Product Line Engineering (PLE)

Produktlinienentwicklung
Von
Christoph Kuberczyk

Gliederung

1. Was ist PLE?
2. Motivation
3. Entwicklung einer Produktlinie
4. Darstellungen von Produktlinien
5. Nachteile von PLE
6. Erweiterungen von PLE

Was ist PLE?

- Entwicklung einer Familie von Softwareprodukten, aufbauend auf einer Plattform, die die Gemeinsamkeiten („Core Assets“) mehrerer Produkte zusammenfasst
- „Software by Design, not Coding“

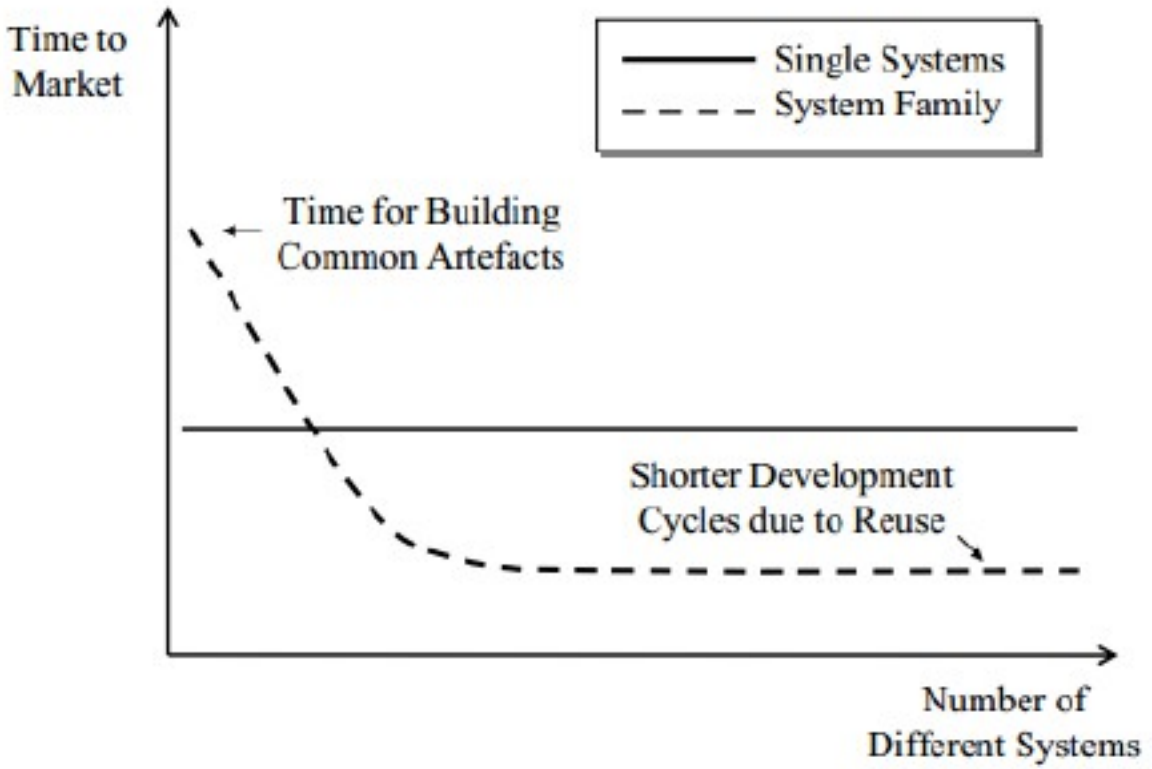
Core Assets sind...

- Architekturen
- Softwarekomponenten
- Domänenmodelle
- Use-Cases
- Testfälle
- usw.

