht weniger
- Gute Marktkenntnis ist zwingend für Erfolg der Methode
- Güte des Scoping und Modeling lassen sich beim Architecting ablesen

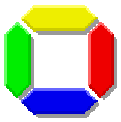
Bewertung Forschungskooperation

- Vertrauensbasis war da
- Erfahrung aus anderen Projekten floß ein
- Anpassungen an Methode opportun
- Rollenwechsel im Laufe des Projekts
- Praktische Probleme unabwendbar?
- Wechsel der Denkweise wertvoller als Techniken
- Aus unserer Sicht lohnt es sich

Anforderungsmanagement und Produktfamilien

Dr. Klaus Pohl

Anforderungsmanagement und Produktfamilien



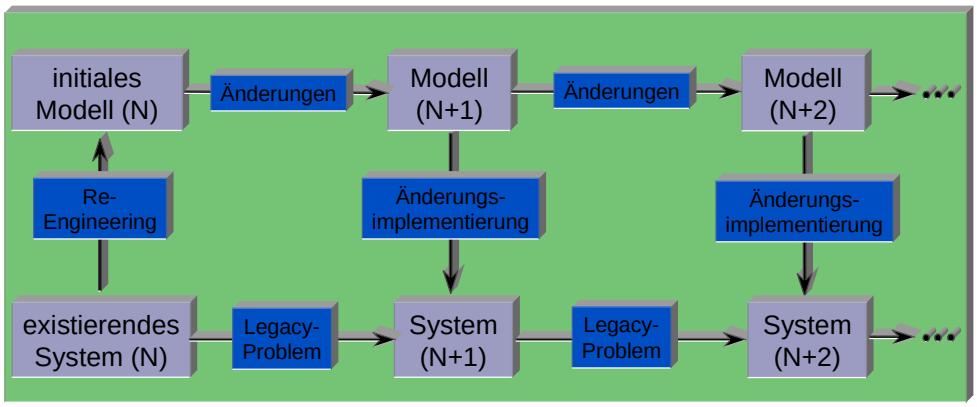
Klaus Pohl, Mark Strembeck
Software Systems Engineering
Universität Essen

Gezielte Wiederverwendung durch Software-Produktfamilien,
Kaiserslautern, 09. November 2000

Vortragsübersicht

- Requirements Engineering: Kurzer Überblick
- Szenarien im Requirements Engineering
- Szenario-basiertes Requirements Engineering für Produktfamilien

Requirements Engineering -- kontinuierliche Tätigkeit



Erhebe und validiere **Modell** der gegenwärtigen **Realität**

\+ definiere **Vision** für die Änderung

\+ schätze **Hindernisse** und **Technikfolgen** ab

= **establishing vision in context**

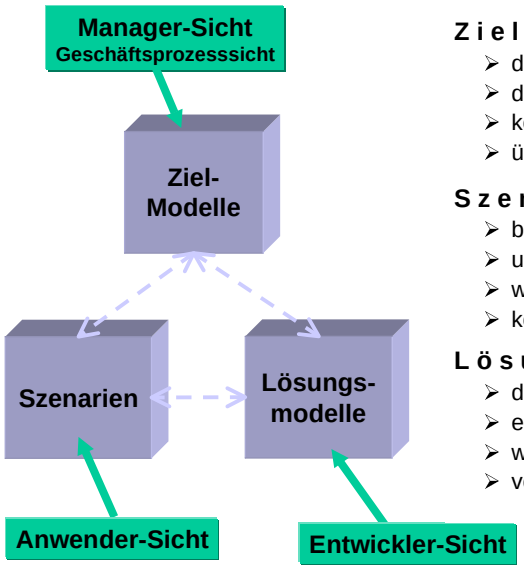
Requirements Engineering als „sozialer“ Prozess

- Requirements müssen gewonnen werden (existieren nicht „per-se“)
  - Problemverständnis entwickeln (Anwendungsdomäne)
  - Wünsche und Einschränkungen erkennen (Begeisterungsfaktoren)
  - Einbeziehung der richtigen Leute (Kosten, Akzeptanz, etc.)

➔ **RE ist ein Lern- und Verhandlungsprozess**

- Unterschiedliche Prozessbeteiligte haben verschiedene Sichten, z.B.
  - Entwickler
  - Nutzer
  - Kunde (Geldgeber)
  - Architekt
  - Tester

Drei Arten von Anforderungsmodellen



Zielmodelle,
- definieren Mehrwert des Systems
- definieren Intentionen (meist ohne Kontext)
- konkurrierende Ziele
- überschaubar

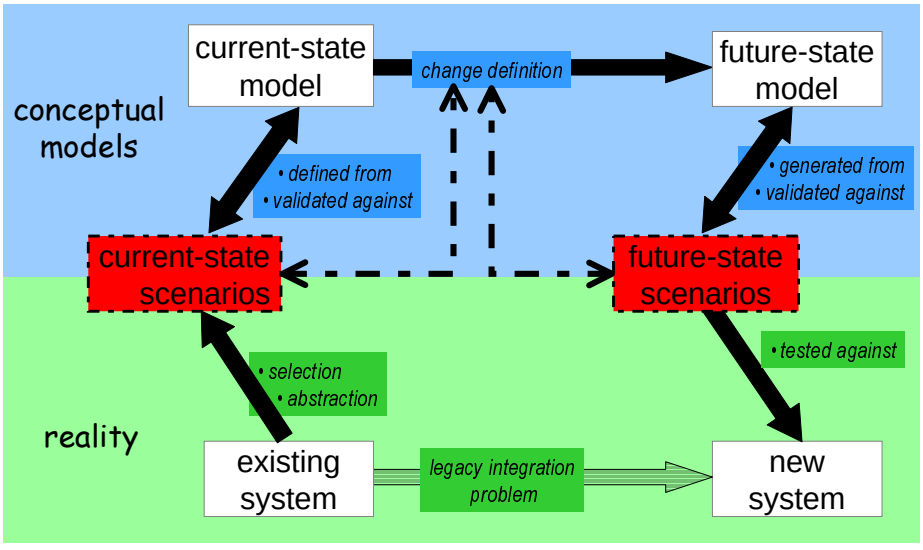
Szenarien,
- beschreiben konkrete Systemverwendung
- unterschiedliche Abstraktion
- widersprüchlich, unvollständig, unpräzise
- konkrete Beispiele für Mehrwert

Lösungsmodelle,
- definieren intendierte Lösung
- einheitliche Abstraktion; komplexe Modelle
- widerspruchsfrei
- verschiedene Systemsichten

(bspw. Daten,Funktion,Verhalten)

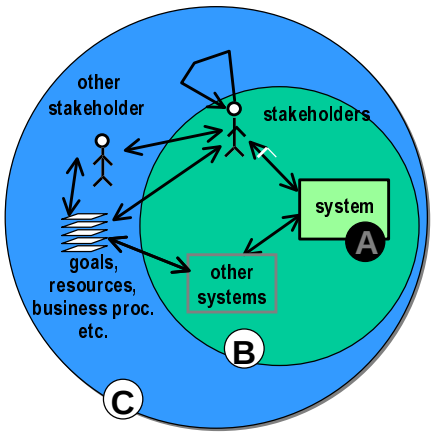
Szenarien als „Middle Level“ Abstraktionen

[Carroll, *Scenario-Based Design* , Wiley&Son, 1995]



Verschiedene Arten von Szenarien

[Kyng 95; Pohl and Haumer, REFSQ 98]



A System Internal Scenarios
context of the system is not considered

B Interaction Scenarios
usage scenarios; interaction between system and environment

C Environmental Scenarios
B + contextual information and interaction

Szenarien sind umfassender als Use Cases

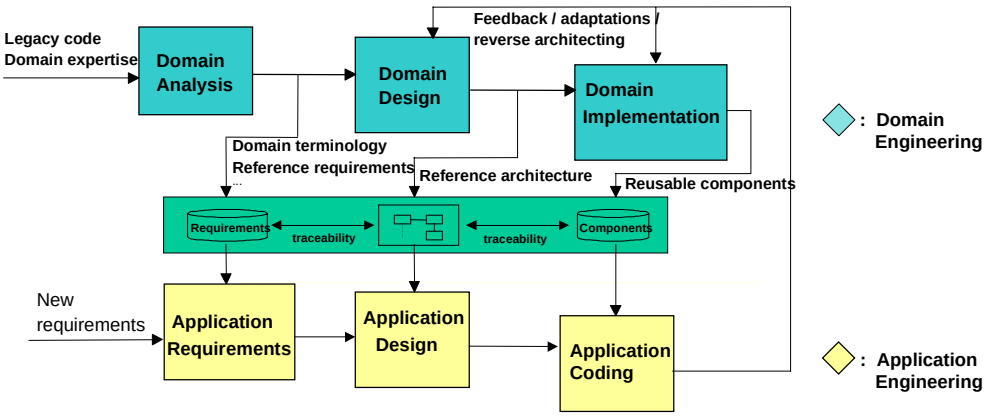
Use Cases = B Szenarien; können auch als A Szen. eingesetzt werden

Aktueller Stand der industriellen Szenarienverwendung

[Weidenhaupt, Pohl, Jarke, Haumer, IEEE Software, 1998]

- Untersuchung von 15 europäischen Software-Entwicklungsprojekten
- Der Einsatz von Szenarien ermöglicht/vereinfacht, u.a.
  - die Verfeinerung abstrakter Modelle
  - eine Komplexitätsreduktion
  - Verwendung als Einigungs- und Konsistenzhilfe
  - die Definition statischer Objektmodelle
- Probleme bei der Verwendung von Szenarien
  - Bezug zu anderen Entwurfs-Artefakten häufig nicht klar definiert
  - Eine methodische Unterstützung fehlt
  - Wiederverwendung noch nicht unterstützt
  - Wartung zu teuer, daher oft out-of date

