Gedanken zur C**i**AO Entwurfsmethodik

Daniel Lohmann
lohmann@cs.fau.de

13. Dezember 2004

1 Über dieses Dokument

Bei den Überlegungen zu C**i**AO stoßen wir immer wieder auf das Problem uns über unsere Vorstellungen von Methoden, Begriffen und Zielen zu unterhalten und die Zusammenhängen zwischen diesen klar zu sehen. Dieses liegt zu einem nicht unerheblichen Teil an den noch unklaren Begriffen und Konzepten.
Dieses Dokument soll, im Kontext von C**i**AO, sowohl zur Begriffsklärung dienen als auch grundlegende Konzepte und Vorgehensweisen dokumentieren. Es ist als „lebendes Dokument“ gedacht, soll also als Diskussionsgrundlage dienen. Mit dem Ziel, dass der Inhalt stetig ergänzt und geschärft wird. Es mag daher zunächst noch als eine recht lose Sammlung von Ideen, Definitionen und Feststellungen erscheinen.[1]

2 Modellierung mit Merkmalsmodellen

2.1 Einleitung

Dieses Kapitel soll noch einmal in Erinnerung rufen, was sich hinter Begriffen wie *Domäne*, *Merkmalsmodell*, *Merkmalskonfiguration* etc. eigentlich verbirgt und wie die Begriffe zusammenhängen. Es soll ferner die Grenzen der Methodik und mögliche Alternativen aufzeigen.

2.2 Merkmalsmodelle

Ein Merkmalsmodell beschreibt ein *Konzept*. Ein Konzept ist ein Begriff, der eine einzelne, abstrakte, zu modellierende Domäne benennt. Es ist abstrakt, da es anhand von *generischen Merkmalen* (=Eigenschaften und Attribute) beschrieben wird, die optional, zwingend oder alternativ sein können. Damit beschreibt ein Merkmalsmodell eine *wohldefinierte und endliche* Menge von Ausprägungen (=Konfigurationen) und zwar anhand der Gemeinsamkeiten und Unterschiede dieser Ausprägungen. Jede Ausprägung stellt eine konkrete *Instanz* des Konzepts dar. Anders gesagt: Ein Merkmalsmodell beschreibt eine Familie, jede Instanz ist ein Familienmitglied.

[1] Will sagen: Nicht erschrecken, wenn es zunächst wie ein Müsli-Paper aussieht :->

2.3 Merkmale

Ein Merkmal ist eine wesentliche Eigenschaft oder ein wichtiges Attribut eines Konzeptes. Wesentlich bzw. wichtig sind vor allem solche Merkmale, durch die sich die Abgrenzung der einzelnen Familienmitglieder voneinander ergibt. Merkmale müssen begrifflich der Domäne entstammen, die durch das Konzept beschrieben werden soll. Ein Merkmalsmodell ist damit inheränt auf eine Domäne beschränkt.

2.4 Instanziierung

Eine Konzeptinstanz wird dargestellt durch eine Konfiguration, d.h. eine *vollständige* und *gültige* Menge von Merkmalsselektionen. Der Prozess der *Selektion* von Merkmalen (=Bindung von Merkmalen) kann sich über mehrere Phasen erstrecken, er kann sich für einzelne Merkmale bis zur Laufzeit verzögern. Unvollständige Merkmalsselektionen führen zu *partiellen Konzeptinstanzen*. Die durch eine partielle Konzeptinstanz beschriebene Menge an Konfigurationen ist eine (echte) Teilmenge des Konzepts.

3 Probleme

3.1 Part-of Beziehungen

Die Beschränkung eines Merkmalsmodells auf eine Domäne bereitet oft Verständnisschwierigkeiten, zumal die resultierenden Instanzen wiederum Merkmale einer anderen Domäne sein können. So beschreibt ein Merkmalsmodell für das Konzept Container eine Familie von Containerabstraktionen anhand von Merkmalen wie Zugriffscharakteristik, Speicherplatzbedarf etc. Ein Merkmalsmodell für das Konzept Containerbibliothek enthält hingegen Merkmale wie Vector oder Liste, die jeweils *benannte Instanzen* des Container-Konzeptes sind. Nichtsdestotrotz sind es zwei verschiedene Domänen, die hier beschrieben werden: Was dem einen seine Merkmale, sind dem anderen seine Instanzen. Es ist also prinzipbedingt nicht möglich ein Merkmalmodell aufzustellen, in dem eine Container-*Bibliothek* anhand von Container-*Merkmalen* beschrieben wird.

