Software-Komponenten

Th. Letschert

THM

University of Applied Sciences

Einführung

- Komponenten: flexibel verwendbare Bausteine einer Architektur
- Thema der Veranstaltung: eine funktionale Sicht auf Komponenten

Software-Komponenten: Thematik

Was ist eine (Software-) Komponente

„A software component is a unit of composition with

- contractually specified interfaces and
- explicit context dependencies only.“

Clemens Szyperski
Component Software – Beyond Object Orientation , Addison-Wesly 2002

„A software component is an architectual entity that

- encapsulates a subset of the system's functionaltity and /or data
- restricts access to that subset via an explicit defined interface, and
- has explicitly defined dependencies on its required execution context.“

Taylor, Medvidovic, Dashofy: Software Architecture, Wiley 2010

Was ist eine (Software-) Komponente

Eine **Software-Komponente**

- Ist wiederverwendbar und ersetzbar
- Hat ein „Außen“ und ein „Innen“
- Verwirklicht die Prinzipien der *Kapselung*, *Abstraktion* und *Modularität*
- Trägt zu einer Ausführungsumgebung bei:
  - durch *Schnittstellen* über die Dienste anderen Komponenten *angeboten* werden und
  - *Schnittstellen* anderer Komponenten, über die deren Dienste *benutzt* werden
- ...

Arten von (Software-) Komponenten

logische und physikalische und Komponenten

- Logisch: Komponenten im Sinne der Programmiersprache:
  - Klasse, Paket, Funktion, …
- Physikalisch: Komponenten als Datei oder Archiv:
  - Übersetzungseinheit, class-Datei, ausführbare Datei, Objektbibliothek, ...

Arten von (Software-) Komponenten

Klassifikation entsprechend dem Lebenszyklus der Software

- Quellcode-Entwicklung
- Compilation
- Zusammenstellen ablauffähiger Einheiten
- Einsatz (Deployment, Installation)

nach ihr können unterschiedliche Arten von Komponenten unterschieden werden:

- **Modul**
  Einheit der Quellcode-Entwicklung, Einheit auf der Ebene der Sprache
- **Übersetzungseinheit**
  Besondere Einheit auf der Ebene der Sprache
- **Assembly**
  Zusammenstellung übersetzter Einheiten zu einer ausführbaren Anwendung
- **Plugin**
  Eine Komponenten einer ausführbaren Anwendung die zur Laufzeit ersetzt werden kann.

Arten von (Software-) Komponenten

Lebenszyklus von Software und Komponenten

- Übersetzungseinheiten sind die wichtigsten Komponenten der Entwicklung

  Der Compiler überwacht, eventuell von Code-Konventionen unterstützt (C/C++), die Schnittstellen der Komponenten (Typen / Signaturen)

- Beispiel Java:
  - Jar-Dateien sind die Assemblies: ausführbare - und Bibliothekseinheiten
  - Linking und Loading findet zur Laufzeit auf Klassenebene statt

    Der Class-Loader ist die Basis aller dynamischen (Laufzeit-) Komponententechnologien!

    Alle Abhängigkeiten zwischen Klassen werden zur Laufzeit gelöst

- Beispiel C:
  - Objektbibliotheken sind die Assemblys
  - Binden und Laden findet auf der Ebene der Bibliotheken (hauptsächlich) zur Bindezeit statt

Thematik dieser Veranstaltung

wir behandeln nur Logische Komponenten

- die von der Sprache konzeptionell unterstützt werden,
- explizite statische Schnittstellen haben,
- die weitgehend durch das Typ-System beschrieben werden können und
- ihre konzeptionelle Basis hauptsächlich in der funktionalen Programmierung haben

Software-Komponenten: Thematik

Wichtige Themen die hier nicht behandelt werden

physikalische Komponenten / Deployment / Plugins ...

relevante Basis-Konzepte hier u.a.

- **Reflection**
- **Klassenlader**
- **Java Module System**
- **DI-Frameworks (DI: Dependency-Injection)**
- **…**

bietet Material für eine eigene Veranstaltung

Funktionale Modularisierung / Funktionen als Komponenten

Funktionen und Werte sind die (einzigen) Komponenten eines Softwaresystems

„This is the key to functional programming's power - it allows greatly improved modularisation. It is also the goal for which functional programmers must strive - smaller and simpler and more general modules, glued together with the new glues we shall describe."

„The first of the two new kinds of glue enables simple functions to be glued together to make more complex ones."

„The other new kind of glue that functional languages provide enables whole programs to be glued together. [..] [Layz evaluation] makes it practical to modularize a program as a generator that constructs a large number of possible answers, and a selector that chooses the appropriate one. [..] Lazy evaluation is perhaps the most powerful tool for modularization in the functional programmer's repertoire."

aus *Why Functional Programming Matters* **von John Hughes**
http://www.cse.chalmers.se/~rjmh/Papers/whyfp.html

Funktionale Modularisierung / Funktionale Modularisierung und Typen

Die Schnittstellen der Komponenten eines Softwaresystems werden durch ihre Typen beschrieben.

Typen beschreiben die möglichen Werte die ein Ausdruck zur Laufzeit annehmen kann.

Sie beschreiben damit deren Einsatzfähigkeit in einem beliebigen Kontext.

Erwartung an die Teilnehmer

Die Teilnehmer sollten vertraut sein mit

- **Objektorientierte Programmierung**
- **Software-Technik**
- **Funktionale Programmierung (auf Einsteiger-Niveau)**
- **Programmiersprache Scala (auf Einsteiger-Niveau)**

The Answer to the Ultimate Question of Life,
the Universe, and Everything
is
~~42~~ ! ~~Function~~ ~~Monad~~ Math