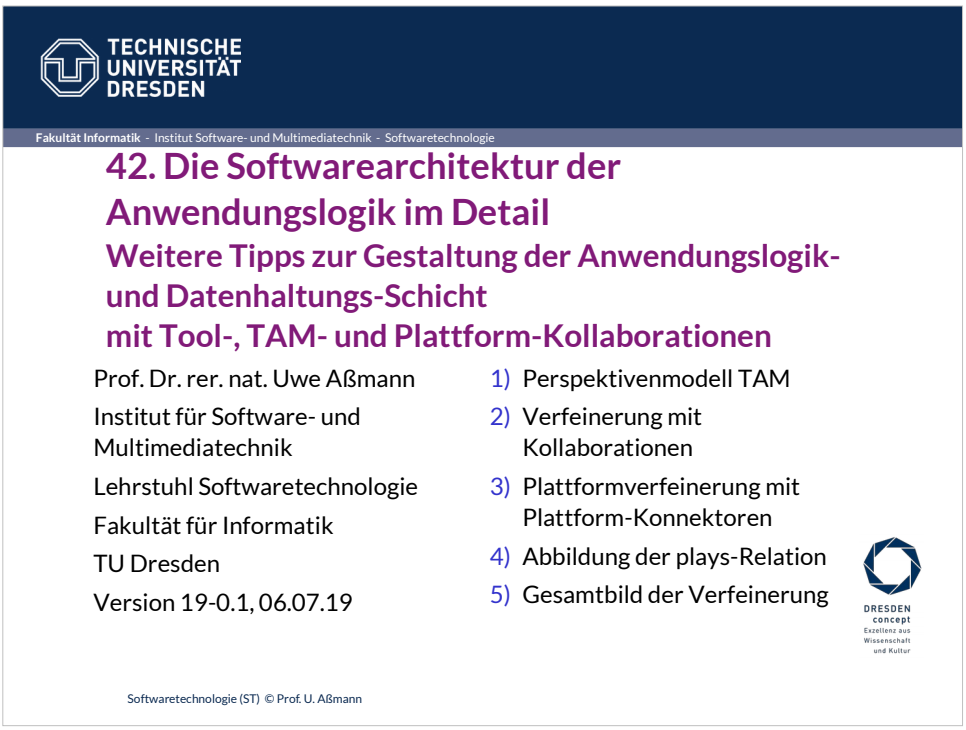


- Parallelen zum Fachgebiet der Architektur:
  - Architekten sind an der Nahtstelle zwischen Kunde und Baufirma.
  - Schlechter Architekturentwurf kann nicht durch gute Bauqualität kompensiert werden.
  - Es gibt Architektur-Spezialisten für bestimmte Anwendungsgebiete.
  - Es gibt "Schulen", die bestimmte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

Einschreibung in das Softwarepraktikum WS 2019/20

- Einschreibung erfolgt für ALLE Studenten (auch für ISTler) ab 08. Juli 2019 über www.jexam.de
- Einschreibung **befristet bis 04. August 2019**
  - Praktikum Softwaretechnologie - **Intern** (Standardfall, Tutor ist Kunde)
  - Praktikum Softwaretechnologie - **Extern**(Firma oder andere Institution als realer Kunde)
- Achtung!
  - **jexam**: Schauen Sie in die Lehrveranstaltungen des **Wintersemesters 2019/20!**
  - Entweder in das externe oder in das interne Praktikum einschreiben! Wer keinen externen Praktikumsplatz erhält, bekommt automatisch einen internen Praktikumsplatz.
  - Wer sich in das Praktikum bis zum 04. August 2019 nicht eingeschrieben hat, kann wahrscheinlich nicht teilnehmen, sich jedoch auf der Nachrückliste einschreiben!

Teil IV - Objektorientierter Entwurf (Object-Oriented Design, OOD)

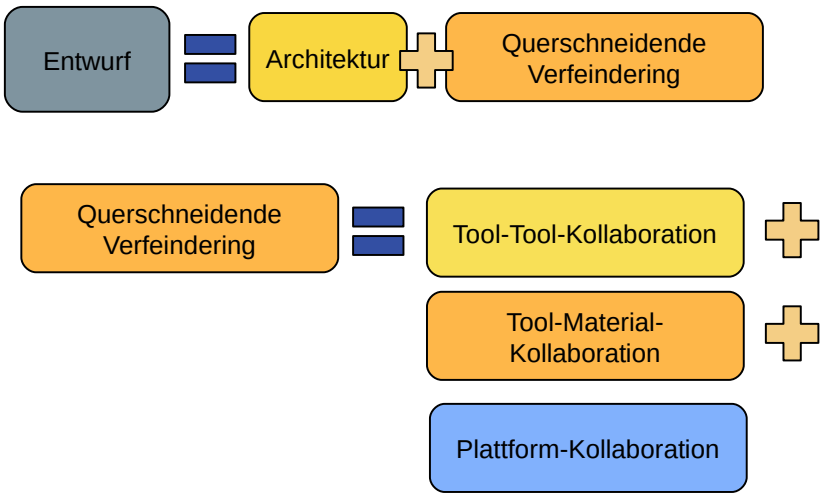
1) 40: Überblick
2) 41: Einführung in die objektorientierte Softwarearchitektur
   1) Architekturprinzipien, Architekturstile, Perspektivenmodelle
   2) Modularität und Geheimnisprinzip
   3) BCD-Architekturstil (3-tier architectures)
3) **42: Schichtenarchitektur im Detail**
   1) **Tool- und TAM-Kollaborationen**
   2) **Plattform-Kollaborationen: Verfeinerung mit querschneidender Objektanreicherung**
4) 43: Architektur interaktiver Systeme
5) 44: Punktweise Verfeinerung von Lebenszyklen
   - Verfeinerung von verschiedenen Steuerungsmaschinen

Literatur

- Obligatorisch:
  - D. Riehle, H. Züllighoven. A Pattern Language for Tool Construction and Integration Based on the Tools&Materials Metaphor. PLOP I, 1995, Addison-Wesley.
  - OSGI Technical White Paper. www.osgi.org
- Fakultativ:
  - Heinz Züllighoven. Object-oriented construction handbook - developing application-oriented software with the tools and materials approach. dpunkt.verlag, 2005, ISBN 978-3-89864-254-5.

Der Feldherrnhügel des Entwurfs

- Wer Architektur und querschneidende Verfeinerung beherrscht, beherrscht den Entwurf.



42.1 Identifikation von Tools, Materials, zur Einordnung von Klassen in die Schichten

Ein Vorschlag für die Konnektion von Anwendungslogik und Datenhaltung

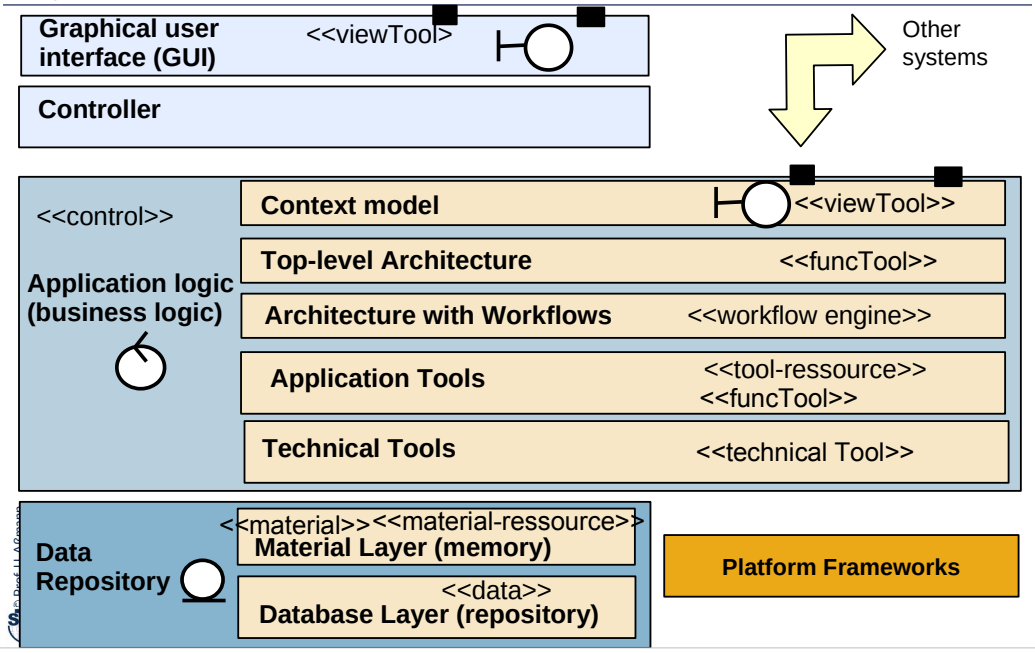
Was wird interaktiv (asynchron) aufgerufen?

