Documenting Early Architectural Assumptions in Scenario-Based Requirements

Dimitri Van Landuyt

Eddy Truyen

Wouter Joosen

KATHOLIEKE UNIVERSITEIT LEUVEN DistriNet RESEARCH GROUP

ibbt Security Department
Security, Privacy & Trust of E-*

Transition to architecture

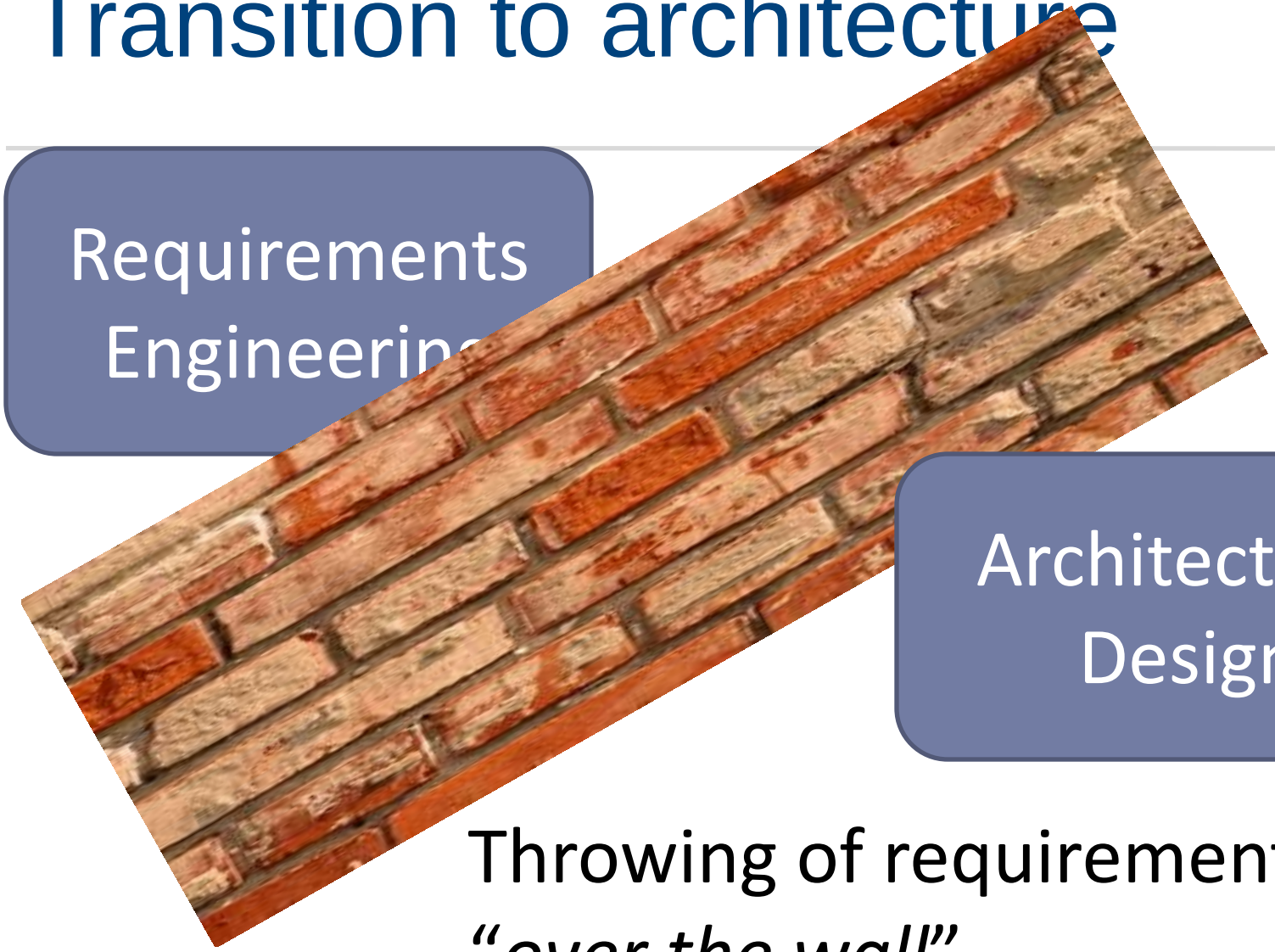
Requirements Engineering

??

