Introduction

PA1410
Software Architecture and Quality
(old course code PA1308/PA1309)

Mikael Svahnberg[1]

[1]Mikael.Svahnberg@bth.se
School of Computing
Blekinge Institute of Technology

Tuesday 25th June, 2013

Outline

A Word about Myself

Formalia
- Course Homepage
- Course Chater
- Literature
- Schedule
- Assignments

Introduction to Software Architectures

Next Step

A Word about Myself

- Associate Professor Mikael Svahnberg
- Master's degree in Software Engineering @ BTH, 1998
- PhD in Software Engineering @ BTH, 2003
- Thesis title: "Supporting Software Architecture Evolution – Architecture Selection and Variability´´
- Research Topics: Software Evolution, Software Architectures, Architecture Evaluation and Selection, Quality Attributes, Requirements Engineering, Requirements Engineering Decision Support, Empirical Software Engineering Research

Course Homepage

- It's Learning: PA1308(PA1309) lp1 Ht12
- Bitbucket `https://bitbucket.org/mickesv/pa1308-software-architecture-and-quality/wiki/Home`

Course Charter

The course comprises the following elements:

- Quality aspects in software and software architecture.
- Architectural styles, languages and patterns.
- Methods for architectural design and evaluation.
- Component-based software engineering.

Aims and Learning Outcomes

On completion of the course the participant will:

- Be able to clearly express an in-depth insight in the area of software architecture (standards, key concepts, software quality definitions etc.) and be able to name and describe a number of key issues related to this area.
- Be able to clearly express an in-depth insight of quality in software, and how this is realised in quantifiable goals.
- Be able to independently, both on a theoretical level and in practice, select between a number of architectural styles, languages and patterns depending on the requirements, and discriminate between them.
- With an attention to details be able to create and document a software architecture consisting of several views and taking several different concerns into consideration.

Literature

Main Literature

L. Bass, P. Clements, and R. Kazman.
Software Architecture in Practice, Third Edition.
Addison-Wesley Publishing Co., Reading MA, 2012.

Reference Literature

F. Buschmann, C. Jäkel, R. Meunier, H. Rohnert, and M. Stahl.
Pattern-Oriented Software Architecture - A System of Patterns.
John Wiley & Sons, Chichester UK, 1996.

J. O. Coplien and G. Bjørnvig.
Lean Architecture for Agile Development.
John Wiley & Sons, Chichester UK, 2010.

M. J. Fowler.
Patterns of Enterprise Application Architecture.
Addison-Wesley, Boston MA, 2003.

R. Pirsig.
Zen and the art of Motorcycle Maintenance.
William Morrow, 1974.

Literature

Additional Literature


J. Bosch.
Design & Use of Software Architectures - Adopting and Evolving a Product Line Approach.
Addison-Wesley, Harlow UK, 2000.


C. Hofmeister, R. Nord, and D. Soni.
Applied Software Architecture.
Addison-Wesley, Reading MA, 2000.


M. Shaw and D. Garlan.
Software Architecture — Perspectives on an Emergin Discipline.
Prentice Hall, Upper Saddle River NJ, 1996.


C. Szyperski.
Component Software – Beyond Object-Oriented Programming.
Addison-Wesley, Harlow UK, 1998.

Schedule, 2013

Date	Title
2013-09-03 Tue	L01 Introduction
2013-09-06 Fri	L02 Quality Attributes and Architecture Decisions (Note time 10-12)
2013-09-10 Tue	L03 Global Analysis
2013-09-12 Thu	L04 Documenting Software Architectures
2013-09-17 Tue	L05 Styles and Patterns
2013-09-19 Thu	S01 Peer Evaluations (NOTE time: 13-17)
2013-09-24 Tue	L06 Architecture Evaluation and Transformations
2013-09-26 Thu	L07 Formal Specifications for Documentation and Evaluation
2013-10-01 Tue	L08 Architectures for Different Purposes
2013-10-03 Thu	L09 Reality Check
2013-10-22 Tue	S02 Presentations of Architectures
2013-10-24 Thu	S02 Presentations of Architectures

The Secret Formula to Pass this Course

- Read the *course book(s)*.
- Do the *self study packages* to verify that you have understood.
- Attend the lectures to confirm your understanding, get a broader view, and to ask questions.
- If you still have questions: use the *discussion forum* or mail the teachers.
- the course book, the lectures, and the self study packages all feed into the *assignments*.
- Study the *assignment rubrics* so you know what is expected of you.
- Start with the assignments early, work regularly on them during office hours.
- Do a risk assessment early and plan your work accordingly.

