44) Querschneidende Verfeinerung mit Plattformkonnektoren

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 13-1.0, 06.07.13

1) Plattformobjekte- und konnektoren
2) Abbildung der Integrates-Relation
3) Gesamtbild der Verfeinerung

Objektorientierter Entwurf (Object-Oriented Design, OOD)

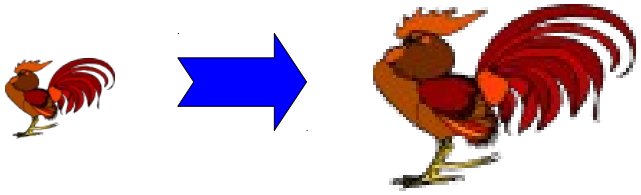
1) Einführung in die objektorientierte Softwarearchitektur
   1) Modularität und Geheimnisprinzip
   2) Entwurfsmuster für Modularität
   3) BCED-Architekturstil (3-tier architectures)
2) Verfeinerung des Entwurfsmodells zum Implementierungsmodell
   1) Anreicherung von Klassendiagrammen
   2) Verfeinerung von Lebenszyklen
   3) Querschneidende Verfeinerung mit Object Fattening
3) Objektorientierte Rahmenwerke (frameworks)
4) Softwarearchitektur mit dem Quasar-Architekturstil

Obligatorische Literatur

- OSGI Technical White Paper. www.osgi.org

Objektanreicherung (Wdh.)

- **Objekt-Anreicherung (Object fattening) durch Unterobjekte** ist ein Verfeinerungsprozess, der an ein Kernobjekt aus dem Domänenmodell Unterobjekte anlagert, die
  - Teile ergänzen (Teile-Verfeinerung)
  - Rollen ergänzen (Konnektor-Verfeinerung), die Beziehungen klären zu
    - Plattformen (middleware, Sprachen, Komponenten-services)
    - Komponentenmodellen (durch Adaptergenerierung)
- Ziel: Entwurfsobjekte, Implementierungsobjekte



- Achtung. Wir nähern uns, nach vielen einzelnen Schritten, dem Höhepunkt der Vorlesung:

> Querschneidende **Objektanreicherung** ist der entscheidende Schritt bei der Verfeinerung von den Analyse- und Entwurfsmodellen zum Implementierungsmodell und zur Implementierung.

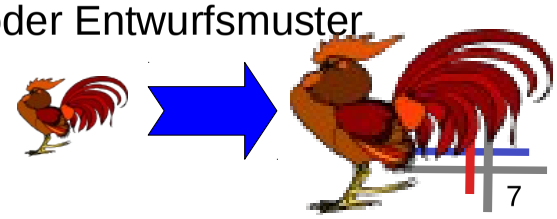
- Gründe:
  - Der objekt-orientierte Software-Entwicklungsprozess startet mit einer Simulation der realen Welt durch Objekte, die zu Systemobjekten erweitert werden und dabei durch technische Informationen angereichert werden müssen

44.1 Objektanreicherung mit Plattforminformation (Querschneidende Verfeinerung für Plattformen)

.. Verfeinerung durch Integration von Unterobjekten..

Plattformanreicherung – Weitere Schritte im Entwurf

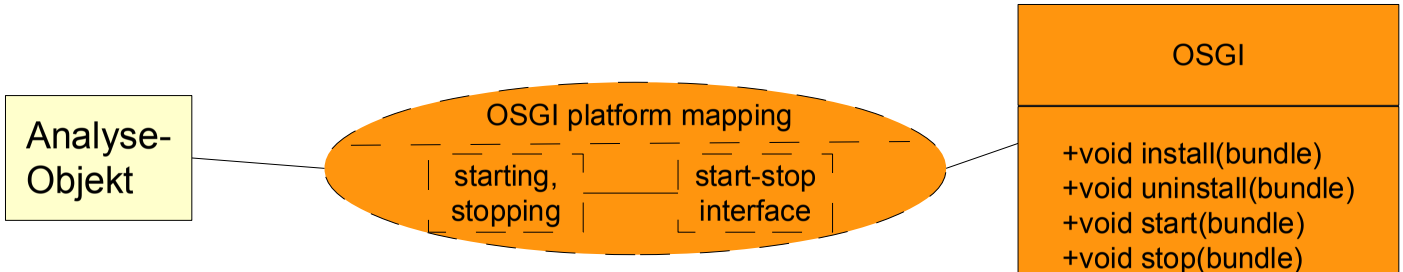
- Teile- und Rollenverfeinerung startet schon in der Analyse
- Bei Entwurfsobjekten kommt **Plattformanreicherung** hinzu:
  - Finden von **Plattform-Konnektoren (-team),** die plattform-fundierte Unterobjekte anlagern, die das spezifische Verhalten bezüglich eines Plattformobjektes kapseln
    - **Plattformfähigkeiten (platform abilities, platform-founded types)** bilden fundierte Typen, die die die Beziehungen zu Plattformen klären
    - **Komponentenadapter (component-model-founded adapters)** klären die Beziehung zu Komponentenmodellen
- Ziel im Entwurf: Implementierungsobjekte ableiten
  - Rollen ergänzen, die Beziehungen klären zu
    - Plattformobjekten (middleware, Sprachen, Komponenten-services)
    - Komponentenmodellen (durch Adaptergenerierung)
  - Realisierung der Integrationsrelation
- Einfache Implementierung durch Teams oder Entwurfsmuster



Plattformobjekte und -konnektoren

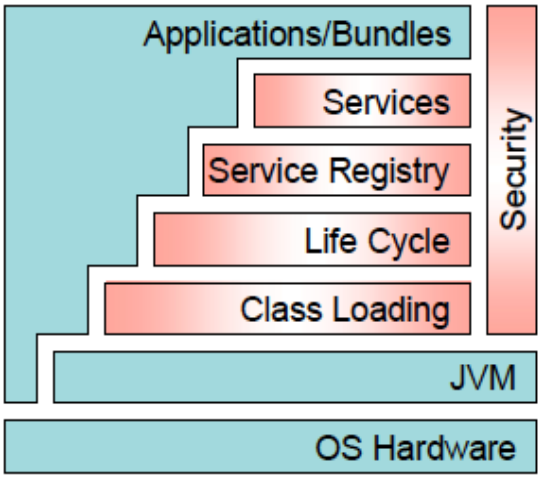
- Ein **Plattformobjekt** ist ein Objekt einer Systembibliothek, auf die eine Software angepasst werden muss
  - Bietet Schnittstelle an bzgl. bestimmter Funktionalität, z.B. abstrakte Maschine (Interpretierer)
  - Variable: je nach Maschine, Middleware, Betriebssystem, Datenbank, Programmiersprache unterschiedlich ausgeprägt
- Die Kollaboration mit der Plattform wird durch einen Konnektor zum Plattformobjekt, dem **Plattformkonnektor (Plattform-Kollaboration)**, ausgedrückt
- OSGI: Komponentenplattform www.osgi.org
  - im Handy, 5er BMW, in Eclipse 3.0, Shell home automation HomeGenie
  - Ein *bundle (Komponente)* paketiert verschiedene Klassen