Architectural Design

Transition to architecture

Requirements Engineering



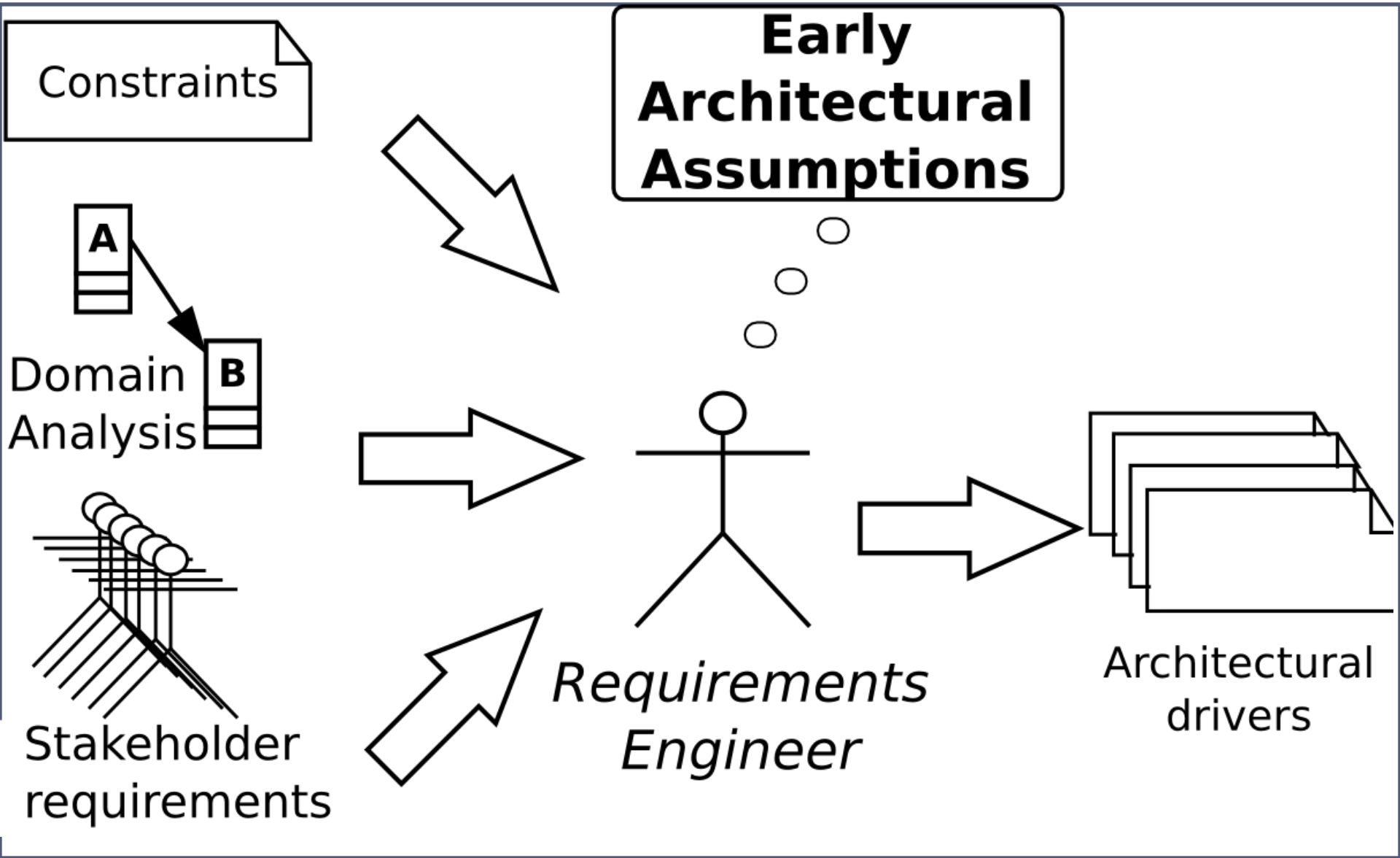
Architectural Design

Throwing of requirements *"over the wall"*

Transition to architecture

- More interaction leads to higher efficiency
  - Cf. Twin Peaks model to SE **[1]**, co-evolution, …
  - Faster convergence to a solution **[7]**
- Example: **early architectural assumptions**

Transition to architecture



