IT University of Copenhagen

**Software Architecture**

Session #10

# Architecture Reconstruction

Assoc. Prof. Mircea Lungu

github.com/mircealungu/reconstruction

## Meta

This and following three lectures

- Are material that you don't find in the SAiP textbook
- Is going to be very practical
- Will give you the chance to do a bit of coding for program analysis
- The basis for your individual report
- Have inspired several of your colleagues to choose thesis projects

Feedback & Questions

- Anonymous form
- Email: mlun@itu.dk
- PR on the .md version of the slides on GH if you see bugs

## Imagine ...

- Onboarding on a new system
- Buying a software company
- Having to do
    - a risk assessment for security
    - an architectural evaluation

-- What would be nice to have in all these circumstances but we almost never have?

## Imagine ...

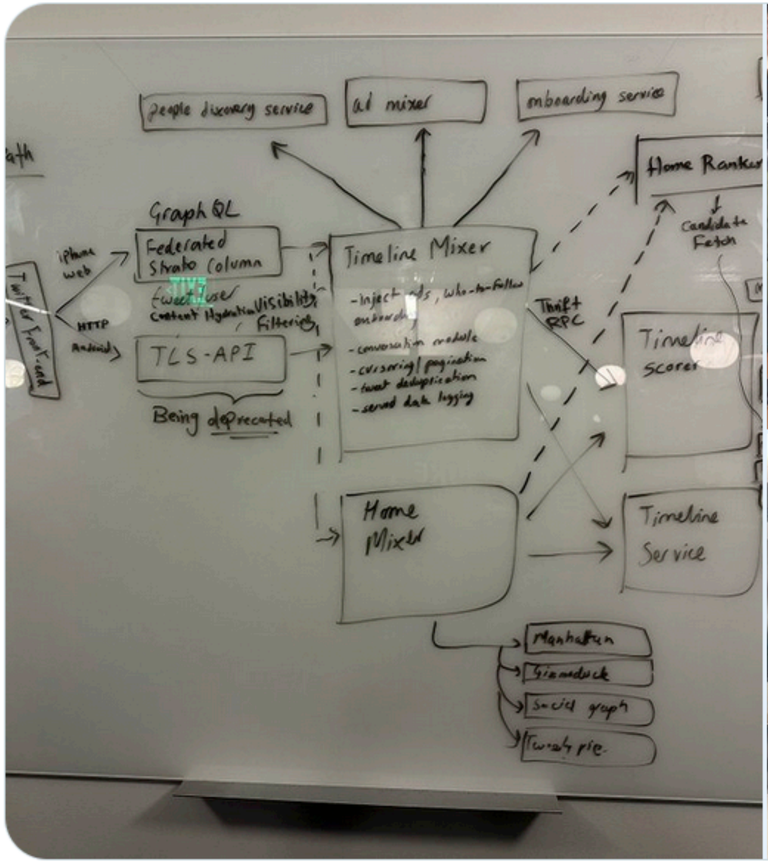
- Onboarding on a new system
- Buying a software company
- Having to do
  - a risk assessment for security
  - an architectural evaluation

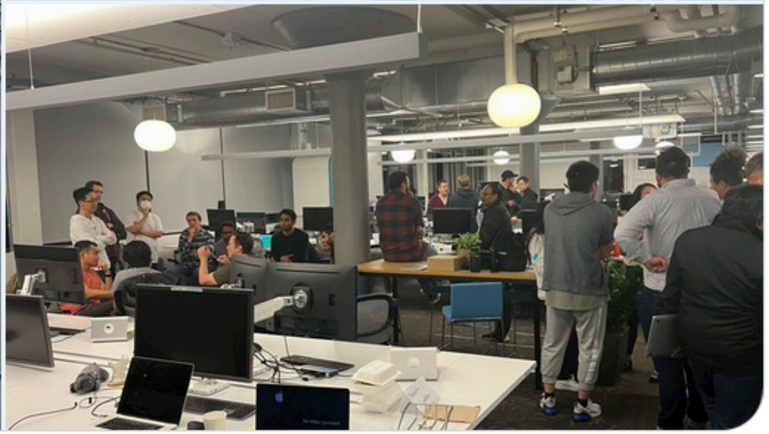
-- What would be nice to have in all these circumstances but we almost never have?