Was ist passiv?

Was muss belegt werden?

Welche Klasse wird in welche Schicht eingeordnet?

Vorausschau:
Q8: Verfeinerte BCED-Schichtung eines Systems mit TAM

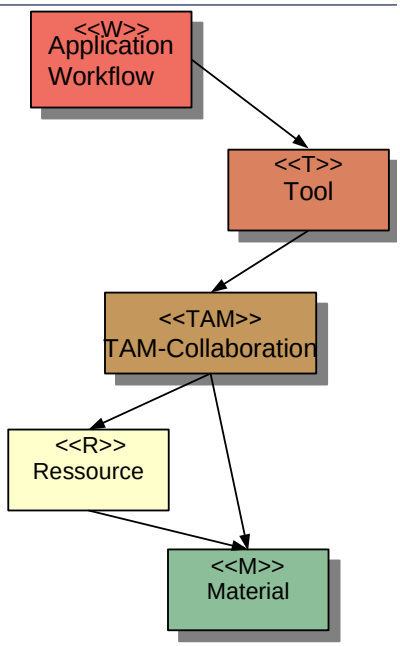


Perspektivenmodell TAM: Trennung von aktiven und passiven Komponenten

Tools-and-Materials [Züllighoven] ist ein Perspektivenmodell, das folgende Aspekte in einem Profil definiert:

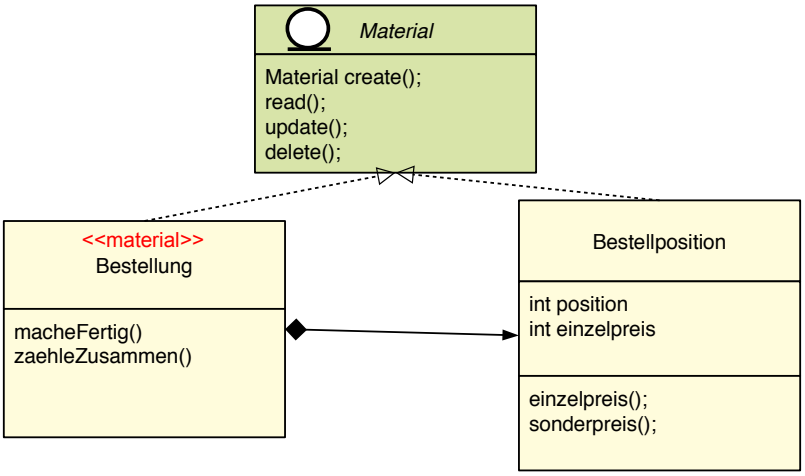
1) Tools (aktive Prozesse, Kommandoobjekte)
2) Ressources (belegbar)
3) Materials (passive Daten, Schicht E)
4) TAM-Collaboration
5) Workflows koordinieren Tools

- Klassen, Module, Komponenten, Pakete sollten mit diesen Aspekten qualifiziert werden



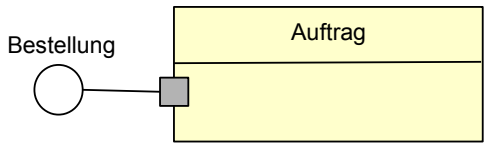
Material-Klassen und -Schnittstellen

- **Materialobjekte** sind passiv, d.h. werden von außen aufgerufen und geben den Steuerfluss nach außen hin zurück
  - Liegen in der Datenablage-Schicht (D)
- Materialobjekte können komposit sein (Muster Composite)
- Materialien folgen der CRUD-Schnittstelle (create, read, update, delete)



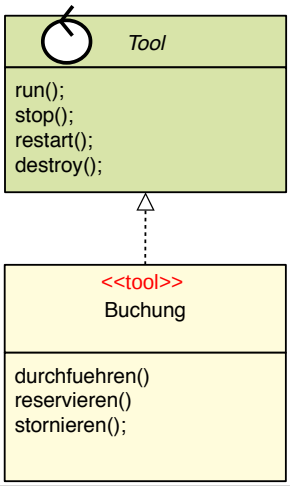
Material-Klassen und -Schnittstellen

- Materialien können in Ports auftauchen



Tool-Klassen und -Schnittstellen

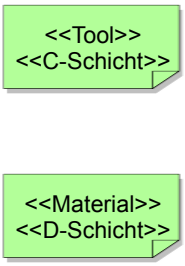
- Toolobjekte sind I.d.R. aktiv, besitzen eigenen Steuerfluss (thread, process)
- Tools liegen in der Anwendungslogik (C-Schicht)



[Züllighoven]

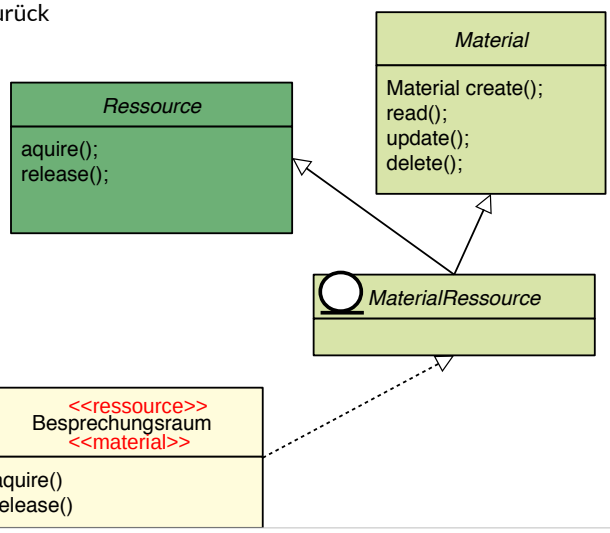
Das TAM-Gesetz (Tools-Materials Gesetz)

Trenne Tool- von Material-Klassen, denn sie gehören auf verschiedene Schichen des Systems.



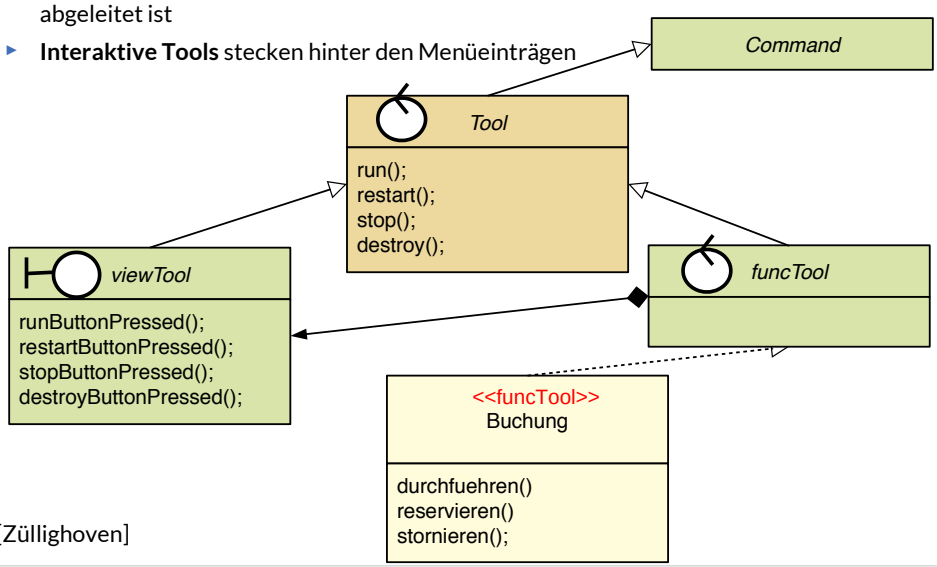
Ressourcen-Klassen und -Schnittstellen

- **Resourcenobjekte** sind Materialien, die vor Nutzung zu *belegen* sind, d.h. sie müssen vor Nutzung alloziert und nach Nutzung freigegeben werden
- **Materialressourcen** sind passiv, werden von außen aufgerufen und geben den Steuerfluss nach außen hin zurück