Prof. U. Aßmann, Softwaretechnologie, TU Dresden



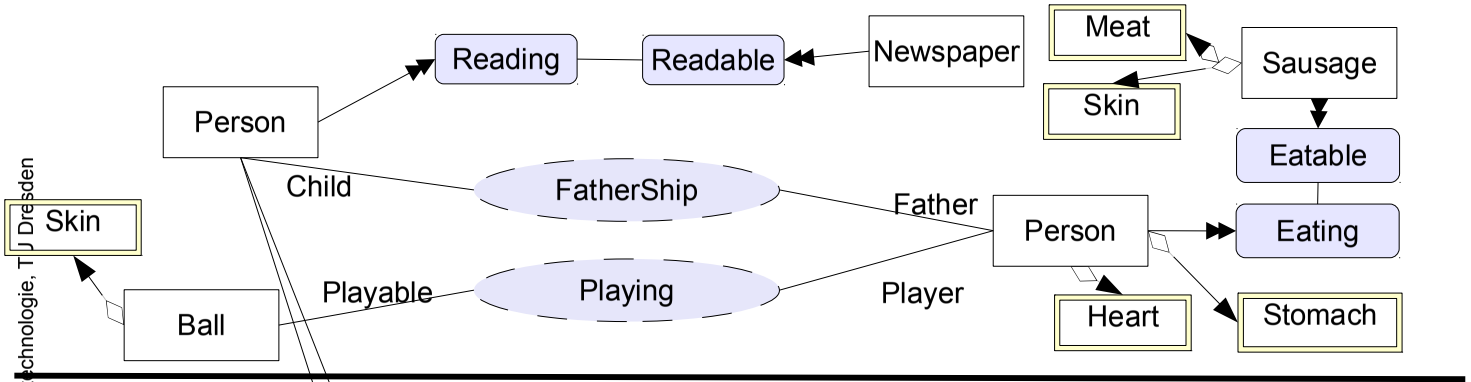
Plattformobjekt OSGI

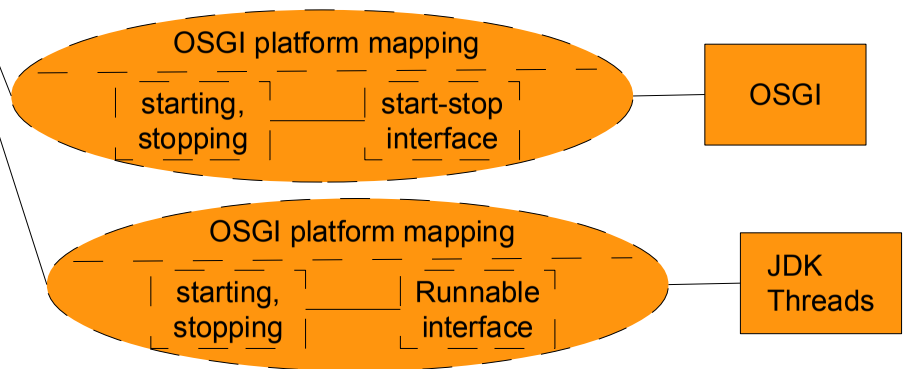
- OSGI bietet 5 Schnittstellen (rot)
  - Klassenlader (für Ersetzung von bundles)
  - Lebenszyklus (life cycle) von *bundles* (Paketen von Klassen, mit zip gepackt und verschickt)
  - Register (service registry): dient zum Registrieren von Bundles und ihren Zuständen
  - Dienste (services) verschiedener Art
  - Sicherheitsfunktionalität
- [OSGI Technical White Paper]



Mit Verfeinerung durch Plattform-Konnektoren (platform fattening)

- Plattform-Konnektoren beschreiben die Beziehungen zu Plattformobjekten sowie die Interaktion der Anwendungsobjekte mit ihnen (orange; Analyse-Konnektoren: lila)
- Plattformobjekte können als Alternativen existieren (hier OSGI, JDK threads) für die Plattform "Lebenszyklus"





