Spezifikation von Fachkomponenten mit UML 2.0

Jörg Ackermann
Universität Augsburg

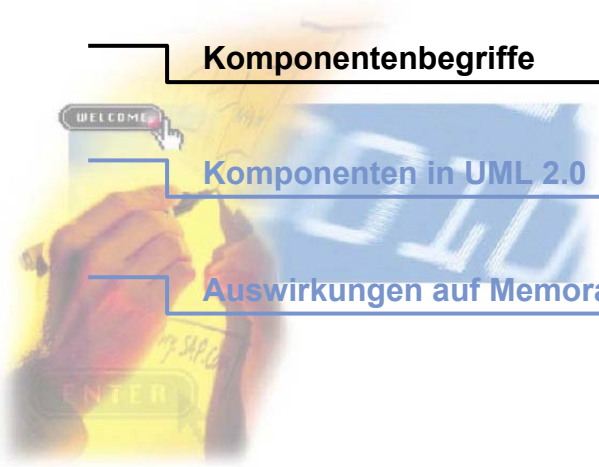
Einleitung

- **UML 2.0 bietet deutlich bessere Möglichkeiten zur Modellierung und Spezifikation von Komponenten**

Dieser Beitrag:

- **stellt die relevanten Konzepte von UML 2.0 vor**
- **diskutiert, wie UML 2.0 im Memorandum berücksichtigt werden könnte**

Agenda



Komponentenbegriffe

Komponenten in UML 2.0

Auswirkungen auf Memorandum

Komponentenbegriffe

- **Definition in UML 2.0 [OMG2003, S. G-574]:**

  A component is a “modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. …”

- **Definition aus dem Memorandum [Turo2002, S. 1]:**

  „Eine *Komponente* besteht aus verschiedenartigen (Software-) Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohldefinierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar ist.“

Komponentenbegriffe – Gemeinsamkeiten (1)

Abgeschlossenheit

- **Definition in UML 2.0:**

  A component is a "modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. …"

- **Definition aus dem Memorandum:**

  „Eine *Komponente* besteht aus verschiedenartigen (Software-) Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohldefinierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar ist."

Komponentenbegriffe – Gemeinsamkeiten (2)

Kapselung

- **Definition in UML 2.0:**

  A component is a "modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. …"

- **Definition aus dem Memorandum:**

  „Eine *Komponente* besteht aus verschiedenartigen (Software-) Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohldefinierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar ist."

Komponentenbegriffe – Gemeinsamkeiten (3)

Verwendung wohldefinierter Schnittstellen

- **Definition in UML 2.0:**

  A component is a “modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. …”

- **Definition aus dem Memorandum:**

  „Eine *Komponente* besteht aus verschiedenartigen (Software-) Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohldefinierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar ist.“

Komponentenbegriffe – Unterschiede

- **Unterschiede ergeben sich aus unterschiedlicher Sichtweise**

- **UML:**
  - Komponenten als Teile von Systemen
  - Definition in Bezug auf Systeme (Teil eines Systems, austauschbar)

- **Memorandum:**
  - fokussiert auf Komponenten an sich (wiederverwendbar, kombinierbar)
  - bringt zusätzlich betriebswirtschaftliche Aspekte ein (vermarktbar)

Arten von Komponentenmodellen

- **Unterscheidung zwischen drei verschiedenen Arten von Modellen [Fowl1999]:**
  - (software-unabhängige) konzeptionelle Modelle
  - Spezifikationsmodelle
  - Implementierungsmodelle
- **Modellarten fokussieren auf unterschiedlichen Aspekten bei der Beschreibung von Komponente und deren Umfeld**

Komponentenarten im UML

- **UML unterscheidet zwischen logischen und physischen Komponenten**
  - Beispiele für logische Komponenten: Businesskomponenten, Prozesskomponenten
  - Beispiele für physische Komponenten: EJB-Komponente, CORBA-Komponente, COM+-Komponente, .NET-Komponente
- **Keine nähere Erläuterung dieser Unterscheidung**
- **Interpretation anhand der angegebenen Beispiele:**
  - Verwendung logischer Komponenten in konzeptionellen Modellen und in Spezifikationsmodellen
  - Verwendung physischer Komponenten in Implementierungsmodellen

Konsequenzen für Memorandum

- **Einsatz von UML-Komponenten**
  - Weitgehende Übereinstimmung der Komponentendefinition in UML und Memorandum (auf technischer Ebene)
  - Einsatz von UML-Komponenten im Memorandum möglich
- **Komponentenarten im Memorandum:**
  - keine explizite, weitere Unterscheidung von Komponenten
  - Allerdings: Fokus auf Außensicht der Komponente, keine Beschreibung der Implementierung
- **Konsequenz für Einsatz von Modellen im Memorandum:**
  - Geeignete Modellart: Spezifikationsmodelle
  - Geeignete Komponentenart: logische Komponenten

