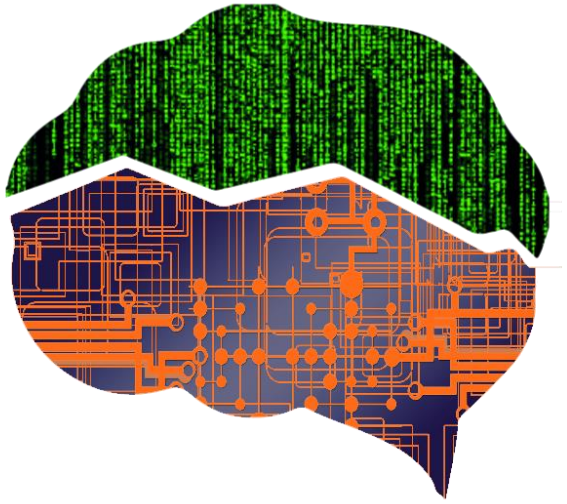
Software Product Line Engineering

Big Picture

Christian Kästner (Carnegie Mellon University)
Sven Apel (Universität Passau)
Norbert Siegmund (Bauhaus-Universität Weimar)
Gunter Saake (Universität Magdeburg)



Bauhaus-Universität Weimar

What have we learned?

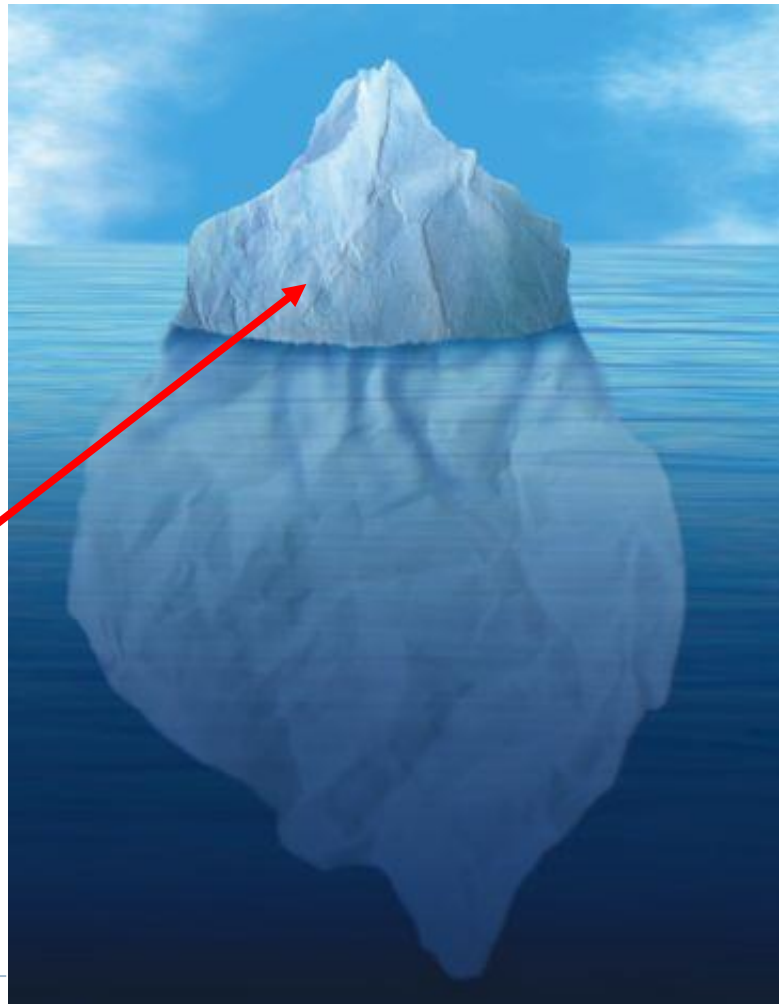
- Basic on software product line engineering
- Classic implementation (parameter, #ifdefs, framework, components…)
  - …and their limitations
- Direct language support for features
  - Collaborations, rolls, aspects, etc.
- Handling of feature interactions
- Non-functional properties
- …