Plattform CORBA: CORBA:Object

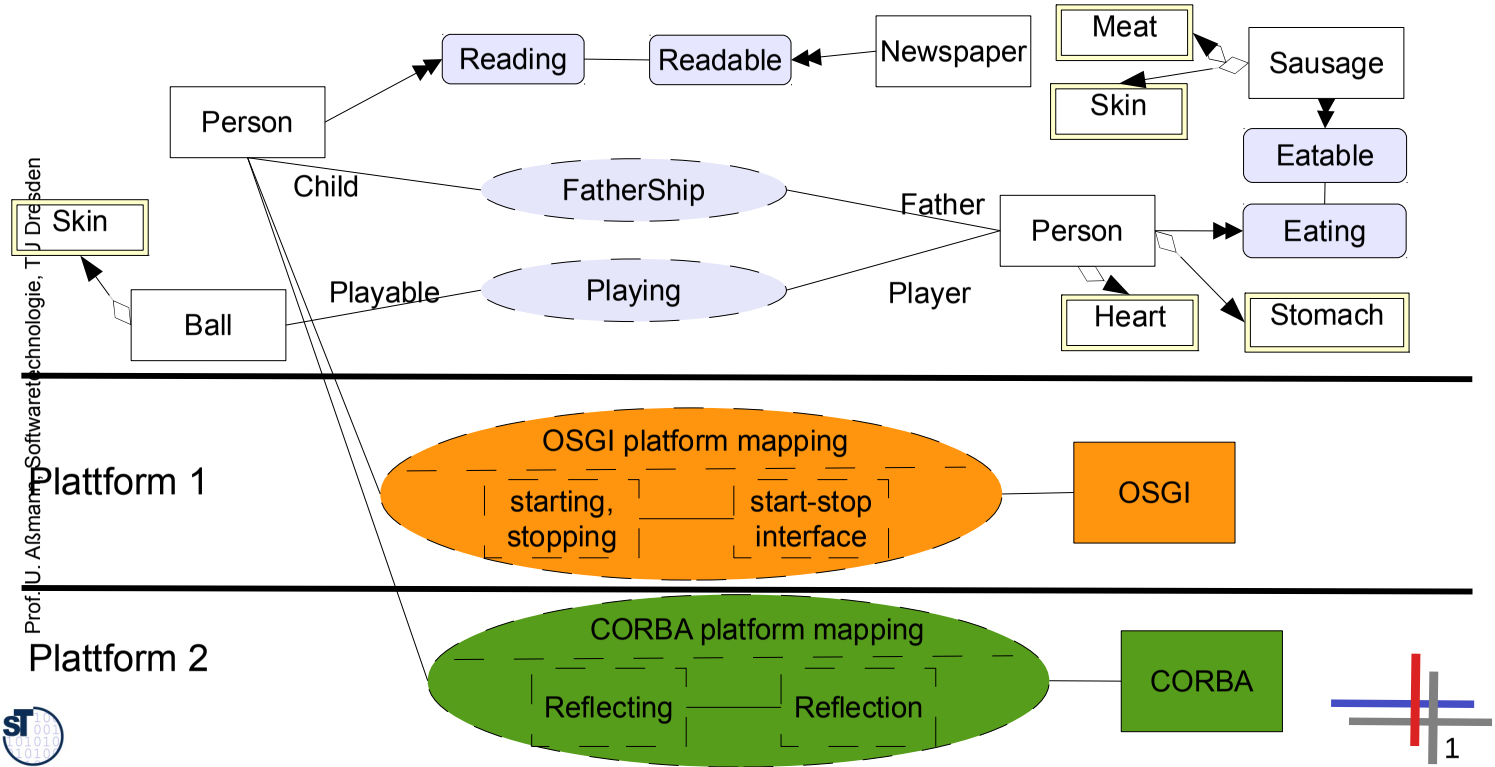
- CORBA bildet eine Komponentenplattform für heterogen programmierte Systeme

CORBA:Object
get_implementation get_interface is_nil is_a is_equivalent create_request duplicate release ....

- In der Klasse CORBA:Object wird elementare Funktionalität einer CORBA Komponente definiert
  - heterogen benutzbar über viele Sprachen hinweg
- CORBA untertützt Reflektion:
  - get_interface liefert eine Referenz auf ein “Schnittstellenobjekt”
  - get_implementation eine Referenz auf eine “Implementierung” (Klassenprototyp)

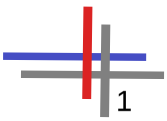
Mit Verfeinerung durch mehrere Plattform-Konnektoren verschiedener Plattformen

- Plattform-Verfeinerung kann auf verschiedenen Stufen ablaufen, und somit verschiedene Plattformen behandelt werden
- Plattformkonnektoren werden stufenspezifisch eingesetzt und können gegen Varianten ausgetauscht werden



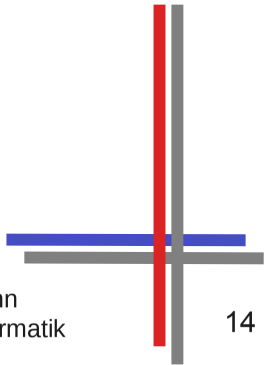
Das Portabilitätsgesetz

Kapselt man Plattformabhängigkeiten in einen Plattformkonnektor, können sie leicht ausgetauscht werden und die Software wird portabel.



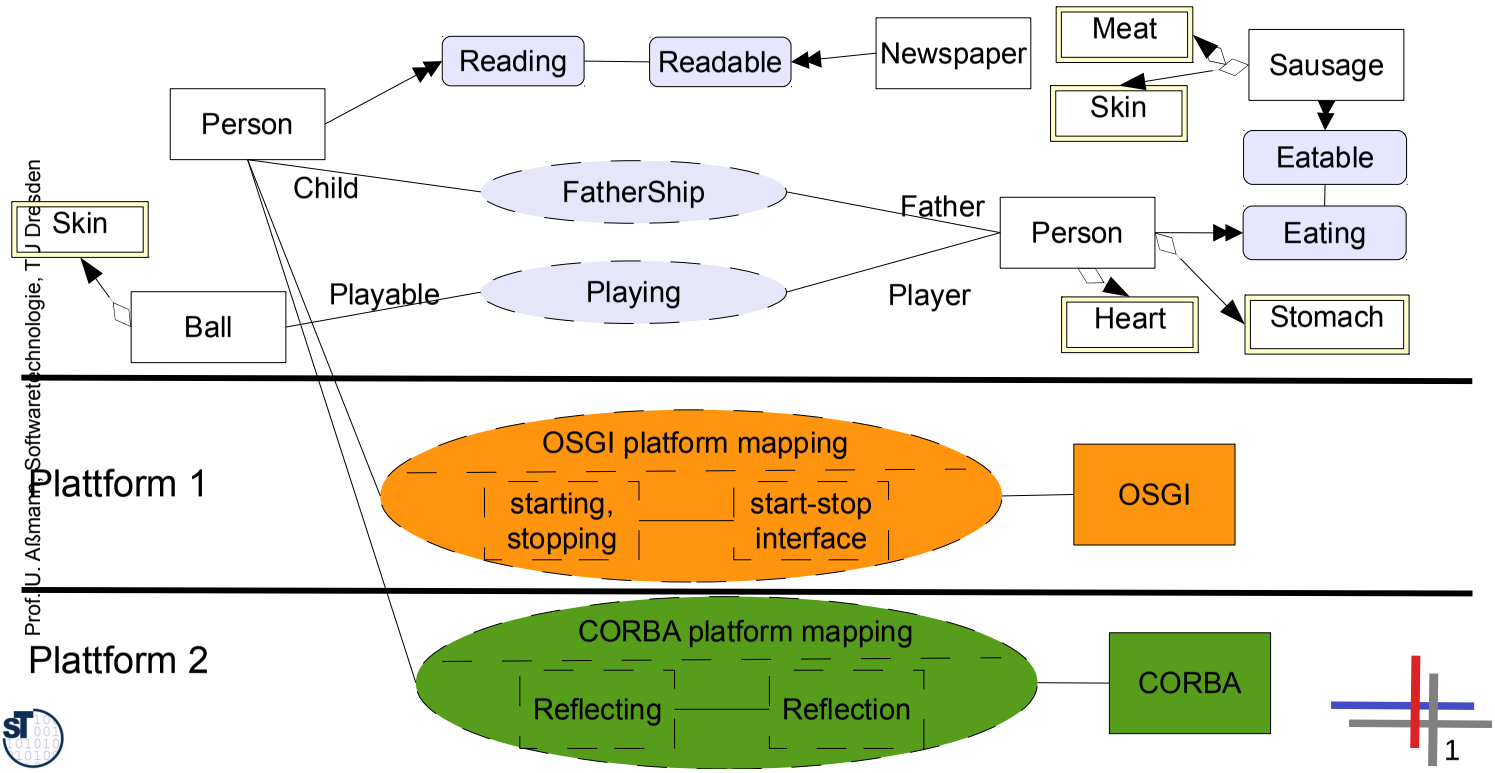
44.2 Abbildung der Integrationsrelation auf klassische Programmiersprachen

.. in der Implementierung ..