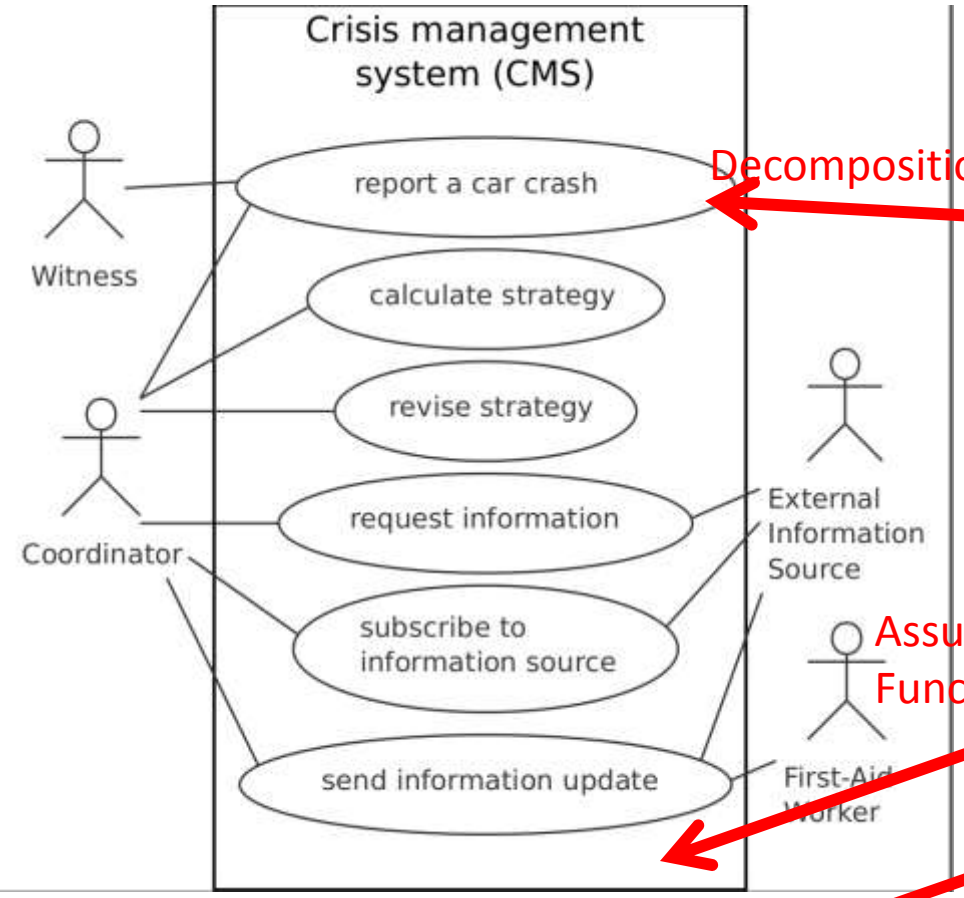
Early architectural assumptions

- **Early**
  - before real architectural decisions have been made
- **Architectural**, about
  - Initial decompositions (~ logical view), initial system elements (~ deployment view), behaviors (~ process view), ...
- **Assumptions**
  - ≠ stakeholder requirements, technical or project constraints, ...
  - Made by technical stakeholder such as requirements engineer

