Betriebssystemtechnik

Operating System Engineering (OSE)

Software Produktlinien

Eingebettete BS Entwicklung heute

Betriebssysteme für eingebettete Systeme

Spezielle Hardware
Spezielle Anwendungen
→
Markt mit > 100 RTOS
> 50% Eigenentwicklungen

- „das Rad wird neu erfunden“
  - auch die selben Fehler werden wiederholt
- oftmals bietet **ein** BS Hersteller **mehrere** Systeme an
  - mit getrennter Code-Basis
  - getrieben durch die speziellen Anforderungen seiner Kunden

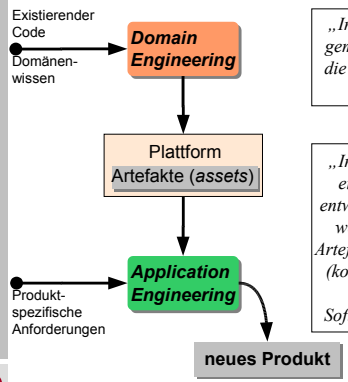
Betriebssystem-Produktlinien

Die Idee der Software-Produktlinien ist die ***„organisierte Wiederverwendung“*** [1].

- **Pro**
  - Systemkomponenten sind besser erprobt, d.h. bessere Qualität
  - kürzere Entwicklungszeiten, d.h. geringere Kosten
- **Contra**
  - Initialer Mehraufwand
  - Konsistenzwahrung der aus der Produktlinie abgeleiteten Produkte
  - Komplexere Verantwortlichkeiten
  - Änderungen am Entwicklungsprozess

Organisierte Wiederverwendung

aktive Gestaltung einer gemeinsamen Plattform
für aktuelle und künftige Produkte



*„Im **Domain Engineering** werden die gemeinsamen und variablen **Artefakte**, die Bestandteile der **Plattform** werden, entwickelt.“*
[1]

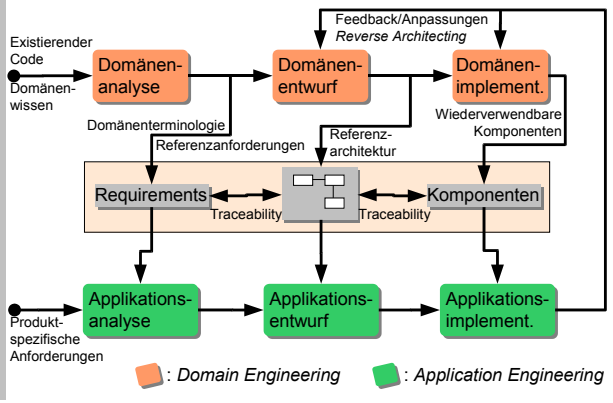
*„Im **Application Engineering** werden einzelne **Produkte** der Produktlinie entwickelt bzw. abgeleitet. Die Produkte werden so weit wie möglich aus den Artefakten der Plattform zusammengefügt (konfiguriert), so dass nur in geringem Maße produktspezifische Softwareentwicklung notwendig wird.“*
[1]

Was ist eine Domäne?

„An area of knowledge“ [2]

- abgegrenzt
- enthält Konzepte und eine Terminologie
- enthält Wissen die Prozesse zum Bau konkreter Systeme
- definiert durch alle, die ein Interesse an der Domäne haben
  - Manager, Marketing, Entwickler, Hersteller, Vertragspartner, Standardisierungsgremien, Investoren, Kunden, Endnutzer, ...
- Beispiele:
  - Börseninformationssysteme der MONEY MAKER AG
  - Motorsteuerungssoftware für Diesel Motoren
  - AVR Wetterstationsanwendungen des Lehrstuhls 4
  - Betriebssysteme für Mindstorm Roboter

Software-Produktlinienentwicklung



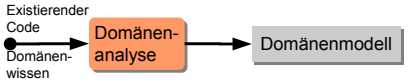
Referenzprozess für die Software-Produktlinienentwicklung [1]

Domain Engineering

...ein drei-phasiger Prozess.