RE im Produktfamilien-Kontext



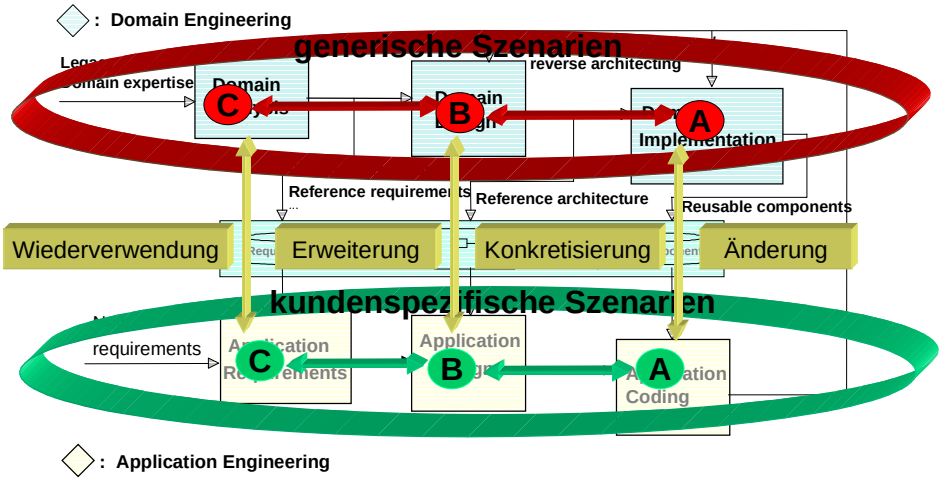
- PF Entwicklung ist auf **Wiederverwendung** und Anpassung ausgelegt
- ein generisches Artefakt existiert in **mehreren** Anwendungen (oft angepasst)
- Änderungen sind häufiger als in normaler Entwicklung
  - jede kundenspez. Anwendung führt zu Änderungen

Probleme von RE im Produktfamilien-Kontext

- Ermitteln von Anforderungen an kundenspezifische Lösungen
  - sollten möglichst bestehende Funktionalität verwenden
  - sollten möglichst wenig Implementierungsaufwand verursachen
- Kunden über Möglichkeiten der Produktfamilie informieren

- Ansatz:
  - Verwendung von Nutzungsszenarien
    - erläutern Funktionalität der Produktfamilie
    - Änderungen werden sofort als Inkremente erfasst und lassen sich somit leichter einordnen
    - Fortentwicklung und Anpassung der Architektur (Evolution)
    - Ableitung geeigneter Testfälle und -szenarien

Lösungsansatz: Szenarien als zentrale Artefakte

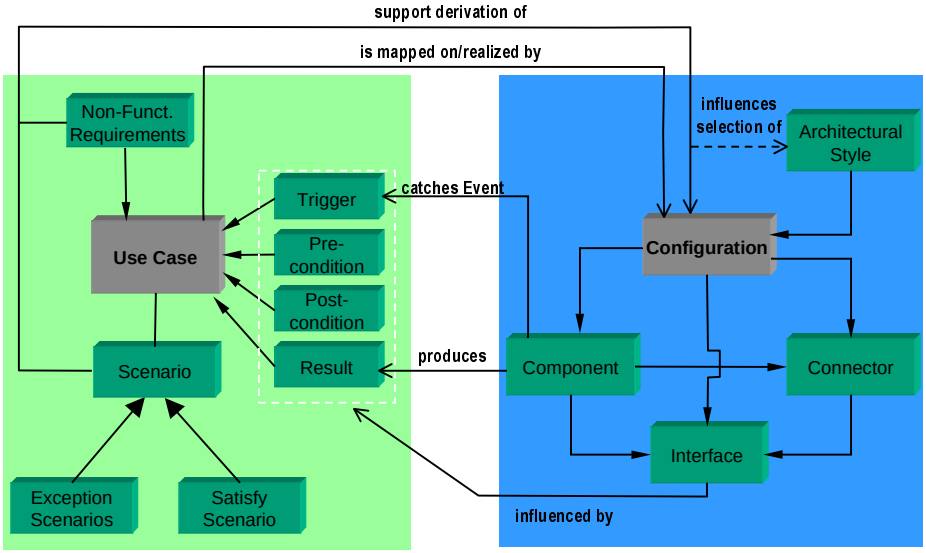


- generische Szenarien beschreiben Produktfamilien-Funktionaliät
- kundenspezifische Szenarien beschreiben Kundenwünsche
  (aus generischen Szenarien „abgeleitet“)

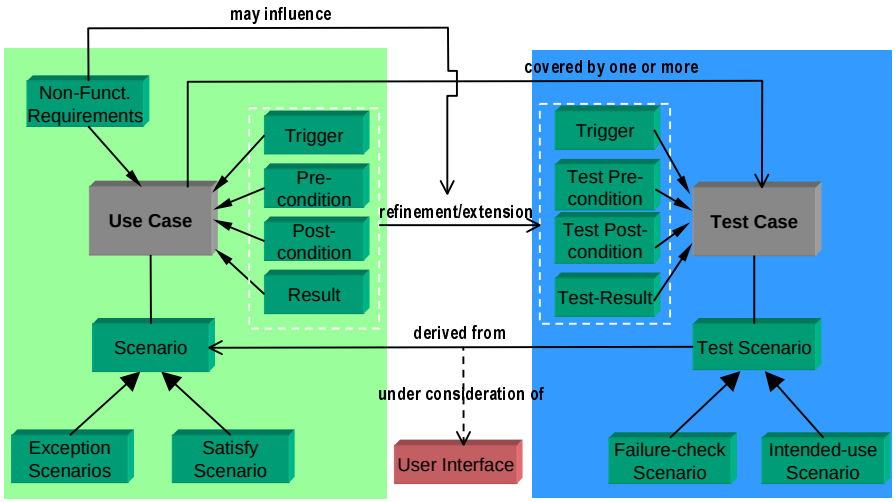
Gewinnung von kundenspezifischen Anforderungen

- Vermittlung der „Leistungsfähigkeit“ der Produktfamilie durch Nutzungsszenarien
  - Grundfunktionalität
  - Detaillierte Erläuterung der jeweiligen Funktionalität(en)
- Positives Ausnutzen von vorhandenen „Freiheitsgraden“
  - Einbringen von bestehenden Lösungen in Entscheidungsprozesse und somit bessere Ausnutzung von Entscheidungsspielräumen
  - Lenken Kundenwünsche in „machbare“ Bahnen
- Erfassen Kundenwünsche
  - Änderungswünsche an bestehenden Szenarien
  - neue Szenarien

Realisierung von kundenspezifischen Anwendungen -- Änderungsauswirkungen auf Architektur (vereinfacht)



Realisierung von kundenspezifischen Anwendungen -- Änderungsauswirkungen auf Test Cases (vereinfacht)



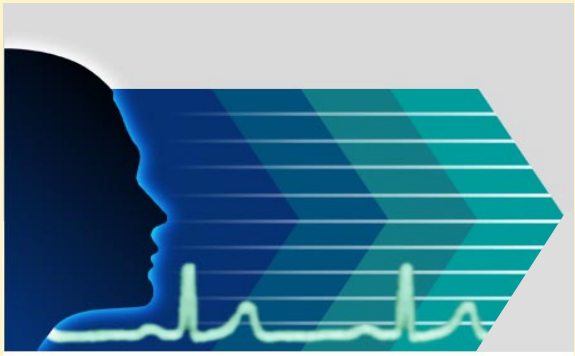
Zusammenfassung

- Szenarien sind bewährtes Mittel zur
  - Einbeziehung des Kunden
  - Modellierung von (funktionalen) Kundenanforderungen
  - Steuerung von Change Management Prozessen
- Unterschied von RE Produktfamilien-Kontext zu sonstigem RE
  - Vermittlung der Funktionalität der PF zum Kunden
  - Wiederverwendung von RE-Artefakten
  - Ermittlung von kundenspezifischen Änderungen und Erweiterungen
  - Unterstützung in der Realisierung
  - positiven Einfluss auf Entscheidungsfreiheiten
  - Unterstützen die Evolution einer Produktfamilie
    - Erkennen von Lücken

Testen und Produktfamilien

Dr. Josef Weingärtner

SIEMENS



Testen und Produktfamilien

Ansätze bei Siemens Health Services für das Testen von Produktfamilien

SIEMENS Medizintechnik

Siemens Products for Health Care

- Electro Medicine — Therapy
- patient monitoring — oncology care systems
- respiratory care — shock wave lithotripsy
- servo-anesthesia — interventional procedures
- telemetry
- electro cardiology
- intracardial measurement / reporting

- Audiology
- hearing instruments
- audiometers
- speech training
- assistive listening devices