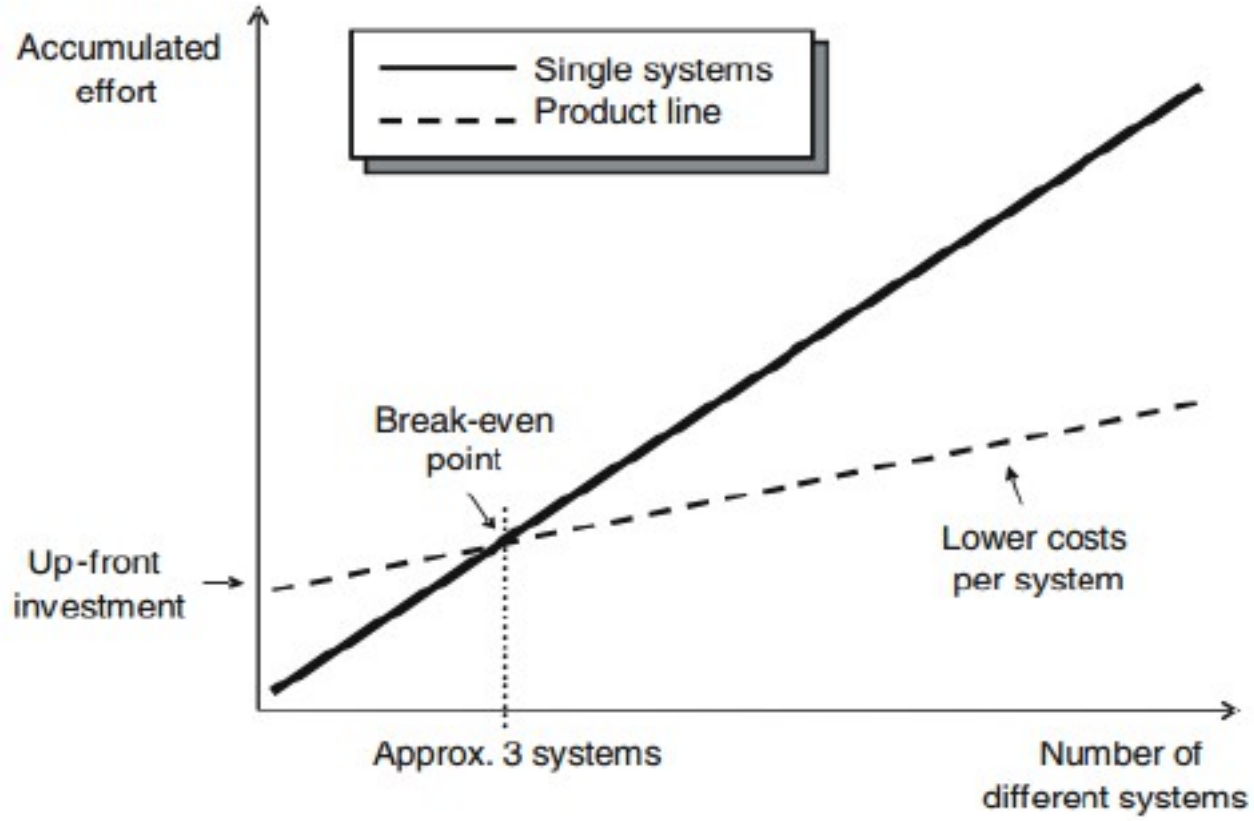
Motivation

- Wiederverwendbarkeit von Software
- Verkürzung der Markteinführungszeit
- Erleichterung der Softwareentwicklung
- Erleichterung des Wartens
- Höhere Kundenzufriedenheit
- Senkung der Entwicklungskosten
- Flexibilität

Vergleich: Time-to-market



Vergleich: Kosten



httpmedia2.giga.de201208samsung_smartphones_patent.jpg

- So könnte eine Familie von Produkten aussehen

Entwicklung einer Produktlinie

Unterscheidung

Domain Engineering (DE):

-„Entwickung für Wiederverwendung“

- Bereitstellung der „Core Assets“(Kernbestandteile) eines jeden Produktes

Application Engineering (AE):

-“Entwickung mit Wiederverwendung“

-Herstellung der konkreten Produkte auf Basis von DE

Domain Engineering

Unterteilt sich in:

1. Product Management
2. Domain Requirements Engineering
3. Domain Design
4. Domain Realization
5. Domain Quality Assurance

Domain Engineering

- 1. Product Management:

  - Bestimmung des Rahmens/Produktportfolios der PL (Scoping). Aufgabe des Managements, da der Rahmen stark von der Marktstrategie abhängt.