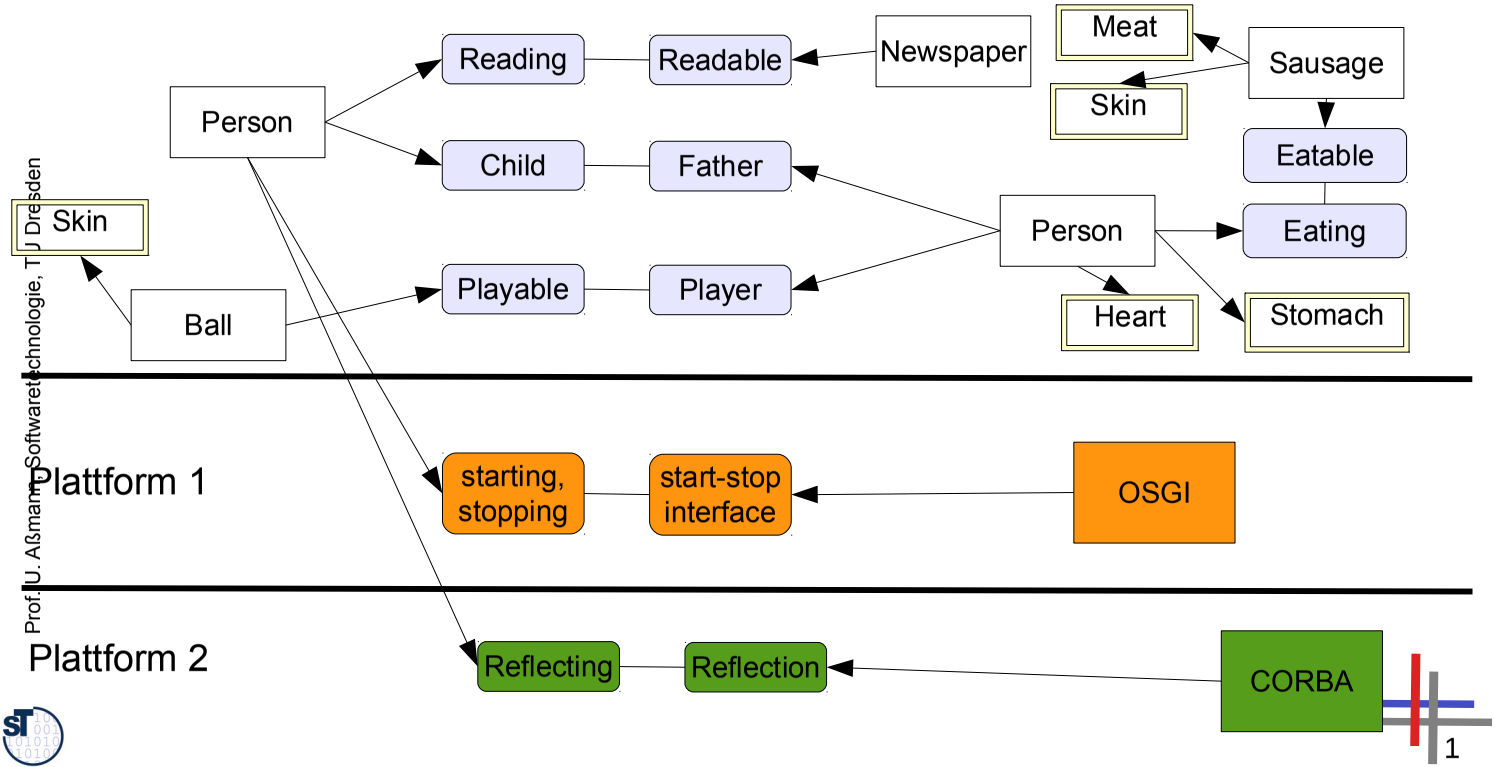
Wie bilde ich “integrates-a” ab?
a) mit einer Rollen-Programmiersprache

- Kollaborationen/Konnektoren und die “integrates”-Relation können verschieden auf eine Programmiersprache abgebildet werden
  1) Durch Rollensprachen wie ObjectTeams; dann liegt die Abbildung im Übersetzer