Feststellung 1: Ein Merkmalsmodell beschreibt und benennt genau eine Domäne und ist begrifflich auf diese Domäne beschränkt. Merkmalsmodelle sind niemals domänenübergreifend.

Feststellung 2: Die Merkmale einer Domäne (eines Merkmalsmodells) sind häufig (benannte) Ausprägungen anderer Konzepte, die ihrerseits wiederum durch Merkmalsmodelle beschrieben sein können.

3.2 Kardinalität von Konzepten

Der Hauptgrund für die oben geschilderten Schwierigkeiten ist das Problem der *Kardinalitäten*. Ein Merkmalsmodell beschreibt mögliche Ausprägungen eines

Konzeptes, macht jedoch keinerlei Aussage über die mögliche und tatsächliche *Koexistenz* verschiedener Instanzen im Zielsystem. Das Konzept Container kann schon allein deshalb keine Containerbibliothek beschreiben.
Feststellung 3: Ein Merkmalsmodell enthält keine Aussage über die Kardinalität eines Konzeptes, d.h. über die mögliche oder tatsächliche Anzahl von (verschiedenen) koexistierenden Konfigurationen des Konzepts im Zielsystem.

Ist die Kardinalität auf der Modellierungsebene undefiniert, so ist sie bezogen auf das Zielsystem doch ein wichtiges Attribut. Tatsächlich bestimmt wird die *mögliche Kardinalität* (= obere Schranke der Kardinalität) durch die verwendeten *Implementierungstechniken* für die Variabilität auf Merkmalsebene. So hat ein Merkmal, das über einen (statischen) Aspekt implementiert wird, implizit die Kardinalität eins, da ein Aspekt auf das Gesamtsystem wirkt und somit nur entweder „vorhanden“ oder „nichtvorhanden“ sein kann. Alle Instanzen des Konzepts im Zielsystem stimmen somit bezüglich dieses einen Merkmals überein. Ein anderes Merkmal, das über Typpolymorphismus implementiert wird, hat hingegen eine mögliche Kardinalität größer als eins; sie entspricht der Anzahl der im Zielsystem vorhandenen Implementierungen des Typs. Die mögliche Kardinalität des Konzeptes ergibt sich dann als das Produkt der möglichen Kardinalitäten der einzelnen Merkmale.
Feststellung 4: Die mögliche Kardinalität eines Konzepts ist das Produkt der möglichen Kardinalitäten seiner Merkmale.

Feststellung 5: Die mögliche Kardinalität eines Merkmals ist bestimmt durch die für das Merkmal (bzw. seine Variabilität) verwendete Implementierungstechnik.

4 Implementierung von Merkmalen

4.1 Part-of Beziehungen

5 Grenzen der Konfigurierbarkeit

5.1 Einleitung

Bezüglich der Konfiguration des CiAO Systems stehen wir of vor der Schwierigkeit, uns darüber klar zu werden, was eigentlich konfigurierbare Merkmale sind und was nicht. Insbesondere „weiche“ Eigenschaften wie Laufzeitverhalten, Robustheit oder Ressourcenbedarf wirken schwer fassbar. Im Folgenden soll versucht werden, Eigenschaftsklassen aufzustellen und eine Idee skizziert werden, wie ein (domänenspezifischer) Konfigurationsprozess aussehen könnte.

5.2 Harte und weiche Eigenschaften

Wir unterscheiden *harte* und *weiche* Eigenschaften des (konfigurierten) Systems.

Weiche Eigenschaften sind keine Eigenschaften einzelner Komponenten und Subsysteme, sondern ergeben sich erst auf der Ebene des Gesamtsystems durch das *Zusammenspiel* aller Komponenten. Es handelt sich also um *emergente Eigenschaften*, die potentiell durch jede einzelne Komponente bzw. ihre Konfiguration beeinflusst werden. Beispiele für weiche Eigenschaften sind:

- Größe
- Laufzeitverhalten
- Robustheit

Dagegen ist Determinismus eher keine weiche Eigenschaft, da ein System genau dann deterministisch ist, wenn alle Subsysteme sich deterministisch verhalten. Damit lässt sich eine derarige globale Forderung bereits auf der Ebene der Subsysteme konfigurieren und ist somit keine echt emergente Eigenschaft. Eine realistische Berechnung der Größe ist hingegen erst auf der Ebene des konfigurierten Gesamtsystems möglich. Größe ist eine vergleichsweise gutmütige weiche Eigenschaft. Zum einen gibt es hier die Möglichkeit, sie durch externe Tools (Compiler/Linker) genau zu berechnen. Zum anderen ist Größe wohldefiniert und ordinalskaliert, wodurch die sich Forderungen leicht überprüfen und Varianten leicht bewerten lassen. Bei Eigenschaften wie Robustheit oder Laufzeitverhalten ist dieses i.a. nicht ohne Weiteres möglich. Definitionen und Skalen für derartige Begriffe sind, wenn es sie überhaupt gibt, oft domänenspezifisch; eine automatische Überprüfung und Bewertung ist oft nicht möglich. In diesen Fällen muss die Abschätzung dann durch einen menschlichen Experten oder ein domänenspezifsches Tool durchgeführt werden.

Harte Eigenschaften sind *konfigurierbare* Eigenschaften, die zu *überprüfbaren/sicheren Aussagen* über das konfigurierte System führen. Sie finden sich erkennbar in der Komponentenimplementierung wieder - allein schon deshalb, weil wir sie vorher irgendwie implementiert haben müssen. Harten CiAO-Eigenschaften liegen damit bewusste Designentscheidungen zugrunde. Beispiele für harte Eigenschaften sind:

- CPU-Typ
- Scheduling-Strategie
- Synchronisations-Mechanismus

6 Der Konfigurationsprozess

6.1 Standardkonfigurierung

Die grundlegende Idee ist, die eigentliche Systemauswahl als einen mehrstufigen Prozess aufzufassen:

Abbildung 1: Prozess der Standardkonfigurierung

1. **Konfiguration** anhand der harten BS-Eigenschaften (manuell und/oder durch Anwendungsanalyse). Das Resultat ist eine Betriebssystem-Konfiguration. Von einigen Komponenten (insbesondere im Hinblick auf weiche Eigenschaften) gibt es jedooch mehrere und aus Sicht der harten Merkmale „gleichwertige“ Implementierungen. Deshalb wird die Konfiguration im allgemeinen *nicht* ausreichen, um die Implementierungskomponenten bereits vollständig zu konfigurieren. Aus der Betriebssystem-Konfiguration leitet sich somit, da unterspezifiziert, *eine Menge* von Komponenten-Konfigurationen (Varianten) ab.

2. **Finetuning** der Komponentenkonfigurationen und Forderung weiterer Komponenten (bei Bedarf).

3. **Bewertung** der weichen BS-Eigenschaften der Varianten. Eventuell müssen für die erforderlichen Daten weitere Tools aufgerufen werden, die z.B. die Varianten übersetzen, darauf statische und dynamische Analysen durchführen oder ähnliches.

4. **Filterung** anhand der Bewertungskriterien und **Auswahl** einer Variante. (Interaktiv mit Hilfe eines speziellen Browsers.)

6.2 Domänenspezifische Konfigurierung

In der oben skizzierten Form bezieht dieser Prozess noch keine Konzepte und Begrifflichkeiten aus der Anwendungsdomäne ein. Das heißt, wir konfigurieren und bewerten hier noch „reine“ C**i**AO-Systeme. Die weitergehende Idee ist nun, diesen Prozess domänenspezifisch zu „klammern“:

1. **Domänenspezifische Konfigurierung** mit Hilfe eines Wizards / Merkmalmodells, durch das die Auswahl anhand von harten Eigenschaften aus der Anwendungsdomäne ermöglicht wird. Das Ergebnis ist eine BS-Konfiguration und eventuell eine (Teil-)Komponenten-Konfiguration.

   (a) **Konfiguration** bei Bedarf, wie oben.

   (b) **Finetuning** bei Bedarf, wie oben

   (c) **Bewertung** wie oben.

Abbildung 2: Prozess der domänenspezifischen Konfigurierung

2. **Domänenspezifische Bewertung** der weichen Eigenschaften aus der Anwendungsdomäne.
3. **Filterung** anhand der Bewertungskriterien und **Auswahl** einer Variante.