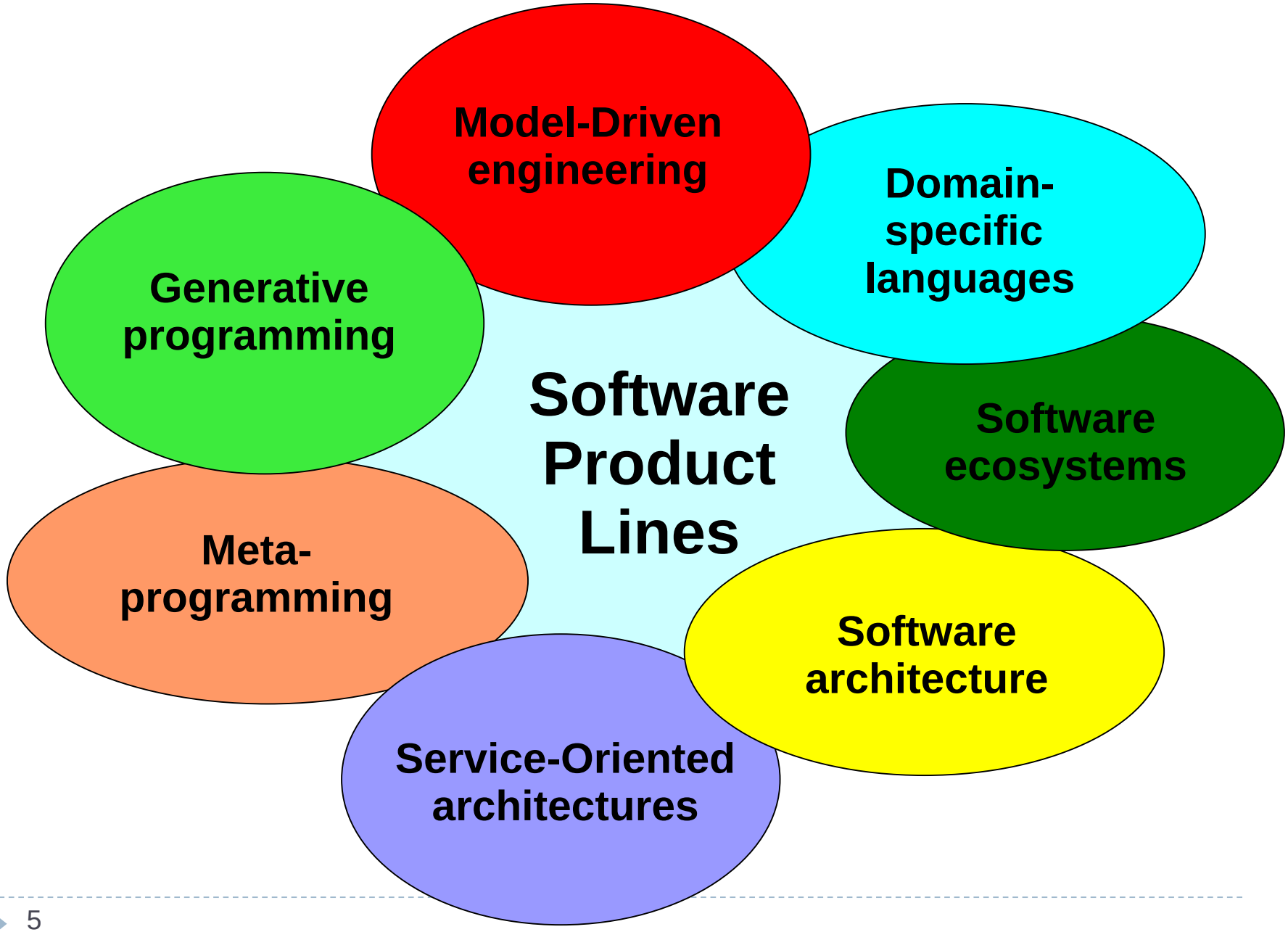
Additional Topics

- Domain analysis, scoping, requirements engineering
- ROI scoring, product management, market analysis, risk management
- Organization
  - Project initialization, financing
  - Organization planning, rolls, responsibilities
  - Process
- Validation and verification of code

Perspective?