- **Domänenanalyse**
  - relevante Informationen sammeln (existierende Systeme, *Manuals*)
  - Bestimmung und Abgrenzung der Domäne
  - Analyse gemeinsamer und unterschiedlicher **Merkmale**
  - Erstellung eines **Domänenmodells**
- **Domänenentwurf**
  - Entwurf einer Architektur für eine Familie von Systemen
  - Entwurf des Prozesses zum Bau konkreter Systeme
- **Domänenimplementierung**
  - Umsetzung der Architektur
  - Erstellung wiederverwendbarer (ggf. variabler) Komponenten

Domänenanalyse



- **Domänenabgrenzung**
  (*domain scoping*)
  - Auswahl und Aufbereitung von Wissen
  - Festlegung einer im Hinblick auf Wiederverwendung erfolgversprechenden Domänenabgrenzung
- **Domänenmodellierung**
  (*domain modeling*)
  - Auswertung der Wissenssammlung, Taxonomien, u.s.w.
  - Erstellung des Domänenmodells als Ergebnis

Das Domänenmodell (1)

*„A domain model is an explicit representation of the **common** and the **variable** properties of the system in a domain, the semantics of the properties and domain concepts, and the dependencies between the variable properties.“*

[2]

Das Domänenmodell (2)

...ist das Ergebnis der Domänenanalyse.

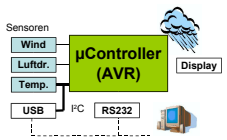
- **Domänendefinition** legt den Scope fest:
  - durch Beispiele, Gegenbeispiele
  - durch Regeln, die bestimmte Systeme oder Merkmale ein- oder ausschließen
- **Domänenlexikon** definiert das Vokabular der Domäne
- **Konzeptmodelle** beschreiben wichtige Konzepte
  - Anwendung geeigneter Formalismen, z.B. Klassen-, Interaktions-, Zustands-, Entity-Relationship-, Datenflussdiagramme, ...
  - sonst textuelle Beschreibungen
- **Merkmalmodelle** definieren eine Menge wiederverwendbarer und konfigurierbarer Anforderungen an die Systeme der Domäne.

Domänendefinition - Beispiel

(stark vereinfacht und verkürzt!)

Domänendefinition: **Lehrstuhl 4 AVR-Wetterstationssoftware**

- Die Domäne umfasst Software für die rechts dargestellte modulare Hardware-plattform. Zusätzlich soll zukünftig auch ein graphisches LCD unterstützt werden.
- Die separat beschriebenen Anwendungs-szenarien „Thermometer“, „Outdoor“, ..., sollen mindestens abgedeckt werden.
- Jede Wetterstationssoftware wird im Flash-Speicher des Controllers ausgeliefert und soll später nicht mehr geändert werden.
- ...



Domänenlexikon - Beispiel

(stark vereinfacht und verkürzt!)

Domänenlexikon: **Lehrstuhl 4 AVR-Wetterstationssoftware**

- **PC Verbindung:** Optionaler Kommunikationskanal zu einem PC, der zur Übertragung von Wetterdaten oder Debug Informationen genutzt werden kann.
- **Sensor:** Teil der Wetterstationshardware zur Messung eines bestimmten Wetterparameter (z.B. Temperatur oder Luftdruck).
- **Aktor:** Teil der Wetterstationshardware, der die gesammelten Wetterdaten weiterverarbeitet (.z.B. LC Display)
- **XML Protokoll:** Einfaches auf XML basierendes Datenformat zum Transfer der Wetterdaten über eine *PC Verbindung*.
- ...

Konzeptmodelle - Beispiel

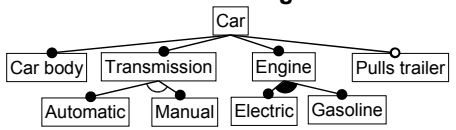
(stark vereinfacht und verkürzt!)

Konzeptmodelle: **Lehrstuhl 4 AVR-Wetterstationssoftware**

- **XML Protokoll:** Folgende DTD beschreibt das XML Format zur Übertragung der Wetterdaten zum PC:
  ```
  <!ELEMENT weather ...>
  ...
  ```
- ...

