Metabeschreibungen

Bei der Erstellung objektorientierter Softwaresysteme können *Metabeschreibungen* benutzt werden, welche größtenteils neue Abstraktionen schaffen und Wiederverwendung fördern:

Slide 1

- Klassenbibliotheken (oder -hierarchien)
- Frameworks
- Patterns (Muster)

Die neuen Abstraktionen werden hier vornehmlich durch die Vererbungsmöglichkeiten geschaffen

Klassenbibliotheken

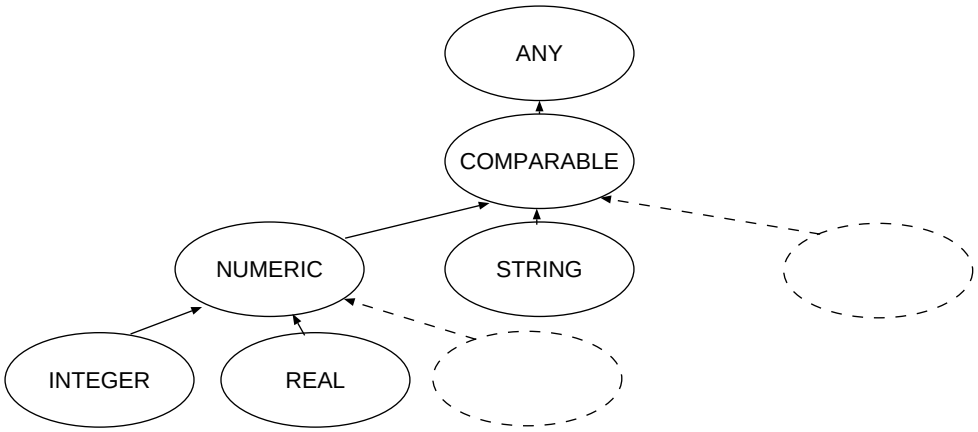
Der systematische Aufbau von Klassenhierarchien durch Vererbung resultiert in Klassenbibliotheken mit folgenden Charakteristika:

Slide 2

- allgemeine, abstrakte Verhaltensbeschreibungen finden sich an der Wurzel der Hierarchie
- durch das Typsystem exisitert ein Regelwerk, durch das sich die Klassen in die Hierarchie einfügen und Erweiterungen möglich sind
- die Klassen an den Blättern der Hierarchie werden vornehmlich durch Instanziierung wiederverwendet
- wichtigste Beziehung zwischen den Klassen ist die Vererbungsbeziehung

Klassenbibliotheken

Slide 3



Frameworks

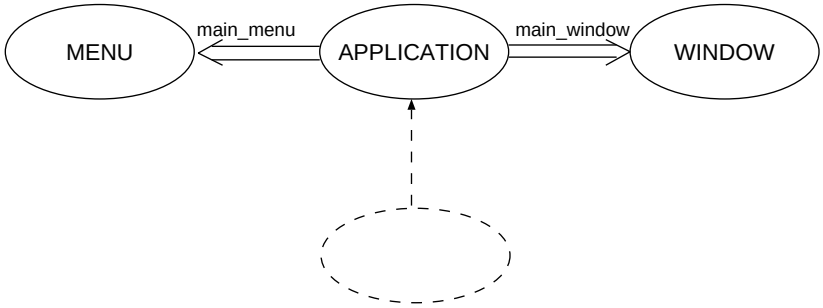
Frameworks stellen eine strukturelle Basis für die Entwicklung eines Systems zur Verfügung. Sie haben folgende Eigenschaften:

Slide 4

- strukturelle Eigenschaften werden vornehmlich in vorgegebenen Kunde-Anbieter-Beziehungen festgelegt (*frozen spots*)
- Anpassungen an die aktuellen Anforderungen werden meist in Vererbungsbeziehungen ausgedrückt (*hot spots*)
- Wiederverwendung wird durch die Benutzung der vorgegebenen Struktur erreicht

Frameworks

Slide 5



Patterns

Während Klassenbibliotheken und Frameworks meist fertig zur Verfügung stehen, erlauben es *Patterns,* kleinere Bestandteile eines Systems flexibel zu beschreiben und zu einen Gesamtsystem zusammenzusetzen:

Slide 6

- Patterns sind Frameworks ,,im Kleinen", d.h. sie sind nicht unbedingt auf eine Gesamtlösung ausgerichtet
- sowohl Kunde-Anbieter- als auch Vererbungsbeziehungen sind wichtig
- sie adressieren Wiederverwendung, Unterstützung beim Entwurf und Unterstützung bei der Kommunikation unter Entwicklern

Eigenschaften von Patterns

Patterns besitzen noch eine Reihe von weiteren Eigenschaften:

- Patterns sollen wiederkehrende, spezifische Design-Probleme lösen
- sie dokumentieren existierende, bewährte Praxis

Slide 7

- sie schaffen ein gemeinsames Vokabular und ein Mittel zur Dokumentation
- Patterns helfen, Komplexität zu bewältigen, indem sie Lösungen zur Verfügung stellen, auf denen weiter aufgebaut werden kann

Definition

Patterns lassen sich wie folgt definieren:

- Ein Pattern beschreibt die Lösung für ein wiederkehrendes Design-Problem.
- Die Lösung ist erprobt und generisch

Slide 8

- Ein Pattern umfaßt die beteiligten Komponenten, ihre Verantwortlichkeiten und Verbindungen sowie die Kollaborationen

Pattern-Beschreibung

Die Beschreibung eines Patterns besteht aus drei Teilen:

Kontext: Situation, aus der das Design-Problem entsteht

Problem: wiederholt auftretendes Problem, gegensätzliche Anforderungen

Slide 9

Lösung: überprüfte Lösung für das Problem, welche die Anforderung in Einklang bringt

Klassen von Patterns

Patterns können in drei Gruppen eingeteilt werden:

- Architektur-Patterns
- Entwurfs-Patterns

Slide 10

- Idiome

Architektur-Patterns

Architektur-Patterns versuchen, das Gesamtsystem durch eine Menge vordefinierter Subsysteme zu strukturieren und die Verbindungen und Rollen der Subsysteme zu beschreiben.