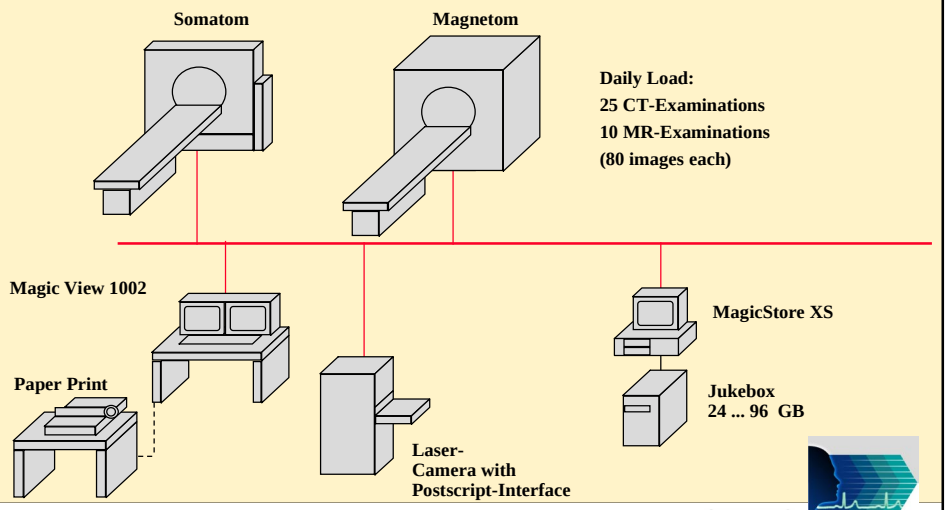
SIEMENS Health Services

SIEMENS medical solutions health services

- Durch den Erwerb von Shared Medical Systems (SMS) und die Zusammenführung der Geschäftsaktivitäten werden in optimaler Weise die Kompetenzen und regionalen Stärken zweier führender Unternehmen im Gesundheitswesen vereint: SMS als ein führender Anbieter auf dem medizinischen Dienstleistungs- und IT-Markt und Siemens medical solutions als das Unternehmen mit der breitesten medizintechnischen Produktpalette.
-
- Statistische Daten (1998/99)
- Umsatz — 8 Mrd DM
- Mitarbeiter (ohne SMS) — 19.200

SIEMENS Health Services

Siemens health services PACS
SIENET Entry Level Solution - CT/MR



SIEMENS Health Services

Testen und Produktfamilien

•**Agenda**

– “Konventionelles Produkttesten an einem Beispiel”

– Übergang zum Testen von Produktfamilien
  • Erweiterung des V-Modells
  • Ziele
– Ausblick

SIEMENS Health Services

Testen und Produktfamilien

•**Hintergrund**

– Das Testen gewinnt für Produktfamilien noch stärkere Bedeutung

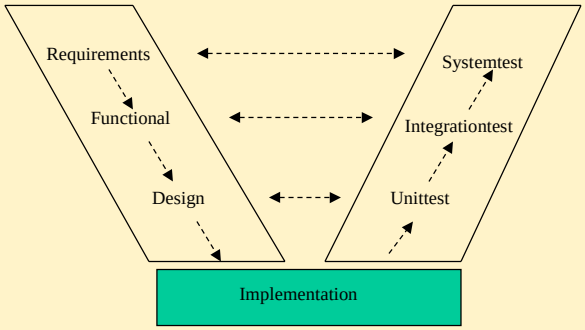
– Die Komplexität der Tests nimmt zu

• **“Gute gegenwärtige Praxis”:**

– Der Entwicklungsprozess (incl. Test activities) folgt dem V-Modell, respektive dem Spiralmodell (ein maßgeschneidertes V pro Spirale)

V-Modell and SystemTest

Entwicklungsmodell bei Siemens Health Services PACS für die Entwicklung eines SW Produkts



Inhalt

- Einige Daten zu Produkt und Projekt
- Zulassungsverfahren für medizintechnische Geräte
- Planung und Controlling der Tests
- Tips

Daten zum Produkt

- *Projekt: Entwicklung einer gemeinsamen SW-Basisplattform für alle künftigen medizintechnischen Großgeräte*
- ***Wichtige Features: Gemeinsame SW-Teile aller Modalitäten (Bsp.: Patientenregistrierung, Bildanzeige, grundlegende Bildverarbeitungsfunktionen, 3D-Funktionen, etc.)***

Daten zum Produkt

Daten zum Projekt

- ***Entwicklungsstart 1992***
- ***objektorientierte Entwicklung in C++***
- ***zuerst unter UNIX; 1995 Übergang auf Windows NT***
- ***ca. 180 Entwickler in Erlangen, USA und Indien***
- ***Davon für Integrations und Testaufgaben: ca. 60***
- ***SW-Größe: ca. 1,5 MBLOC C++ Produktcode, ca. 1,5 MBLOC C++ Testercode***
- ***Anteil sicherheitskritischer Code-Teile: ca 5%***

Zulassungsverfahren für medizintechnische Geräte

- ***Allg. Anforderungen an die SW-Entwicklung***
  - ***Gesetzliche:***
    - ***FDA (GMP)***
    - ***CE-Kennzeichnung***
  - ***Siemens Interne und reifegradbezoge:***
    - ***CMM / OPAL***
- ***Durch:***
  - ***sichergestellt durch die MED Qualitätsrichtlinien zur Erstellung von Produktsoftware sowie des Nachweises deren Einhaltung***

Anforderungen an den Test des Produktes

- ***Produktspezifische Q-Ziele***
- Zu erreichen für den Final release:
- keine neuen Fehler durften in 16-stündigem Test des „final build“ pro funktionalem Paket/Service im System- und Integrationstest entstehen

- Kriterien, die für Phasen-Übergänge zu erreichen waren: das kontinuierliche Testen mußte berücksichtigt werden (e.g. IT->ST: keine CHARMs pro funktionalem Paket älter als 10 Tage)

Anforderungen an den Test des Produktes

- Performance **Test Date:** _____
- **Component Arch./Networking** **Tester Signature:** _____
- **Functionality** Exporting to MOD
- **Requirement** Average Time for S1<= x/Image
- Average Time for S2<= y/Image
- Average Time for S3<= z/Image
- **Conditions S1** **Result:** _____
- **Pass: yes ☐ no ☐**
- **S2** **Result:** _____
- **Pass: yes ☐ no ☐**
- **S3** **Result:** _____
- **Pass: yes ☐ no ☐**
- **Original images: Images have not been stored before into local database with changes ("save as"), I.e. they are not stored panned or rotated or flipped and zoom = 1.0**

- **Test Data 1 Patient D**
- **Test Steps 1. Open Job Control**
- **2. Confirm Save "Yes to All": start measurement**
- **3. Wait for "Completed" in Job Control: measure elapsed time**

Anforderungen an den Test des Produktes

- ***Anforderungen an den SW-Test***
  - Testumfang
    - Modultests
    - Unittests für sicherheitsrelevante Einheiten
    - Integrationstest
    - Systemtest
    - Workflow (Use Case) Szenarien Tests
    - Streßtests
    - Performance Tests

Anforderungen an den Test des Produktes

- ***Anforderungen an den SW-Test***
  - Spezifikation und Reproduzierbarkeit
    - Modultests: Automatisiert im KM-Archiv, soweit möglich;
    - Unittests für sicherheitsrelevante Einheiten: Unittest-Spezifikationen
    - Integrationstest: Integrationstest-Spezifikationen
    - Systemtest-Spezifikationen
    - Workflow (Use Case) Szenarien: Workflow-Testspezifikationen
    - Streßtests: Use-Cases (GUI driven) + memory consumption
    - Performance-Tests: Template

Anforderungen an den Test des Produktes

- Testthemen/Testaufgaben/Testnachweise
  - Modultests: Automatisiert im KM-Archiv: Logfiles; Schriftliche Bestätigung des Modulverantwortlichen;
  - Unittests für sicherheitsrelevante Einheiten: Unittest-Protokolle mit Testsummary
  - Integrationstest: Integrationstest- Protokolle mit Testsummary
  - Systemtest- Protokolle mit Testsummary
  - Workflow (Use Case) Szenarien: Workflow-Test- Protokolle mit Testsummary
  - Streßtests: Logfile + Summary in Form einer zeitlich fortgeschriebenen Statistik
  - Performance-Tests: Protokolle basierend auf dem Template; Summary in Form einer zeitlich fortgeschriebenen Statistik

Anforderungen an den Test des Produktes

- ***Anforderungen an den SW-Test***
  - Organisation der Tests, Zeitablauf
    - Modultests: Automatisierte Tests nach jedem Build; nicht automatisierte Tests am Phasenende;
    - Unittests für sicherheitsrelevante Einheiten: Unittests am Phasenende
    - Integrationstests in IT-Phase
    - Systemtests in ST-Phase
    - Workflow (Use Case) -Tests ab IT-Beginn;
    - Streßtests ab IT-Beginn
    - Performance-Tests ab IT-Beginn

Forderungen bzgl. Testtechniken

- *Welche Testtechniken sind angemessen?*
  - *formale Verifikation: nein*
  - *whitebox-test: Auf Module- und Unittestebene*
  - *Teststufen: Modultest, Unittest, Integrationstest, Systemtest, Workflow-Test, Streßtest, Performance-Test*
  - *nötige/eingesetzte Tools*
    - *Modultests: PC-Lint(statische Codeprüfung), Boundschecker/Purify (Memory Leaks)*
    - *Unittests: Protokolle*
    - *Integrationstest: Protokolle*
    - *Systemtest: Protokolle*
    - *Streßtests: Visual Test*
    - *Performance-Tests: Kamera-Messungen, Stop-Uhr, Timer-Aufrufe*

Planung und Controlling der Tests

- Inhaltliche Festlegungen in einem Testplan
- Phasencontrolling: Projektplan
- Testfortschrittskontrolle: Tägliche Meetings mit Testverantwortlichen ab IT-Beginn; Wöchentliche Projektstatusbesprechungen mit allen Verantwortlichen
- Modultests: Kontrolle der Logfiles / Abschluß durch Modulverantwortliche / Gruppenleiter; Abschluß durch Review
- Unittests für sicherheitsrelevante Einheiten: Unittests am Phasenende: Abschluß durch Review
- Hauptreviews zu bestimmten Phasen
- Codefreeze mit überprüfbaren Kriterien

