# Software reference architectures: related architectural concepts, challenges, and (new?) domains

Matthias Galster

Department of Computer Science and Software Engineering

May 6, 2015

UC
UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

# Goals

Relate RA to other architectural concepts

Discuss high-level, non-technical challenges related to the design and use of RA

Explore domains that may benefit from RA



RA = “Reference Architecture” or “Reference Architectures”

# Take-away messages

"Reference architecture" is a fuzzy concept.

Reliance on RA may limit flexibility and innovation.

We may explore RA for cross-cutting domains.

This talk has three parts.

# Part I – architectural concepts

Software framework

Reference model

Reference architecture

Architecture framework

# Starting point: reference architecture

- Reusable architectural knowledge
  - Generic artifacts, standards, design guidelines, styles, vocabulary, etc.
  - Architectural best practices
- Not a highly specialized set of requirements
  - Used as starting points to specialize for own requirements
  - Used in and across organizations in a domain
- Why
  - Standardization, interoperability
  - Speed up development

E. Nakagawa et al.: *Reference architecture and product line architecture: a subtle but critical difference*. ECSA 2011
P. Kruchten: *The rational unified process: an introduction*. 2004
S. Angelov et al.: *A classification of software reference architectures: analyzing their success and effectiveness*. WICSA 2009
J. Governor et al.: *Web 2.0 architectures - what entrepreneurs and information architects need to know*. 2009

# Architecture frameworks?

# Architecture frameworks

- Focus on creating and using architecture description
  - Set of conceptually related viewpoints (according to ISO/IEC/IEEE 42010)
- Structure thinking and architecture description: layers or views
- Could be part of software reference architecture description
  - Generic and common vocabulary
  - But maybe not (application, development) domain-specific?
  - Often defined for high-level domains, free of detailed domain information

ISO/IEC/IEEE 42010

D. Emery, R. Hillard: *Every architecture description needs a framework: expressing architecture frameworks using ISO/IEC 42010*. WICSA 2009

G. Muller, P. van de Laar: *Researching reference architectures and their relationship with frameworks, methods, techniques, and Tools.* CSER 2009