-- **Up to date architectural documentation**

**Elon Musk**
@elonmusk

Just leaving Twitter HQ code review





link to original tweet

## Discussion

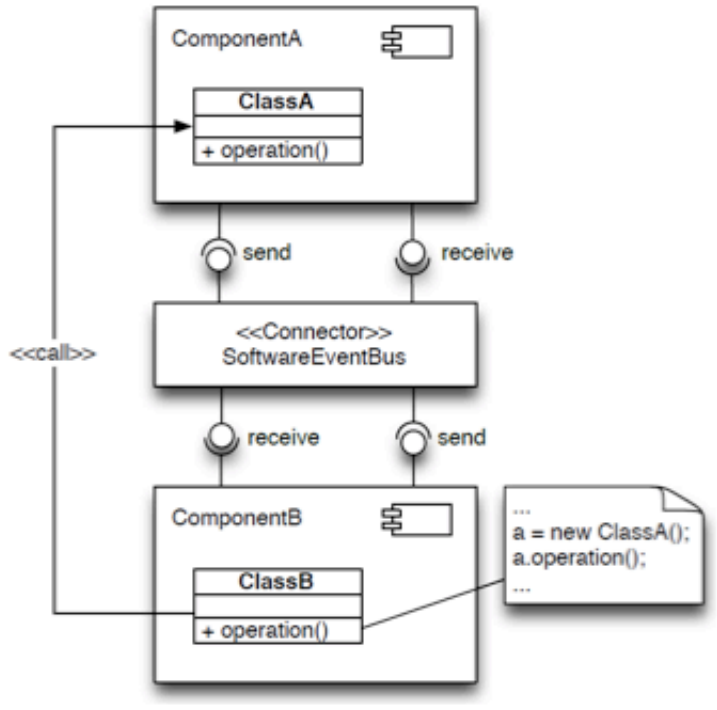
Have you seen architectural documentation for every system?

- No, *Why is it missing?*
- Yes?
  - Is it up to date?
  - *No? Why not?*

## Why is Architectural Documentation Obsolete?

- Hard to maintain
- Link (traceability ) between architecture and code is often not obvious
- No perceived value for the customer
- Because developers make decisions and changes
  - that are not aligned with the original vision => **architectural drift**
  - that go against prescriptive architecture => **architectural erosion**

## Architecture Erosion Example



What could be the cause of erosion here?

Why would it be a problem?

# How to Keep Architectural Documentation up to Date?

1 / **Enforcing architectural constraints**

- special DSLs and tools for architecture constraints definition (e.g. Dictō)
- some are implemented as Unit Tests (e.g. ArchUnit)

## How to Keep Architectural Documentation up to Date?

1 / **Enforcing architectural constraints**

- special DSLs and tools for architecture constraints definition (e.g. Dictō)
- some are implemented as Unit Tests (e.g. ArchUnit)

2 / **Generating architectural diagrams from code**

- as opposed to drawing them in Powerpoint
- we'll see techniques for doing this
- interesting research direction & thesis topic

# How to Keep Architectural Documentation up to Date?

1 / **Enforcing architectural constraints**

- special DSLs and tools for architecture constraints definition (e.g. Dictō)
- some are implemented as Unit Tests (e.g. ArchUnit)

2 / **Generating architectural diagrams from code**

- as opposed to drawing them in Powerpoint
- we'll see techniques for doing this
- interesting research direction & thesis topic

3 / **Reconstructing the Architecture**

- and ideally follow up with one of the previous two

## Architecture Reconstruction (AR)

a.k.a: *architecture recovery* (the two are used interchangeably)

(def.) **A reverse engineering approach that aims at reconstructing viable architectural views of a software application** [1]

- reverse engineering?

[1] Ducasse & Pollet, Software Architecture Reconstruction: a Process-Oriented Taxonomy

## Reverse Engineering

**(def.)** the process of analyzing a subject system to identify the system’s components and their interrela- tionships and create representations of the system in another form or at a higher level of abstraction. (Demeyer et al., Object Oriented Reengineering Patterns, Chapter 1.2)

Note:

- idenitfy
  - components
  - relationships
- higher level of abstraction

Relation with architecture recovery?