Planung und Controlling der Tests

- Integrationstests in IT-Phase: Abschluß durch Review
- Systemtests in ST-Phase: Abschluß durch Review
- Workflow (Use Case) -Tests ab IT-Beginn: Periodische Durchführung, gesteuert durch Projektplan;
- Streßtests ab IT-Beginn: Periodische Durchführung, Beginn ab Stabilitätserreichung der Software; Korrekturzyklen auf System- und Modullevel;
- Performance-Tests ab IT-Beginn: Periodische Durchführung

Planung und Controlling der Tests

- ***Testendekriterium: ab wann galt Test als "bestanden"***
- Modultests: Kontrolle der Logfiles: Fehlerfreier Lauf / Entwickler und Gruppenleiter bestätigen, daß Modultest erfolgreich durchgeführt und gefundene Fehler beseitigt wurden
- Wichtige Voraussetzung: Effizientes, toolgestütztes Change Request Management (CHARM)
- Unittests für sicherheitsrelevante Einheiten: Unittestprotokolle sind ausgefüllt; gefundene Fehler behoben;

SIEMENS Health Services

Planung und Controlling der Tests

- ***Testendekriterium: ab wann galt Test als "bestanden"***
- Integrationstests: Testprotokolle sind ausgefüllt; gefundene Fehler behoben;
- Systemtest: Testprotokolle sind ausgefüllt; gefundene Fehler behoben;
- Workflow (Use Case) -Tests ab IT-Beginn: Testprotokolle sind ausgefüllt; gefundene Fehler behoben;
- Streßtests ab IT-Beginn: Streßtest läuft im Toleranzbereich fehlerfrei (bzgl. Leaks und Laufzeitfehler);
- Performance-Tests ab IT-Beginn: Ziele erreicht; bei nicht erreichbaren Zielen ´Placet´ der Requirements-Ersteller vorliegend

– ***Schulung von Testern und Entwicklern: On the Job***

SIEMENS Health Services

Projektrückblick: Testergebnisse

- ***Welche Test-Ergebnisse wurden gesammelt***
  - Fehlerraten:
    - wurden wöchentlich in Statistikauswertungen gesammelt und in Statusbesprechung verfolgt
  - Testprotokolle
  - Testaufwand in Personenjahren
    - ca. 18 MJ

Projektrückblick: Ziele und Ergebnisse

- ***Welche Ziele konnten erreicht werden***
  - Q-Ziele
    - Man kam den Q-Zielen sehr nahe
  - Testziele
    - Geplante Testziele wurden bzgl. Durchführung erreicht
- ***Welche Ziele konnten nicht / nicht voll erreicht werden***
  - Beispiel Performance
    - Grund: Architektonische Randbedingungen nicht lösbar
  - Beispiel 0 Fehler in 16 Stunden Test
    - Grund: Software zu komplex; Parametrierbarkeit

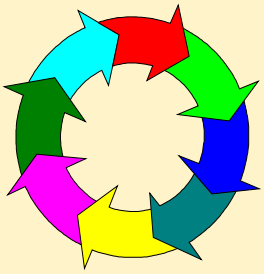
Projektrückblick: Ziele und Ergebnisse

- Welche Maßnahmen könnten dies das nächste mal verbessern
  - Stärkerer Tool-Einsatz, vor allem in frühen Phasen
  - Frühere Schulung
  - Verantwortlichkeiten für Fehlersuche / Reaktion auf Performance- und Stresstest-Resultate früher festlegen
  - Erhöhen der Qualität und Abdeckung von Codereviews

Tips und Lehren

- Tips für Testmanager
  - noch mehr Gewicht auf Fehlervermeidung in frühen Phasen; noch intensivere Einbindung der Tester in frühe Phasen
  - Gute Statistiken: zeigen den Stand des Projektes sowie die voraussichtliche Dauer bis zu einer Freigabe
  - vermeiden sollte man:
    - Aufkommen von ´finger-pointing´
- Die wichtigste Lehre aus dem Testbetrieb
  - Das Herzstück ist ein leistungsfähiges Konfigurationsmanagement
  - Sehr gutes Fehlermanagement

USE CASES and SYSTEM SCENARIOS



- Some Basic Principles
- Intended Tool use
- Goals
- Further preconditions
- Next steps

Übergang zum Testen von Produktfamilien

Idee:

1) Erweiterung des V-Modells für den Teil des “system family requirements engineering”
2) Erweiterung des V-Modells für den Test des “system family engineering”-Teils
3) Aufsetzen eines zweiten erweiterten V-Modells. Innerhalb dieses Modells werden Szenarien und Requirements top down eingefügt. These Szenarien und Requirements werden auf der jeweiligen Ebene des V-Modells verfeinert. Dadurch erhält man ein Modell, bei dem auf der rechten Seite die Tests der Szenarien nur noch durchgeführt werden müssen.

- Dieses Vorgehen wird in einer Kooperation mit der Universität Essen aufgesetzt.

Übergang zum Testen von Produktfamilien

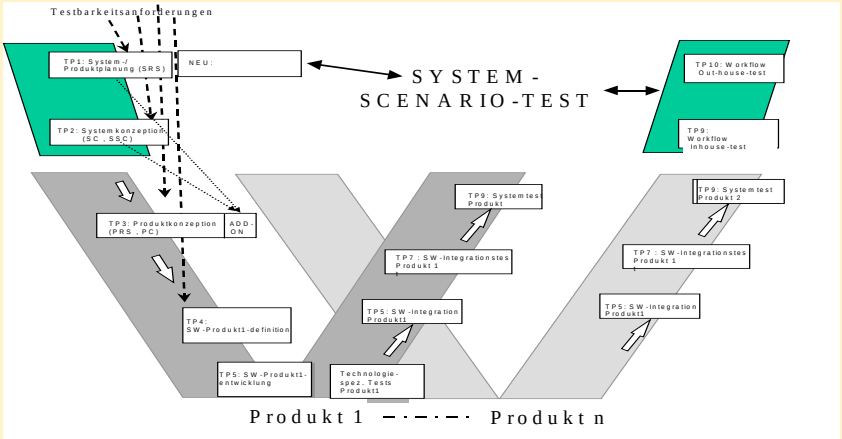
Idee:

4) Alle “good practices” der Vergangenheit für das Testen der Produktfamilien beibehalten

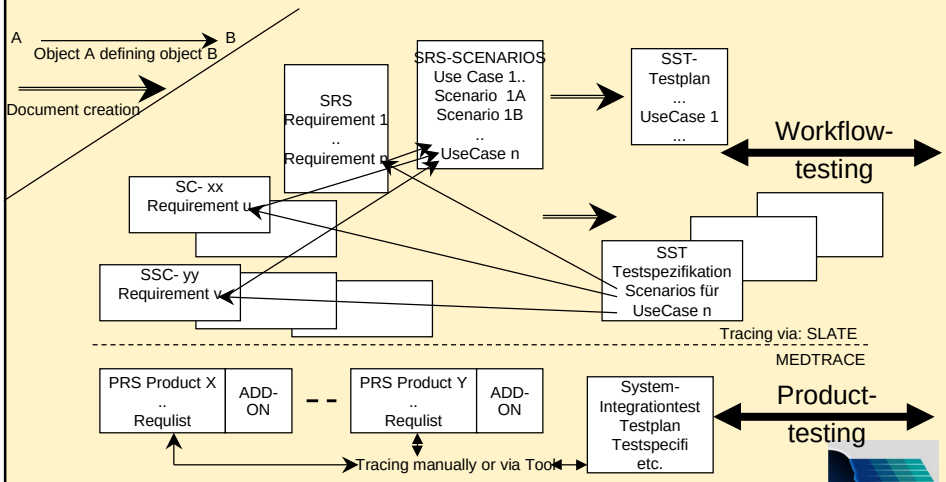
5) Gemeinsamer Systemtest der Produktfamilien

6) Systemtest gemäß “Feldeinsatz”

Erweitertes V-Modell



System-Szenario-Test (SST)

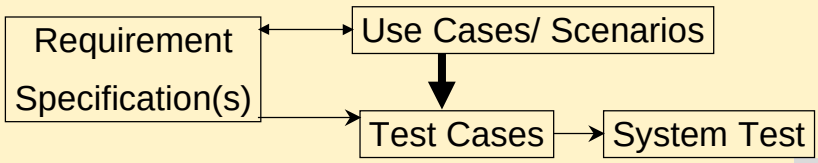


Testen und Produktfamilien

- Nicht nur “end-user Szenarien” werden betrachtet (Sondern genaus Szenarien zur Installation und Wartung)
- Auch Stress-Test-Szenarien werden berücksichtigt

Testen und Produktfamilien

- Wiederverwendung
- Durch geeignetes Tracing der Requirements / Szenarien und Testfälle durch alle Ebenen kann die Wiederverwendung von Testfällen für wiederverwendete Komponenten sichergestellt werden (Teil der Kooperationsarbeit mit Universität Essen)



Definitions 1

- USE CASES (UC)
  Use Cases are what happens when actors interact with the system. One or more actors use the system to achieve a desired goal. By recording all the ways our system is used („cases of use“ or Use Cases) we can accumulate all the goals or requirements of our system. Use cases are goals that are archived by acting according well defined scenarios.Every Use Case shall be described in a fixed format (SLATE generable, at the moment plain text see page 9) in the SPICA SRS-SYSTEM SCENARIOS.
  To Every UC at minmum one Requirement of the SRS SPICA is linked.
- SCENARIOS (SC)
  Scenarios consist of a sequence of steps to achieve the goal of a UC (see above) , each step in a scenario is a sub (or mini) goal of the one Use Case. Scenarios shall not be refined by scenarios and can therefore directly be used as input for System-Workflowdescriptions and -testspecifications.
  .

.

.