# Viewpoints



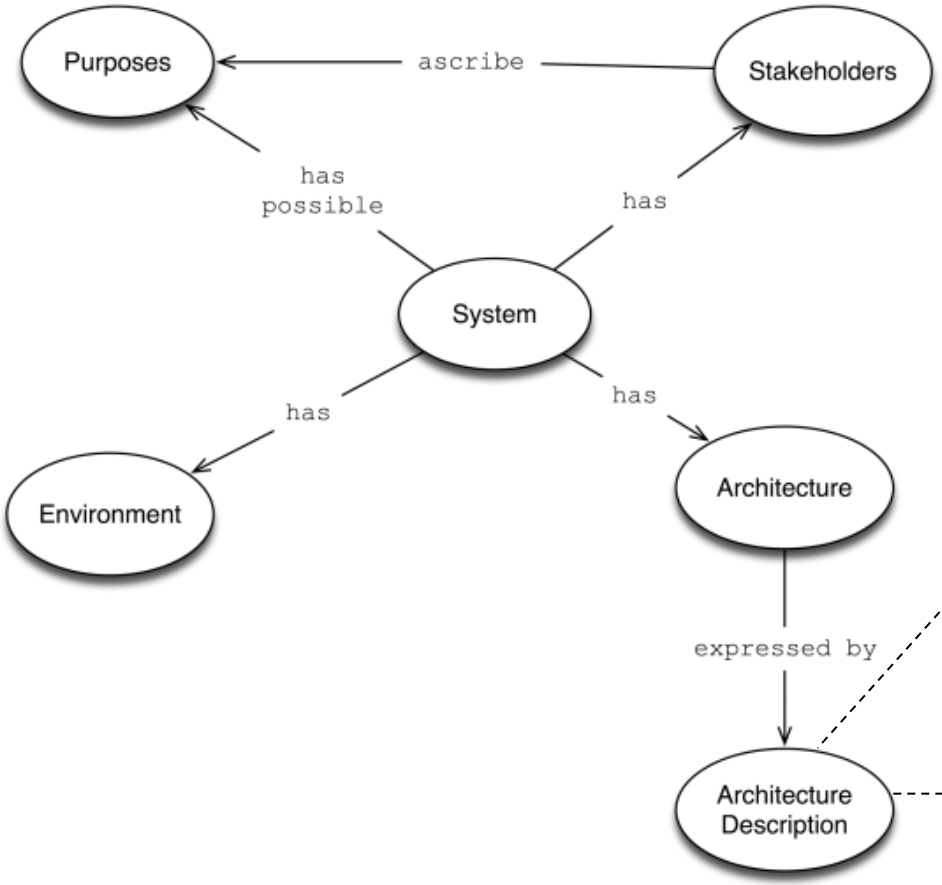
ISO/IEC/IEEE 42010

# Viewpoints

- Used to create architecture views

- Views
  - Concrete architecture from perspective of different stakeholders
  - Frame concerns of those stakeholders

- Viewpoints
  - Conventions for describing architecture-related concerns
  - Set of stakeholders holding those concerns
  - Set of model kinds
  - Correspondence rules between different architectural models