Tool-Klassen und -Schnittstellen

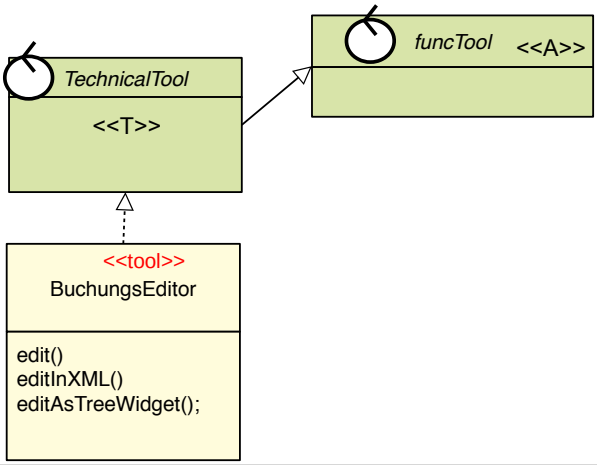
- Toolobjekte haben einen interaktiven Teil (viewTool, boundary, view) und einen ausführenden, funktionalen Teil (funcTool, "model"), der aus dem Command-Pattern abgeleitet ist
- **Interaktive Tools** stecken hinter den Menüeinträgen



[Züllighoven]

Technische Tool-Klassen und -Schnittstellen

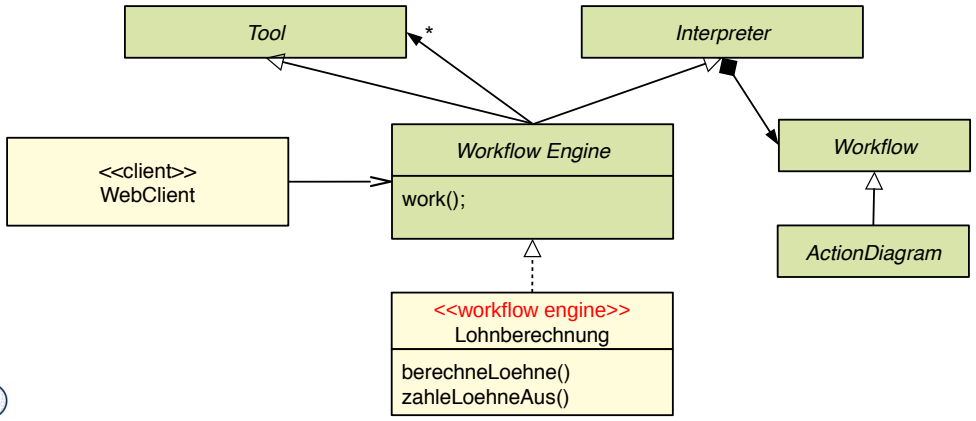
- **Technische Tools** sind funktionale Tools, die eine technische Funktionalität tragen, die nicht anwendungsunspezifisch ist
  - Bsp.: Editor, Lister, Inspektor, Browser, Verschlüsseler, Komprimierer, Optimierer
- Technische Tools verwalten das Material und bilden eine C-Teilschicht direkt über der Materialschicht



[Züllighoven]

Workflow-Engine-Klassen und -Schnittstellen

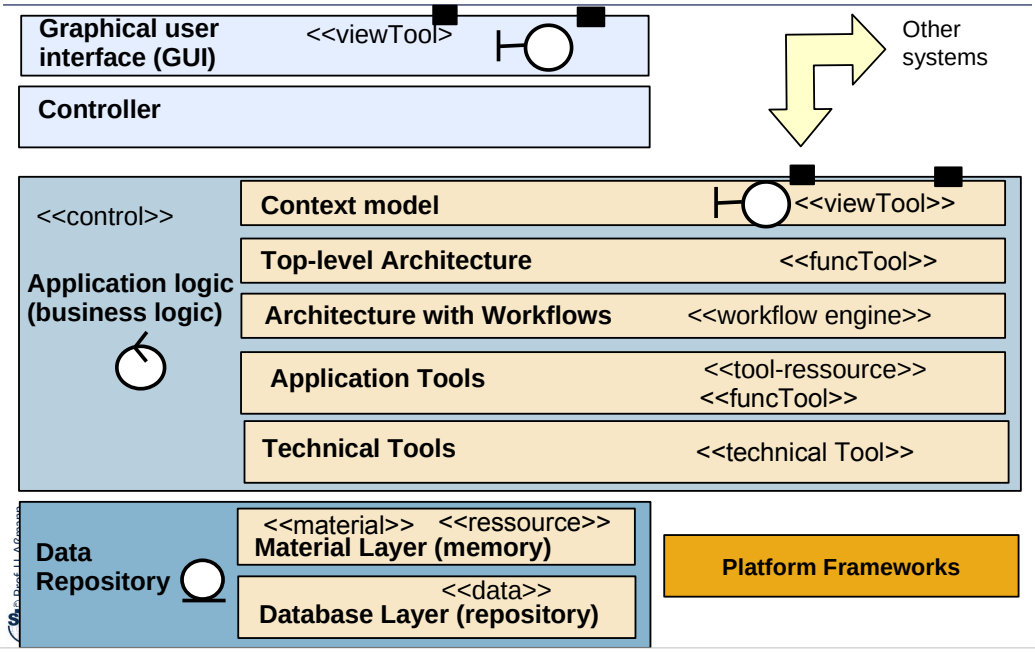
- Def.: Eine **Workflow-Engine** ist ein funktionales Tool, das einen komplexen Arbeitsablauf (Workflow) abarbeitet (interakiv oder batch) und andere Tools ansteuert
  - Workflow-Engines rufen andere Tools auf, setzen also auf ihnen auf
- Workflows können durch Aktionsdiagramme (Aktivitätendiagramm, Statechart, BPMN) beschrieben werden
- Workflow-Engines gehören zu C-Schicht



Frage: Wie ordnet man TAM-klassifzierte Objekte den BCED-Schichten zu?

- Die TAM-Klassifikation erlaubt uns, Klassen bestimmten Schichten der Anwendung zuzuordnen.

Q8: Verfeinerte BCED-Schichtung eines Systems mit TAM



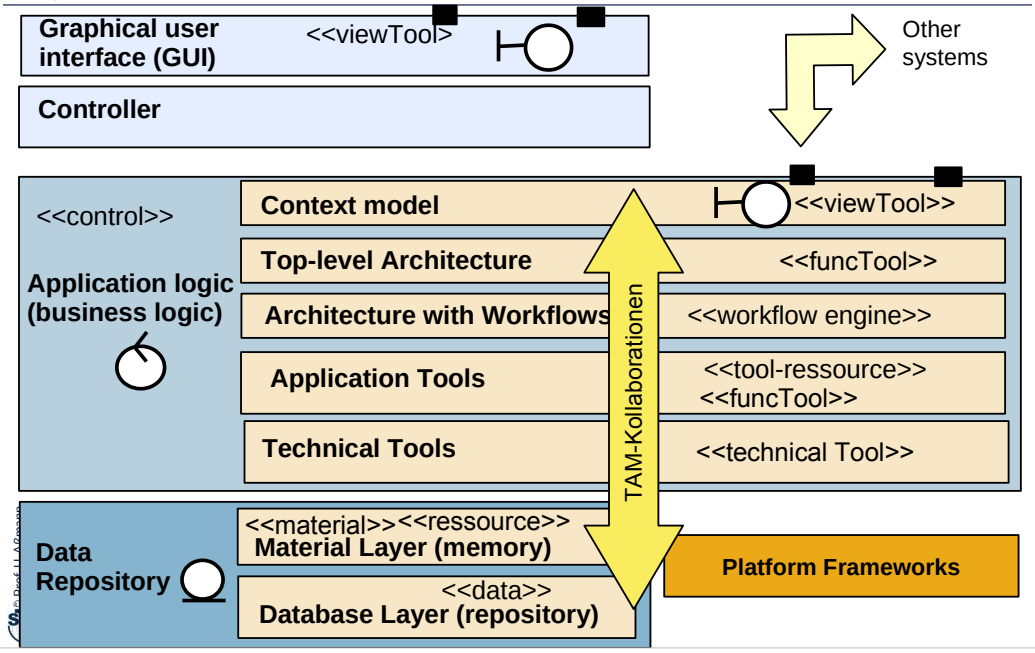
42.2 Querscheidende Verfeinerung in der BCED-Schichtenarchitektur mit TAM