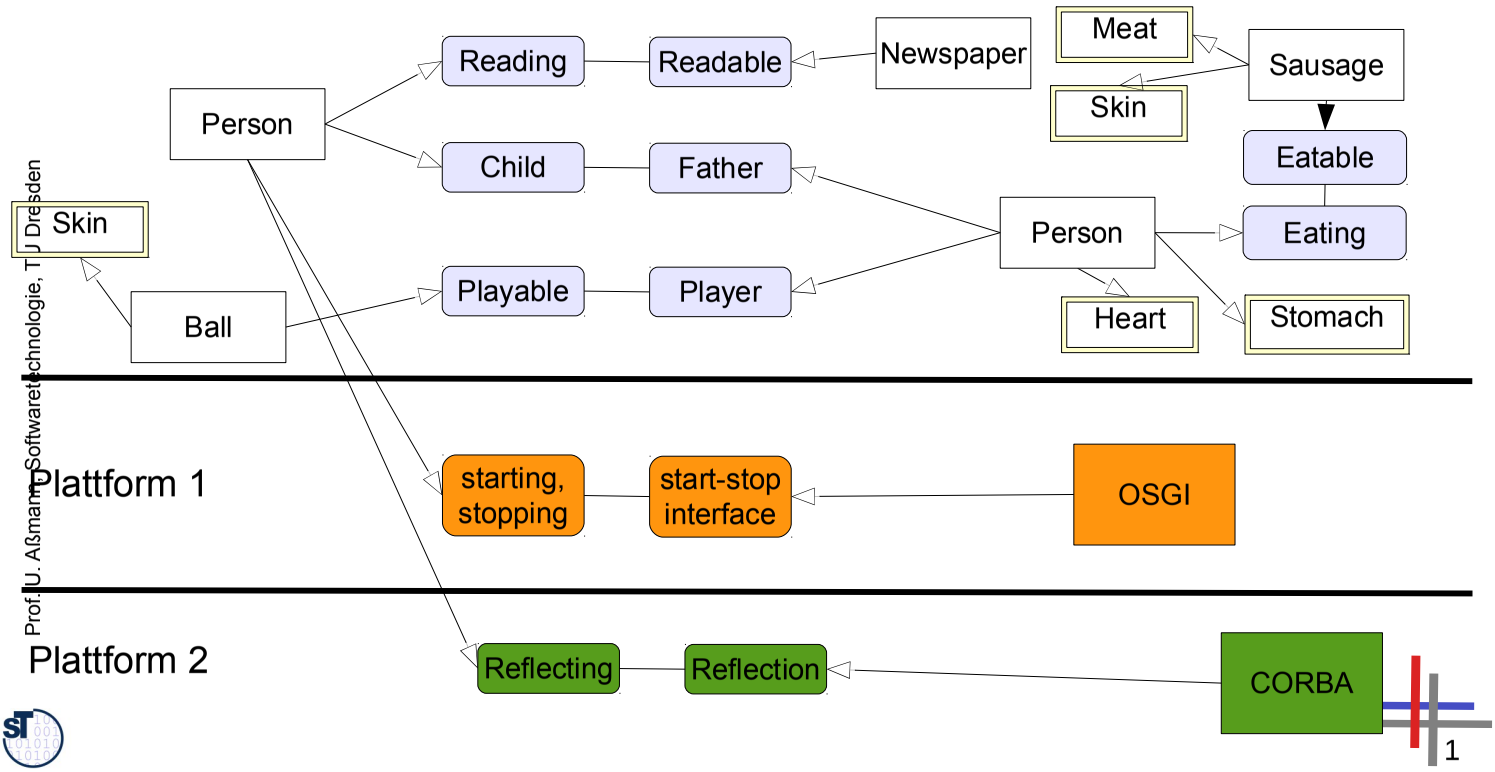
b) Wie bilde ich “integrates” durch Delegation ab?

- Ersetze alle “integrates”, “plays”, “mandatory-part”, etc. durch Delegationen
- Einfach, allerdings splittert man alle logischen komplexen Objekte in unzählige Implementierungsobjekte auf (siehe Vorlesung “Design Patterns and Frameworks”)
- Statische Komposition der Initial- und Terminal-Botschaften nötig



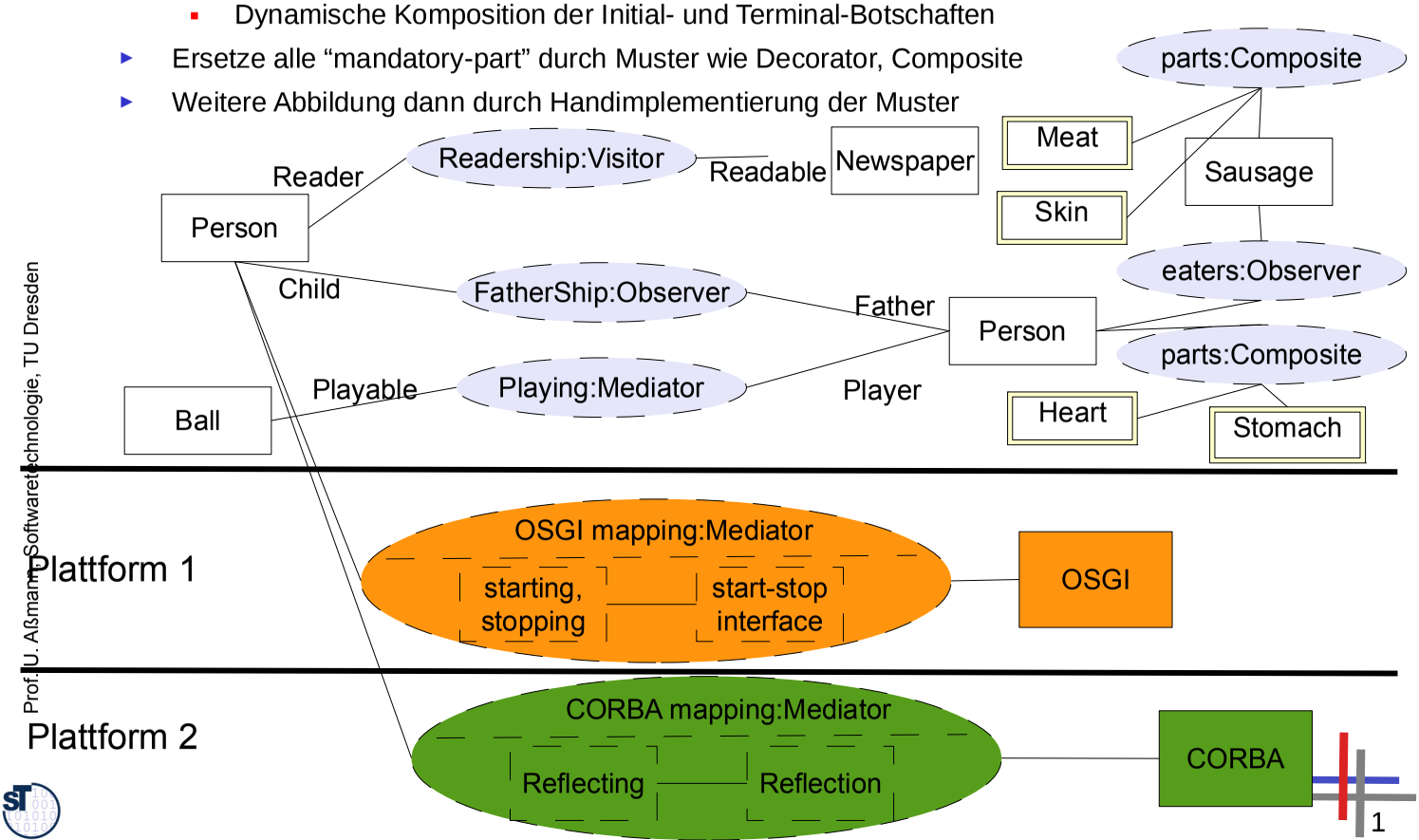
c) Wie bilde ich “integrates” durch Vererbung ab?

