Software Architecture [H07Z9a]

Introduction and organization

Wouter Joosen
Dimitri Van Landuyt

DistriNet, Dept. of Comp. Science, KU Leuven

Welcome

- Course instructors
  - Wouter Joosen
  - Dimitri Van Landuyt
- Labs and project support
  - Alex van den Berghe,
  - Laurens Sion,
  - Stef Verreydt,
  - Willem Verheyen

Overview of this lecture

(1) Situating the course

Software Engineering courses

- “Methodiek van de informatica” (MI)
  - Starter level programming course
- “Object-gericht programmeren” (OGP)
  - Advanced programming course – OO concepts
- “Ontwerp van Softwaresystemen” (SWOP/OSS)
  - First software engineering course (software design)
- “Vereistenanalyse voor complexe softwaresystemen”
  - Requirements analysis
- …. **Software Architecture**
  - Software engineering continued – *completed*?
- “Advanced methods for software architecture” (*non-mandatory course*)

Differences with "Software Design" (or an equivalent course)

- We further expand our view on the development life cycle
  - We work on requirements & analysis
  - We work on "early design": Software Architecture
- We evaluate architecture & design artifacts before implementation
  - No "*compiler comfort*"
- Emphasis on addressing non-functional requirements
  - Security, performance, availability, modifiability, fault tolerance, extensibility, adaptability, etc *... more on these later*

Differences with a "standard" software development course

Reality check:

- The requirements exist, to a large extent
- The target system
  - is an **extension** of an existing software system,
  - has to **interoperate** with existing software systems
- The technological context cannot be determined from scratch
  Greenfield software development is rare
- We cannot ignore economics: cost, effort: software is typically more expensive than expected

(2) Course objectives

(2) Course objectives

- Broaden the scope: establish an understanding of the **entire software engineering process** in realistic software production settings
- To elaborate on the **role of analysis/evaluation** in this context, and to elaborate on the role of **software architecture** in the software engineering process
- To study the **creation** and **extension** of software architecture, and THUS to focus on **non-functional requirements**

Secondary objectives/additional skills

- Team work, team organization, collective decision-making, brainstorming, ...
- Time management, planning, deadline-driven
- Analytic, top-down reasoning, addressing ambiguity, creating abstractions that work
- Communicating about technical decisions, motivating, discussing trade-offs
  - Incl. presenting, writing, feedback
- Modeling, abstractions, using existing abstraction techniques (UML)
- Self-assessment and peer review, being your own worst critic, someone else's constructive critic

(3) Course organization

Modus operandi

- **Project-driven work**
  - experience course, intensive (**!**)
  - no emphasis on theory
- **Teamwork**
  - team size: 3
  - time management, coordinated effort
- **Quality control**
  - systematic delivery and feedback

Course activities

- 12 Lectures - Thursdays
  - Details on Toledo (Course Information)
- 9 lab sessions - Mondays
  - 8 sessions to work on project assignment (team),
  - 1 session on Visual Paradigm /modeling skills (individual)

  - Starting on **Feb 21**

Course outline

- **Part 1** – Getting to know the application case, **requirements**
  - Architecturally-significant requirements (ASRs)
  - Quality attribute scenarios (NFRs)
- **Part 2** – Evaluating a **software architecture**
  - Understanding, reading models, identifying trade-offs
  - Definition, context, notation
- **Part 3** – Extending a **software architecture**
  - Methodology & known principles/practices/tactics
  - Modeling
  - Making trade-off decisions

“Phase 1”