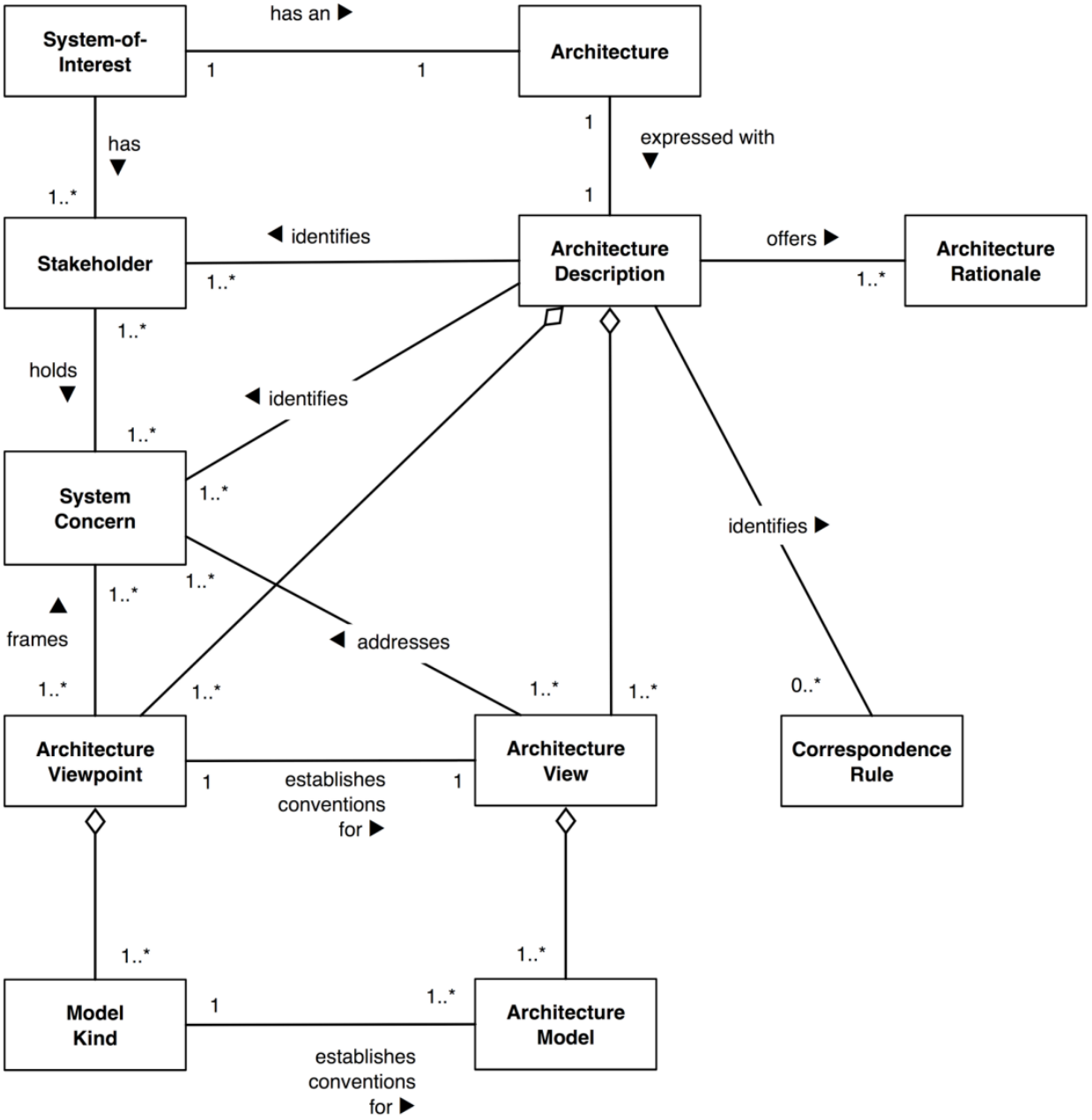



**Artifacts include**

- views,
- viewpoints,
- model kinds,
- models,
- stakeholders,
- concerns,
- decisions,
- etc.



ISO/IEC/IEEE 42010






ISO/IEC/IEEE 42010

So how are viewpoints “similar” to reference architectures?

Provide **generic architecture knowledge** to **document** , but without “**implementation**”

# Software frameworks

```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author john doe <doe.j@example.com>
 */
HelloButton()
{
  JButton hello = new JButton( "Hello, wor