- Ersetze alle “integrates”, “plays”, “mandatory-part”, etc. durch Vererbung
- Einfach, allerdings braucht man Mehrfachvererbung oder “mixin inheritance”
- Statische Komposition der Initial- und Terminal-Botschaften nötig



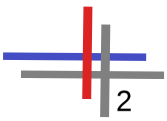
d) Wie bilde ich “integrates” durch Implementierungsmuster ab?

- Ersetze alle “integrates”, “plays”, etc. durch Muster wie Observer, Visitor
  - Dynamische Komposition der Initial- und Terminal-Botschaften
- Ersetze alle “mandatory-part” durch Muster wie Decorator, Composite
- Weitere Abbildung dann durch Handimplementierung der Muster

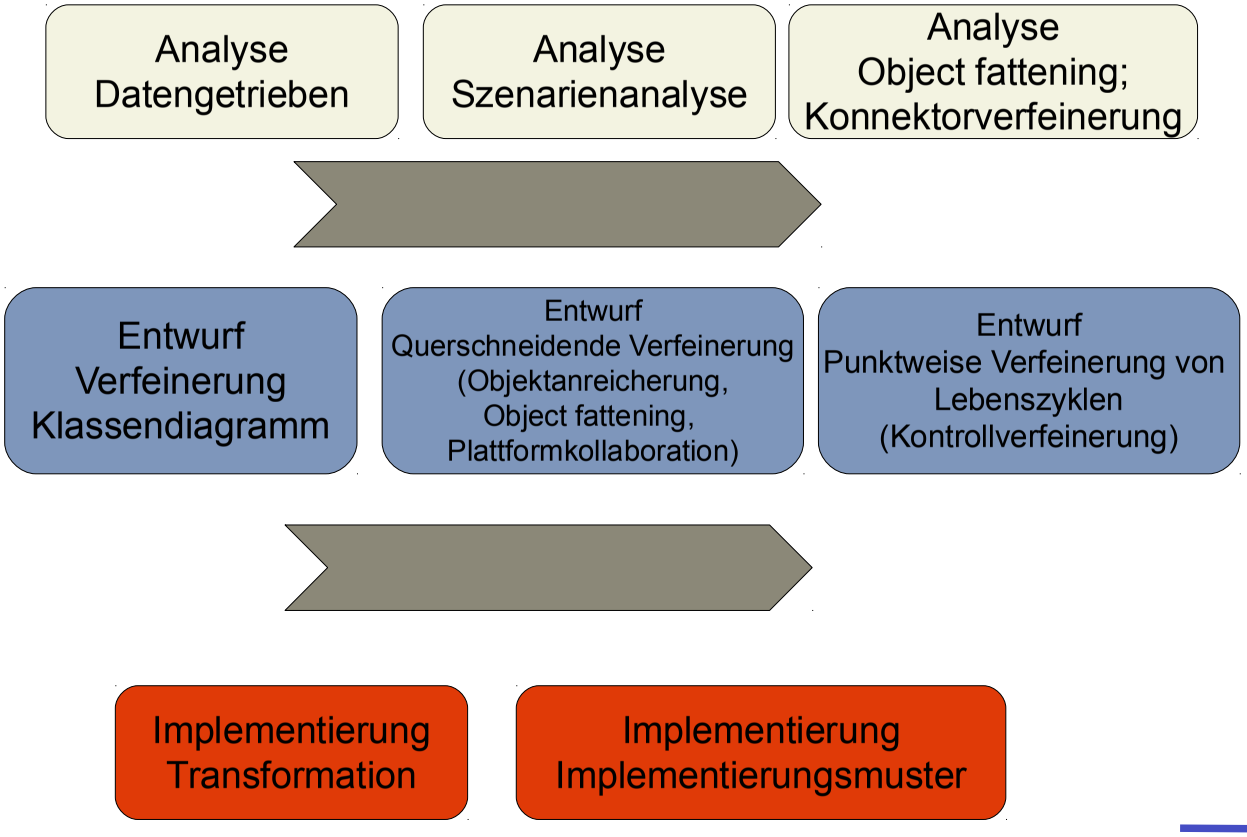


d) Wie bilde ich “integrates” durch Transformation ab?

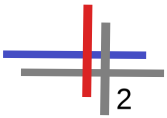
- Ersetze alle “integrates”, “plays”, etc. durch *Transformationsregeln*
- Führt auf *Modellgetriebene Architektur (model-driven architecture, MDA)*
- Weiter in der Softwaretechnologie-II



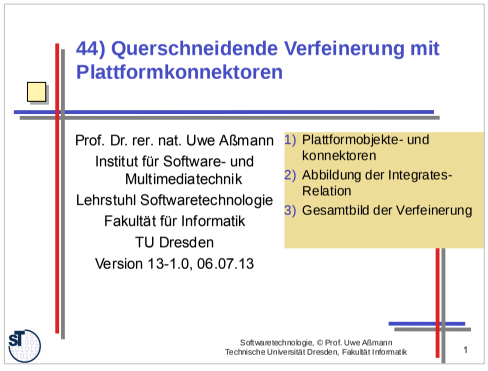
44.3 Gesamtbild der Verfeinerung



Prof. U. Aßmann, Softwaretechnologie, TU Dresden



The End



- Parallelen zum Fachgebiet der Architektur:
  - Architekten sind an der Nahtstelle zwischen Kunde und Baufirma.
  - Schlechter Architekturentwurf kann nicht durch gute Bauqualität kompensiert werden.
  - Es gibt Architektur-Spezialisten für bestimmte Anwendungsgebiete.
  - Es gibt "Schulen", die

Objektorientierter Entwurf (Object-Oriented Design, OOD)

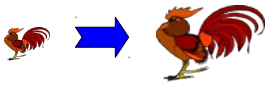
1) Einführung in die objektorientierte Softwarearchitektur
   1) Modularität und Geheimnisprinzip
   2) Entwurfsmuster für Modularität
   3) BCED-Architekturstil (3-tier architectures)
2) Verfeinerung des Entwurfsmodells zum Implementierungsmodell
   1) Anreicherung von Klassendiagrammen
   2) Verfeinerung von Lebenszyklen
   3) Querschneidende Verfeinerung mit Object Fattening
3) Objektorientierte Rahmenwerke (frameworks)
4) Softwarearchitektur mit dem Quasar-Architekturstil

Obligatorische Literatur

- OSGI Technical White Paper. www.osgi.org

Objektanreicherung (Wdh.)