- 2. Domain Requirement Engineering:

  - Umfasst alle Aktivitäten, um gemeine und variable Anforderungen zu entwerfen, verhandeln,dokumentieren,validieren,managen

Domain Engineering

- 3. Domain Design:

  -Entwurf der Architektur und der Komponenten
- 4. Domain Realization

  - Implementation/Schaffen der core assets
- 5. Domain Quality Assurance

  - formale Verifikation, statische Analyse
    dynamisches Testen

Application Engineering

1. Application Requirements Engineering
2. Application Design
3. Application Realisation
4. Application Quality Assurance

Application Engineering

- 1. Application Requirement Engineering:

  - konkrete Anforderungen an ein Produkt, Verwendung der Domain Requirements und Nutzen derer Variabilität

- 2. Application Design:

  - Entwerfen der Architektur für konkrete Produkte, aufbauen auf Domain Design

Application Engineering

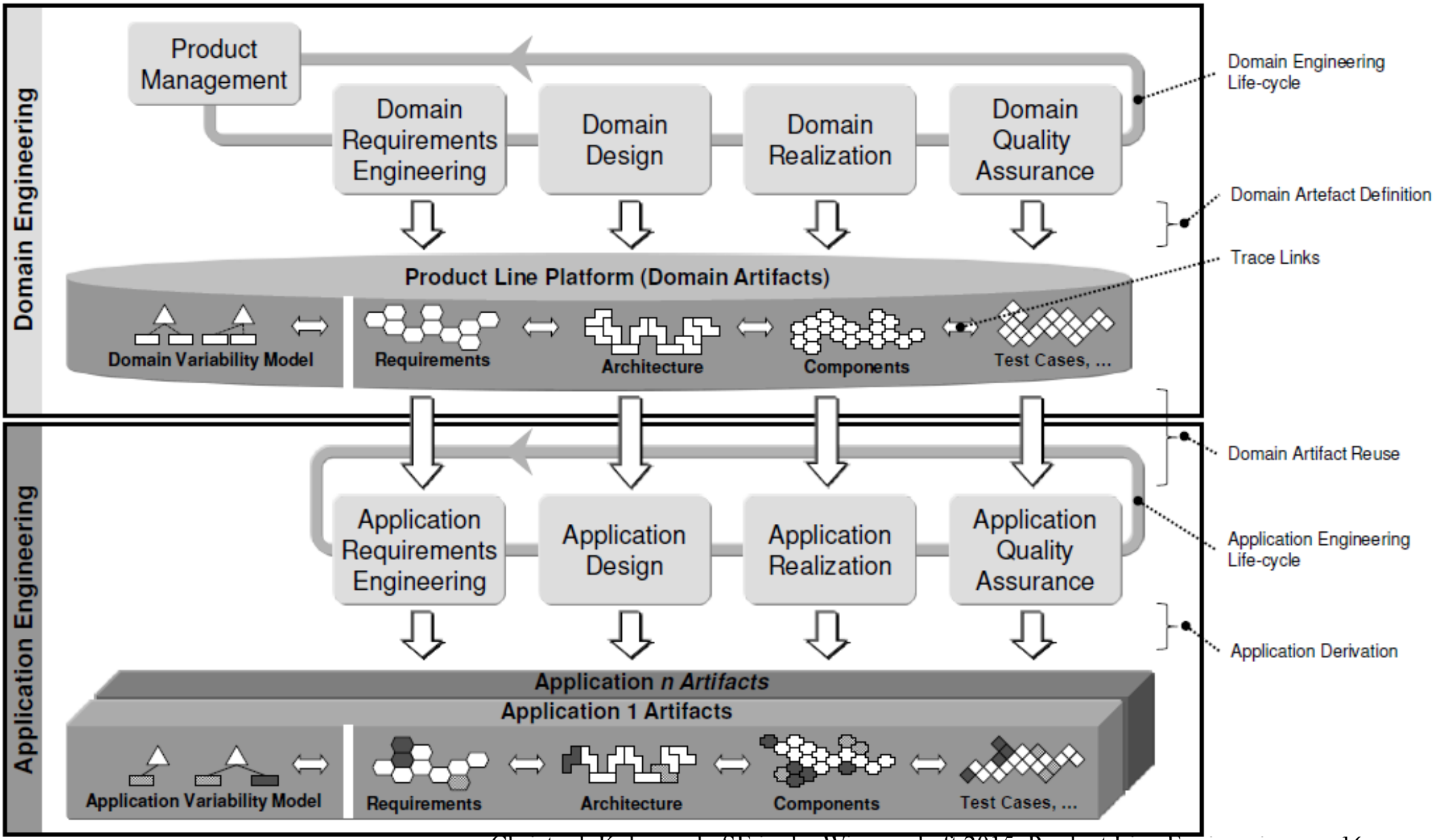
- 3. Application Realization:

  - Konkrete Implementation aufbauend auf dem Application Design

- 4. Application Quality Assurance:

  - Hauptsächlich dynamisches Testen

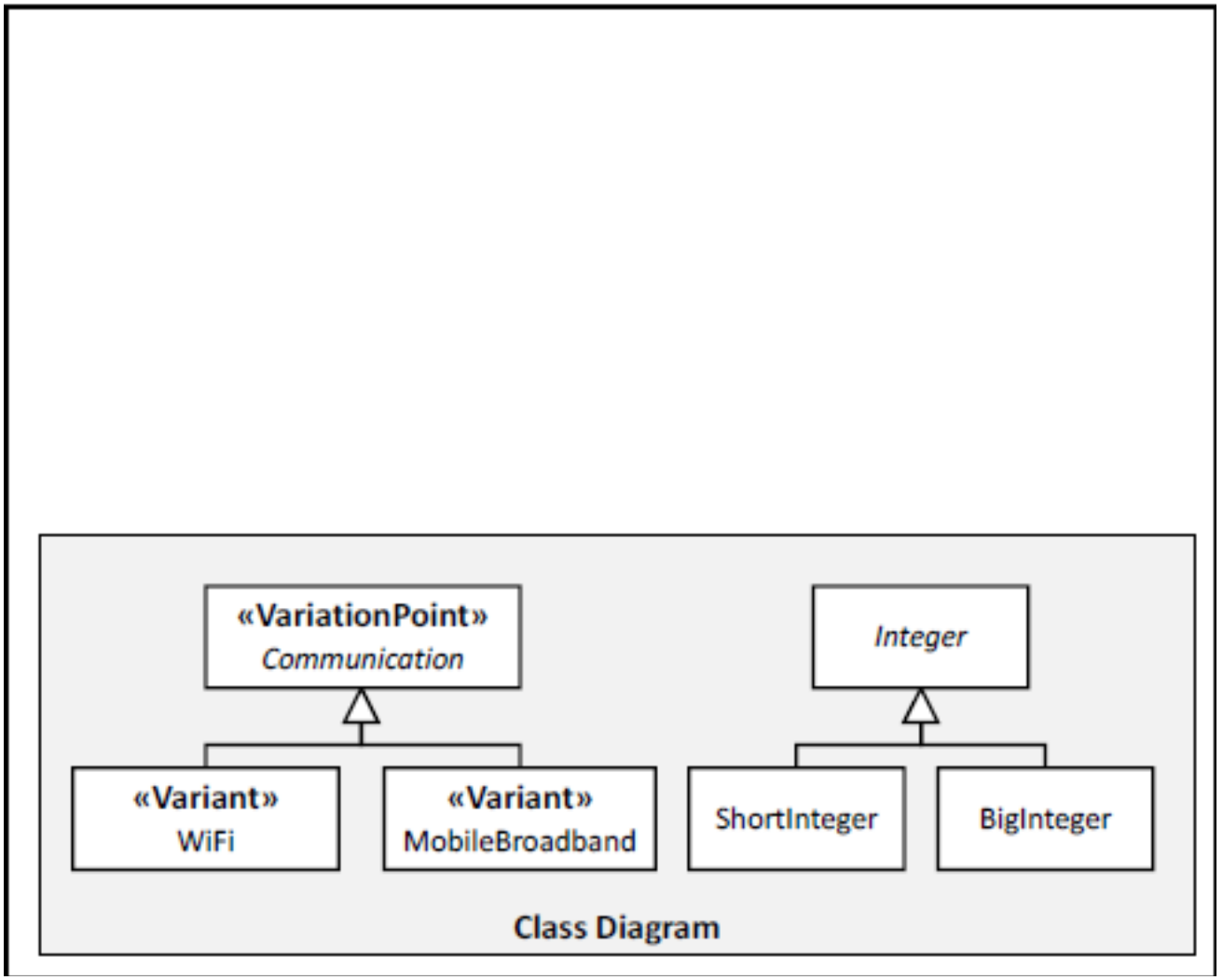
  - Nutzen von Use-Cases aus der Plattform


Domain Engineering
Product Management
Domain Requirements Engineering
Domain Design
Domain Realization
Domain Quality Assurance
Domain Engineering Life-cycle
Domain Artefact Definition
Trace Links
Product Line Platform (Domain Artifacts)
Domain Variability Model
Requirements
Architecture
Components
Test Cases, ...
Domain Artifact Reuse
Application Engineering
Application Requirements Engineering
Application Design
Application Realization
Application Quality Assurance
Application Engineering Life-cycle
Application Derivation
Application n Artifacts
Application 1 Artifacts
Application Variability Model
Requirements
Architecture
Components
Test Cases, ...