  hello.addActionListener( new HelloBtnList

  // use the JFrame type until support for t
  // new component is finished
  JFrame frame = new JFrame( "Hello Button"
  Container pane = frame.getContentPane();
  pane.add( hello );
  frame.pack();
  frame.show();              // display the fra
}
```

# Software frameworks

- About implementation, code
  - Generic functionality that can be adapted by user-written code
  - Reusable "software environment"

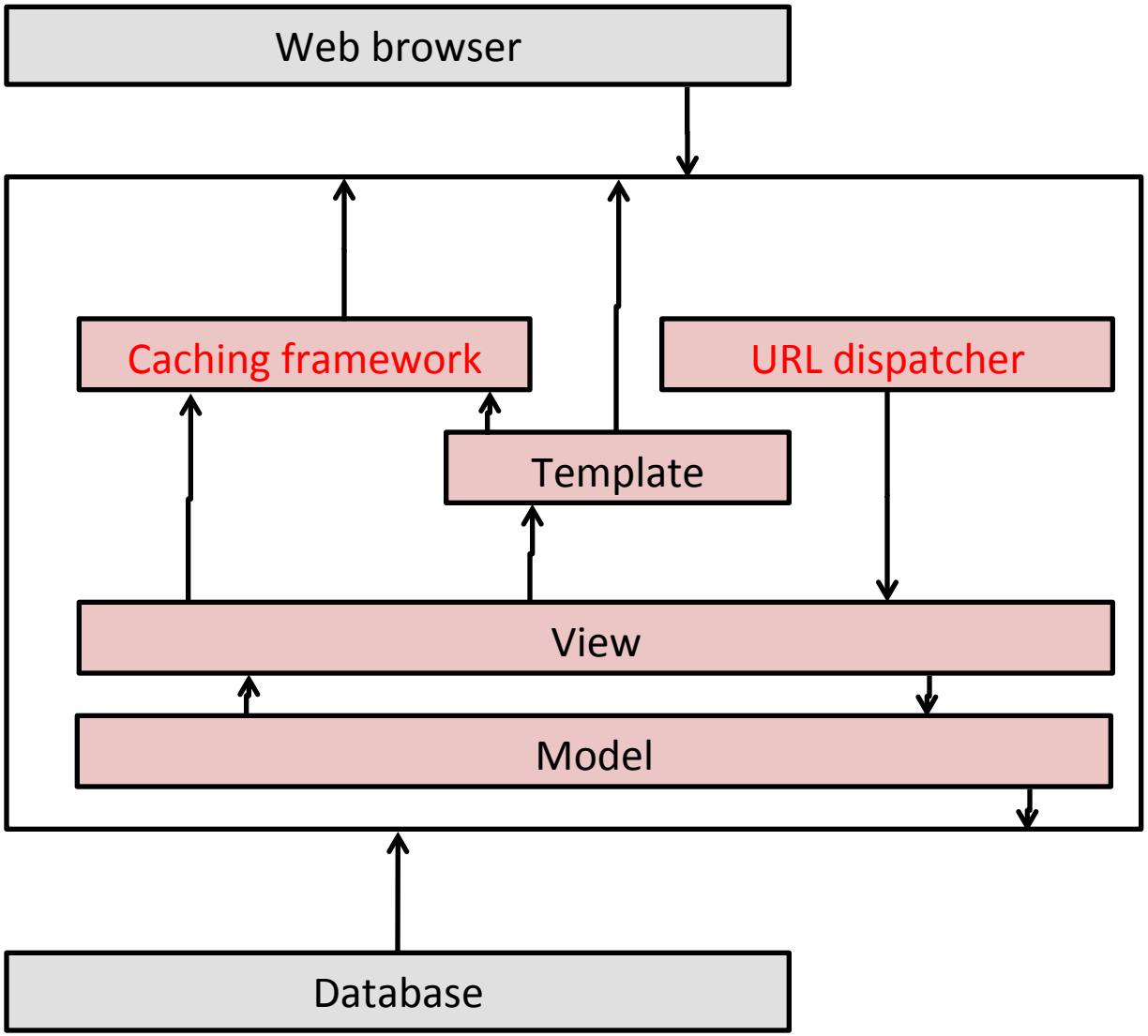
- Can include compilers, code libraries, tools or API's