- How do the topics of the course relate to software development in general?




Model-Driven engineering
Domain-specific languages
Generative programming
Software Product Lines
Software ecosystems
Meta-programming
Software architecture
Service-Oriented architectures

Software Ecosystems

- Open platforms
- Everyone can supply extensions
- Tailor-made systems

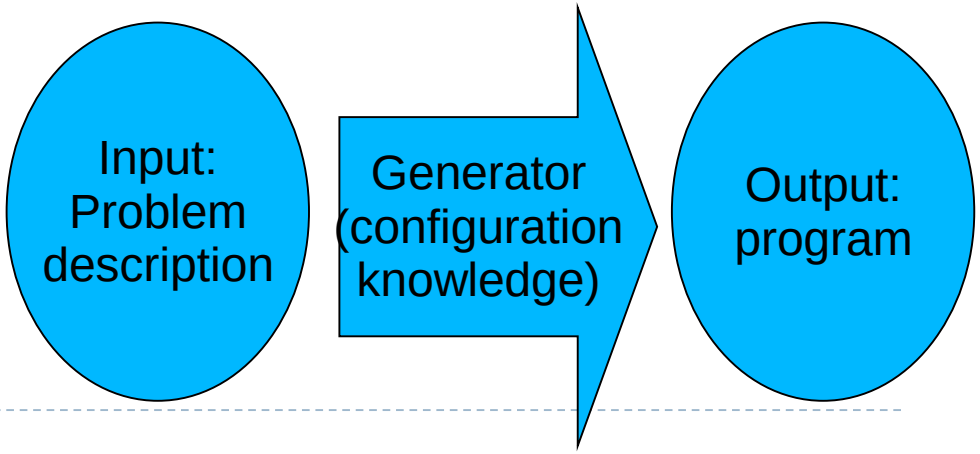
- Closed variability modeling and product line analyses not possible

Generative Programming

- Given a specification software will be generated automatically based on existing code fragments

  Generative programming (GP) is a style of computer programming that uses automated source code creation through generic classes, prototypes, templates, aspects, and code generators to improve programmer productivity.

- Application scenarions
  - Form generators
  - Compiler-Compiler
  - Parser generators
  - Query optimization
  - …



Generative Programming II

- Connection to the lecture
  - Feature selection ⇔ Abstract input program
  - Feature modules ⇔ Code skeletons / patterns
  - Feature composition ⇔ Generation
- GP can do also…
  - More complex input than simple feature selection (e.g., form description, grammar, domain-specific language.,…)
  - Any king of generator
  - Meta-programming (programs manipulate programs)

Meta-programming

- Programs manipulate programs
- Separation in levels
  - Basic program, 1-n meta-level (meta programs)
- Reach program efficiency via abstraction
- GP with program as input ➔ Meta program
- AOP & FOP as meta-programming interpretable

Metaprogramming (MP) is the writing of computer programs that write or manipulate other programs (or themselves) as their data or that do part of the work during compile time that is otherwise done at run time. In many cases, this allows programmers to get more done in the same amount of time as they would take to write all the code manually.

Domain-specific Languages I

- Principle of abstraction
  - State the problem and its solution in a suitable language
  - Generator/Interpreter generates target code
- Advantages: less redundancy, better readability, less technical details, easy to learn, better error messages

```
Set camera size:
    400 by 300 pixels.
Set camera position:
    100, 100.
Move 200 pixels right.
Move 100 pixels up.
Move 250 pixels left.
Move 50 pixels down.
```

A domain-specific language (DSL) is a programming language designed for, and intended to be useful for, a specific kind of task. This is in contrast to a general-purpose programming language, such as C, or general-purpose modeling languages like UML.

