46 Softwarearchitektur mit dem Quasar-Architekturstil

Prof. Dr. U. Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
http://st.inf.tu-dresden.de
Version 11-0.1, 10.07.11

- Optionales Material

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

1) Einführung in die objektorientierte Softwarearchitektur
   1) Modularität und Geheimnisprinzip
   2) Entwurfsmuster für Modularität
   3) BCD-Architekturstil (3-tier architectures)
2) Verfeinerung des Entwurfsmodells zum Implementierungsmodell (Anreicherung von Klassendiagrammen)
   1) Verfeinerung von Operationen
   2) Verfeinerung von Assoziationen
   3) Verfeinerung von Vererbung
3) Verfeinerung von Lebenszyklen
   1) Verfeinerung von verschiedenen Steuerungsmaschinen
   2) Querschneidende Verfeinerung mit Chicken Fattening
4) Objektorientierte Rahmenwerke (frameworks)
5) Softwarearchitektur mit dem Quasar-Architekturstil

Sekundäre Literatur

- Johannes Siedersleben (ed.). Quasar: Die sd&m Standardarchitektur. http://www.openquasar.de
- Johannes Siedersleben. Moderne Softwarearchitektur. Umsichtig planen, robust bauen mit Quasar. dpunkt-Verlag, 2004.

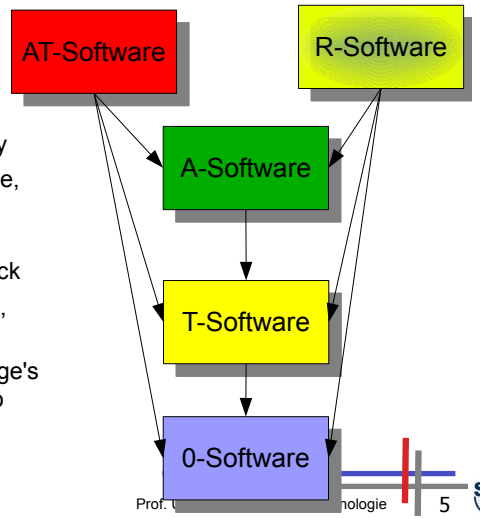
Quasar

- An architectural style of SD&M, the leading German software house for individual software
  - Software categories (blood groups)
  - Component orientation
  - A-TI-I architectural style
  - Component-oriented development process
- Bisher kannten wir 2 Aspekte von Software, Architektur und Anwendung. Jetzt unterscheiden wir zusätzlich *Technik*

Software Blood Groups (Blutgruppen)
(Softwarekategorien nach Wiederverwendbarkeit)

- 0: independent of application and technology
  - JDK collections, C++ STL, GNU regexp
- A: application- or domain-related. Stems from domain model.
  - Client, Customer, ...
- T: technology-oriented APIs, independent of application, but not of technology
  - JDBC, CORBA CosNaming
- AT: depending on application and technology
  - To be avoided: hard to maintain, to reuse, to evolve
- R: for representation changes of business objects into external representations and back
  - Serialization, deserialization, encryption, decryption, packing, unpacking
  - Transporting an object from one language's representation to anothers (e.g., Java to Cobol)

- USES relationships:



Architectural Components

- 0-interfaces contain only technical types (strings, collections etc)
  - well reusable
- A-interfaces contain domain types (account, bill,..)
  - A-components live in the Application-Logic and the Database tier
  - Hard to reuse
- T-interfaces provide techical APIs
  - Necessary everywhere
- R-interfaces contain both, because they change representation
  - Are necessary in the middleware and data layer
  - Special kind of A
  - Can often be generated from specifications, hence not reusable, but re-generatable
    - XML tools, e.g., XMI (model interchange)
    - OMG MOF tools (Model-driven development)

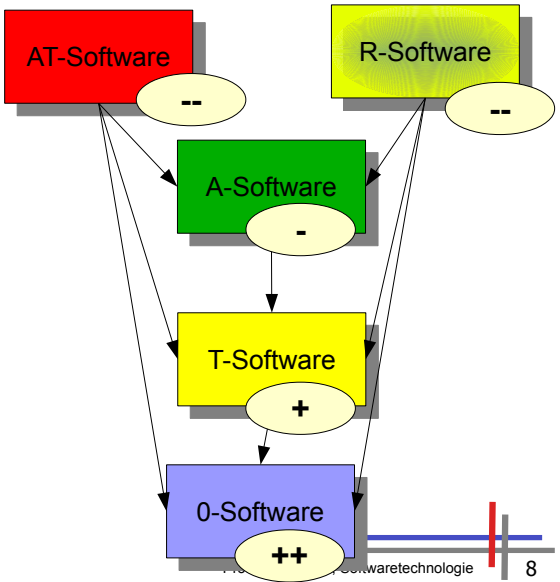
Zweck der Blutgruppen

- Zweck der Blutgruppen ist es, Anwendung und Technik möglichst weit voneinander zu trennen, um sie getrennt besser wiederverwenden zu können.

Siedersleben's Blutgruppen-Gesetz:
Jede Schnittstelle, Klasse, und Komponente gehört genau zu einer Softwarekategorie.

Wiederverwendbarkeit der Blutgruppen

- Die Wiederverwendbarkeit der Gruppen nimmt von 0 nach AT hin ab.
  - Technisch orientierte Komponenten sind leichter wiederzuverwenden
  - Anwendungsspezifische schwerer.
  - Problemfall AT
- Die Blutgruppen durchziehen alle Schichten der BCED-Architektur
  - Auf jeder Ebene gibt es Technik, Applikation, Repräsentation



Blutgruppen-Gesetze

- Der Aufruf von A-Komponenten aus T-Komponenten heraus ist gefährlich
  - Die azyklische USES-Beziehung von A nach T wird zerstört
  - Es entsteht AT-Software
- Blutgruppen-Kalkül:
  - A+0 = A
  - T+0 = T
  - A + T = AT

- Aufruftabelle:

	0	A	T	R
0	0	A	T	R
A		A	AT	
T			T	
R				R

Siedersleben's AT-Gesetz:
Mischen von A und T führt immer zu sehr schlecht wiederverwendbarer Software.

Was haben wir gelernt?

- Jenseits der Begriffe Architektur/Anwendung und BCED kann man die Softwarekomponenten in Blutgruppen einteilen (A, T, 0, R, AT)
- Vermeide das Vermischen von bestimmten Gruppen (A und T), denn AT-Software ist schlecht wiederverwendbar
- Sortiere alle Pakete/Komponenten in der BCED-Architektur nach Blutgruppen und vermeide Vermischung!

The End