- In contrast to regular libraries
  - Define program control flow
  - Extensibility (e.g., through overriding or specialization of code)
  - May not allow their code to be modified
  - I.e., have their own architecture

# Examples

- ASP.NET
- MonoRail
- Google Web Toolkit
- Node.js
- CherryPy
- Django
- Ruby on Rails

# Software frameworks have architecture

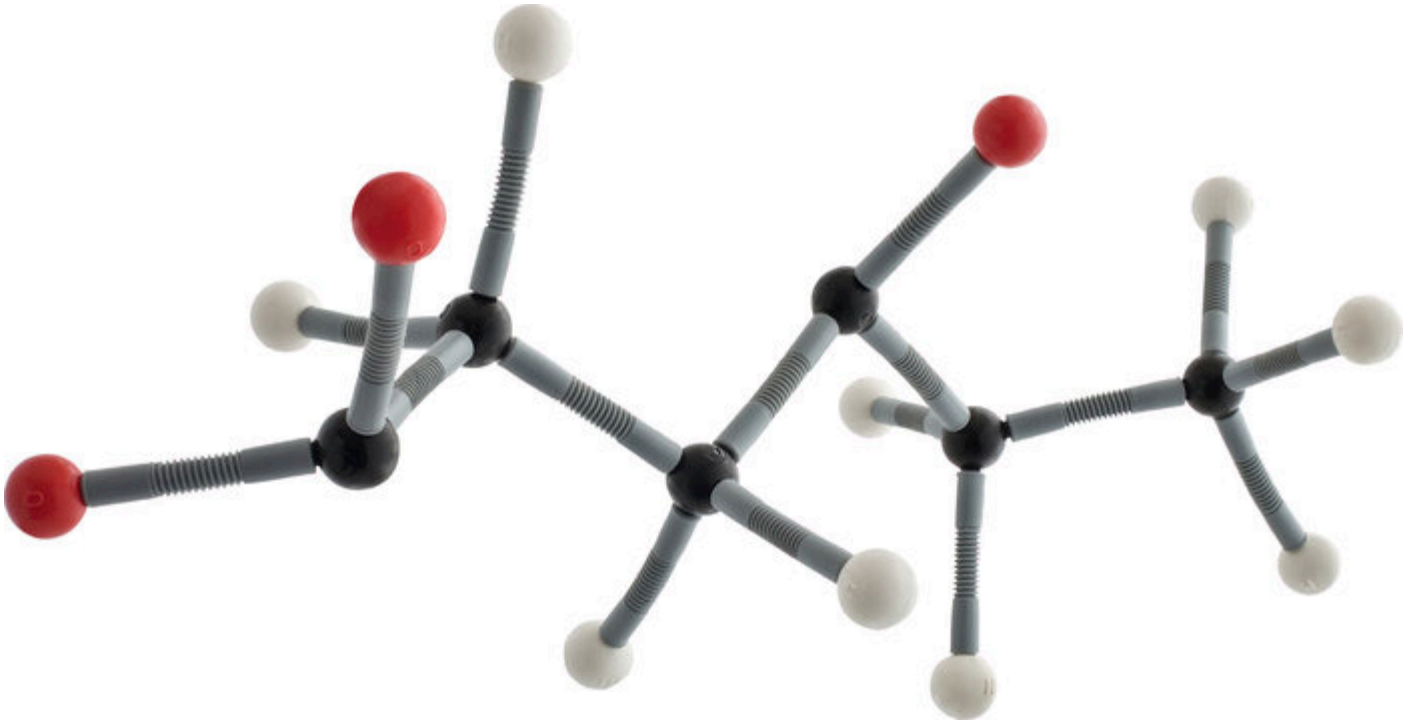



**Django**

- Model: describes data
- Template: how user sees data (e.g., html pages)
- Views: what users see
- Controller: URL dispatcher