.. Verfeinerung durch TAM-Kollaborationen

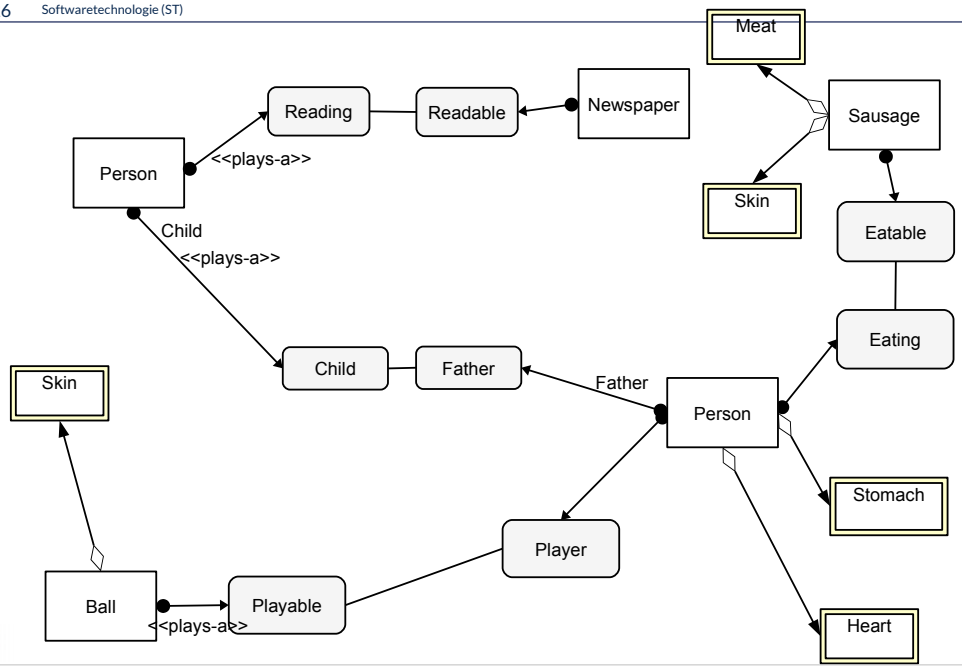
Tools gehören zur Anwendungslogik

Materialien und Ressourcen in die Datenhaltung

Q8: Verfeinerte BCED-Schichtung eines Systems mit TAM

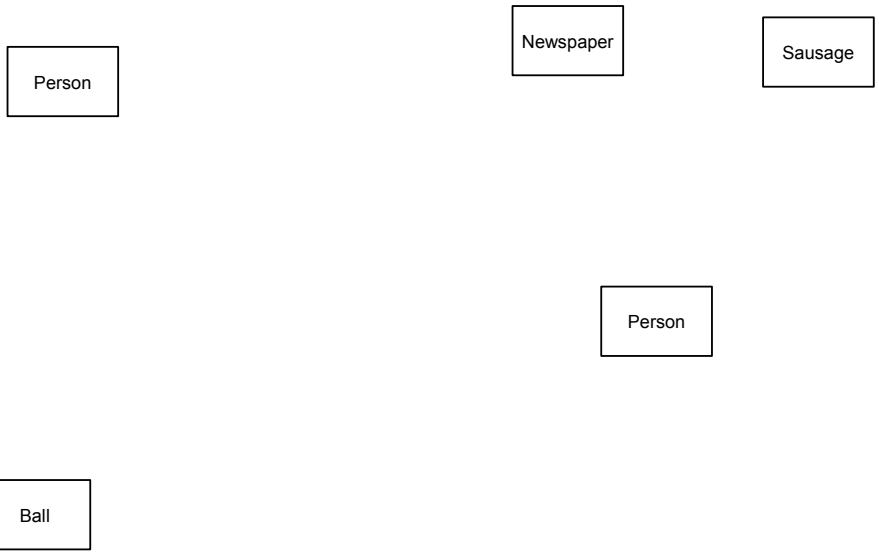


Personen-Analysemodell mit Rollenobjekten und Teilen – Wie teile ich das auf die Schichten auf?



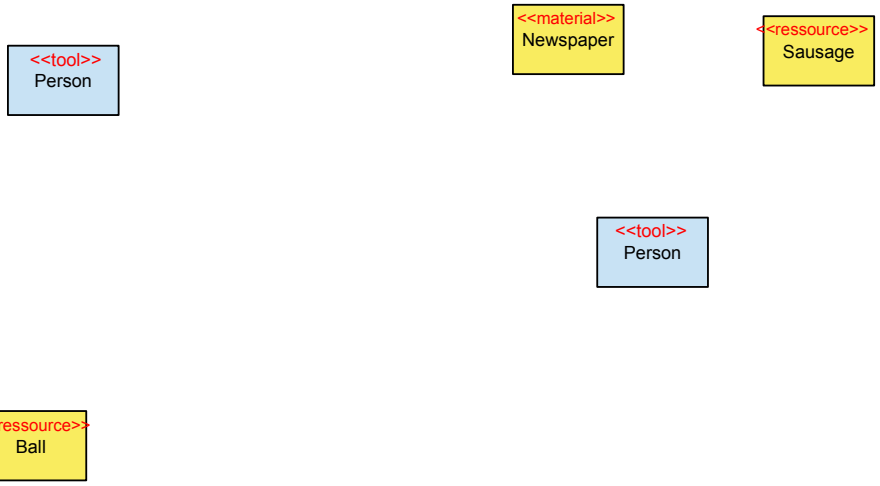
Mit Verfeinerung durch Integration von Unterobjekten (Objektanreicherung, Object Fattening)

- Rohzustand: Identifikation der natürlichen Typen (in dem Domänenmodell)



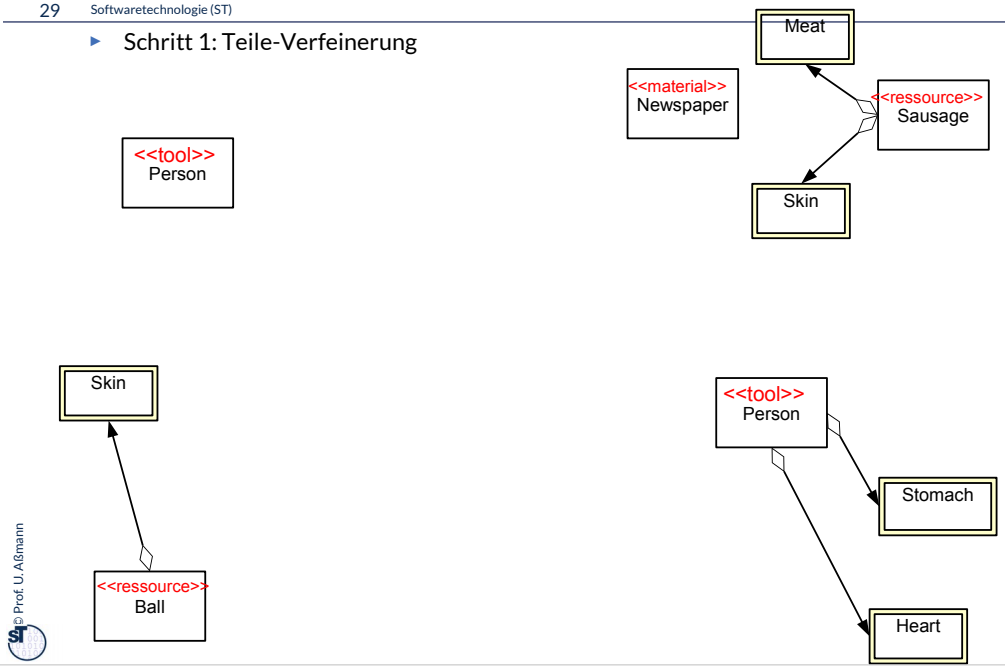
Mit Klassifikation von Tools and Materials

- Bestimme Tools, Workflows (C-Schicht, Application Logic)
- Bestimme Materials, Ressourcen (D-Schicht)



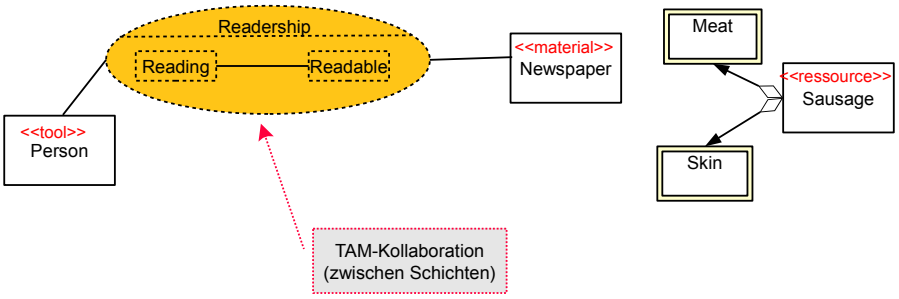
Mit Verfeinerung durch Integration von Unterobjekten (Object Fattening)