Die Bereiche, aus denen die Architektur-Patterns stammen, decken dabei die typischen Anwendungsstrukturen ab:

Slide 11

- Eingabe-Verarbeitung-Ausgabe
- Interaktive Systeme
- Verteilte Systeme

Beispiele von Architektur-Patterns

Filter: bestehen aus Datenquelle, Filter und Datensenke

Schichtenarchitektur: Verteilung von Funktionalität auf unterschiedliche Schichten mit definierten Übergabepunkten

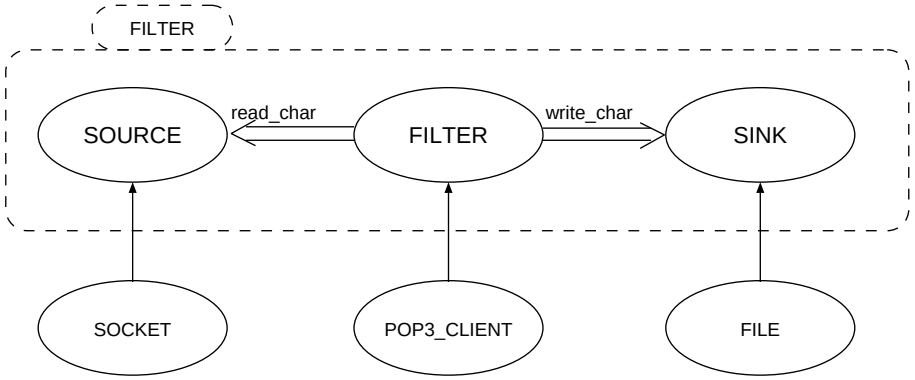
Slide 12

Model-View-Controller: Trennung von Datenhaltung, Präsentation und Interaktion

Microkernel: Verteilung von Diensten und Schaffung einer Kommunikationsinfrastruktur

Beispiel: Filter-Pattern

Slide 13



Entwurfs-Patterns

Auf der Ebene von Subsystemen in Software-Architekturen liegt das Anwendungsfeld für Entwurfs-Patterns (Design-Patterns). Die Bereiche, die Entwurfs-Patterns abdecken, sind sehr vielfältig, darunter sind:

Slide 14

- Dekomposition von Strukturen
- Organisation von Arbeitsabläufen
- Zugriffssteuerung
- Kommunikation

Beispiele für Entwurfs-Patterns

Teil-Ganzes: Aufteilung in kleinere Bestandteile und Definition der Kooperation

Master-Slave: Aufteilung und Zusammenführung von Arbeitsabläufen

Proxy: Beschreibung eines Stellvertreters und der Delegation

Slide 15

Client-Server: bestehen meist aus Client, Nachrichtenverteiler und Server, dienen der Aufgabenverteilung

Idiome

Idiome versuchen, Probleme zu lösen, die nicht aus dem Systementwurf resultieren, sondern durch die Implementierung entstehen (z.B. Benutzung einer bestimmten Programmiersprache):

Slide 16

- Programmierstil
- Dokumentationsstil
- Unzulänglichkeiten von Programmiersprachen

Beispiele für Idiome

Referenz-Zähler: Unterstützung von Speichermanagement in C++

Assertions: Simulierung von Vor- und Nachbedingungen in C++ durch Makros

Wiederholte Vererbung: Implementierungs- und Interface-Vererbung in Eiffel

Slide 17

Einrückungsregeln: Kennzeichnung von Code-Blöcken durch Formatierung des Quelltextes

Pattern-Systeme

Pattern-Systeme beschreiben eine Menge von Patterns, welche gemeinsam verwendet werden können:

Slide 18

- möglichst alle Kategorien sollen abgedeckt sein (dadurch Abdeckung möglichst vieler Anwendungsfälle)
- Definition der Zusammenhänge zwischen den Patterns
- Beschreibung der Anwendung und Implementierung der Patterns
- möglichst uniformer Aufbau der Beschreibungen

Verwendung im Entwicklungsprozeß

Patterns implizieren kein neues Vorgehen im Entwicklungsprozeß. Trotzdem strukturieren sie den Ablauf beim Design:

Slide 19

- die Grundstruktur des Systems wird durch ein oder mehrere Architektur-Patterns festgelegt
- die Subsysteme werden mit Entwurfs-Patterns beschrieben
- sich wiederholende Probleme und Aufgabenstellung bei der Implementierung werden mit Idiomen bearbeitet

Implikationen aus der Verwendung

Bei der Verwendung von Patterns sollten jedoch zwei Punkte beachtet werden:

- Patterns schaffen keine neuen Abstraktionen, sie fördern keine neuen Lösungen

Slide 20

- die Verwendung von Patterns soll sich an den Problemen orientieren und nicht umgekehrt