Software frameworks are **generic** and provide means for **implementing** (rather than describing) software systems

# Reference models



# Reference models

- Abstract framework or domain-specific ontology
- Interlinked set of clearly defined concepts
- Division of functionality with data flow between the pieces
  - Decomposition of problem into parts that cooperatively solve problem
- Reference architecture
  - Reference model mapped onto software elements
  - Software elements implement functionality of reference model

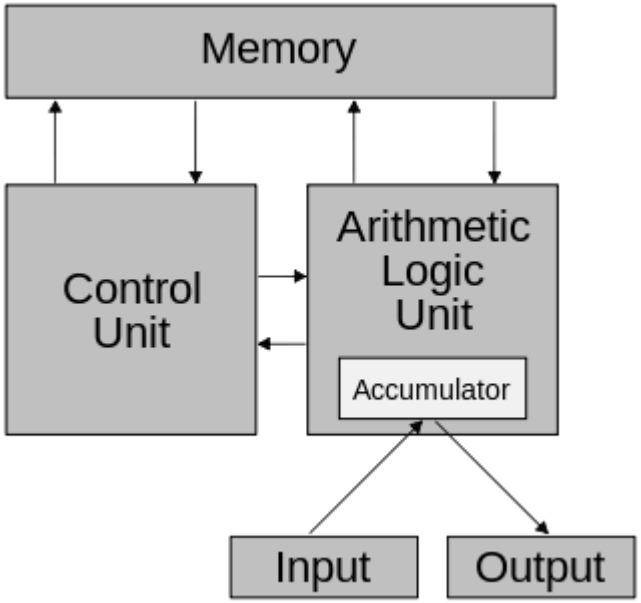


L. Bass et al.: *Software architecture in practice*. 2003.

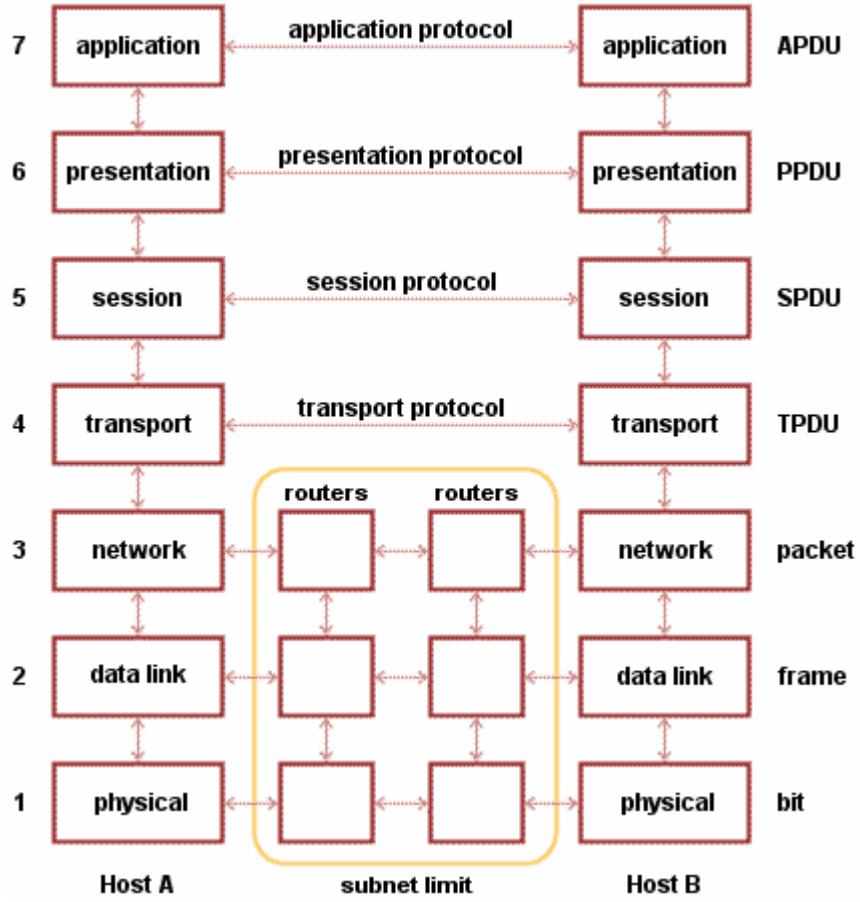
Whereas reference modeling divides functionality, RA is mapping of that functionality onto system decomposition