- Schritt 1: Teile-Verfeinerung



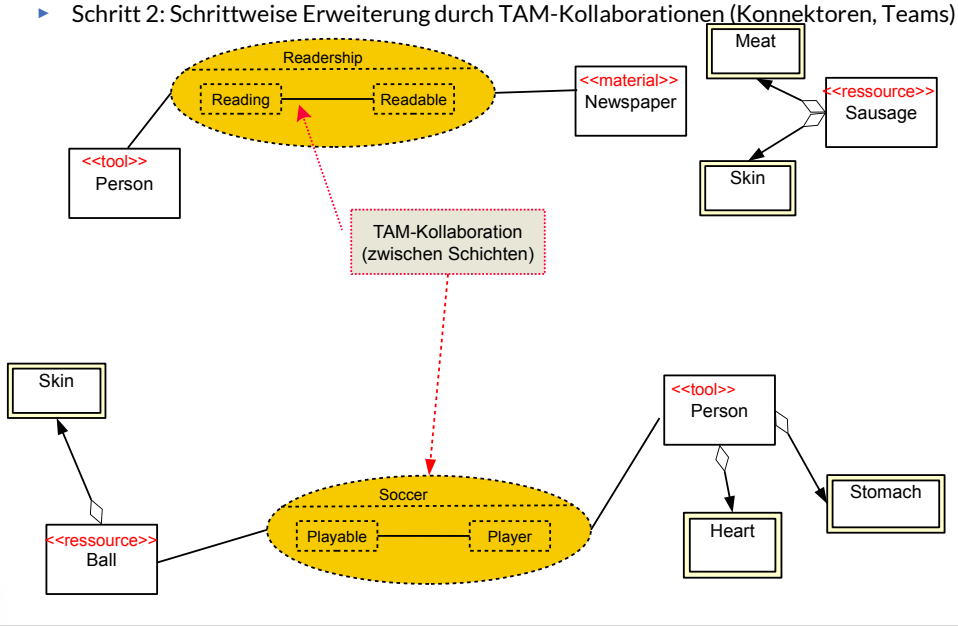
Mit Querschneidender Verfeinerung durch TAM-Kollaborationen

- Schritt 2: Schrittweise Erweiterung durch **TAM-Kollaborationen**



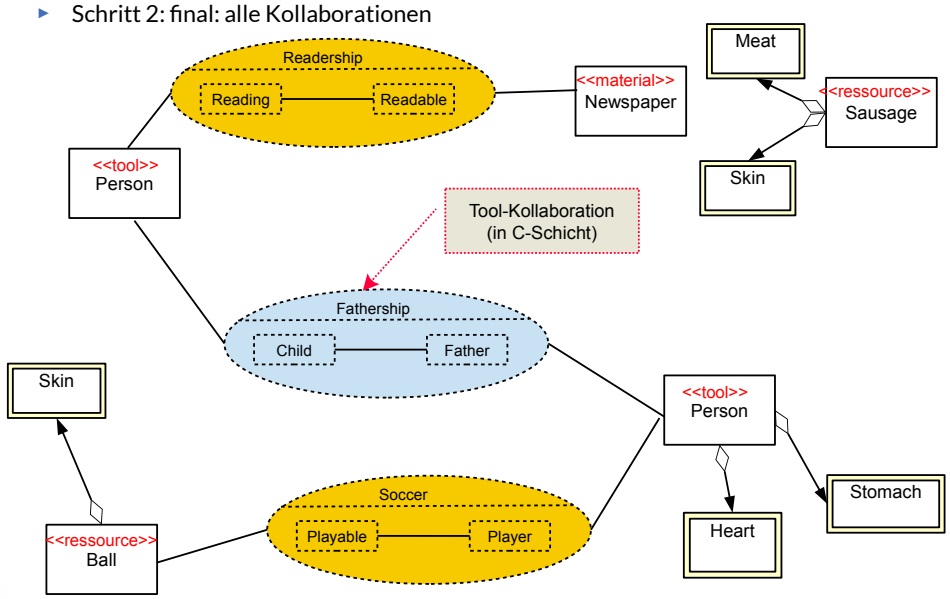
Mit Querschneidender Verfeinerung durch TAM-Kollaborationen

- Schritt 2: Schrittweise Erweiterung durch TAM-Kollaborationen (Konnektoren, Teams)

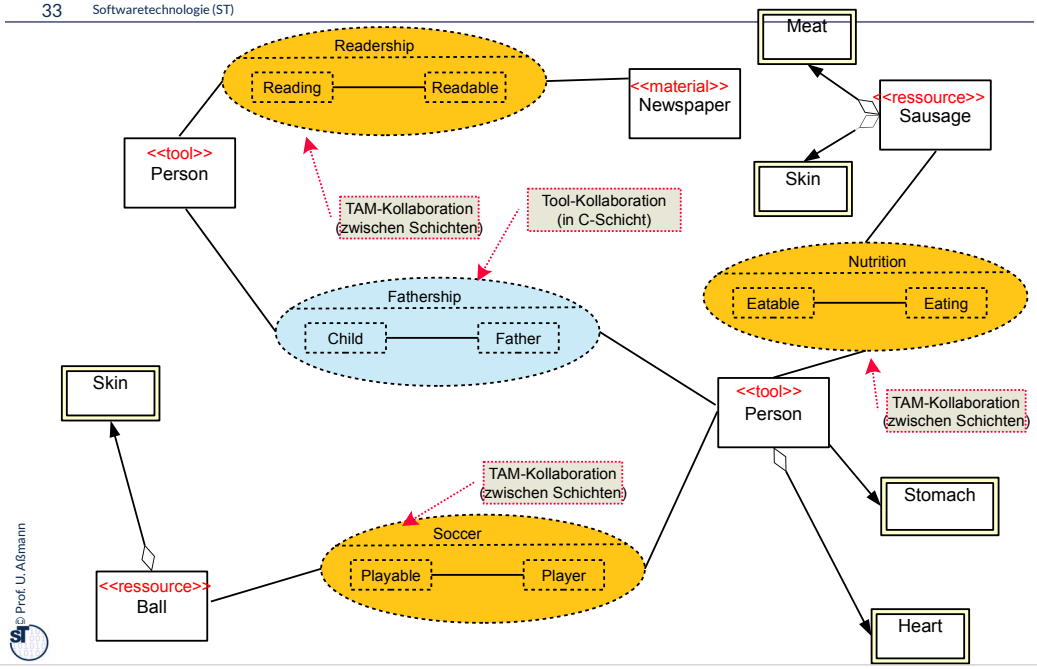


Mit Querschneidender Verfeinerung durch Tool-Tool-Kollaborationen

- Schritt 2: final: alle Kollaborationen



Analysemodell – Angereichert durch Einziehen von querschneidenden Kollaborationen



42.3 Plattformanpassung mit Plattformkollaborationen

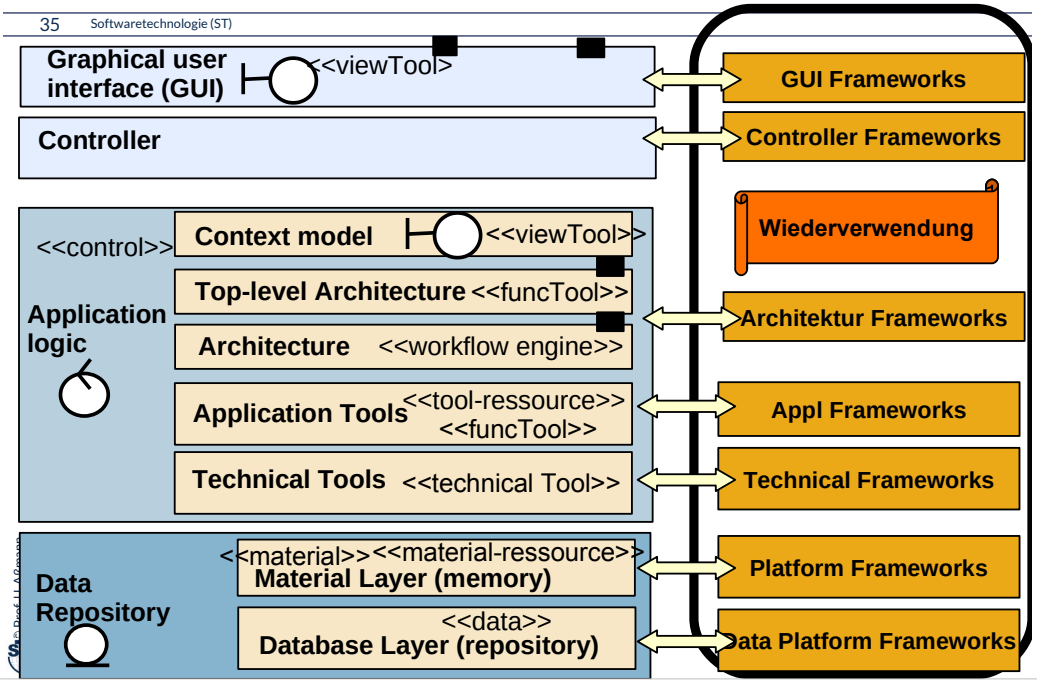
Verfeinerungsbeispiel für Anpassung auf Plattformen

.. Verfeinerung durch Integration von Unterobjekten..