Domain-specific Languages II

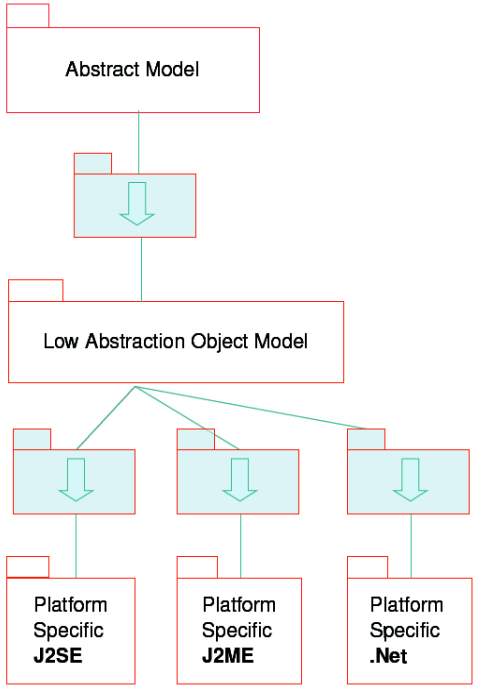
- Query languages
- Unix-Shell scripts
- Tabular calculation programs
- Regular expressions
- Document and data description languages
- Graph description languages
- Languages to describe forms
- etc.

Domain-specific Languages III

- DSL as an input to GP
- Connection to product lines:
  - Feature model ⇔ DSL
  - Feature ⇔ language construct in a DSL
  - Feature composition ⇔ translation process

Model-Driven Engineering I

- Software is described by models
- Abstract models are translated to concrete ones
- Transformation can be realized in multiple steps
- Model at the end of the transformation chain represents the implementation of the system (in form of code)