# Examples

von Neumann architecture
for sequential computing




OSI layer reference model

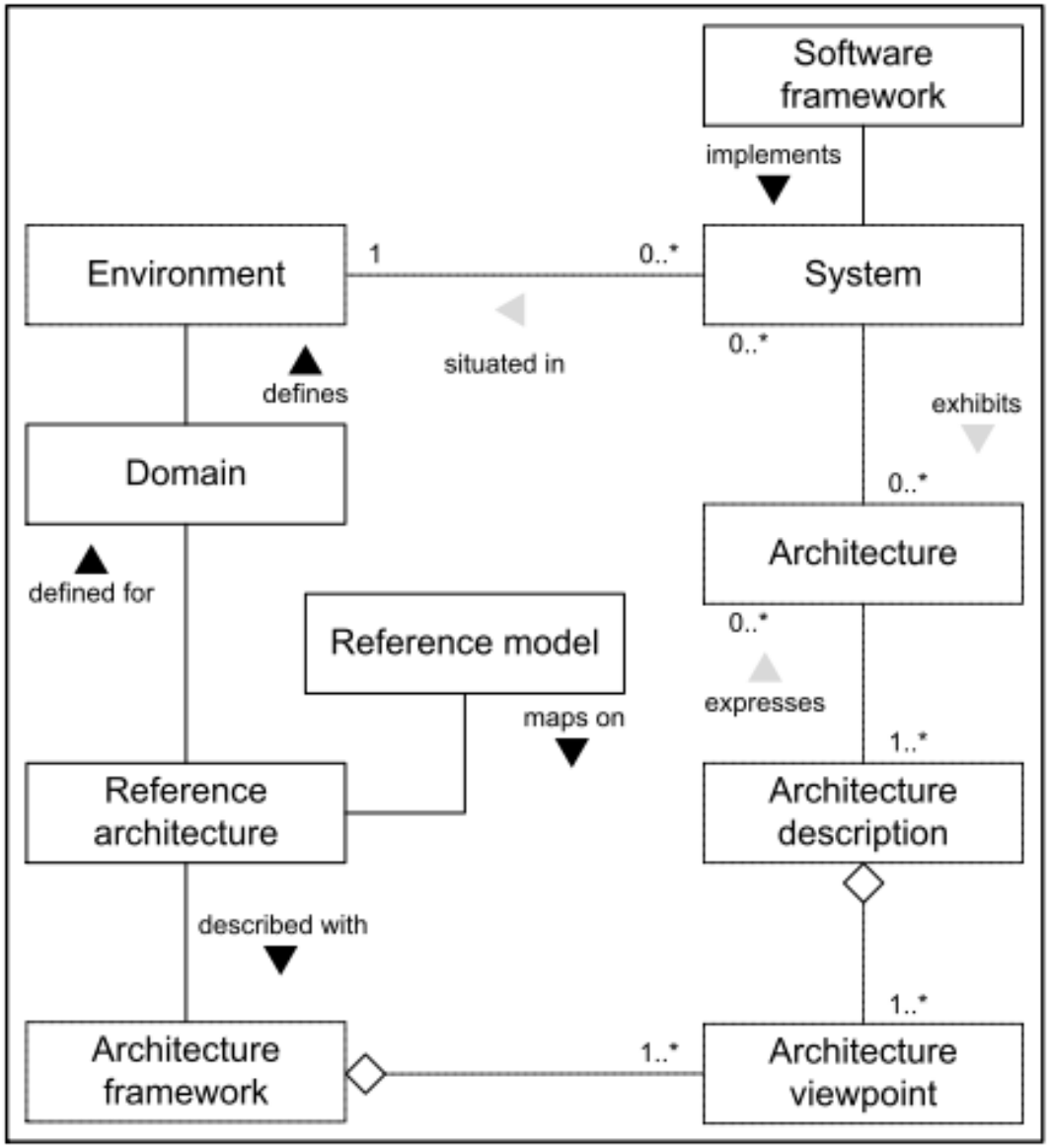


# Product line architecture

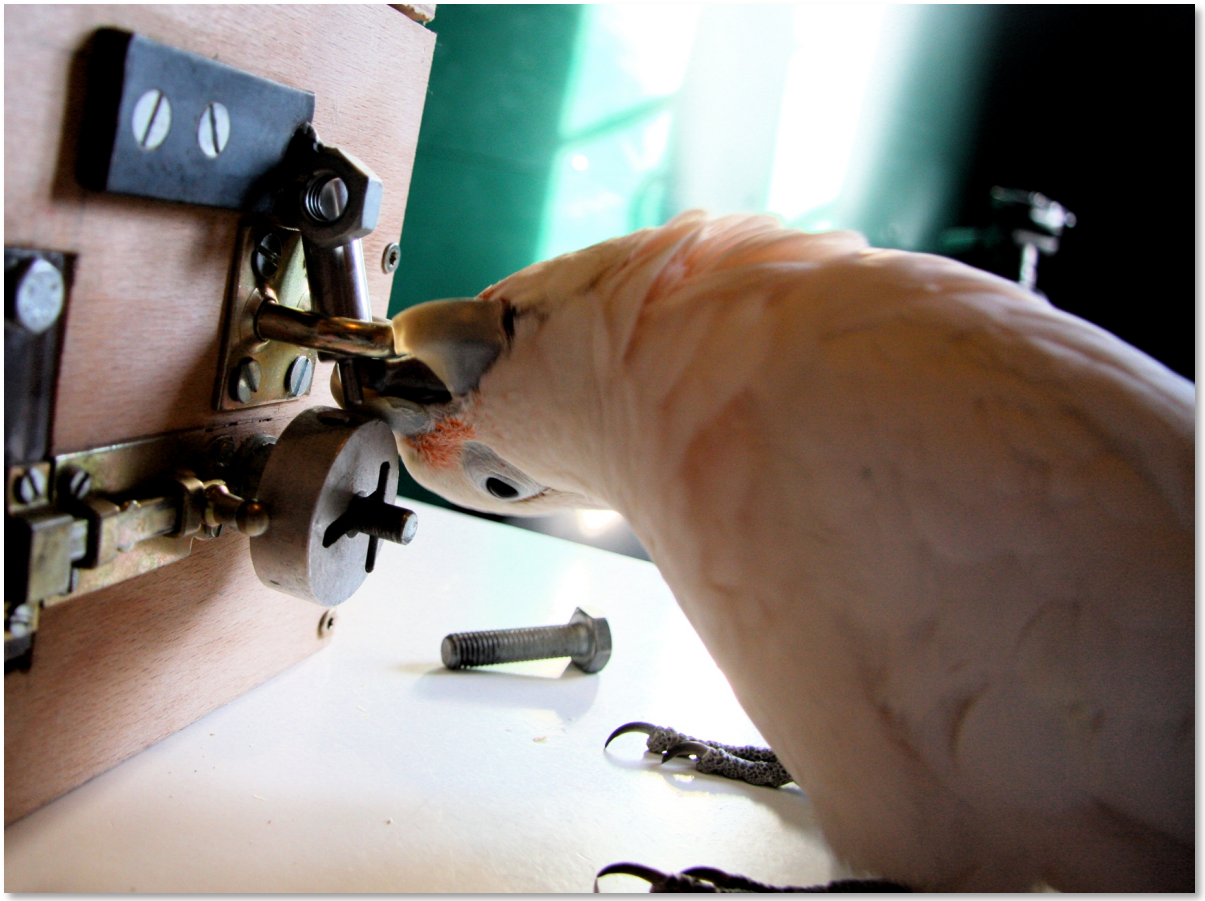
- Reference architecture = product line architecture?
- Product line architecture one type of reference architecture?
- Product line architecture provides more product information
  - Rather than domain information
  - Makes variability between products of a product line explicit

# What about reference **implementations**?

# How things may fit together



# Part II – challenges



# Challenges

- Other work on benefits, problems, pro's, con's

- Examples
  - Reference architectures for multiple product lines
    - Nakagawa and Oquendo
  - Constraints for the design of variability-intensive service-oriented RA
    - Galster et al.
  - How reference architectures are used in industry
    - Angelov et al.