Merkmalmodelle (*feature models*) [3]

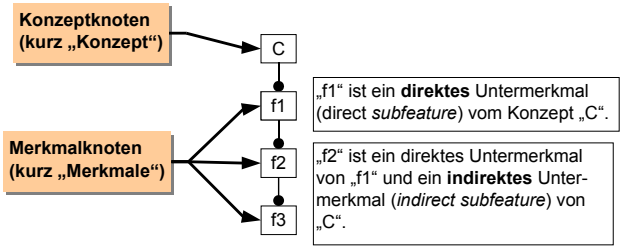
- erlauben, die Gemeinsamkeiten und Unterschiede der Systemvarianten explizit zu machen
  - drücken Konfigurierbarkeit aus
  - sind unabhängig von Implementierung und Architektur
- bestehen aus einem **Merkmaldiagramm:**



- beinhalten noch **weitere Informationen**:
  - kurze Beschreibung und Begründung jedes Merkmals, Interessenten, Beispielsysteme, Einschränkungen, Verfügbarkeit und Bindung, *Open/Closed* Attribut, Priorität

Merkmaldiagramme (*feature diagrams*)

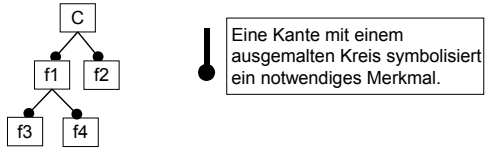
...sind gerichtete azyklische Graphen. Man spricht auch bei Merkmaldiagrammen von **Eltern- und Kindknoten** bzw. -merkmalen. Sie beschreiben die unterschiedlichen möglichen **Instanzen eines Konzepts**.



Es gibt eine Reihe verschiedener **Arten von Merkmalen**.

Notwendige Merkmale

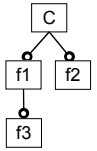
... (*mandatory features*) gehören zur Beschreibung eines Konzepts, wenn ihr Elternknoten zur Konzeptbeschreibung gehört. Der Wurzelknoten (das Konzept selbst) gehört immer zur Konzeptbeschreibung.



Da alle Merkmale von „C“ notwendig sind, gibt es hier nur eine Variante: { C, f1, f2, f3, f4 }

Optionale Merkmale

... (*optional features*) können zur Beschreibung eines Konzepts gehören, (nur!) falls auch der Elternknoten zur Konzeptbeschreibung gehört.

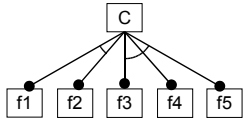


Eine Kante mit einem unausgemalten Kreis symbolisiert ein optionales Merkmal.

Eine Instanz von „C“ kann folgende Beschreibungen haben: {C}, {C, f1}, {C, f1,f3}, {C, f2}, {C,f1, f2} oder {C,f1,f2,f3}, aber **NICHT** {C,f3}

Alternative Merkmale

... (*alternative features*) treten immer in Gruppen auf. Wenn der Elternknoten einer Gruppe zur Konzeptbeschreibung gehört, ist genau eines der alternativen Merkmale auch Teil der Beschreibung.

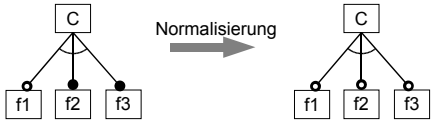


Ein unausgemalter Bogen kennzeichnet eine Gruppe alternativer Merkmale.

Eine Instanz von „C“ kann folgende Beschreibungen haben: {C, f1,f3}, {C, f1,f4}, {C, f1,f5}, {C,f2,f3}, {C,f2,f4} oder {C,f2,f5}

Optionale alternative Merkmale

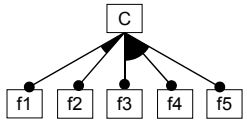
... (*optional alternative features*) unterliegen einer Normalisierung. Daher sind entweder alle oder kein Merkmal einer Gruppe alternative Merkmale optional.