Definitions 2

- REQUIREMENTS (R)
  Are defining the wishes,intentions etc. of stakeholders concerning the system. Requirements can be defined on different documentation levels. Every UC has to be linked to one requirement in minimum (normally the “Use Case Requirement”) but should also be linked to every requirement, which is covered by the SCs associated to the UC (i.e. DICOM-Datatypes, Login/Logout, userspecific requirements etc.) . This feature shall be used to increase effectiveness of testing (Implizit testing of requirements).
  SYSTEM-SCENARIO-TESTS (SCT)
  Based on UCs and SCs descriptions, System-Scenario-Tests are possible which are the only tests on system-level and include installation- and maintenance-tests,
- ACTORS
  Actors can be human or not human (computer-system reactions on certain events such as autorouting/prefetching)

Intended Tool Use

- SLATE USE

  The structured descriptions of UCs and SCs is / shall be input in a database to allow the use of the Requirement Engineering Tool “SLATE”.
  Based on this modelling description in the slate-generated System concept Rev0.1 already UseCase and Scenarios are inserted, for the SCs an EXCEL-Template is available for Slate-import (intended automated import)

- If necessary, the used procedure/template has to be modified (Slate compatible)

Testen und Produktfamilien
Ziele

- Erreichen eines ersten Teilschritts zu einem “Use Case getriebenen” Software Entwicklungsprozesses
- Innhaltlich treffender System-Workflow-Test ist trotz der steigenden Komplexität möglich
- Minimisierung der Test-Aufwände ist möglich durch Mittesten von ursprünglich “nicht so hoch priorisierten” Requirements
- Verwendung eines “Requirement Engineering Tools” auch für den Test, um die Erzeugung von Testfällen und die Requirement “Traceability” zu unterstützen
- Vorgehen ermöglicht die Planung von “work flow tests” (Ressourcen, Daten)

Template Use Case

➢Goal
..... Text ...

➢Defining Actors
..... Text ...

➢Description
..... Text ...

➢Defining Scenarios
..... Text ...

Template Scenario

		Scenario	
		Goal	
		Precondition	
		Postcondition	
		Actor	

#	from	event	to	description	remark/condition
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					

Nächste Schritte

- Teams festlegen
- “Es einfach tun”
- Regelmäßiges Controlling

Validation and Testing

Testen und Produktfamilien

- **Agenda**
  - "Konventionelles Produkttesten an einem Beispiel"
  - Übergang zum Testen von Produktfamilien
    - Erweiterung des V-Modells
    - Ziele
  - Ausblick

Ein Change Management Prozess für ein Reuse Object Repository

Dr. Annette Schreiber

S

Ein Change Management Prozeß für ein Reuse Object Repository

Annette Schreiber
Siemens AG

S

Inhalt

1. Begriffe und Definitionen
2. *Reuse Repository*
3. Rollen
4. Change Management Prozeß

1. Begriffe und Definitionen

Reuse Object (RO)

= Software - Asset
(z.B. Komponente, Code-Segment,..)

+ *RO-environment*
(z.B. Information, Spezifikationen, *Application Guide*, Test Module, RO-Evaluation ...)

➨ Speicherung und Verwaltung im sog. *Reuse Repository*

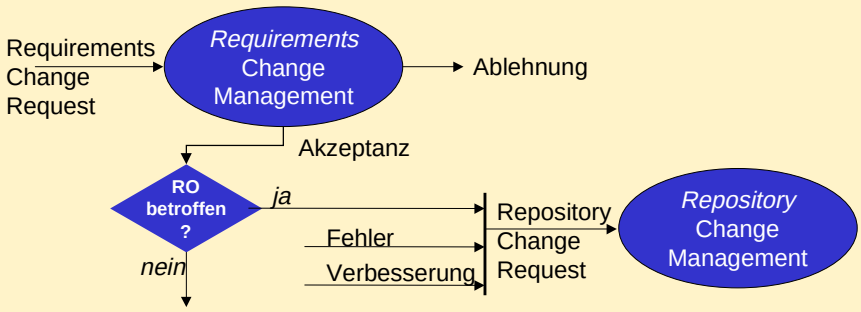
➨ Unterscheidung in verschiedene “Generizitätsstufen” (z.B. voll-generisch, kundenspezifisch, ...)

1. Begriffe und Definitionen

Repository Change Management

➨ Change Management für ROs, die in einem *Reuse Repository* gespeichert werden

➨ *Repository* Change Management ↔ *Requirements* Change Management :



S

1. Begriffe und Definitionen

Change Request

Typen:

- *Requirements Specification Change*
  - Kunden-Requirements
  - System-Requirements
  - Produktfamilien-Requirements (z.B. Anforderung der Wiederverwendbarkeit in verschiedenen Produkten / für verschiedene Kunden)
- *Improvement* (z.B. von “unbrauchbaren” ROs)
- *Critical Error* (Fehler, der aufgetreten ist und verbessert werden muß)
- *Sleeping Error* (Fehler, der vom Kontext der Wiederverwendung abhängig ist)

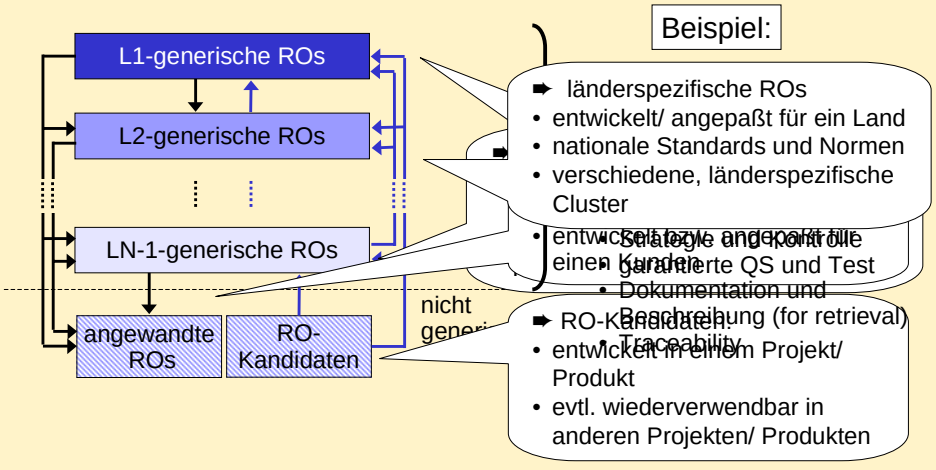
Initiator:

- Marketing, Product Management, Produktdefinitionsteam
- Entwicklungsteams
- Domänenexperten (*R-group*)

S

2. Das Reuse Repository

= administrierte, organisierte und kontrollierte Menge aller ROs



S

3. Rollen

R- Group (Reuse Group)

Experten-Gruppe

Zusammensetzung:

Experten von allen Domänen (Architektur, Anwendung, ...)

Aufgaben:

- Definition einer Plattform Architektur
- Definiton der ROs und ihrer Umgebung
- Beratung des “R-CCB”
- Unterstützung der Projekte
- Technische Kontrolle

S

3. Rollen

R- CCB (Reuse Change Control Board)

Entscheidungsgremium

Zusammensetzung:

Manager mit personeller und finanzieller Verantwortung

Aufgaben:

- strategische Entscheidungen bzgl. Inhalt, Prozesse, Architektur
- Entscheidungen bzgl. Änderungen, Prioritäten von Änderungen, Releases von ROs
- Entscheidungen bzgl. der Kostenzuteilung
- Entscheidungen bzgl. des Personals
- Festlegung der Verantwortlichkeiten (z.B. für ROs, Administration, Durchführung der Änderungen, Qualitätssicherung, Test)

S

3. Rollen

Repository-Administrator – Administration und Pflege des Reuse Repositories

Aufgaben:

- Installation und Administration der Toolumgebung
- Umsetzung des Reuse Repositories mittels eines Tools
- Administration der Error/ Change Reports
- administrative Überwachung der Change Requests (Vollständigkeit der Informationen, Fortschrittskontrolle, ...)
- Berichte über den Zustand und aktuelle Vorgänge des Reuse Repositories; Mitteilung und Bekanntgabe von Änderungen des Reuse Repositories

S

3. Rollen

RO-Responsible – verantwortlich für ein RO

Aufgaben :

- Pflege und Wartung eines ROs
- Unterstützung der Projekte bei der Wiederverwendung des ROs
- Mitglied der R-Group bei Bedarf

Change Request Responsible – verantwortlich für einen Change Request

Aufgaben :

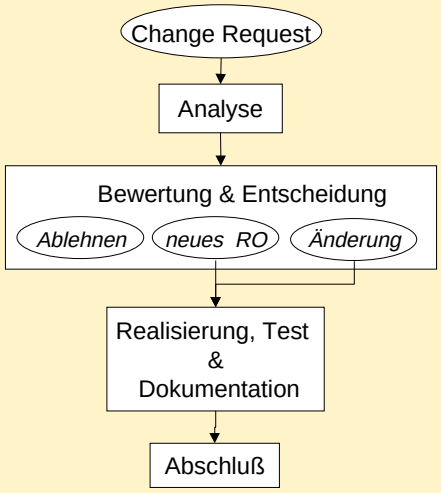
- administrative Bearbeitung eines Change Requests
- verantwortlich für die Realisierung des Change Requests

⇒ Rolle, die innerhalb der Bearbeitung eines Change Requests von unterschiedlichen Personen wahrgenommen werden kann