Agenda

Komponentenbegriffe

Komponenten in UML 2.0

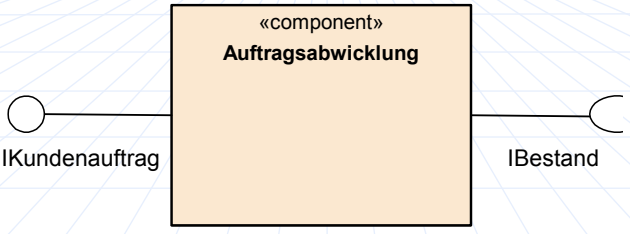
Auswirkungen auf Memorandum

Komponenten in UML 1.4 und UML 2.0

- **UML 1.4 enthält Modellierungskonstrukt *Komponente***
  - Aber: nur für Modellierung physischer Komponenten vorgesehen
  - Für logische Komponenten werden (als Workaraound) andere Modellierungskonstrukte verwendet, z.B.:
    - Subsysteme [Andr2003]
    - Klassen mit dem Stereotyp «comp spec» [ChDa2001]
- **Bedeutendste Änderung in UML 2.0: Konstrukt *Komponente* auch für logische Komponenten**
  - Ergebnis: deutlich bessere Möglichkeiten zur Spezifikation von Komponenten

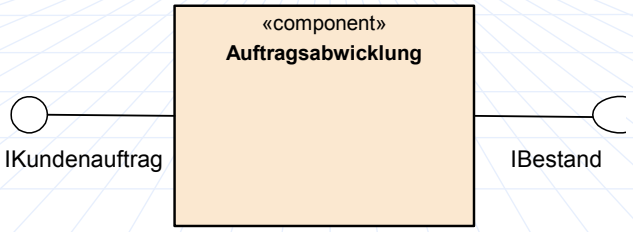
Komponenten in UML 2.0 (1)

- **Darstellung einer Komponente: Rechteck mit Stereotyp «component»**
- **Eigenschaften einer Komponente:**
  - Im UML Metamodell ein Subtyp von Klasse
  - Kann daher Attribute und Methoden haben
  - Kann optional eine interne Struktur haben
  - Kann eine Menge von Ports zur Verfügung stellen



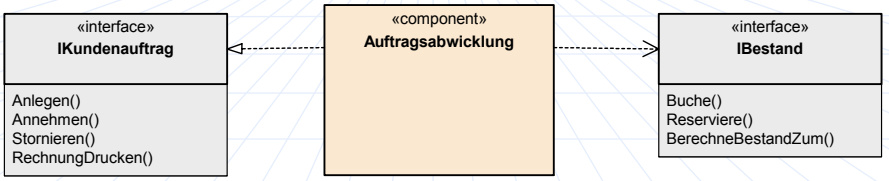
Komponenten in UML 2.0 (2)

- **Externe Sicht einer Komponente = Black-Box-Sicht: Komponente plus ihre Schnittstellen**
- **Schnittstellen einer Komponente:**
  - Komponente kann mehrere Schnittstellen haben
  - Graphische Unterscheidung zwischen bereitgestellten und benötigten Schnittstellen
  - Im Bild: *IKundenauftrag* = bereitgestellte Schnittstelle, *IBestand* = benötigte Schnittstelle



Komponenten in UML 2.0 (3)

- **Für detailliertere Darstellung der Schnittstellen: Verwendung des Konstrukts *Schnittstelle* («interface»)**
  - Schnittstelle kann so inklusive ihrer Operationen (mit Parametern) und eventuell vorhandener Attribute modelliert werden
  - Unterscheidung zwischen bereitgestellt und benötigt durch Art des Pfeils



Komponenten in UML 2.0 (4)

- **Optional: Versehen einer Komponente mit Ports**
  - Port = Schnittstelle mit eigenem Bezeichner
  - Dient z.B. zur Unterscheidung verschiedener Zugangspunkte, die technisch dieselbe Schnittstelle verwenden
- **Optional: Verbinden eines Zustandsdiagramms mit einer Schnittstelle (oder einem Port)**
  - expliziert dynamische Bedingungen (Aufrufreihenfolge von Operationen)
- **Interne Sicht einer Komponente = White-Box-Sicht: Darstellung der internen Struktur und der Realisierung**
- **Außerdem: eine Reihe von Detailverbesserungen**
  - insbesondere bei der OCL (siehe dazu auch Tagungsband)

Agenda

Komponentenbegriffe

Komponenten in UML 2.0

Auswirkungen auf Memorandum

Argumente pro UML 2.0 im Memorandum

- **Eine der Zielstellungen des Memorandums: Einsatz bekannter und weit verbreiteter Notationstechniken**
- **UML entwickelt sich mehr und mehr zum Standard für Modellierung und Spezifikation von Komponenten**
  - Enthält daher auch viele Verbesserungen für Komponenten

➢ **Memorandum sollte UML 2.0 an allen geeigneten Stellen einsetzen**
  - Nur so ist eine breite Akzeptanz des Memorandums zu erreichen