Eine Instanz von „C“ kann folgende Beschreibungen haben: {C}, {C,f1}, {C,f2} oder {C,f3}

Oder-Merkmale

... (or-*features*) treten immer in Gruppen auf. Wenn der Elternknoten einer Gruppe zur Konzeptbeschreibung gehört, ist auch jede nicht leere Untermenge der Gruppe Teil der Beschreibung.

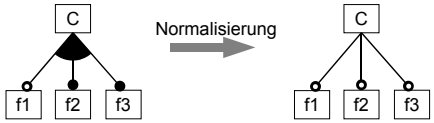


Ein ausgemalter Bogen kennzeichnet eine Gruppe von Oder-Merkmalen.

„C“ hat zwei Gruppen mit Oder-Merkmalen (f1,f2 und f3,f4,f5). Damit können $(2^2\text{-}1)*(2^3\text{-}1)$ oder 21 verschiedene Beschreibungen abgeleitet werden.

Optionale Oder-Merkmale (?)

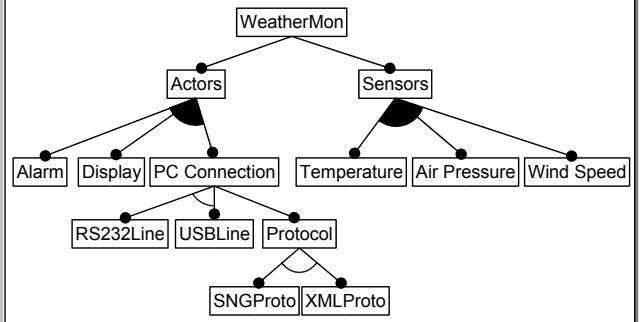
... (*optional or-features*) unterliegen einer Normalisierung. Daher werden optionale Oder-Merkmale nicht verwendet.



Effektiv sorgt ein optionales Oder-Merkmal dafür, das die Menge der Merkmale der Gruppe auch leer sein darf.

Merkmaldiagramm - Beispiel

Merkmaldiagramm: **Lehrstuhl 4 AVR-Wetterstationssoftware**



Weitere Informationen zu Merkmalen

...runden das Merkmalmodell ab.

- **Beschreibung:**
  - kurze Beschreibung der Semantik wie: „Alarm – Eine Wetterstation mit Alarm kann Wetterdaten auf Grenzwerte überprüfen und bei Über- bzw. Unterschreitung einen Alarm signalisieren“
  - ggf. geeignete Formalismen nutzen (Pseudocode, Gleichungen,...)
- **Begründung:**
  - Warum ist das Merkmal im Modell, wann sollte es gewählt werden? Z.B. Verweis auf ein Anwendungsszenario.
- **Interessenten:**
  - Welche Personen/Programme benötigen das Merkmal?
  - Wichtige Frage, um unnötige Variabilität zu vermeiden.

Weitere Informationen zu Merkmalen

- **Beispielsysteme:**
  - ggf. bekannte Systeme, die ebenfalls das Merkmal aufweisen, nennen
- **Einschränkungen:**
  - Beziehungen zwischen Merkmalen, die nicht durch das hierarchische Merkmaldiagramm erfasst werden können, z.B. „Debugging“ **erfordert** „PC Verbindung“, „X“ **hat einen Konflikt mit** „Y“
- **Verfügbarkeit und Bindung:**
  - Wann, wo und durch wen kann das Merkmal benutzt werden?
  - Ist das Merkmal statisch oder dynamisch gebunden?