Self Study Packages

- We provide a number of self study packages.
- These are related to the lectures and assignments.
- They are designed to provide additional insights.
- We do not provide answers to these packages; they are to be seen more as pointers to where *you* may find more information or practice.
- We do not require you to submit these for marking, we only hope that you may learn something from them.
- By doing them (individually), you will gain a better understanding of how to address the assignments.

Assignments

- There are five assignments in the course; four team assignments and one individual:
  - A01: Architecture Decisions: Factors, Issues, Strategies, Conceptual View
  - A02: (Individual Assignment) Personal Reflections
  - A03: Architecture Design: Architecture Transformation, Module View, Execution View
  - A04: Formal Specification: AADL
  - A05: Architecture Change: Modify architecture as a response to change requests.
- You are expected to work in *groups of four* for the team assignments.
- More information is available on the course homepage.

Assignment Deadlines

Date	Title
2013-09-13 Fri, at 08:00	A01 Deadline
2013-09-20 Fri, at 08:00	A02 Deadline
2013-10-04 Fri, at 08:00	A03 Deadline
2013-10-18 Fri, at 08:00	A04 Deadline
2013-11-01 Fri, at 08:00	A05 Deadline
2013-11-29 Fri, at 08:00	A01-A05 Resubmission 1
Autumn 2014	A01-A05 Resubmission 2

Assignment Rubrics

- The assignments are assessed according to a rubric
- There is one rubric for each assignment; there are several criteria in each rubric; there are four different levels for each criterion.
- For each level and criterion, there is a textual description. This description provides generic feedback on what you can improve on an item. In addition, we *may* provide additional feedback that is specific to your assignment.

Example

If the feedback reads: “A1TFT: 2”, this shall be interpreted as:
Technological Factor Tables *Factors are listed, together with an assessment of their Flexibility & Changeability and Impact. Some important factors are missed. The Flexibility, Changeability and/or Impact are inaccurate for some.*

And thus you need to improve e.g. by identifying the missing factors, and make sure that you consistently specify flexibility, changeability, and impact correctly. By reading the levels above, you may find more hints as to what is missing.

Resubmissions

- Iff you failed A01, the version you submit with A03 will be used for reassessment.
- Iff you failed A03, the version you submit with A05 will be used for reassessment.
- A02, A04, and A05 are resubmitted after the course (see assignment deadlines).

In a resubmission, please:

- highlight changes you have made!
- discuss, where applicable, the changes you make
- remember to bring your updates forward so that the current assignment reflects your changes as well.

Assignment Cooperation

- Cooperate if you want and think you can.
- **However:** Each group must hand in a report that is uniquely produced by them.
- Use your own experience when writing the assignments
- Discuss
- Reflect
- **DO NOT COPY/CHEAT/PLAGIARISE**

Where to draw the line?

Example Assignment

The teacher says: “Choose company A, B, or C. You must investigate the advertising campaign which that company used in the past two years. Write a report that evaluates the campaign’s imåpact and make recommendations for future campaigns in that company. *Do your own work. Hand in an individual report.*

Suppose 3 students do what is listed below and they do it in this order:

1. The three students discuss the task with other students.
2. They look at past examples of similar student reports. They discuss together what is good and bad about the other students’ work.
3. Each one chooses company B then discovers the other two have done the same. They decide to discuss ideas.

Where to draw the line?

4. They all three decide to do a bit of research on advertising campaigns in general. They all look for information but agree to really go into depth on one aspect (one researches how to measure impact, another looks at design, another looks especially at cost etc.) Everyone makes notes from their research.
5. They report orally on no. 4 (above) *['Here is what I found out']*. They tell each other useful sources of information and which general sources were especially good.
6. They exchange research notes on what they have found so far, including sources.
7. The one person of the three who is really good at information retrieval collects information on company B's advertising campaign(s). He shares what he finds.

Where to draw the line?

8. One person organises the report structure, makes headings and gives others a copy.
9. They all share out the writing. Each person writes two sections, using the shared notes from (6) above. Everyone contributes ideas to the 'conclusions' section and agree what to write.
10. They combine all the sections. Each student takes the combined draft away in electronic format. Each student, working alone, writes 'over the top' of the others' work. No person changes more than 5% of a fellow student's work.
11. Each student submit his or her final report and signs a statement that this is *'an individual report and this is my own work'*.

Introduction to Software Architectures

- What is it?
- Why bother?
- I am a games student. Why should I bother?