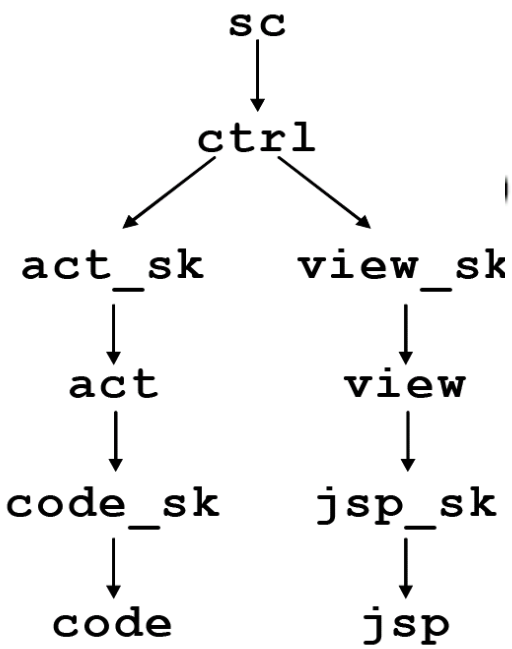


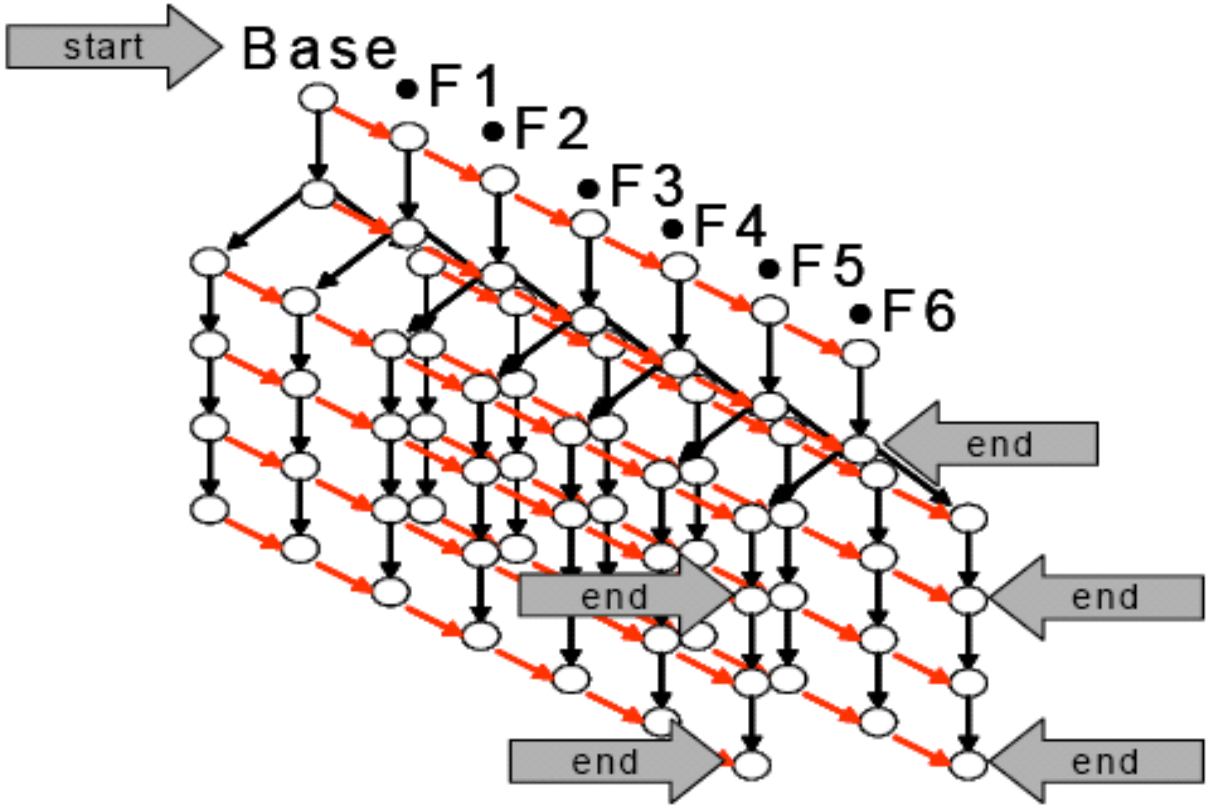
Model-Driven Engineering (MDE) or Model-Driven Development (MDD) refer to the systematic use of models as primary engineering artifacts throughout the engineering lifecycle. MDE/MDD can be applied to software, system, and data engineering. Models are considered as first class entities.

Model-Driven Engineering II

- MDE is a form of GP
  - Model as input, new model or code as output
- MDE as DSL
  - Modeling language for a specific domain, e.g., BPEL
- Connection to product lines from different views
  - A) Feature selection as model that will be transformed to source code (as in GP/DSL)
  - B) Features can refine models

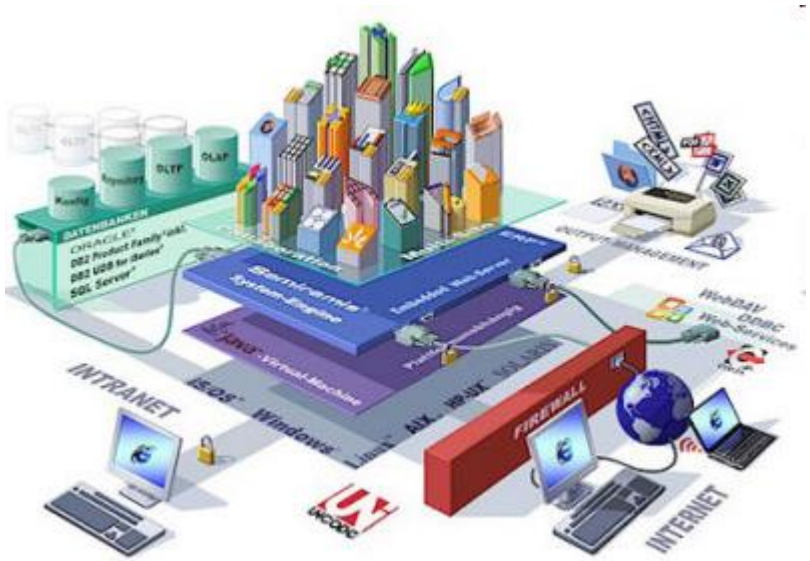
Feature-Oriented Model-Driven Development