Weitere Informationen zu Merkmalen

- **Open/Closed Attribut:**
  - gibt an, ob weitere variable Untermerkmale erwartet werden
  - z.B. könnte die Liste der „Sensoren“ der Wetterstation u.U. mit dem „Open“ Attribut versehen werden.
  - kann im Merkmaldiagram durch „...“ gekennzeichnet werden
- **Prioritäten:**
  - relevant z. B. für die Implementierungsreihenfolge
  - ggf. können Konflikte mit Einschränkungen auftreten

Der Prozess der Merkmalmodellierung

> *„Feature Modeling is the activity of modeling the common and the variable properties of concepts and their interdependencies and organizing them into a coherent model referred to as feature model.“*

[2]

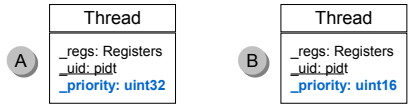
- Welche „**Merkmalquellen**“ gibt es?
- Welche **Strategien** kann man anwenden?
- Welche **generellen Schritte** gehören zur Merkmalmodellierung?

Merkmalquellen

- Existierende und potenzielle Interessenten
- Domänenexperten und Literatur
- Existierende Systeme
- Bereits existierende Modelle
- Modelle, die während der Entwicklung entstehen (Feedback)

Strategien zur Merkmalidentifizierung

- Domänenterminologie auf Variabilität untersuchen
  - Beispiel: „Die PC Verbindung kann per RS232 oder USB realisiert sein.“
  - auch Implementierungseigenschaften werden Merkmale, falls sie für Interessenten von Bedeutung sind
    - Beispiel: „auf Plattform X soll auf die Benutzung von Gleitkommadatentypen möglichst verzichtet werden“
- Konzeptmodelle unterschiedlicher Quellen vergleichen
  - Beispiel:



Strategien zur Merkmalidentifizierung

- „*feature starter sets*“ verwenden
  - Betrachtung des Systems aus bestimmten Blickwinkeln
  - Ausnutzung bekannter Merkmale
  - Beispiel: Eine Wetterstationssoftware benötigt Fehlerbehandlung, da sich z.B. defekte Hardware unerwartet verhalten könnte (**Blickwinkel**). Fehler werden **im Allgemeinen** durch die Überprüfung von Vor- und Nachbedingungen festgestellt. Es können auch Invarianten...“
  - Andere typische Blickwinkeln sind „Speicherverwaltung“, „Synchronisation“, „Persistenz“, ...
- in allen Entwicklungsphasen nach Merkmalen suchen
  - manchmal ergibt sich Variabilität erst bei der Betrachtung von Architektur- oder Implementierungsmodellen

Strategien zur Merkmalidentifizierung

- Merkmale vorausschauend identifizieren
  - mehr Merkmale identifizieren als implementiert werden sollen
  - erleichtert unerwartete Erweiterungen
  - für die durch die Implementierung unterstützte Domäne erfolgt eine weitere Abgrenzung

Ablauf der Merkmalmodellierung

Folgende Aktivitäten werden in kleinen, schnellen Schritten zyklisch durchgeführt („*micro-cycles*“):

1. Gemeinsamenkeiten von Instanzen notieren
2. Unterschiede zwischen Instanzen notieren
3. Merkmale in Merkmaldiagramme organisieren
4. Merkmalkombinationen u. -interaktionen analysieren
   - Abhängigkeiten und Konflikte ermitteln
   - neue Merkmale finden, auf die man erst durch Analyse der Kombinationen kommt
5. weitere Informationen zu Merkmalen aufzeichnen

Ausblick

- Durchführung einer Domänenanalyse als Übung
  - pure::variants als unterstützendes Werkzeug
- Referenzarchitektur für die Komponenten einer eingebetteten Betriebssystem-Produktlinie
  - „Domänenentwurf“
- Untersuchung verschiedener Techniken zur Umsetzung von Variablität in der Implementierung der Komponenten
  - „Domänenimplementierung“

Literatur

[1] G. Böckle, P. Knauber, K. Pohl, K. Schmid (Hrsg.). *Software-Produktlinien*. dpunkt.verlag, 2004. ISBN 3-89864-257-7.

[2] K. Czarnecki und U.W. Eisenecker. *Generative Programming – Methods, Tools, and Applications*. Addison-Wesley, 2000. ISBN 0-201-30977-7.

[3] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. *Feature-oriented Domain Analysis (FODA) Feasibility Study*. Technical Report, CMU/SEI-90-TR-2, 1990.