## Reverse Engineering vs. Reengineering?

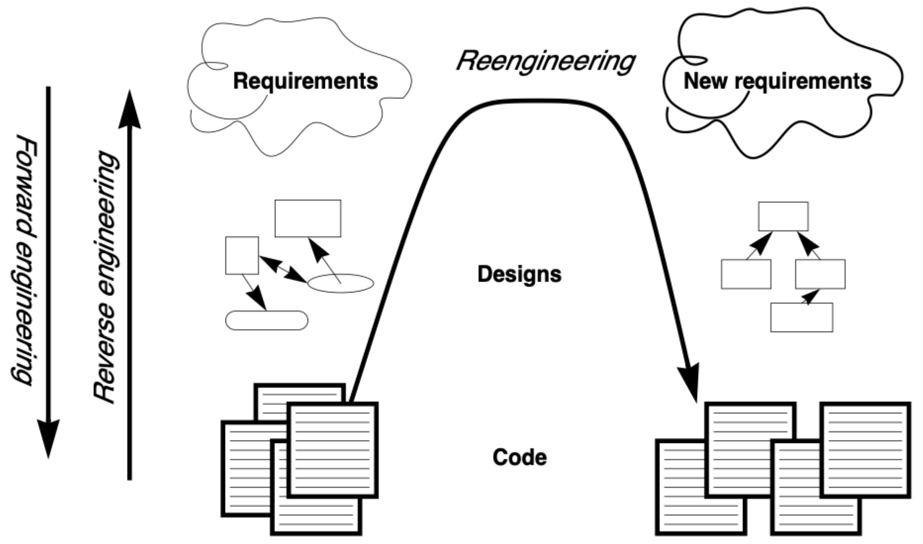


Figure 1.1: Forward, reverse and reengineering

“ Reengineering is the **examination and alteration** of a subject system to reconstitute it in a new form” (Demeyer et al., Object Oriented Reengineering Patterns, Chapter 1.2)

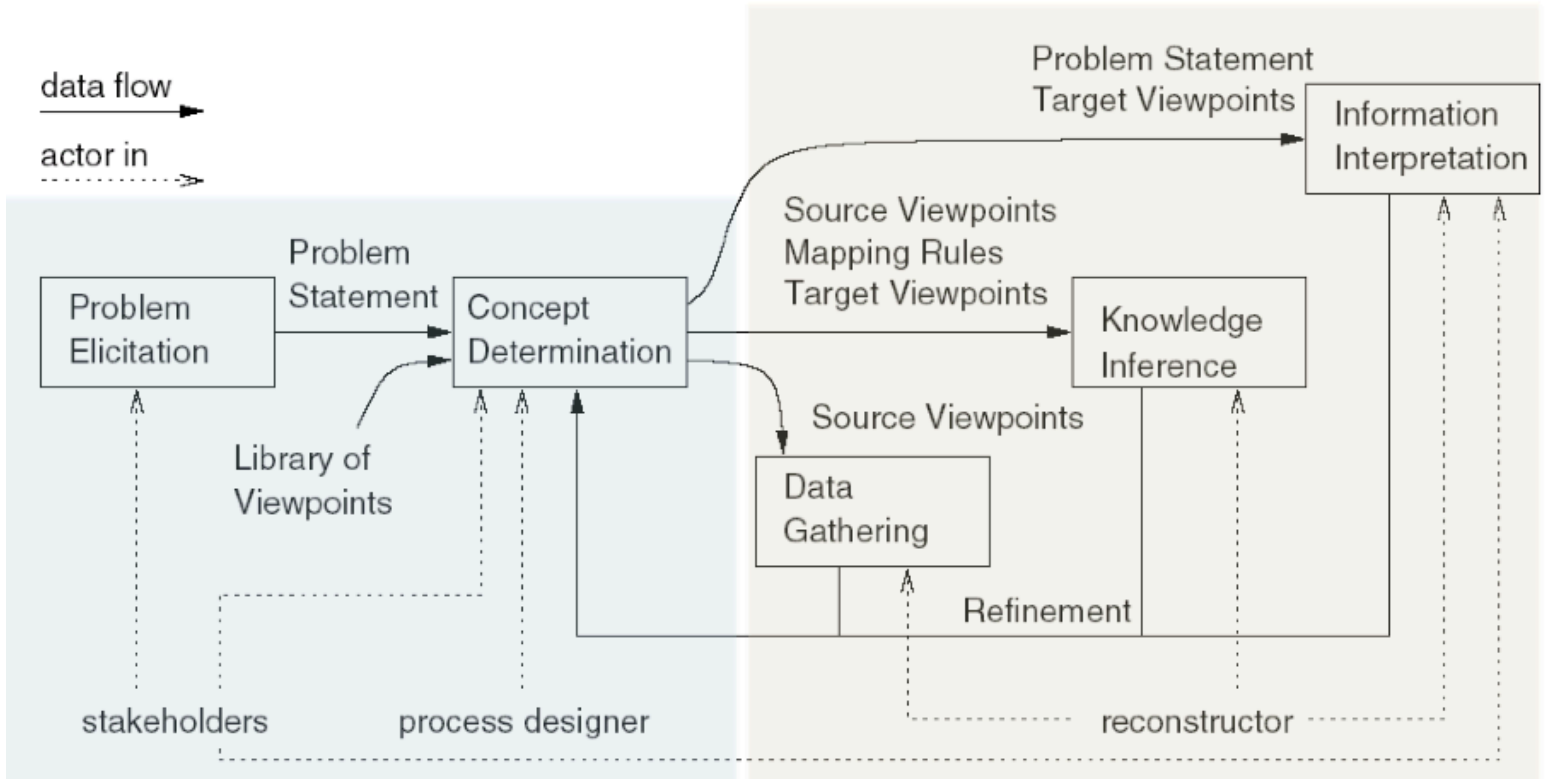
Relation with AR?