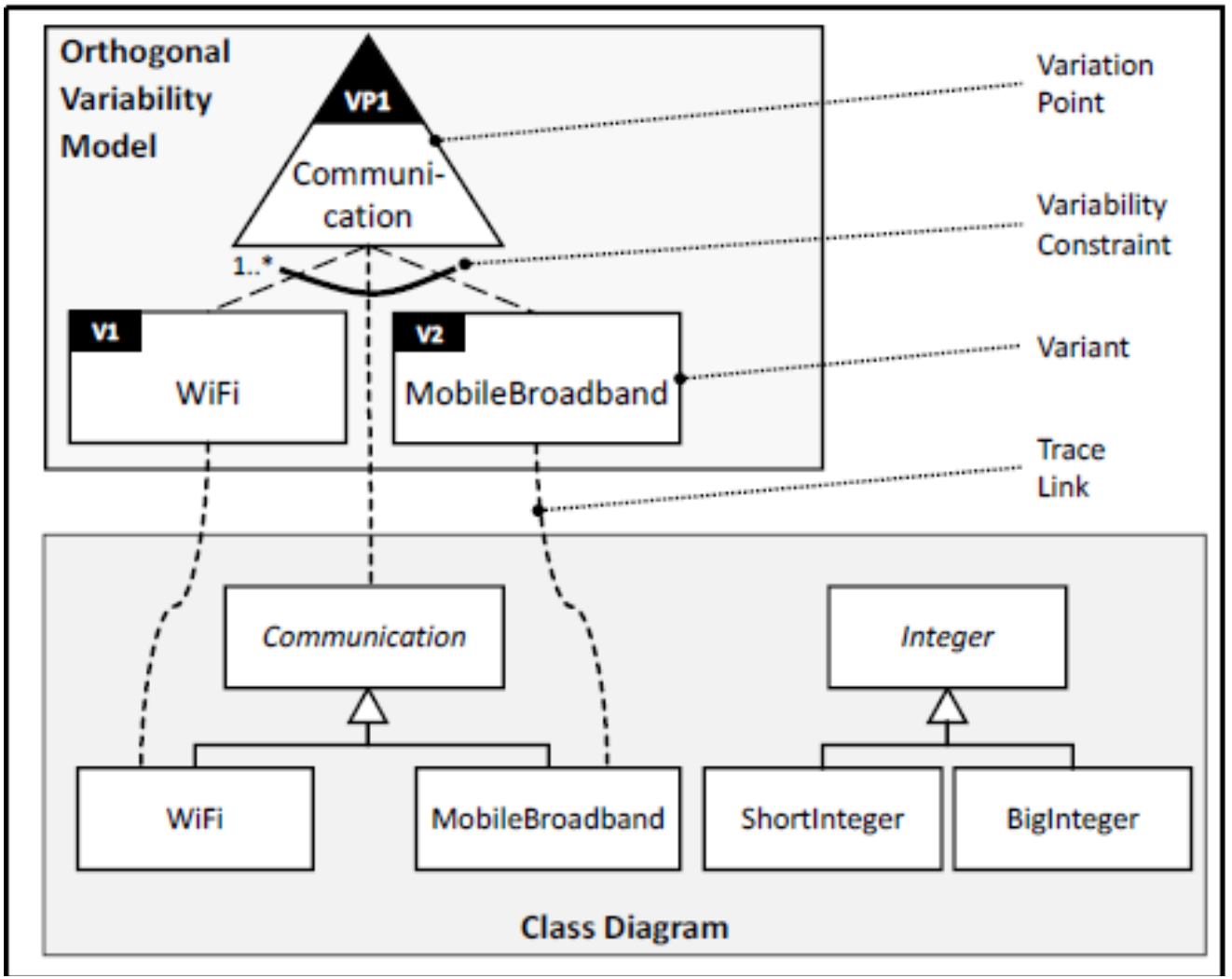
Ansätze

- Proactive approach (Big-Bang): Zunächst Domain Engineering, dann Application Engineering
- Reactive approach: Zunächst nur die wichtigsten Domain Artifacts, die übrigen während der AE
- Reengineering-driven approach: Bestehende Anwendungen und Systeme werden in die Produktlinie integriert