S

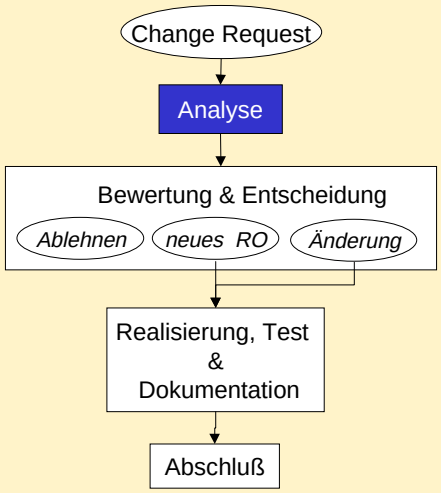
4. Change Management Prozeß

Überblick



S

4.1 Change Management Prozeß – Analyse



S

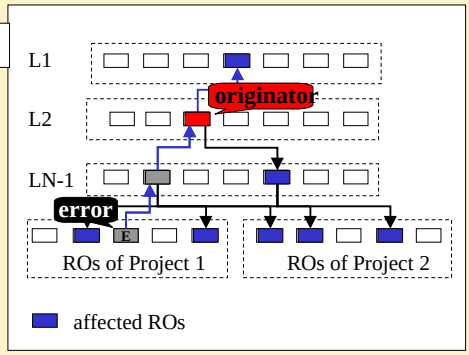
4.1 Change Management Prozeß – Analyse

Vor-Analyse des Change Requests

Festlegung des CR-Responsible

Identifizierung des verursachenden ROs

Identifizierung aller betroffenen ROs und Projekte



S

4.1 Change Management Prozeß – Analyse

Vor-Analyse des Change Requests

Festlegung des CR-Responsible

Identifizierung des verursachenden ROs

Identifizierung aller betroffenen ROs und Projekte

Integration aller Stakeholder im weiteren Prozeß

Grobentwurf verschiedener Lösungen

Kostenabschätzungen

CR-Analysis-Report

Stakeholder: RO-Responsibles von betroffenen ROs, PLs/ SWTLs von Projekten, ...

Verschiedene Möglichkeiten der Integration:

- Benachrichigung oder
- Erweiterung der R-Group

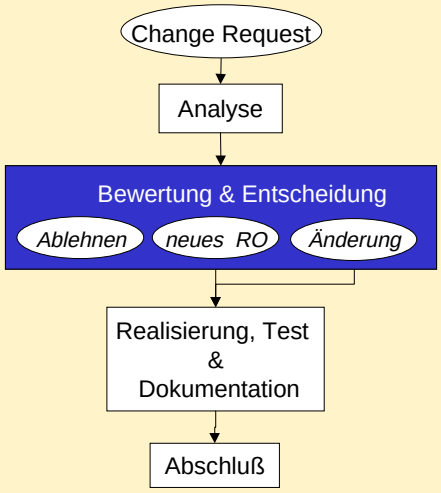
beinhaltet:

⌘ Kosten zur Durchführung der Änderung

⌘ Kosten zur Adaption aller betroffenen ROs und Projekte

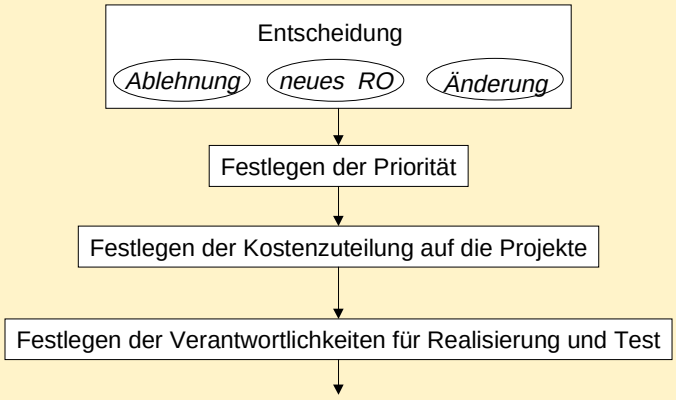
S

4.2 Change Management Prozeß – Bewertung



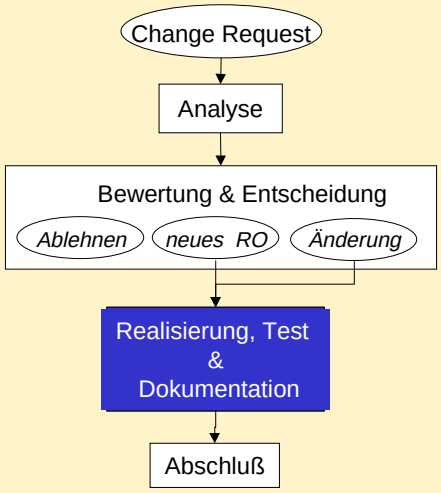
S

4.2 Change Management Prozeß – Bewertung



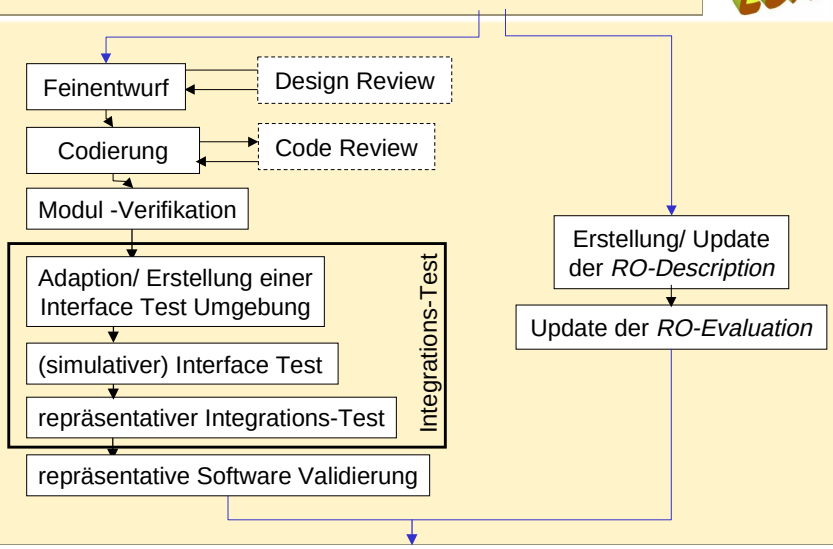
S

4.3 Change Management Prozeß – Realisierung



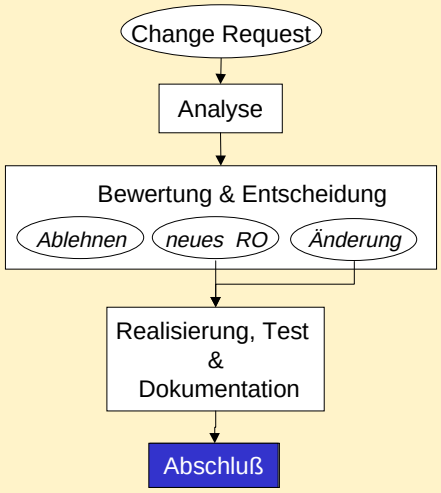
S

4.4 Change Management Prozeß – Realisierung



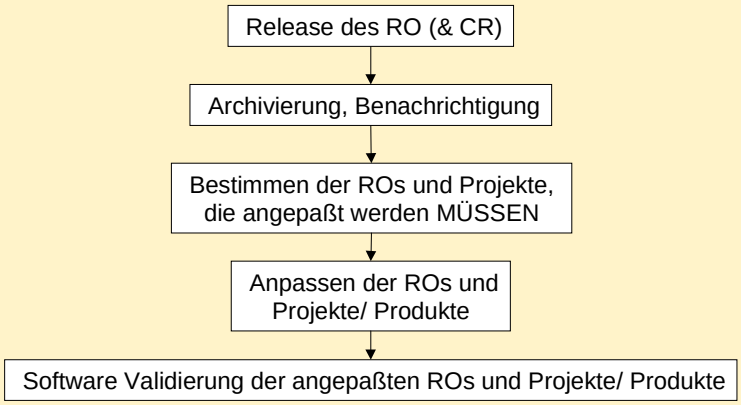
S

4.5 Change Management Prozeß – Abschluß



S

4.5 Change Management Prozeß – Abschluß



5. Zusammenfassung und Ausblick

Zusammenfassung:

Definition eines Prozesses für das Change Management eines Reuse Repositories bestehend aus Software-Assets (ROs)

Ausblick:

➨ Integration des Reuse Repository Change Management innerhalb des Standard-Entwicklungs-Prozesses:

- Harmonisierung der Rollen: z.B. Koordinierung der Rollen des Requirements-CCBs und des R(euse)-CCBs ?
- Entwicklung *with reuse*
- Entwicklung *for reuse*

➨ Erweiterung des Reuse von Software-Assets auf allgemeine Assets

Eine Fallstudie für den Produktlinien-Ansatz bei Bosch: Fahrzeug-Umfeldsensorik

Dr. Stefan Ferber

ITEA
Information Technology for European Advancement

Eine Fallstudie für den Produktlinien-Ansatz bei Bosch: Fahrzeug-Umfeld-Sensorik

Dr. Stefan Ferber
Robert Bosch GmbH
Forschung und Vorausentwicklung
Software Technologie FV/SLD
Frankfurt am Main

BOSCH 

Übersicht

- Einführung
  - Was sind Produktlinien?
  - Warum Produktlinien bei Bosch?
  - Gewachsene Produktlinien
  - Geplante Produktlinien mit der Produktlinien-Technologie
- Fallstudie
  - Ziel der Fallstudie
  - Anwendungen der Fahrzeug-Umfeld-Sensorik
  - Basistechnologie
  - Arbeitsergebnisse
  - Bewertung
- Zusammenfassung

BOSCH 

Einführung
Definition Produktlinie

- Eine Produktlinie ist eine Gruppe von softwareintensiven Produkten
  - die eine gemeinsame Basis an Feature haben und ein Marktsegment adressieren
  - die eine gemeinsame Architektur haben
  - die unter Verwendung gemeinsamer Kernkomponenten (Assets) entwickelt werden
- Beispiel:

Produktlinie → **Marktsegment**

Autoradio/CD-Spieler Geräte

Markt der Unterhaltungsgeräte im Auto

Einführung
Produktlinien-Technologie

- neues Entwicklungsparadigma für softwareintensive Systeme
- systematische Verwendung von Software Assets, um Produkte einer Produktlinie zu „entwickeln“
  - zusammenzubauen
  - zu generieren
  - zu modifizieren
  - zu konfigurieren
- systematische Entwicklung solcher Assets
- strategische, umfassende Wiederverwendung zur Erreichung von Geschäftszielen und Marktpositionen.
- ökonomischen Ausnutzung von Gemeinsamkeiten in Produkten
- Asset ist „eine wertvolle Sache“
  - Software Assets–Beispiele:
    - Anforderungsdokumentation und Anforderungsanalyse
    - Architektur, Design
    - Software–Komponenten (Module, Klassen, ...)
    - Frameworks
    - Entwicklungsprozesse, Methoden, Werkzeuge, Änderungsprozesse
    - Testspezifikationen, Testdaten, Testpläne

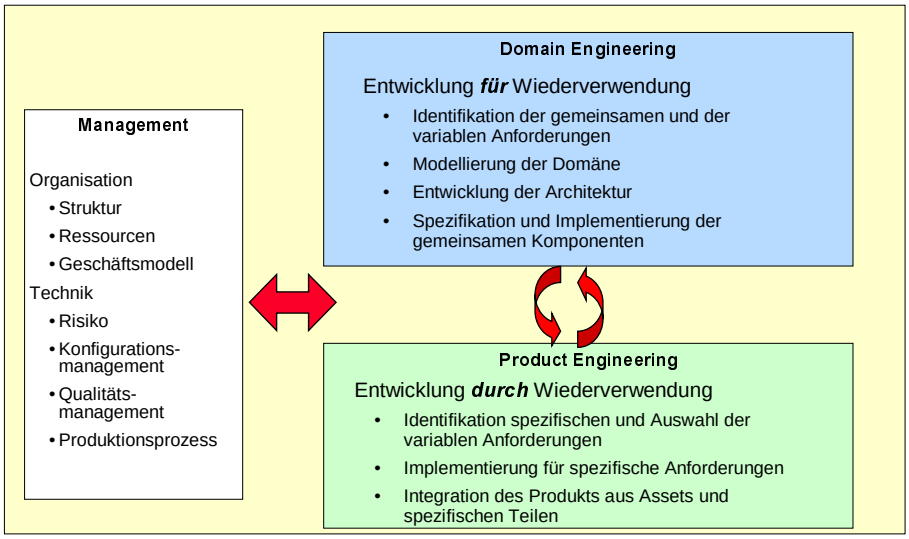
Produktlinien bei Bosch
Gewachsene Produktlinien

- Motivation der gewachsenen Produktlinien und Plattformen
  - Gleiche Basisfunktionalität für unterschiedliche Fahrzeughersteller
  - Hoher Kostendruck
- Keine explizite Produktlinien-Architektur
- Keine explizite Untersuchung der Varianten und Variabilitäten
- Produktlinien-Wissen implizit in den Köpfen der Mitarbeiter
- Beispiele:
  - Motorsteuerungen
  - Getriebesteuerung
  - Fahrdynamik
  - Radio
  - Navigationssysteme

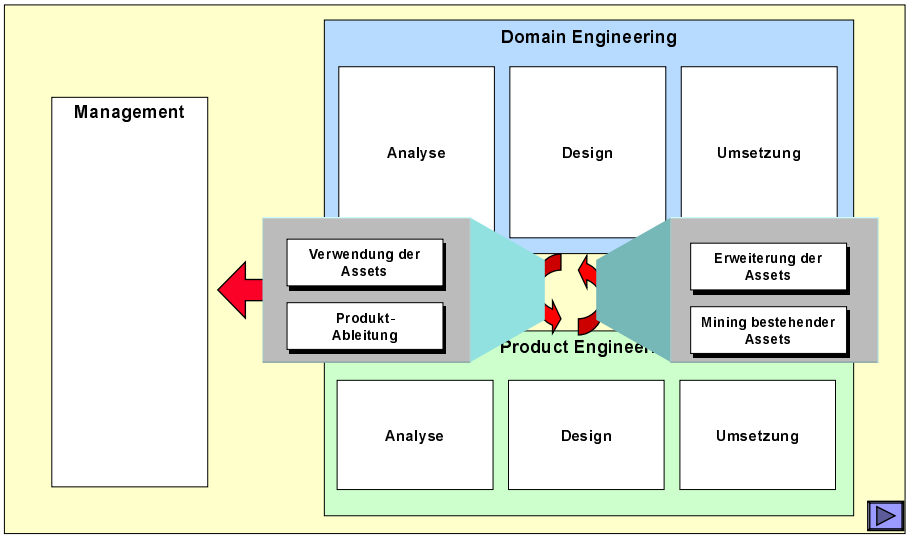
Produktlinien bei Bosch
Geplante Produktlinien

- Ziele
  - Qualitativ hochwertige und zeitstabile Assets
  - Einfache, kostengünstige und schnelle Produktmontage
  - Geringerer Aufwand beim Produkttest
  - „Return of Investment“ der Assets
  - Domänenwissen institutionalisieren
- Systematische Systemfamilie-Entwicklung
  - Gemeinsamkeiten identifizieren
  - Varianten und Variabilitäten berücksichtigen
  - Produktlinien-Architektur (Hardware+Software) erstellen
  - Assets für die Produktentwicklung bereitstellen
  - Produktentwicklung soweit wie möglich durch „Montage“ von Assets
- Methodik
  - Domain Engineering

Produktlinien-Technologie
Prozessfamilien



Produktlinien-Technologie
Schnittstellen Domain/Product Engineering

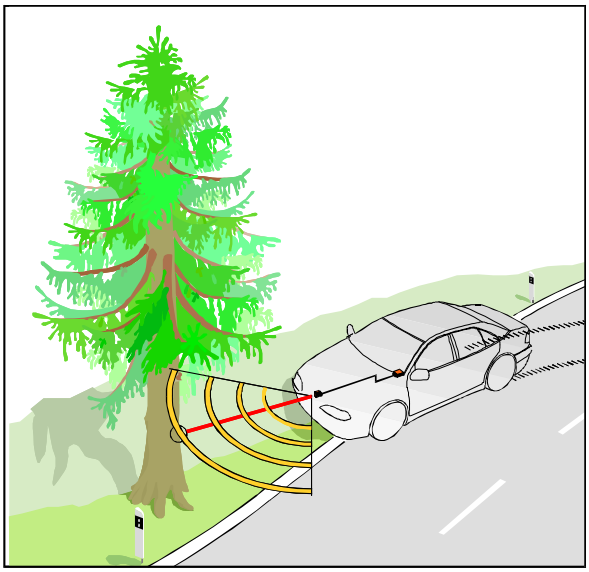


Fallstudie für den Produktlinien-Ansatz: Fahrzeug-Umfeld-Sensorik

- Fahrzeug-Umfeld-Sensorik: eine typische Produktlinie von Bosch für das Geschäftsfeld Kraftfahrzeugausrüstung
- Genügt der Produktlinien-Ansatz den besonderen Anforderungen der Informationstechnologie im Automobil
  - Zuverlässigkeit?
  - Sicherheit?
  - Vertraulichkeit?
  - Komplexität?
  - Kosten?
- Führt Domain Engineering zu
  - einer höheren Software- oder Hardware-Wiederverwertung?
  - kürzerer Entwicklungszeit?
  - höherer Qualität?
  - geringeren Entwicklungs- und Produktkosten?

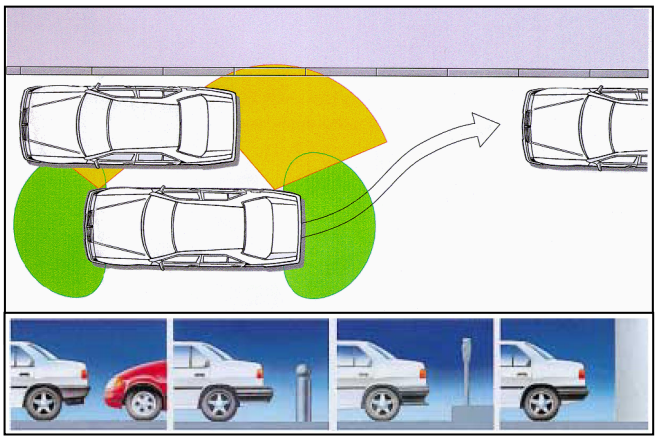
Fahrzeug-Umfeld-Sensorik Anwendung Pre-Crash-Detection

- Erhöhung der Insassensicherheit
- Überwacht relative Geschwindigkeit und Beschleunigung auftauchender Objekte
- Vorhersage von Zusammenstößen
- Vorhersage der Zeit, des Ortes, Aufprallgeschwindigkeit und -richtung
- Beeinflusst Airbags, Gurtstraffer oder andere Aktoren