E. Y. Nakagawa, F. Oquendo: *Perspectives and challenges of reference architectures in multi software product line.* MultiPLE 2013
M. Galster et al.: *Constraints for the design of variability-intensive service-oriented reference architectures - An industrial case study*. IST 2013
S. Angelov et al.: *Software reference architecture: exploring their usage and design in practice.* ECSA 2013

# In this talk

- High-level challenges related to design and use of software RA

- Much has been said about advantages and benefits of RA
  - Challenges here maybe research questions for future work?
  - Or scenarios in which RA are difficult to apply?

# Incremental and iterative development

- RA provide already made high-level design decisions
  - May limit solution space when exploring solution alternatives
  - Unclear what impact of RA is and how RA could be utilized most
- Agile frameworks such as Scrum
  - Projects driven by requirements as user stories, based on value
  - Requirements maintained on a product and sprint backlog
  - Product planning happens at the beginning of each sprint
- RA provides constraints which sprint planning must consider
  - Partially defined design in the RA
  - Do RA contradict or complement agile values, principles and practices?

# Global development and markets

- Reference architecture: standardization
  - Between / across products in an application or technology domain
- Today's markets are not locally restricted
  - Products target private or institutional customers around the world
  - Ensure compliance with regulations in diverse markets
    - In particular in regulated industries such as medical devices or avionics
- Designing RA that comply with many regulations and policies from different countries and domains may be challenging
  - Even harder when considering multi-disciplinary solution approaches

# Competitive markets, innovation

- RA provide partial solutions for products in a domain
- Software organizations compete through innovation
  - Many successful companies are innovative companies
  - Target new market opportunities, independent of current ideas
- But: time to market can be the difference between project success and failure
- May need "light-weight" RA
  - Balance potential for innovation and reduction of development effort
  - Commoditized, differentiating, innovative functionality

J. Bosch: *Achieving simplicity with the three-layer product model*. IEEE Computer 2013
H. Olsson et al.: *Scale and responsiveness in large-scale software development*. IEEE Software 2014

# Practical relevance

- Design, evaluation and maintenance of RA should result in RA that are relevant and applicable in practice

- Empirical foundation
  - RA must be based on a sufficient number of real-life phenomena, and on well-known and proven principles
    - Address real stakeholder interests
    - Building blocks derived from the problem domain and real life phenomena
    - Be based on concepts proven in practice

- Empirical validity
  - RA needs to be evaluated to ensure its applicability and validity

# But then…

- Most current initiatives propose RA that lack any validation and target very general problems without a clear description of the application domain

- Many reference architectures described in literature remain at a proposal stage

S. Affonso et al.: *Reference architectures for self-managed software systems: a systematic literature review*. SBCARS 2014
M. Guessi et al.: *Representation of reference architectures: a systematic review.* SEKE 2011

# Part III – emerging domains


# Unmanned Aerial Vehicles (UAV)

- Current hype for “civil” applications
  - Agricultural technology, forestry, emergency responder support, to provide infrastructures in under-developed regions
  - Combines flight control, 3D computer vision, swarm intelligence, wireless communications, networking, power systems
- Commercially airworthy avionics and UAV applications must comply with regulations
  - International and national standards for airworthy software (process and product safety and reliability regulations)
  - Bodies in different countries, e.g., FAA, CAA
- Requirements elicitation? Evaluation?

# Wearable Computing and Smart Homes

- Wearable computers or smartphones became common devices
  - Equipped with different types of sensors and can deploy various health-related applications
  - Smart homes and functional buildings offer automation infrastructures (sensors, actuators, control) to improve energy-efficiency, to assist with the care of people

- Domains: health, home automation, network engineering, wearable and mobile computing, wireless body sensor networks, etc.

# Summary

"Reference architecture" is a fuzzy concept.

Reliance on RA may limit flexibility and innovation.

We may explore RA for cross-cutting domains.