SE Challenges

- Reduce Development Cost – Deliver on time, within budget
- Increase System Quality – . . . with the *right* quality
- Decrease Maintenance Costs
- Reduce Time-to-Market

SE Challenges

- Reduce Development Cost – Deliver on time, within budget
- Increase System Quality – . . . with the *right* quality
  - Functionality
  - Performance
  - Security
  - Reliability
  - Usability
  - . . .
- Decrease Maintenance Costs
- Reduce Time-to-Market

Decisions

- Balancing all stakeholders result in a number of *Business and Technical Decisions*
- Software architecting is about identifying which decisions are necessary, and finding solutions that satisfy all stakeholders.

Decisions *are* the Architecture

I would go as far as to say that these decisions *are* the architecture. . . . The rest is just an instantiation of the architecture.

How can the Architecture Help?

The architecture is a tool for

- Understanding
- Planning
- Communicating
- Predicting and Evaluating Quality
- Identifying Reuse

Influences on Architecture

- Customer Requirements, of course
- Developing Organisation
  - e.g., business goals
  - Organisational structure
  - Available expertise (the architect's experience)
- Technical Environment

The Architecture Business Cycle

The architecture is influenced by stakeholders, developing organisation, technical environment, architect's experience, etc.
Likewise, the architecture influences all of the above.
A software architecture

- Manifests early design decisions
- Constrains an implementation
- Dictates organisational struture
- Impacts change management

A developed system

- Adds to previous experience
- May affect business goals
- Refines development organisation (Process improvement)
- Affects customer requirements
- Provides a reusable software base

A “Good” Software Architecture

- Is based on *conscious* decisions
- Is *evaluated* to ensure that it satisfies the specific goals for the system
- Pays attention to current and future *quality attributes*
- Is well *documented*, with traceability to the architecture decisions
- Features well defined *modules*(components), with well defined *interfaces* and well defined *responsibilities*
- Is restricted to a small set of interaction patterns that are *consistently* used.

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

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

Software Architecture Defined

- First Definition: Boxes and Lines
  - What is the nature of the elements (boxes)?
  - What are the responsibilities of the elements?
  - What is the siginificance of the connections (lines)?
  - What is the significance of the layout?
- Second Definition: Add semantics (provide legend)
  - What is the significance of the layout?
  - What are the interfaces of the elements?
  - How does the architecture operate at runtime?
  - How do we build it?
- Third Definition:

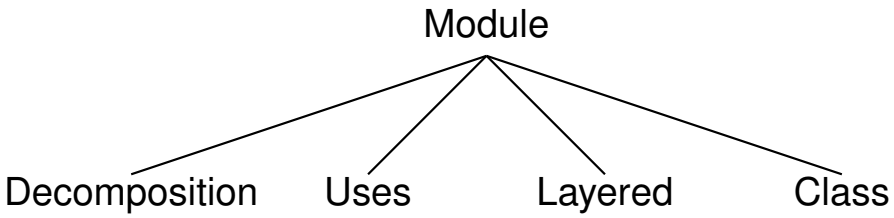
  The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

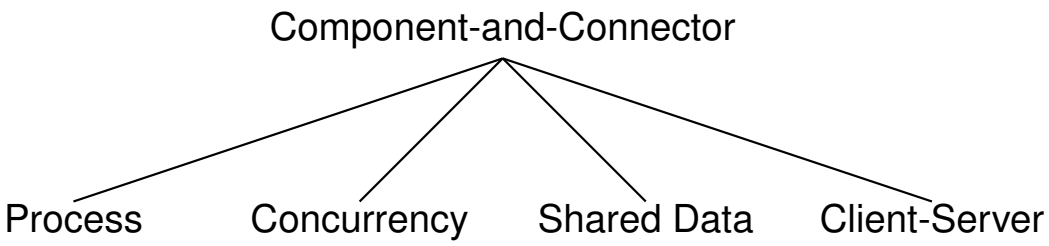
Useful Intermediaries

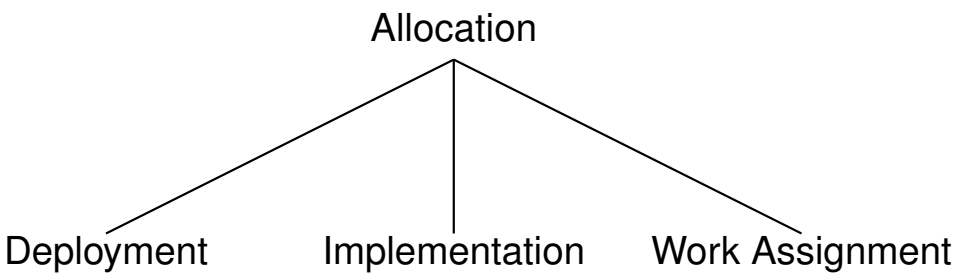
- Architectural Patterns (or Architectural Styles)
  A “constellation” of elements and relationships.
- Reference Model
  The “standard” solution to a known problem
- Reference Architecture
  A mapping of a reference model onto a system decomposition