- **Objekt-Anreicherung (Object fattening) durch Unterobjekte** ist ein Verfeinerungsprozess, der an ein Kernobjekt aus dem Domänenmodell Unterobjekte anlagert, die
  - Teile ergänzen (Teile-Verfeinerung)
  - Rollen ergänzen (Konnektor-Verfeinerung), die Beziehungen klären zu
    - Plattformen (middleware, Sprachen, Komponenten-services)
    - Komponentenmodellen (durch Adaptergenerierung)
- Ziel: Entwurfsobjekte, Implementierungsobjekte



- Achtung. Wir nähern uns, nach vielen einzelnen Schritten, dem Höhepunkt der Vorlesung:

> Querschneidende **Objektanreicherung** ist der entscheidende Schritt bei der Verfeinerung von den Analyse- und Entwurfsmodellen zum Implementierungsmodell und zur Implementierung.

- Gründe:
  - Der objekt-orientierte Software-Entwicklungsprozess startet mit einer Simulation der realen Welt durch Objekte, die zu Systemobjekten erweitert werden und dabei durch technische Informationen angereichert werden müssen

44.1 Objektanreicherung mit Plattforminformation (Querschneidende Verfeinerung für Plattformen)

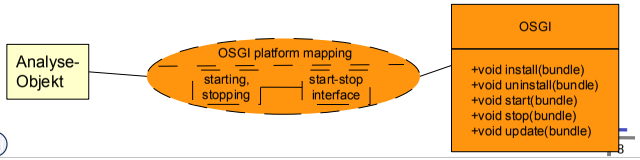
.. Verfeinerung durch Integration von Unterobjekten..

Plattformanreicherung – Weitere Schritte im Entwurf

- Teile- und Rollenverfeinerung startet schon in der Analyse
- Bei Entwurfsobjekten kommt **Plattformanreicherung** hinzu:
  - Finden von **Plattform-Konnektoren (-team),** die plattform-fundierte Unterobjekte anlagern, die das spezifische Verhalten bezüglich eines Plattformobjektes kapseln
    - **Plattformfähigkeiten (platform abilities, platform-founded types)** bilden fundierte Typen, die die die Beziehungen zu Plattformen klären
    - **Komponentenadapter (component-model-founded adapters)** klären die Beziehung zu Komponentenmodellen
- Ziel im Entwurf: Implementierungsobjekte ableiten
  - Rollen ergänzen, die Beziehungen klären zu
    - Plattformobjekten (middleware, Sprachen, Komponenten-services)
    - Komponentenmodellen (durch Adaptergenerierung)
  - Realisierung der Integrationsrelation
- Einfache Implementierung durch Teams oder Entwurfsmuster

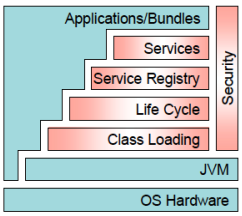
Plattformobjekte und -konnektoren

- Ein **Plattformobjekt** ist ein Objekt einer Systembibliothek, auf die eine Software angepasst werden muss
  - Bietet Schnittstelle an bzgl. bestimmter Funktionalität, z.B. abstrakte Maschine (Interpretierer)
  - Variable: je nach Maschine, Middleware, Betriebssystem, Datenbank, Programmiersprache unterschiedlich ausgeprägt
- Die Kollaboration mit der Plattform wird durch einen Konnektor zum Plattformobjekt, dem **Plattformkonnektor (Plattform-Kollaboration)**, ausgedrückt
- OSGI: Komponentenplattform www.osgi.org
  - im Handy, 5er BMW, in Eclipse 3.0, Shell home automation HomeGenie
  - Ein *bundle (Komponente)* paketiert verschiedene Klassen



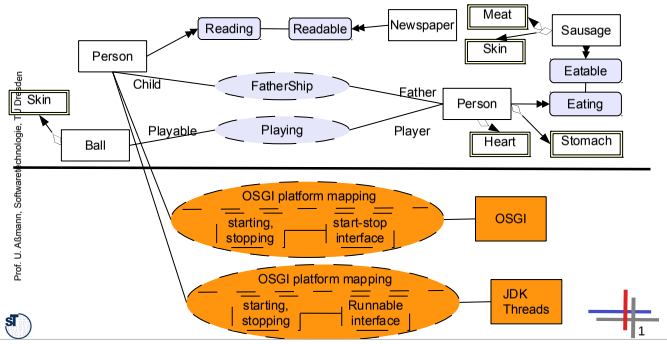
Plattformobjekt OSGI

- OSGI bietet 5 Schnittstellen (rot)
  - Klassenlader (für Ersetzung von bundles)
  - Lebenszyklus (life cycle) von *bundles* (Paketen von Klassen, mit zip gepackt und verschickt)
  - Register (service registry): dient zum Registrieren von Bundles und ihren Zuständen
  - Dienste (services) verschiedener Art
  - Sicherheitsfunktionalität
- [OSGI Technical White Paper]



Mit Verfeinerung durch Plattform-Konnektoren (platform fattening)

- Plattform-Konnektoren beschreiben die Beziehungen zu Plattformobjekten sowie die Interaktion der Anwendungsobjekte mit ihnen (orange; Analyse-Konnektoren: lila)
- Plattformobjekte können als Alternativen existieren (hier OSGI, JDK threads) für die Plattform "Lebenszyklus"



Plattform CORBA: CORBA:Object

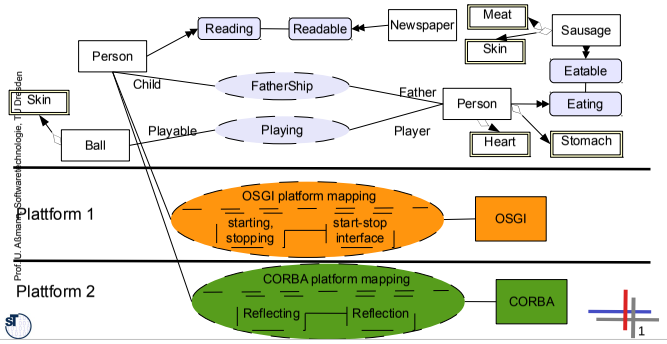
- CORBA bildet eine Komponentenplattform für heterogen programmierte Systeme

CORBA:Object
get_implementation get_interface is_nil is_a is_equivalent create_request duplicate release ....