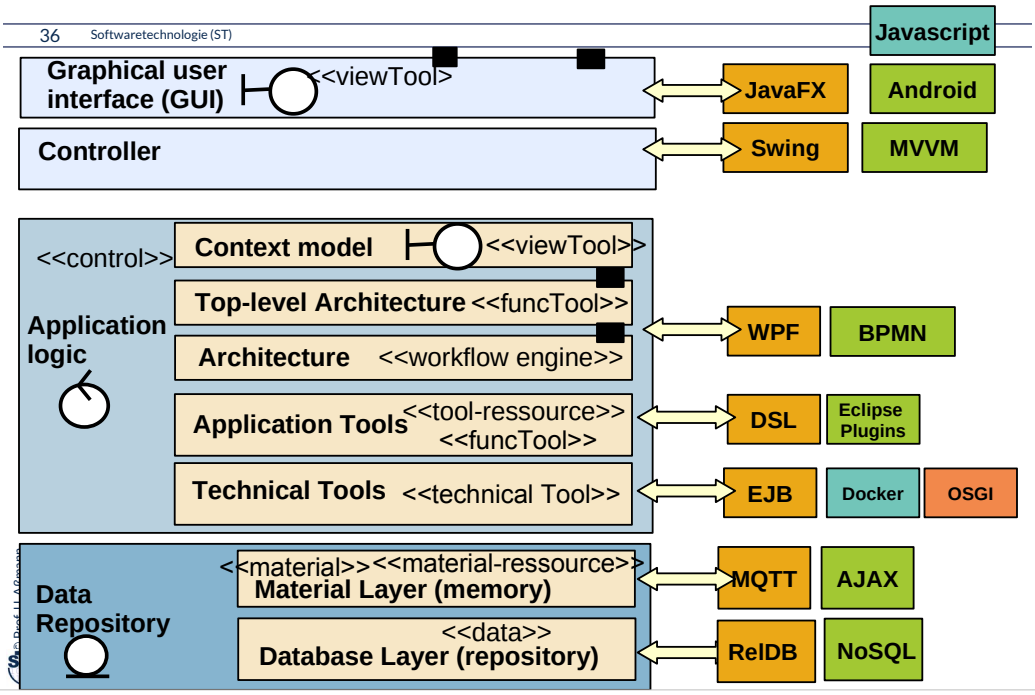
Teile und Rollen zu Plattformobjekten hin

- in 2019 optional -

Q7': Verfeinerte BCED-Schichtung eines Systems mit TAM und Technik-Plattform-Frameworks



Plattform-Wechsel immer und überall – wie beherrschen?

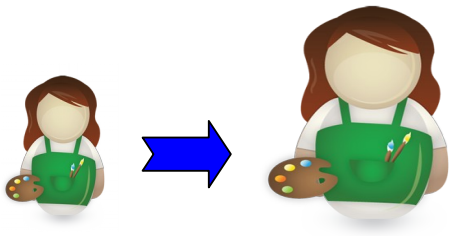


42.3.1 Anpassung von Material-Objekten an Plattformen
Objektanreicherung mit Plattforminformation (Querschneidende Verfeinerung für Plattformen)

.. Verfeinerung durch Integration von Unterobjekten..

Plattform-Verfeinerung

- Def.: **Plattform-Verfeinerung** ist ein Objektanreicherungs-Prozess zur Entwurfszeit, der Konnektoren mit plattform-spezisches Verhalten ergänzt (Plattform-Verfeinerung)
- Die hinzugefügten Kollaborationen und Konnektoren mit ihren Rollen klären Beziehungen zu
  - Plattformen (Betriebssystem, Middleware, Sprachen, Komponenten-services)



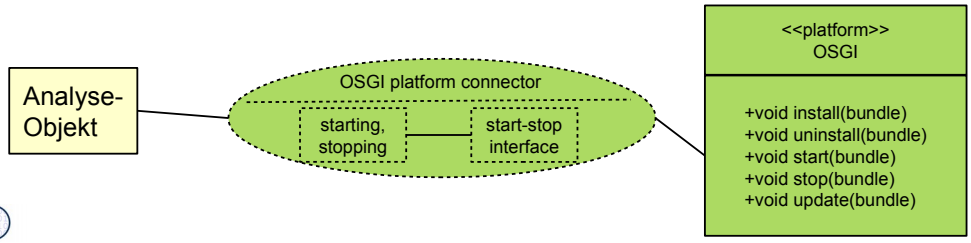
Plattformverfeinerung – Schritte im Entwurf

- Identifiziere jede **Plattform**
  - Bibliotheken und Frameworks, auf denen aufgebaut wird
  - Betriebssystem, Datenbanken, Middleware
- Finde **Plattform-Konnektoren,** mit Rollen-Unterobjekten, 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
- **Realisierung** der Konnektoren und der Integrationsrelation
  - Einfache Implementierung durch Konnektoren oder Entwurfsmuster

•Teile- und Rollenverfeinerung von Großobjekten startet schon in der Analyse und geht im Entwurf weiter

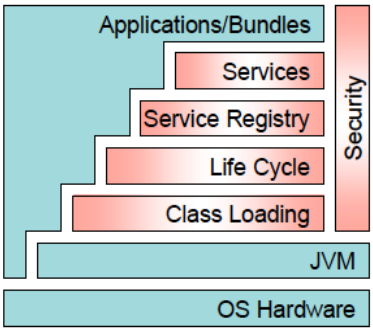
Plattformobjekte und Plattform-Konnektoren

- Def.: Ein **Plattformobjekt** ist ein Objekt eines Plattform-Frameworks, das wesentliche Laufzeitfunktionalität bietet und auf die eine Software angepasst werden muss
  - Bietet Schnittstelle an bzgl. bestimmter Funktionalität, z.B. abstrakte Maschine (Interpretierer)
  - Variabel: je nach Maschine, Middleware, Betriebssystem, Datenbank, Programmiersprache unterschiedlich ausgeprägt
- Def.: Eine **Plattformkollaboration** kapselt die Kollaboration eines Anwendungsobjekts mit der Plattform durch einen Konnektor zum Plattformobjekt
- OSGI: Komponentenplattform www.osgi.org kann in einem OSGI-Objekt gekapselt werden
  - im Handy, 5er BMW, in Eclipse 3.0, Shell home automation HomeGenie
  - Ein *bundle (Komponente)* paketiert verschiedene Klassen



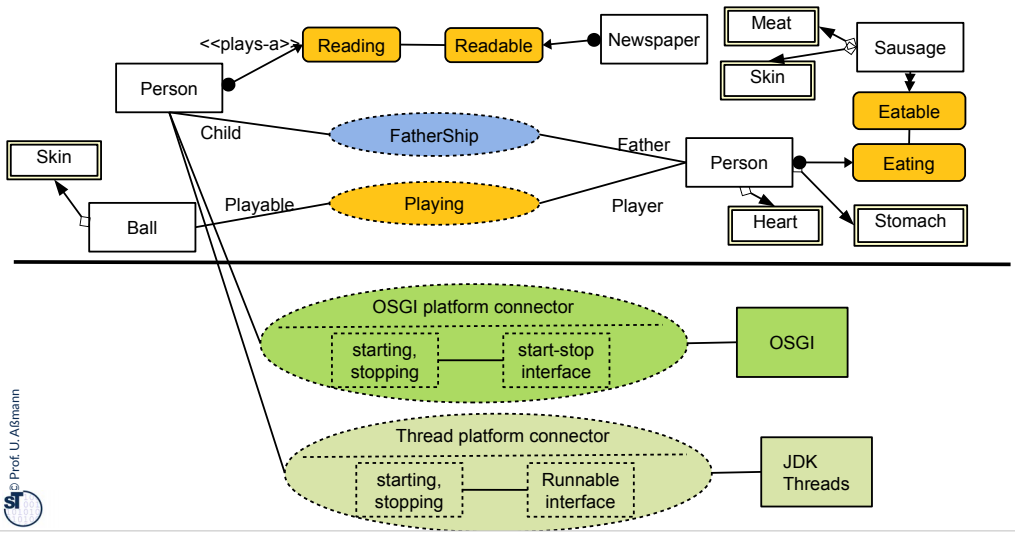
Plattformobjekt OSGI

- 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]