Structures and Views

Bass et al.(2012):

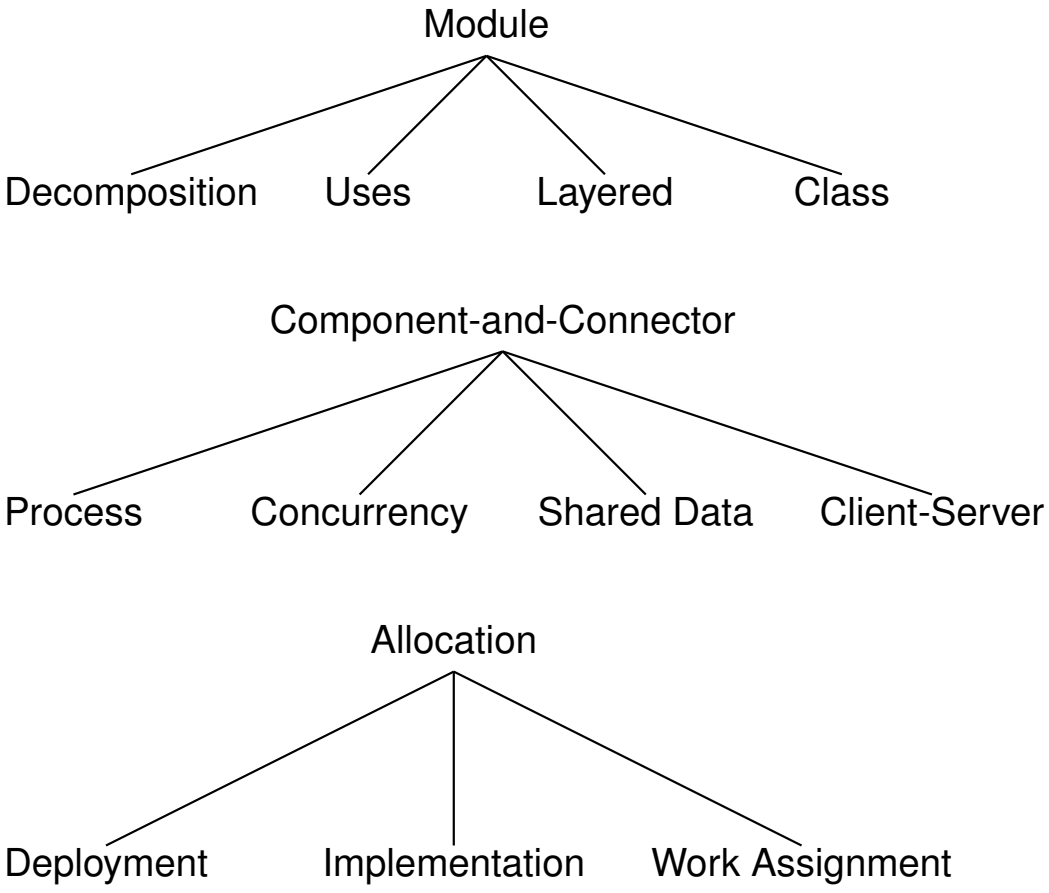




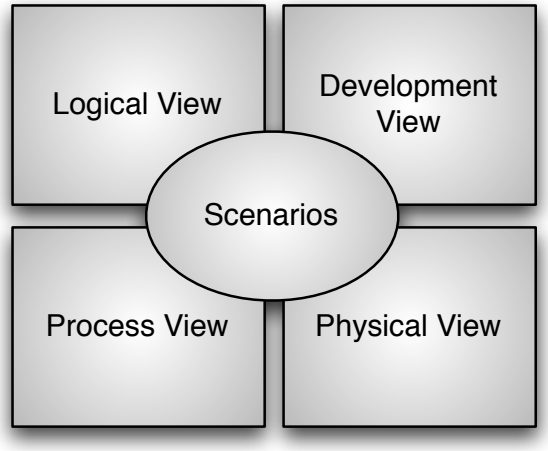


Structures and Views

Bass et al.(2012):