- **Außerdem vereinfacht die UML 2.0 die Spezifikation**

Vorschläge für Memorandum allgemein

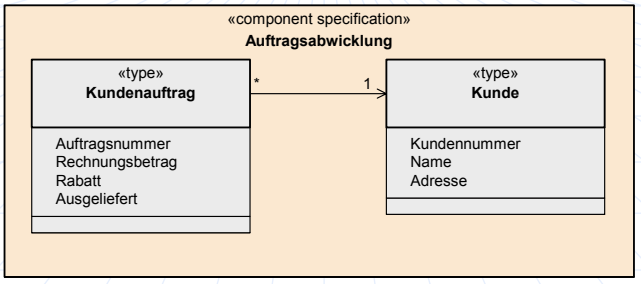
- **Spezifikation enthält ein Diagramm der externen Sicht der Komponente**
  - zeigt Komponente und die bereitgestellten und benötigten Schnittstellen
  - Diagramm gilt für die gesamte Spezifikation und wird keiner Ebene zugeordnet
- **Namenskonvention *Extern* nicht mehr nötig**
  - Unterscheidung anhand des Diagramms
- **Komponente kann mehrere Interfaces haben**
  - Daher Name der Fachkomponente <> Name eines Interfaces

Vorschläge für Schnittstellenebene

- **Einsatz der UML als Notationstechnik**
  - Detaillierte Beschreibung der Schnittstellen in einem speziellen Diagramm
  - Diagramm enthält für alle Schnittstellen: zugehörige Dienste (Operationen) mit ihren Parametern und Ausnahmen
- **Bisher gab es einen Methodenbruch zwischen Schnittstellenebene und Verhaltensebene [Acke2001]**
  - Konsequenz: Spezifikationsobjekte der verschiedenen Ebenen einander nicht eindeutig zuordenbar
  - Bisheriger Workaround: durch Namenskonventionen
- **Konsequenz: einheitliche Verwendung der UML (+ OCL) auf der Schnittstellen-, der Verhaltens- und der Abstimmungsebene**
  - passt auch gut zum Vorschlag von Overhage, einen thematischen Bereich *Technik* mit diesen drei Ebenen zu bilden [Over2002]
  - Falls Schnittstellenbeschreibung in OMG IDL notwendig, bei Bedarf generieren

Vorschläge für Verhaltensebene

- **Oft ist für Spezifikation ein konzeptionelles Schema nötig**
- **Bisher: Verwendung eines Klassendiagramms**
- **Mit UML 2.0: passender als interne Sicht der Komponente modellieren**
  - Entitäten = Klassen mit Standard-Stereotyp «type»
  - Modell dient als Interface Information Model (vgl. z.B. [ChDa2001])
  - Grundlage zur Formulierung von Zusicherungen mit OCL



Vorschläge für Abstimmungsebene

- **Zwei Varianten als Alternativen im Memorandum: temporale OCL-Ausdrücke und Zustandsdiagramm**
  - beide haben ihre spezifischen Vorteile
  - Konformität zur UML wird gewahrt
  - Ausdrucksmächtigkeit wird nicht verringert

Vorteile durch Vorschläge

- **Spezifikation bleibt konform zum Modellierungsstandard UML**
- **Schnittstellen-, Verhaltens- und Abstimmungsebene verwenden einheitliche Notationstechnik**
- **Spezifikation auf diesen Ebenen profitiert von erweiterten Ausdrucksmöglichkeiten von UML 2.0**
- **Auf einige der bisher notwendigen Workarounds kann verzichtet werden**
  - gedachte Komponente *Extern*
  - Namenskonventionen zur Verbindung von Schnittstellen- und Verhaltensebene

Referenzen

- [Acke2001] *Ackermann, J.*: Fallstudie zur Spezifikation von Fachkomponenten. In: *K. Turowski (Hrsg.)*: 2. Workshop Modellierung und Spezifikation von Fachkomponenten., Bamberg 2001, S. 1 – 66.
- [Andr2003] *Andresen, A.:* Komponentenbasierte Softwareentwicklung mit MDA, UML und XML. Hanser Verlag, München 2003.
- [ChDa2001] *Cheesman, J.; Daniels, J.:* UML Components. Addison-Wesley, Boston 2001.
- [Fowl2000] *Fowler, M.:* UML Distilled – A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2000.
- [OMG2003] *OMG (Hrsg.):* Unified Modeling Language: Superstructure, Version 2.0, Stand 10.04.2003. URL: http://www.omg.org/uml, Abruf am 2003-06-10.
- [Over2002] *Overhage, S.:* Die Spezifikation – kritischer Erfolgsfaktor der Komponentenorientierung. In: *K. Turowski (Hrsg.)*: 4. Workshop Komponentenorientierte betriebliche Anwendungssysteme (WKBA 4). Augsburg 2002, S. 1 – 17.
- [Turo2002] *Turowski, K. (Hrsg.).:* Vereinheitlichte Spezifikation von Fachkomponenten. Memorandum des Arbeitskreises 5.10.3 der Gesellschaft für Informatik. Augsburg 2002.