## How To Do Architecture Reconstruction?

Symphony: View-Driven Software Architecture Reconstruction

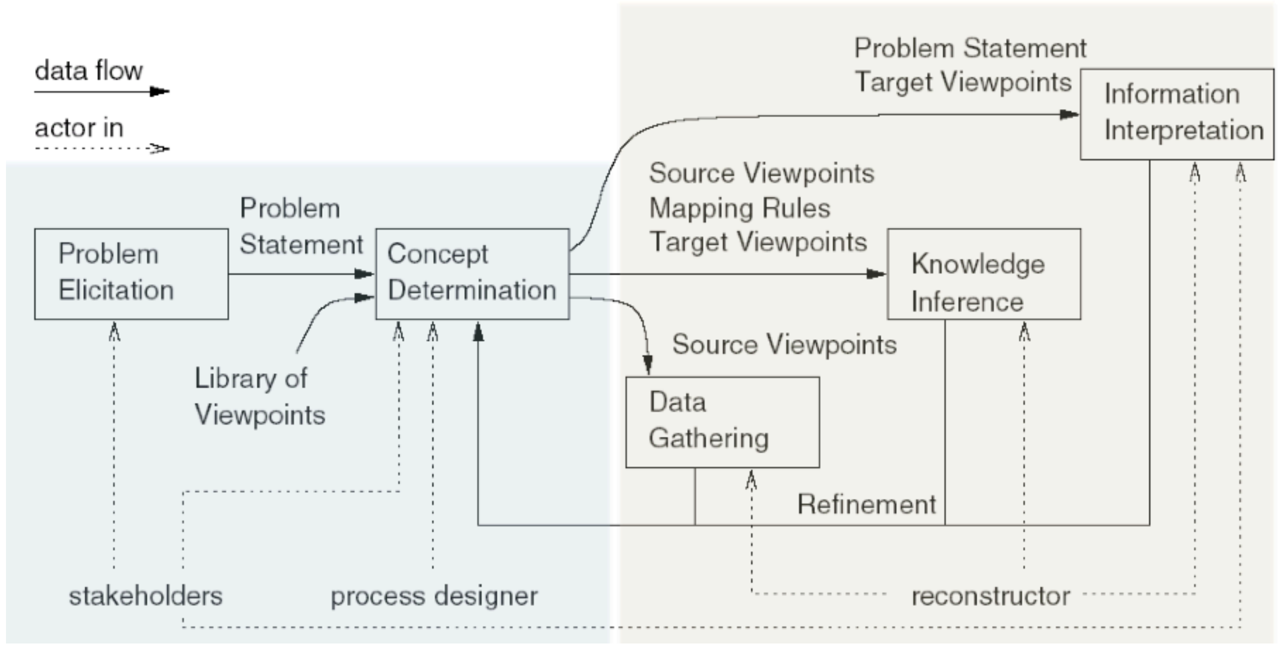
- Paper by Van Deursen et al.
- View-driven approach
- Distinguishes between three kinds of *views*
    1. **Source**
        - view extracted directly from artifacts of a system
        - not necessarily architectural (e.g. see later example)
    2. **Target**
        - describes architecture-as-implemented
        - any of the 3+1 views
    3. **Hypothetical**
        - architecture-as-designed
        - existing documentation
        - presentations