Early architectural assumptions in scenario-based requirements

- Case study: car crash management system (CMS)
  - Collecting information about a car crash
  - Suggesting resolution strategies
  - Coordination of resolution (dispatching help workers, etc)
- Third-party case study used for comparing modeling approaches
  - Used in a MSc course on software architecture
  - Scenario-based requirements: Use Cases and Quality Attribute Scenarios

Early architectural assumptions in scenario-based requirements



Decomposition

Assumed Functionality

Technical constraint

Availability: car crash reporting	
[..]	[..]
Stimulus	The *(sub-)system responsible for receiving emergency calls and forwarding them to an available coordinator* has crashed
[..]	[..]
Response	After detection, system goes in degraded modus: • *Calls are redirected* • *Restart/redeployment of the subsystem [..]*
Response measure	• Does not affect *ongoing witness calls* • Calls are *redirected* within 5s. • [..]

Key observations and problem statement

- Three observations about early architectural assumptions (EAAs)
  1. Documented **implicitly**
  2. Bad **modularity:** many scenarios are based on the same EAAs
  3. **Crosscutting** effects on system (& its requirements)

- **Problem**: this hinders the **navigability** and **accessibility** of requirements body
  - → Hard to navigate *semantic interrelations* between requirements (mental effort)

Problem statement: motivation

- Key early development activities are **hindered by limited navigability**:
  - Consistency management in RE
    - E.g. making changes in one scenario might ripple to others
  - Identification of architectural interaction points
    - ADD **[2,3]**
    - ATAM **[2,4]**
  - Architectural change impact analysis
    - E.g. invalidating an EAA: what's the impact?
- Architectural knowledge management
  - Document the *process*, not only the *end result*

Towards a solution

1. Make EAAs **explicit** and **modular**
   - Sufficiently expressivity modeling formalism to address crosscutting nature of EAAs
   - In ongoing work: we are aspect-oriented modeling techniques for this
2. Provide **process support** in the transition to architecture
   - ➔ Maintain traceability links between EAAs and actual architectural decisions
     - Accept, refine, reject

Summary

- **Early architectural assumptions** (EAAs)
  - (i) *implicit*, (ii) *scattered* and *tangled* and (iii) exert *crosscutting influences*;
  - hinder **key development activities** in the transition to architecture
- Similar to (late) architectural assumptions
  - Shown to have non-trivial impact on software quality **[5,6]**

Questions?

- **[1]** Bashar Nuseibeh. Weaving together requirements and architectures. IEEE Computer, 34(3):115–117, 2001.
- **[2]** L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice. Addison-Wesley, second edition, 2003.
- **[3]** Rob Wojcik, Felix Bachmann, Len Bass, Paul C. Clements, Paulo Merson, Robert Nord, and William G. Wood. Attribute-driven design (ADD), version 2.0. Technical report, Software Engineering Institute, November 2006.
- **[4]** R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere. The architecture tradeoff analysis method. In ICECCS ’98.

Questions?

- **[5]** David Garlan, Robert Allen, and John Ockerbloom. Architectural mismatch, or, why it's hard to build systems out of existing parts. In Proceedings of the 17th International Conference on Software Engineering, pages 179–185, Seattle, Washington, April 1995.

- **[6]** James A. Miller, Remo Ferrari, and Nazim H. Madhavji. An exploratory study of architectural effects on requirements decisions. J. Syst. Softw., 83(12):2441–2455, December 2010.

- **[7]** Dimitri Van Landuyt, Eddy Truyen, and Wouter Joosen. On the modularity impact of architectural assumptions. In Proceedings of the 2012 workshop on Next Generation Modularity Approaches for Requirements and Architecture, NEMARA '12, pages 13–16, New York, NY, USA, 2012. ACM.