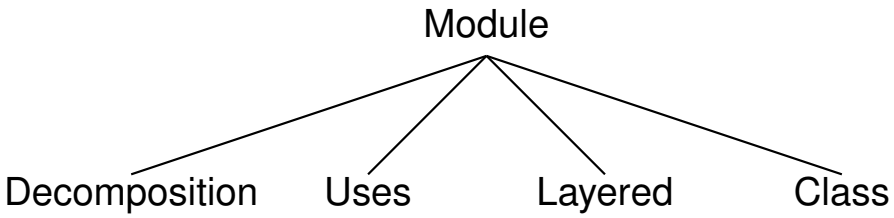


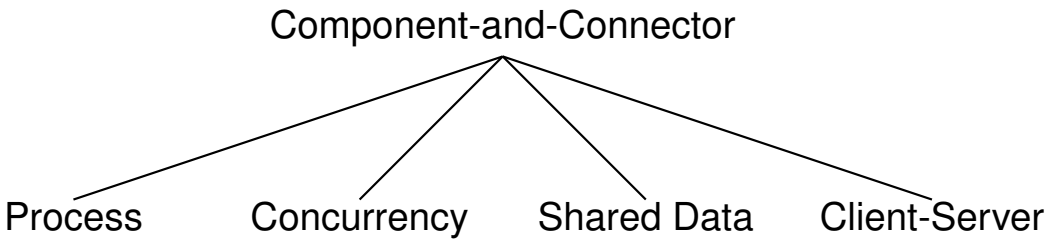
Kruchten(1994):

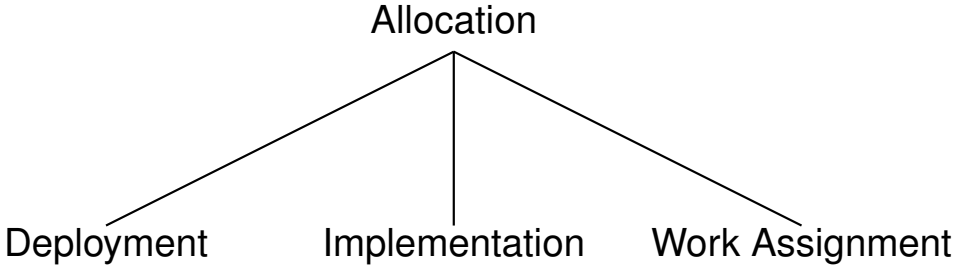


Structures and Views

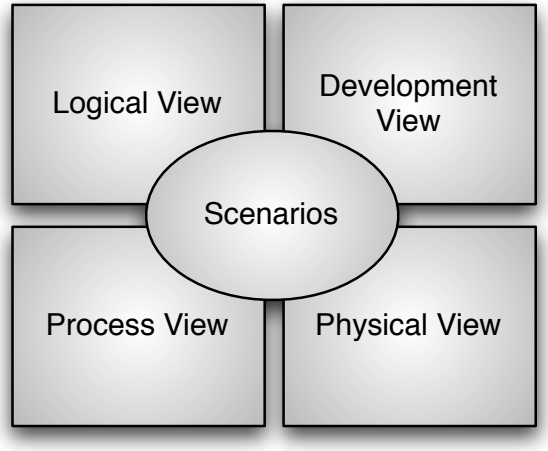
Bass et al.(2012):







Kruchten(1994):



Hofmeister et al.(2000) uses a variant of this:

- Conceptual View
- Module View
- Execution View
- Code View

Next Step

- How do we juggle these different perspectives?
- Where do we start creating our architecture?
- How do we use the architecture to ensure we get the right quality?

A modern game

- Is developed by teams including more than 800 people
  - over a period of several years
  - using a multitude of different development processes
  - requiring that the game is always playable (=no big bang delivery)
- Constitute large online platforms
  - with many, many server platforms from different cloud vendors
  - with requirements not only on performance but also on maintainability, scalability, portability, flexibility
  - that needs to securely interact with other systems (e.g. banking systems)
  - with massive requirements on data CIA (Confidentiality, Integrity, Availability)
  - that need to be able to robustly accept extensions and plug-ins while at the same time protecting the safety of the game platform and the users
  - with massive requirements on system availability

A modern game architecture

- anticipates all of this, and is created as a result of a structured process
- enables early testing of features *as well as* quality attributes
- is flexible so that it can grow and change along with the game and game platform

In this course:

- you may not study a game architecture as such. You have other courses for that.
- you will learn a structured way of analysing a system to identify the forces that are relevant for that particular system.
- you will learn how to analyse a particular architecture and identify where there are shortcomings or opportunities for improvements
- you will get an understanding of quality attributes that are relevant for any software system, but even more so for a modern game.
- you will understand how to balance these quality attributes against each other