# Symphony Stages: Design (blue) & Execution (yellow)



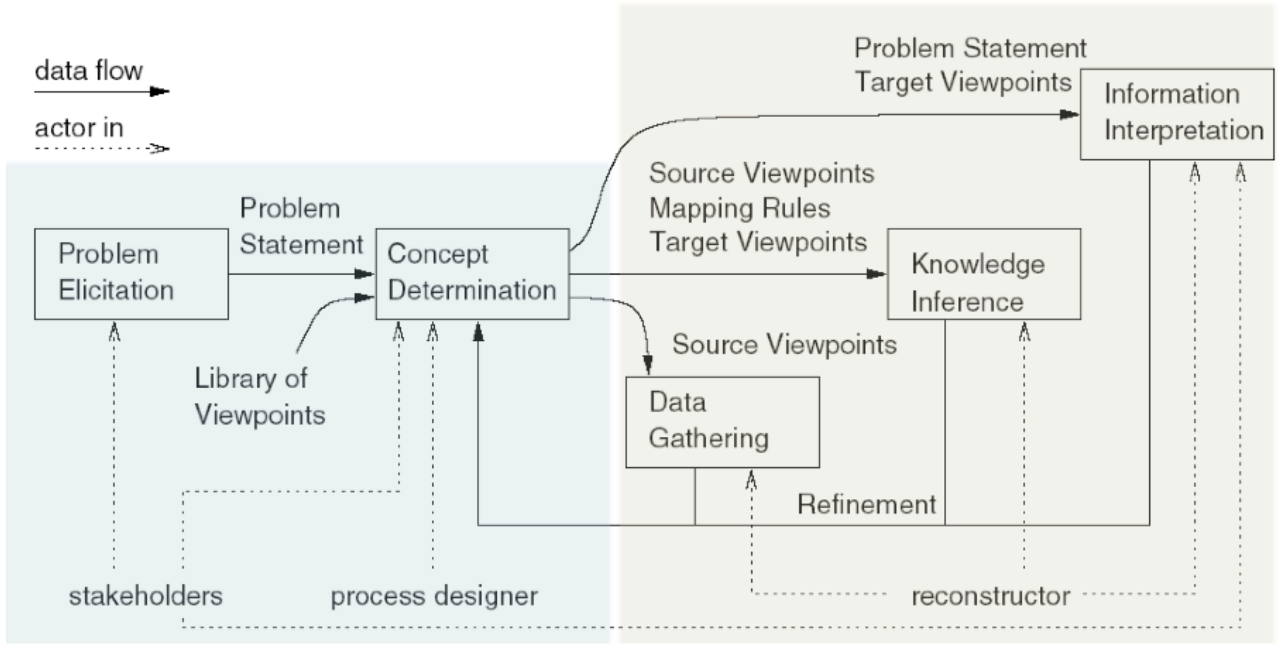
## Desgin: Problem elicitation

- “Business case” for reconstruction
- What is the problem?