Software Architecture I

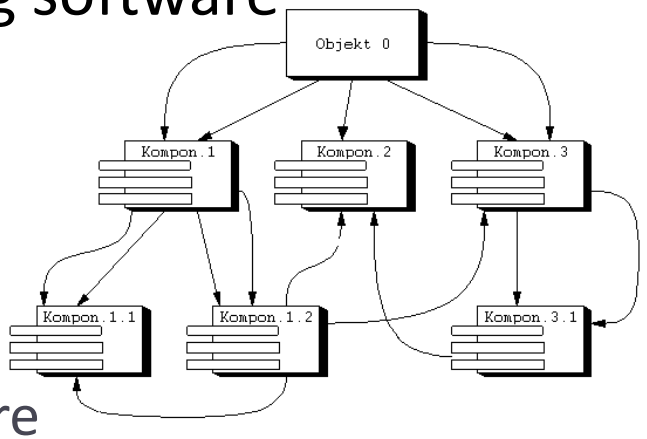
- Principle of abstraction
  - Consider the coarse software structures that are of relevance; ignore the irrelevant details
- Documentation and design are in focus
  - Components, Connectors, UML, Deployment, etc.



The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them.

Software Architecture + Features

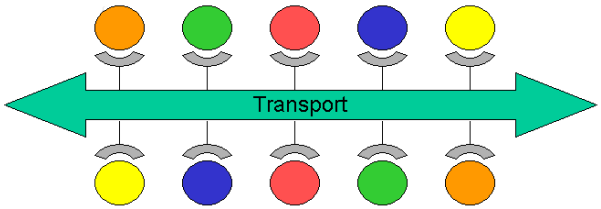
- Feature-orientations is a for structuring software architecture
  - Feature ⇔ Component
  - Glue-Code ⇔ Connector (connects two components)
  - Feature model ⇔ Product line architecture
  - Feature selection ⇔ Product architecture



- Advantage: “Gap” between design and implementation is closed, especially for crosscutting concerns

Service-Oriented Architectures

- Separation of concerns with services
- Services are distributed and language indepedendent
- Different communication and orchestration patterns