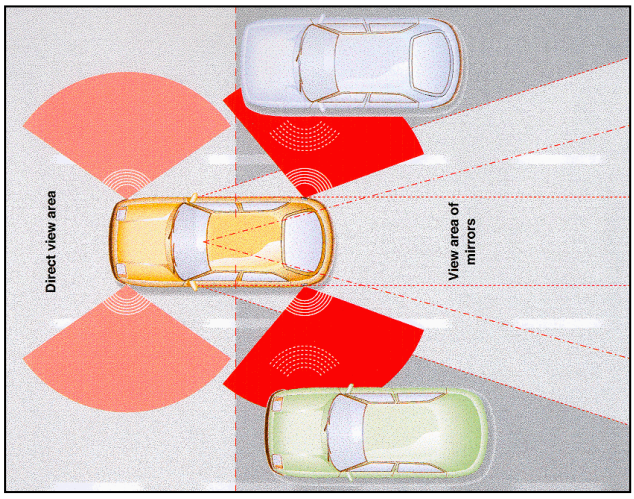
Fahrzeug-Umfeld-Sensorik Anwendung Einparkhilfe

- Verbesserung des Fahrkomforts
- Unterstützt den Einparkvorgang, indem vor Kollisionen gewarnt wird
- Kann Größe einer Parklücke vermessen
- Unterstützt den Fahrer beim Lenken durch Richtungs-vorgaben
- Kann Abstände zum nächsten Hindernis anzeigen

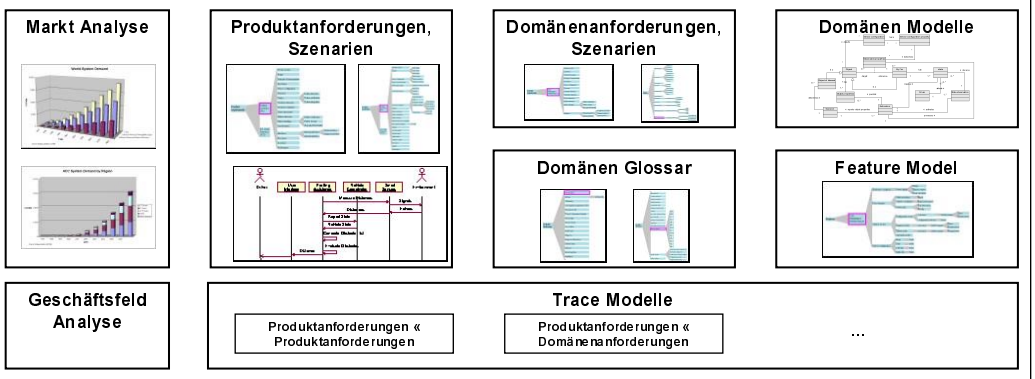


Fahrzeug-Umfeld-Sensorik Anwendung Tote-Winkel Überwachung

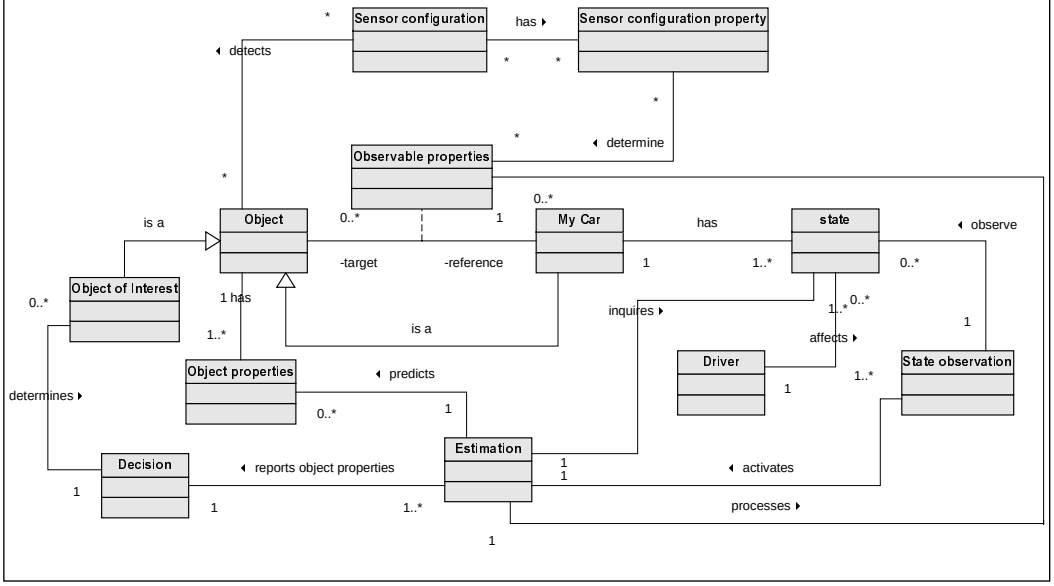
- Verbesserung des Fahrkomforts und der Sicherheit
- Überwacht den Seitenbereich bezüglich parallel fahrender Fahrzeuge
- Zeigt dem Fahrer visuell das Eintreten eines Fahrzeuges in den Toten-Winkel Bereich
- Warnt den Fahrer beim Spurwechsel visuell und akustisch



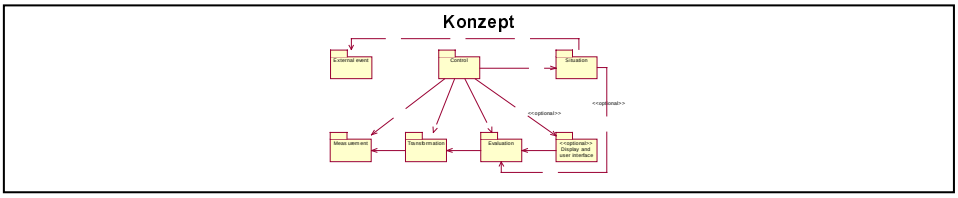
Fallstudie Produktlinien-Technologie
Arbeitsergebnisse: Domänen Analyse

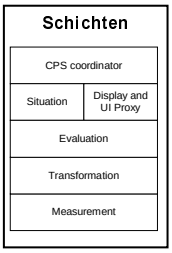


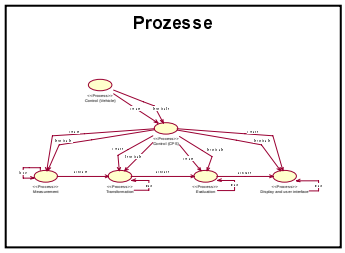
Domänen Analyse
UML Domänen Modell für Objekterkennung

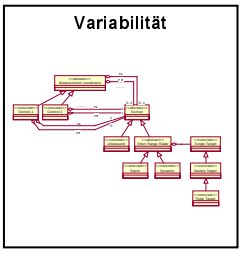


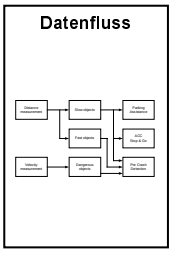
Fallstudie Produktlinien-Technologie
Arbeitsergebnisse Domänen Architektur

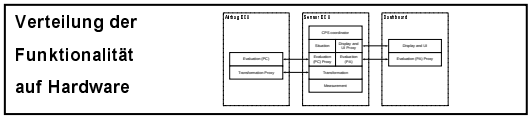


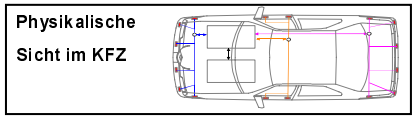












Arbeitsergebnisse Validierung
Beispiele

- „Goal-Question-Metric“ Ansatz für eine empirische Validierung
  1. Sensor sharing to save costs, volume, power, and weight in a car.
     - By counting the available sensors and sensors used by each application, sensor utilization (su) can be computed as:

$$su := \frac{1}{I} \sum_i \frac{n_i}{n}$$

n_i = number of sensors used by application i
n = number of sensors mounted in the car

  2. High reuse is expected to lead to fast product development.
     - Correlation of reuse and time-to-market
     - Reuse measured in LOC
     - Time-to-market measured in days

Fallstudie Produktlinien-Technologie
Vorläufige Ergebnisse

- Stand
  - Ultraschall-Sensoren in der Serienproduktion
  - Radar-Sensoren in der Produktentwicklung
  - Prototypen werden entwickelt und erprobt
- Vorteile
  - Systematische Analyse der Domäne Umfeld-Sensorik
  - Bewusste Entscheidung über den „Scope“ der Produktlinie
  - Frühe Identifikation Gemeinsamkeiten und Unterschiede der Anwendungen (Konzeptphase)
  - Zukünftige Anforderungen und Trends voraussehen und in der Architektur berücksichtigen
  - Wiederverwertung von Software im Sensor und Steuergerät
- Probleme
  - Hohe Anfangsinvestitionen
    - Mitarbeiter schulen
    - Domain Engineering
  - Initiale „Time-to-Market“ länger
  - Schwache Werkzeugunterstützung
  - Innovative Domänen
    - neue Algorithmen
    - keine Markterfahrung

Produktlinien-Technologie
Zusammenfassung

- Produktlinien bei Bosch
  - Hohes Potential für Produktlinien bei der Bosch Kraftfahrzeugausrüstung
  - Bisherige Produktlinien sind historisch gewachsen
  - Architekturbasierte Anwendung des Produktlinien-Ansatzes wird erprobt
  - Kernkompetenz in der Bosch Forschung, zunehmendes Interesse der Geschäftsbereiche
  - Bosch Beteiligung an öffentlich geförderten Projekten
    - PRAISE, ESAPS, CAFE
  - Organisatorische Konsequenzen
- Fallstudie Fahrzeug-Umfeld-Sensorik
  - Domänen Analyse ist aufwendig
  - Kosten- und Bauraumeinsparung für Sensoren und Steuergerät sind
    - *offensichtlich*
    - *zwingend nötig*
  - Andere Vorteile *noch* nicht quantifiziert

Dokumenten Information

Titel:	Gezielte Wiederverwendung durchSoftware-Produktfamilien Vorträge, Diskussionen, Erfahrungsaustausch
Datum:	Dezember 2000
Report:	IESE-090.00/D
Status:	Final
Verteiler:	public