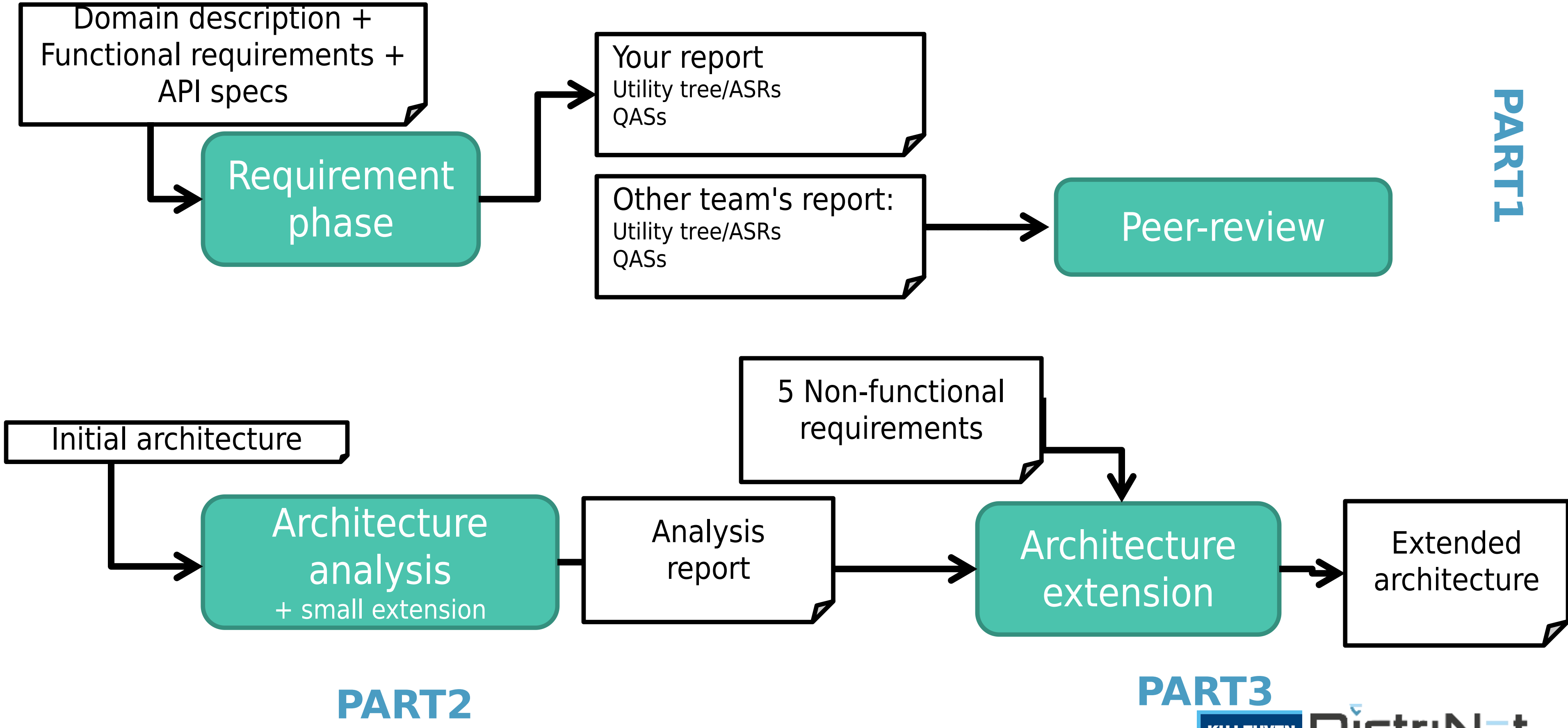
“Phase 2”

Delineated project activities

- Part 1: requirements engineering – focus on ASRs and NFRs
- Part 2: evaluation of initial architecture – specific set of questions and attention points
- Part 3: extension – scoped set of extensions

No blank pages to start from: we provide initial starting points

Timeline of the project



Continuous feedback

- Part 1: Peer review
- Parts 2 and 3: SA Plugin for Visual Paradigm – *some* compiler comfort
- Part 2: initial extension – feedback on modeling and solution
- Part 3: interactive lecture where you present a solution

Continuous support – on-campus

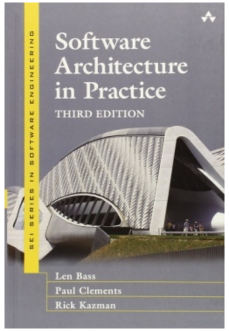
- **Mondays**: Lab sessions = project working sessions
- **Thursdays**: Lectures
  - Theory in function of the assignment
  - Start with project Q&A
- **Other days**: Toledo discussion boards