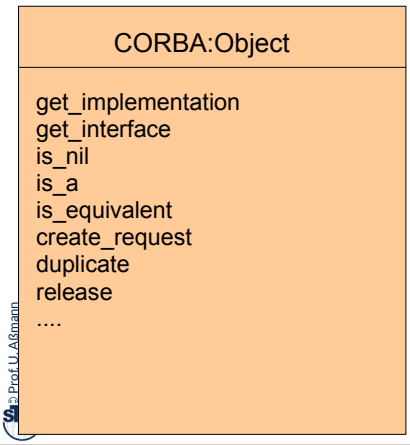
Mit Verfeinerung durch Plattform-Konnektoren (platform fattening)

- 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

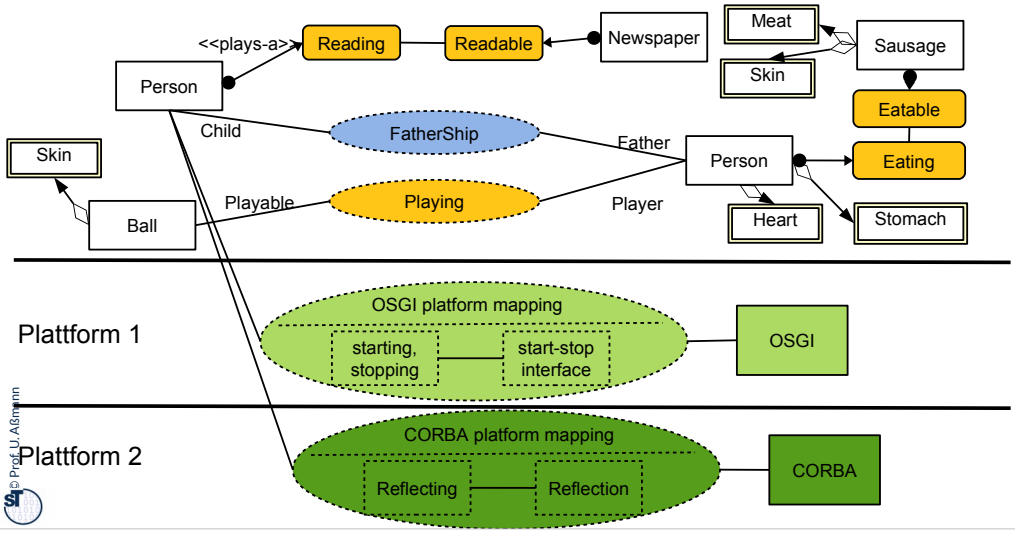
- CORBA bildet eine Komponentenplattform für heterogen programmierte Systeme



- 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
- Plattformkollaborationen werden stufenspezifisch eingesetzt und können gegen Varianten ausgetauscht werden



Das Portabilitätsgesetz

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

Bei einer Portierung auf eine andere Plattform müssen I.d.R. für Datenhaltung und Anwendungslogik getrennt Plattformkollaborationen entwickelt werden.

Verfeinerungsgesetze im Entwurf

Querscheidende Verfeinerung im Entwurf nutzt **Tool-Tool-Kollaborationen**, **Tool-Materal-(TAM-)Kollaborationen, Plattform-Kollaborationen.**

Man kapselt **Tool-Tool-Kollaborationen**, um sie auf der C-Schicht (Anwendungslogik) später variieren und erweitern zu können.

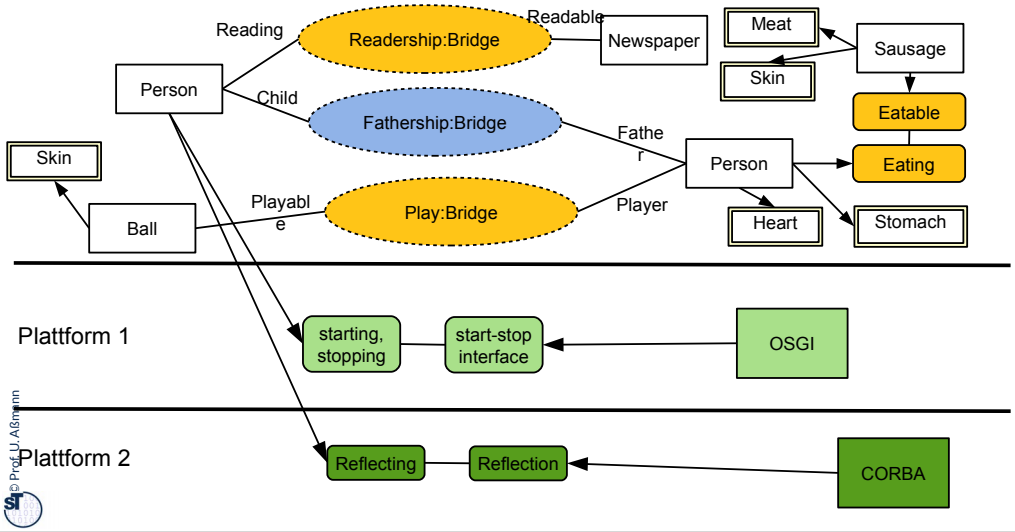
Man kapselt **Tool-Material-Kollaborationen**, um sie zwischen der C-Schicht (Anwendungslogik) und der D-Schicht (Data) später variieren und erweitern zu können.

42.4 Abbildung der Kollaborationen auf klassische Programmiersprachen

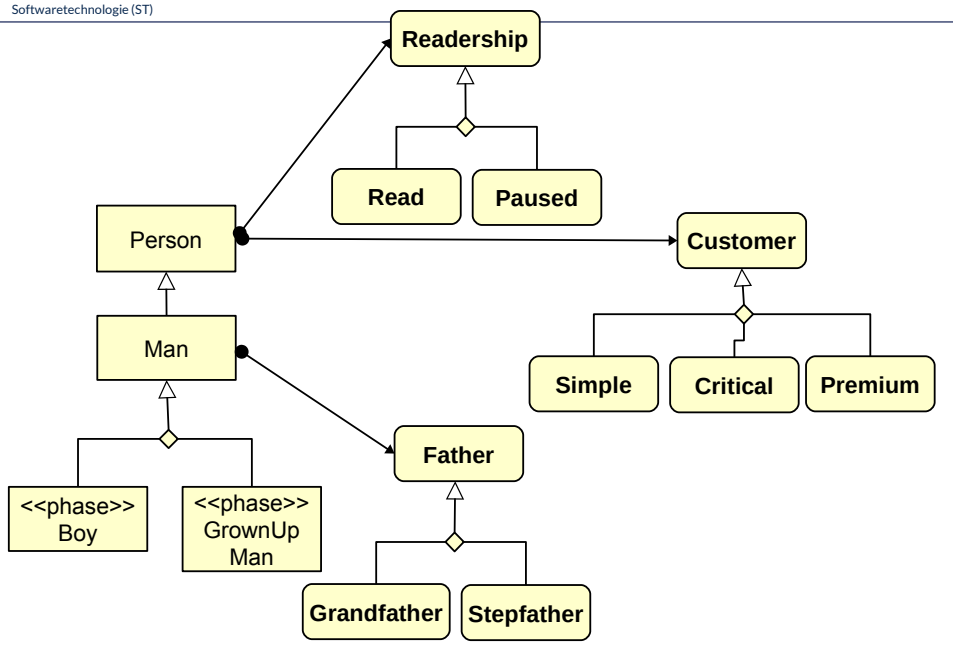
.. in der Implementierung ..

a) Wie bilde ich “plays-a” durch Multi-Bridge (Delegation) ab?

- Ersetze alle “plays”, “mandatory-part”, etc. durch Entwurfsmuster Bridge und Multi-Bridge (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