- In der Klasse CORBA:Object wird elementare Funktionalität einer CORBA Komponente definiert
  - heterogen benutzbar über viele Sprachen hinweg
- CORBA untertützt Reflektion:
  - get_interface liefert eine Referenz auf ein "Schnittstellenobjekt"
  - get_implementation eine Referenz auf eine "Implementierung" (Klassenprototyp)

Mit Verfeinerung durch mehrere Plattform-Konnektoren verschiedener Plattformen

- Plattform-Verfeinerung kann auf verschiedenen Stufen ablaufen, und somit verschiedene Plattformen behandelt werden
- Plattformkonnektoren werden stufenspezifisch eingesetzt und können gegen Varianten ausgetauscht werden



Das Portabilitätsgesetz

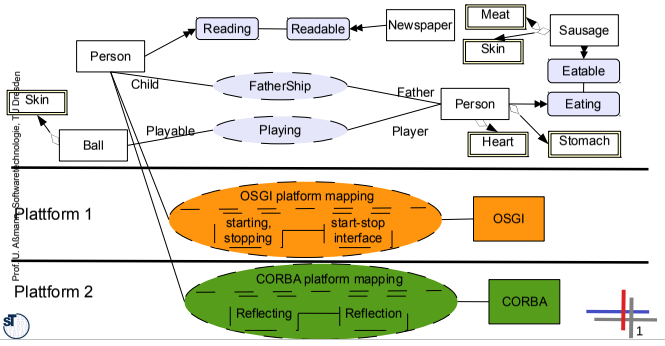
Kapselt man Plattformabhängigkeiten in einen Plattformkonnektor, können sie leicht ausgetauscht werden und die Software wird portabel.

44.2 Abbildung der Integrationsrelation auf klassische Programmiersprachen

.. in der Implementierung ..

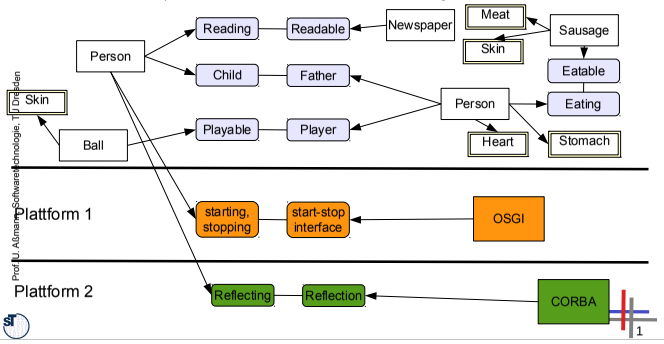
Wie bilde ich “integrates-a” ab? a) mit einer Rollen-Programmiersprache

- Kollaborationen/Konnektoren und die “integrates”-Relation können verschieden auf eine Programmiersprache abgebildet werden
  1) Durch Rollensprachen wie ObjectTeams; dann liegt die Abbildung im Übersetzer



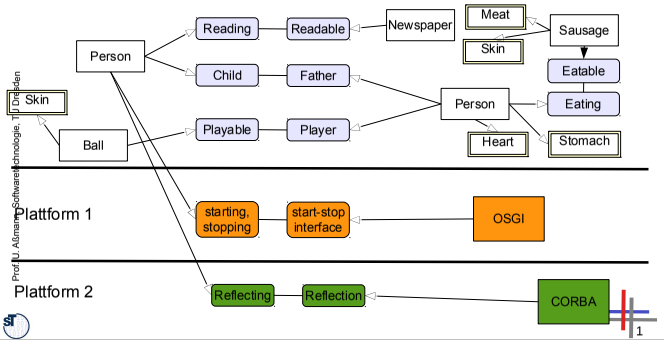
b) Wie bilde ich “integrates” durch Delegation ab?

- Ersetze alle “integrates”, “plays”, “mandatory-part”, etc. durch Delegationen
- Einfach, allerdings splittert man alle logischen komplexen Objekte in unzählige Implementierungsobjekte auf (siehe Vorlesung “Design Patterns and Frameworks”)
- Statische Komposition der Initial- und Terminal-Botschaften nötig



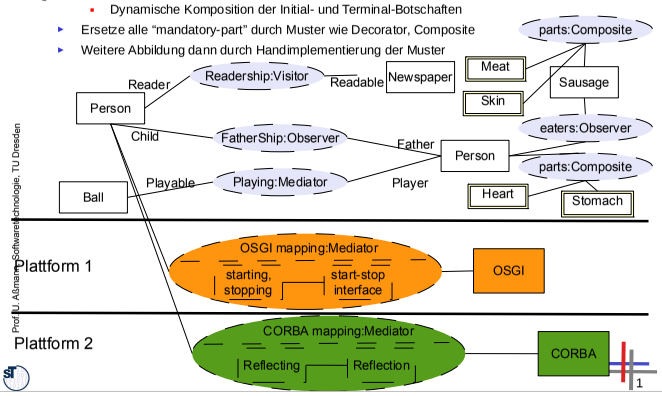
c) Wie bilde ich “integrates” durch Vererbung ab?

- Ersetze alle “integrates”, “plays”, “mandatory-part”, etc. durch Vererbung
- Einfach, allerdings braucht man Mehrfachvererbung oder “mixin inheritance”
- Statische Komposition der Initial- und Terminal-Botschaften nötig



d) Wie bilde ich “integrates” durch Implementierungsmuster ab?

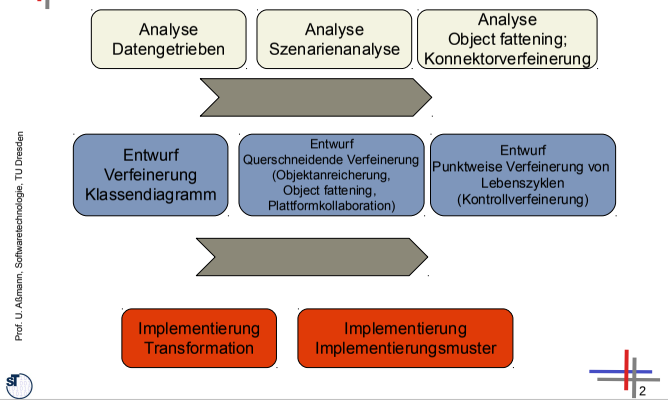
- Ersetze alle “integrates”, “plays”, etc. durch Muster wie Observer, Visitor
  - Dynamische Komposition der Initial- und Terminal-Botschaften
- Ersetze alle “mandatory-part” durch Muster wie Decorator, Composite
- Weitere Abbildung dann durch Handimplementierung der Muster



d) Wie bilde ich “integrates” durch Transformation ab?

- Ersetze alle “integrates”, “plays”, etc. durch *Transformationsregeln*
- Führt auf *Modellgetriebene Architektur (model-driven architecture, MDA)*
- Weiter in der Softwaretechnologie-II

44.3 Gesamtbild der Verfeinerung



The End