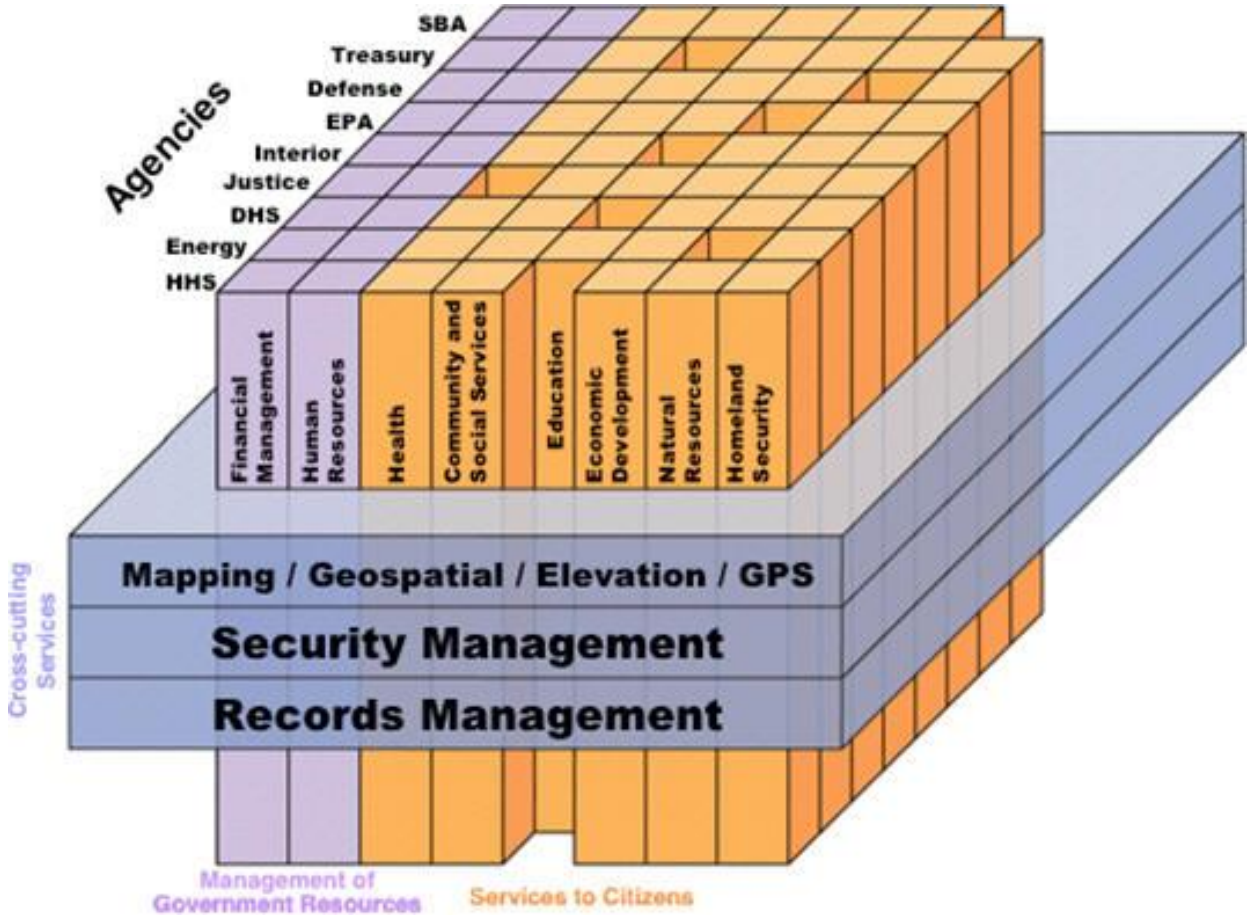


Service Oriented Architecture (SOA) is an architectural style that guides all aspects of creating and using business processes, packaged as services, throughout their lifecycle, as well as defining and provisioning the IT infrastructure that allows different applications to exchange data and participate in business processes loosely coupled from the operating systems and programming languages underlying those applications.

Service-Oriented Architectures + Features

- Connection of SOA and product lines (View #1)
  - Feature model ⇔ SOA
  - Feature ⇔ Service
  - Feature selection ⇔ Service lookup
  - Feature composition ⇔ Service integration
- Connection of SOA and product lines (View #2)
  - A feature affects the implementation of multiple services, i.e., crosscutting
  - Features as a mean to configure services

Crosscutting Features in SOA



Zusammenfassung

- Feature-Orientierung ist ein ganzheitlicher Ansatz zur Software-Entwicklung
- Verwandtschaft und Überlappungen mit mehreren aktuellen Entwicklungen in Forschung und Industrie
- Eine guter Startpunkt um in die Felder “Software Engineering” und “Programmiersprachen” einzutauchen, beruflich oder akademisch

Outlook

- Bachelor and master theses
- Seminars
- Hiwi positions
- Promotion (?)

Literature I (Software product lines in general)

- Paul Clements, Linda Northrop. Software Product Lines – Practice and Pattern. Addison-Wesley. 2002
- Klaus Pohl, Günter Böckle, Frank van der Linden. Software Product Line Engineering. Foundations, Principles, and Techniques. Springer. 2005.
- Günter Böckle, Peter Knauber, Klaus Pohl, Klaus Schmid (Hrsg.), Software-Produktlinien - Methoden, Einführung und Praxis. dpunkt. 2004

Literature II

- K. Czarnecki and U. Eisenecker. Generative Programming - Methods, Tools, and Applications. Addison-Wesley, 2000. [Standard literature of GP]
- J. Bartlett. The Art of Metaprogramming, Part 1: Introduction to Metaprogramming. IBM developerWorks, 2005.
  [Overview of meta-programming]
- M. Mernik, J. Heering, and A. Sloane. When and How to Develop Domain-Specific Languages. ACM Computing Surveys, 37(4), 2005.
  [Overview of DSL-Development]

Literature III

- L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice, Second Edition. Addison-Wesley, 2003
  [Standard literature to software architectures]
- D. Schmidt. Model-Driven Engineering. IEEE Computer, 39(2), 2006.
  [Introduction to MDE/MDD]
- T. Erl. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, 2005.
  [Overview about SOA]