Continuous support – COVID code red

- **Mondays**: Lab sessions: Blackboard Collaborate + Slack channel
- **Thursdays**: Lectures: live-streams
  - Start with project Q&A – discussion boards / Q&A on Thu
- **Other days:** Toledo discussion boards - **!**

Course Materials

- Slide hand-outs (available on Toledo under "Course Documents")
- Books
  - "Software Architecture in Practice 3rd Ed.", Bass, Clements, Kazman
    - Focus on chapters 2, 4, 5, 7, 9
  - "Pattern Oriented Software Architecture (Volume 4)" by Buschmann et al.
    - On Toledo (Course Documents)
- Project support on Toledo and during lab sessions


Software Architecture in Practice
THIRD EDITION
SEI SERIES IN SOFTWARE ENGINEERING
Len Bass
Paul Clements
Rick Kazman

WILEY SERIES IN SOFTWARE DESIGN PATTERNS
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
A Pattern Language for Distributed Computing
Volume 4
Frank Buschmann
Kevlin Henney
Douglas C. Schmidt

Evaluation

- Project and oral exam in *project evaluation sessions*
- Discussion of the project results (team-wise)
- Weights: 25% - 25% - 50%
  - Students must be successful in all parts to pass for the course

Course deadlines

- **Part 1: Requirements**
  - Report: 25% of the score
    - [Feb 17- DL: **Wed, March 09 (noon)**] 3 weeks
  - Peer review, influences the part 1 score
    - [Mar 10 - DL: **Wed, March 23 (noon)**] 2 weeks
- **Part 2: Architecture Analysis**
  - Report + vpp file: 25% of the score
    - [Mar 10 - DL: **Thu, March 31 (noon)**], 3 weeks
- **Part 3: Architecture Extension**
  - Report + vpp file: 50% of the score,
    - [Mar 31 - DL: **Fri, May 13 (noon)**] 6 weeks

Project evaluation sessions affect grades (*obviously*) – possibly individually

How to contact us

1. **Content** - Toledo discussion boards: General | Part1 | VP-UML | Part2 | Part3-4
2. **Team business** - deadlines, practical, organization
   SoftwareArchitecture@cs.kuleuven.be
   - Put **all team members** in CC
3. **Individual/personal issues**: mail to Dimitri Van Landuyt **AND** Wouter Joosen (*ombudsman*)

(decreasing order of preference)

- **Contacts** page on Toledo

(4) Key assumptions

Assumption/expectation I

- We expect students to have executed a software development project (similar to the software design course)
  - It is difficult to create/evaluate/extend a software architecture if you are not (slightly) experienced in building software

Assumption/expectation II

- Master students plan their efforts, and manage the plan
  - The SA project cannot be executed in “one intensive working session”, neither in a very short time window: it requires iteration, reflection, and discussion
  - Team effort, team coordination

Assumption/expectation III

- This course follows a project-centric approach
- We deliver what students need in order to be successful, but:
  - Pro-active attitude required
  - (Most of the) lectures are PART OF the project: lectures provide background knowledge that will help you understand

It is all about building up experience…

(so do not sit back, relax etc. …)

(5) First practical arrangements

Practicalities
1. Toledo

- All students: register to [H07Z9a] Software Architecture
  - .. NOT to [H09B5a], (nor to [H0S00a], or [H09B6a])
  - (should be okay, but pls doublecheck)

Practicalities:
2. Toledo team registration

- To do: form and register a team (team size: **3**)

  https://tolapps.kuleuven.be/Tolinto/event/4365493278_gIl

  - **Lab session, pick a slot** either “10h30”, “13h30” or “16h”
  - Important for load balancing (online support / room capacity)
- **Find team mates**: Discussion board is open, lab session on 21/02
- Deadline: **Wed 23/02**

About teamwork

- Equal roles
  - Everyone should be equally familiar with the end result
  - Same amount of effort spent: keep track of the contributions of each team member
  - Your contributions affect others (!)
  - Compare agendas

- **Problems** and **team issues >** let us know **in time**