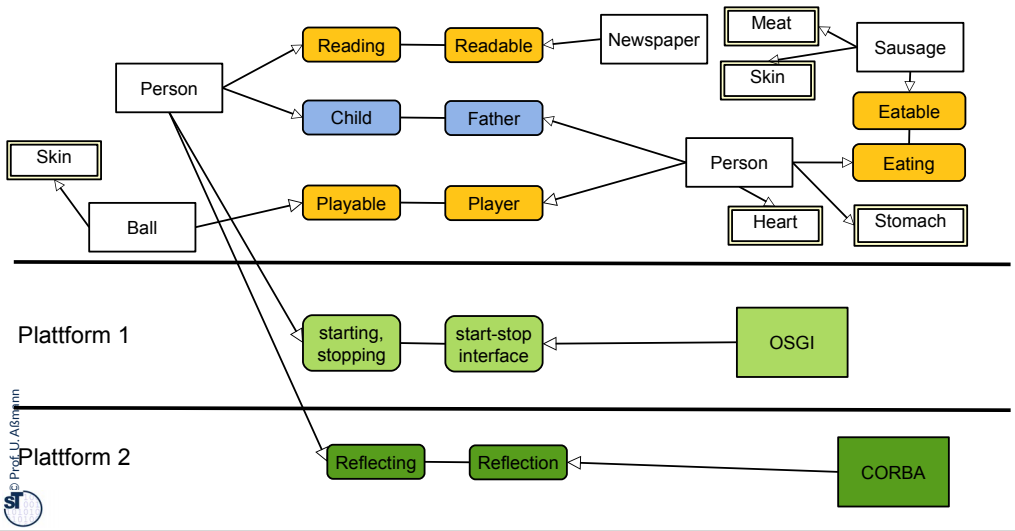


Erinnerung: Realisierung von Rollen mit Multi-Bridge



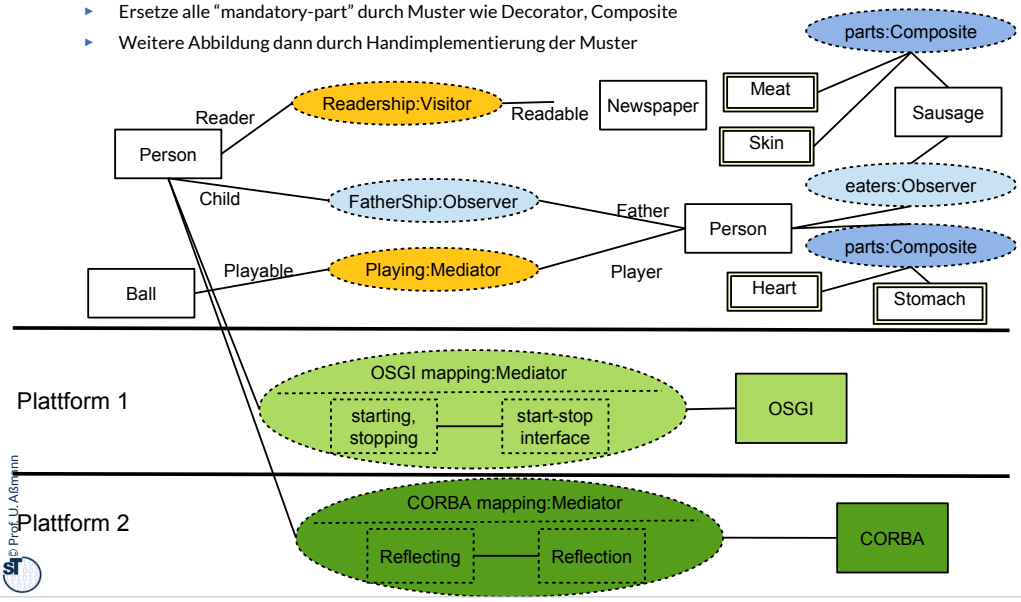
b) Wie bilde ich “integrates-a” durch Vererbung ab?

- Ersetze alle “plays”, “mandatory-part”, etc. durch Vererbung (Mehrfachvererbung oder “mixin inheritance”)



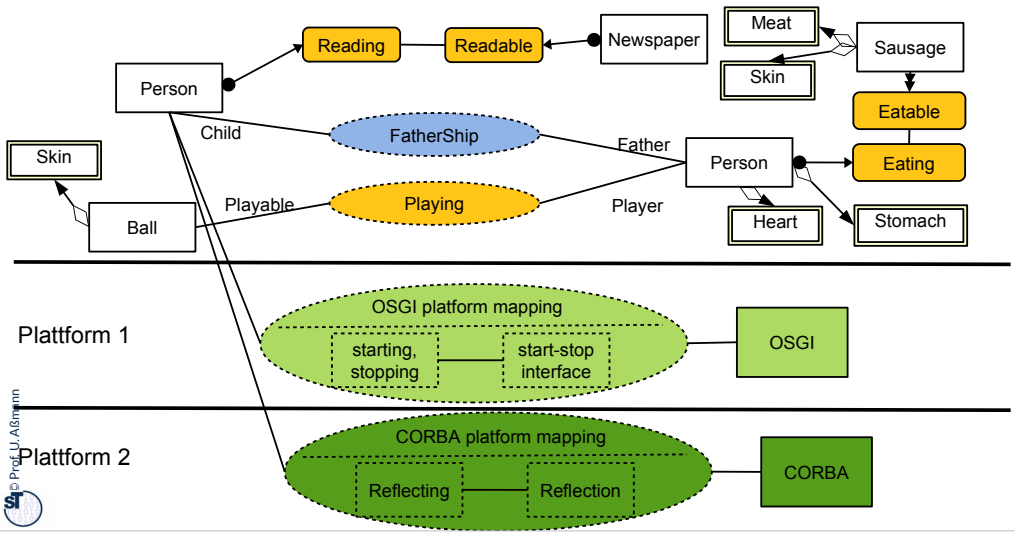
c) Wie bilde ich “plays-a” durch Implementierungsmuster ab?

- Ersetze alle “plays”, etc. durch Muster wie Observer, Visitor
- Ersetze alle “mandatory-part” durch Muster wie Decorator, Composite
- Weitere Abbildung dann durch Handimplementierung der Muster



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

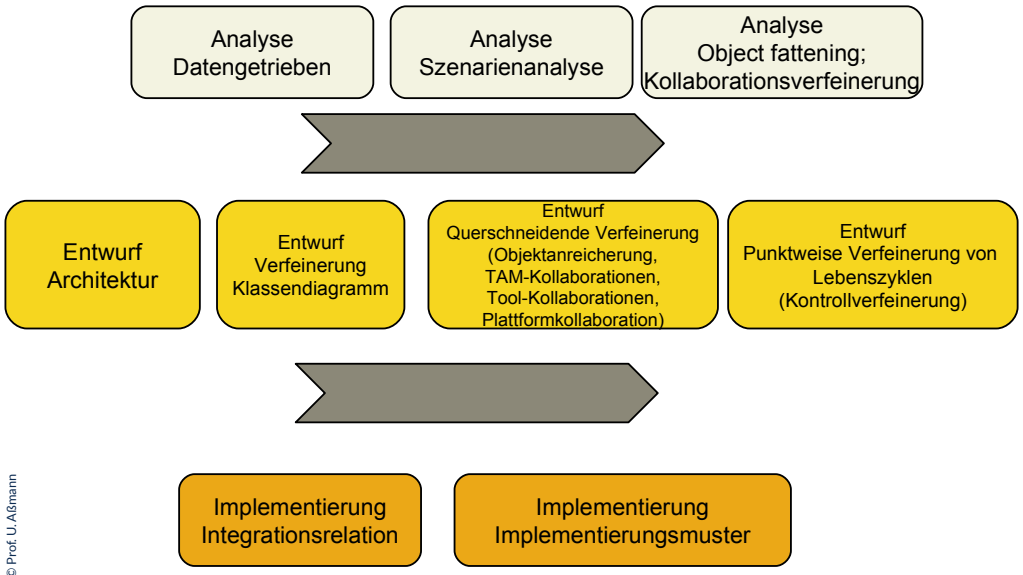
- Kollaborationen und die “plays-a”-Relation können auf eine Rollen-Programmiersprache abgebildet werden
  1) wie ObjectTeams.org; dann liegt die Abbildung im Übersetzer



e) 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

42.5 Gesamtbild der Verfeinerung



The End

- Gehen Sie im Geiste zurück auf die Szenarienanalyse aus Teil III. Nutzen Sie die TAM-Stereotypen, um die dort analysierten Klassen in eine Schicht einzuordnen. Was werden Tools? Was Materials? Was Workflows?
- Warum ist eine Trennung von Tool und Material auf verschiedene Objekte in verschiedenen Schichten sinnvoll?
- Was unterscheidet eine TAM-Kollaboration von einer Tool-Kollaboration?
- Was unterscheidet eine TAM-Kollaboration von einem Tool-Konnektor?
- Was unterscheidet eine TAM-Kollaboration von einer Plattform-Kollaboration?
- Was unterscheidet eine Plattform-Kollaboration von einem Plattform-Konnektor?
- Wie kann man die Materialien testen? Wie die Tools?
- Wieso muss man Ressourcen-Materialen zuteilen und sperren?
- Wann entsteht aus einer Bridge einer Collaboration eine Multi-Bridge?
- Wieso will man eine Software auf andere Plattformen portieren?
- Geben Sie zwei Realisierungen für eine UML-Kollaboration mit “plays-a”-Links an. Vergleichen Sie deren Vor- und Nachteile

Anhang A: Nebenbemerkung

- Integration von Unterobjekten in Kernobjekte kann *zu verschiedenen Zeiten* erfolgen
  - Zur Entwurfszeit
  - Zur Bindezeit
  - Zur Allokationszeit eines Objekts
  - Zur Laufzeit
  - Zur Zeit der Software-Pflege und -Migration