Integrated Variability Model



Orthogonal Variability Model



Nachteile

- Hohe Startkosten
- Begrenzte Ausdrucksmöglichkeiten der Modelle
- Sehr unflexibel (bei unvorhergesehenen Veränderungen der Anforderungen)
- Kann schnell veralten
- Kein Kundenfeedback

httpmedia2.giga.de201208samsung_smartphones_patent.jpg

- So könnte eine Familie von Produkten aussehen

- Können diese Nachteile beseitigt werden?

  - Einbeziehung agiler Methoden in die Entwicklung für höhere Flexibilität

  - Benutzung von DSLs für höhere Ausdrucksfähigkeit der Modelle

PLE und agile Programmierung im Zusammenspiel (APLE)

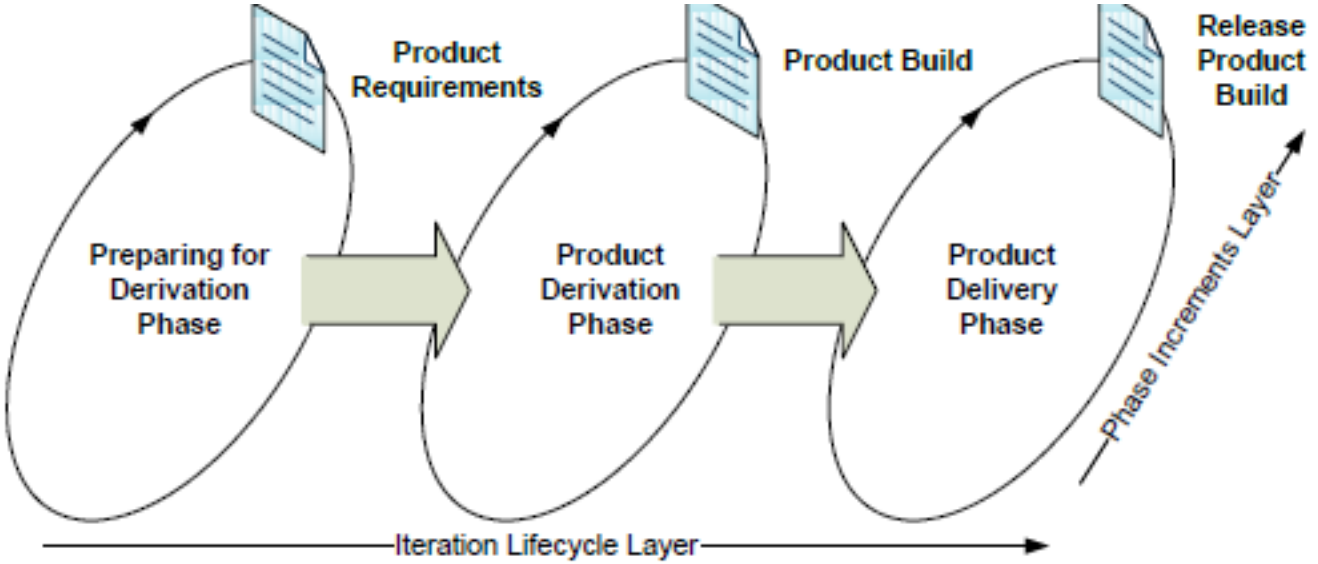
- Gemeinsamkeiten von PLE und agiler Programmierung:
  - geringe Time-to-Market
  - geringe Entwicklungskosten
  - Flexibilität
  - Kundenzufriedenheit als hohes Ziel

Agile Process Model for Product Derivation (A-Pro-PD)

- Integration agiler Methoden beim AE
- Vorteile:

  - Einbeziehung der Kunden in den Gestaltungsprozess

  - geringeres Risiko als beispielsweise bei einer Big-Bang-Herangehensweise


Product
Requirements
Product Build
Release
Product
Build
Preparing for
Derivation
Phase
Product
Derivation
Phase
Product
Delivery
Phase
Phase Increments Layer
Iteration Lifecycle Layer

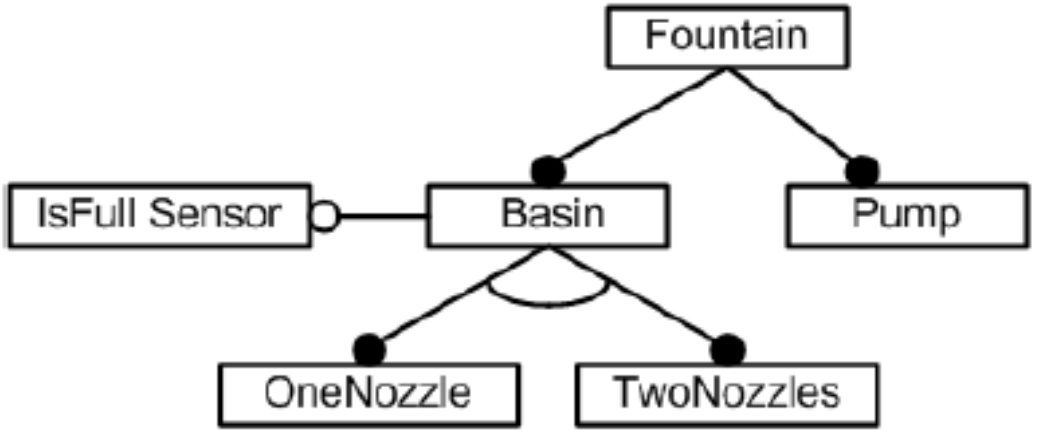
Weitere Möglichkeiten der Einbindung agiler Methoden

- Agile Methoden „als Marktforschung“ während Aufbau der Plattform
- Unter Zuhilfenahme von TDD erweitern der Produktfamilie durch Löschen, Hinzufügen oder Modifikation von „variants“

Einbindung von DSLs

- Produktableitung aufgrund der Baumstruktur eines Variability Models als kontextfreie Grammatik ohne Reduktion erfassen
- Ergänzen dieser Grammatik zu einer DSL

Beispiel (aus „PLE using DSLs“)




```
Fountain -> Basin PUMP
Basin -> ISFULLSENSOR? (ONENOZZLE | TWONOZZLES)
```

Die Grammatik wird transformiert...

```
Fountain -> "fountain" Basin Pump Behavior
Basin -> "basin" IsFullSensor Nozzle*
Behavior -> Rule*

Rule -> "if" Condition "then" Consequence
Condition -> Expression
Expression -> AttrRefExpression | AndExpression |
              GreaterThanExpression | IntLiteral;

AndExpression -> Expression "&&" Expression
GreaterThanExpression -> Expression ">" Expression
AttrRefExpression -> <attribute-ref-by-name>
IntLiteral -> (0..9)*

Consequence -> AttrRefExpression "=" Expression

IsFullSensor: "sensor" ID (full:boolean)?
Nozzle: "nozzle" ID
Pump: "pump" ID (rpm:int)?
```

Beispielprogramm

```
fountain
  basin sensor s
        nozzle n1
        nozzle n2
  pump p
  if s.full && p.rpm > 0 then p.rpm = 0
```

Fazit

- Product Line Engineering sinnvoll bei Herstellung vieler ähnlicher Produkte
- Ist in seiner Reinform für viele Produktlinien allerdings sehr riskant
- Nachteile können durch Modifikationen des Modells bspw. der Kombination mit agiler Programmierung verringert werden

Quellen

- Padraig O´Leary, Fergal McCaffery, Steffen Thiel, Ita Richardson „An Agile Process Model for Product Derivation in Software Product Line Engineering“
- Markus Voelter, Eelco Visser, „Product Line Engineering using Domain-Specific Languages
- Jessica Diaz, Jennifer Perez, Pedro P. Alarcon, Juan Garbajosa „Agile Product Line Engineering – A Systematic Literature Review“
- Christian Berger, Holger Krahn, Holger Rendel, Bernhard Rumpe „Feature-basierte Modellierung und Verarbeitung von Produktlinien am Beispiel eingebetteter Software“
- Andreas Metzger, Klaus Pohl „Software Product Line Engineering and Variability Management: Achievements and Challenges
- Zhonghua He „Einführen einer Softwareproduktlinie“