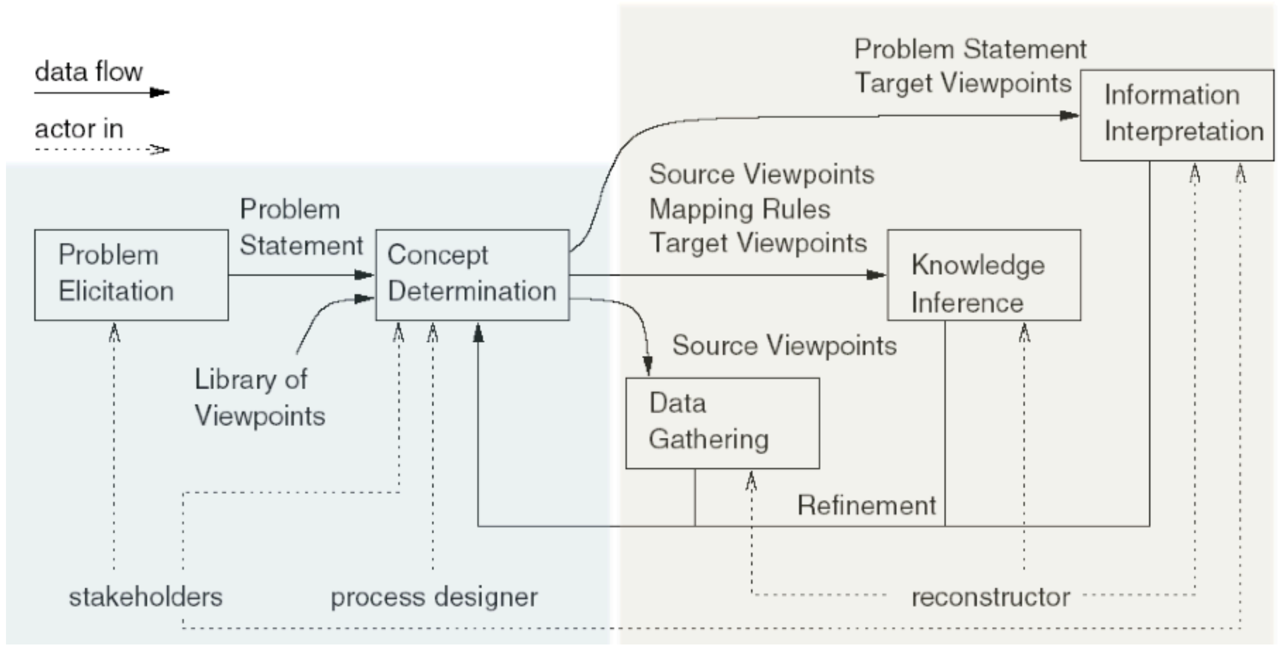
# Design: Concept determination

- What architectural information is needed to solve the problem?
- **Which viewpoints are relevant?**



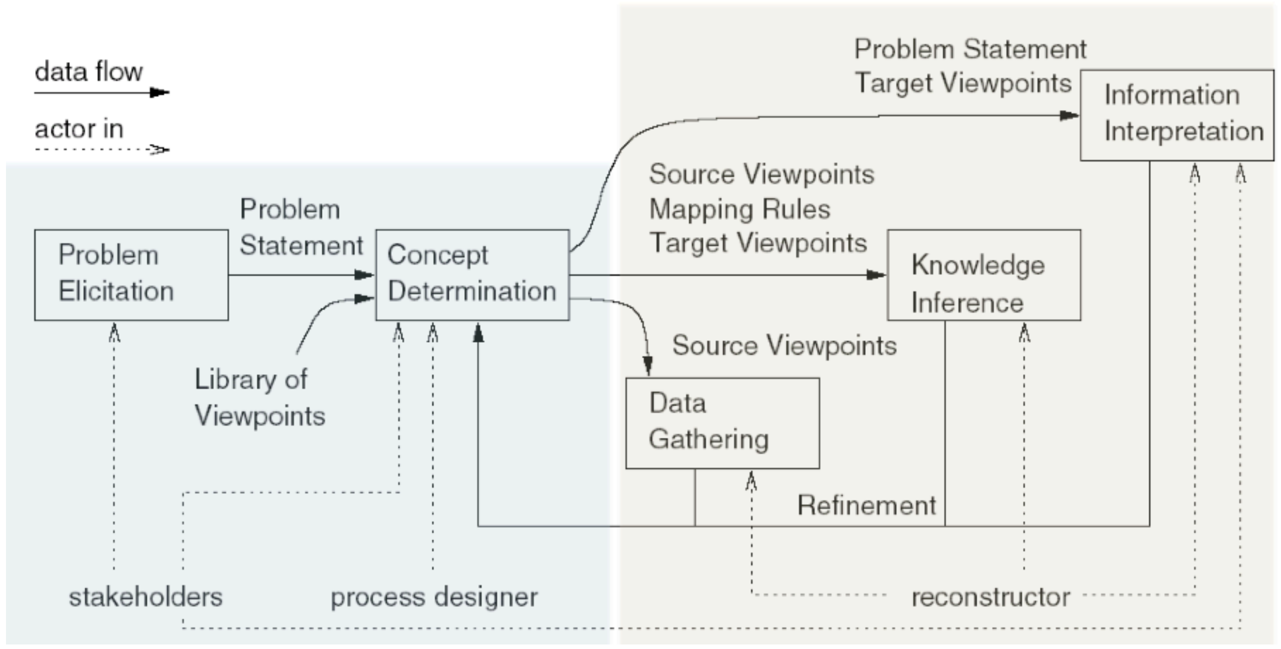
## Execution: Data gathering

- Collecting and extracting low-level source views
- Can involve a multitude of sources



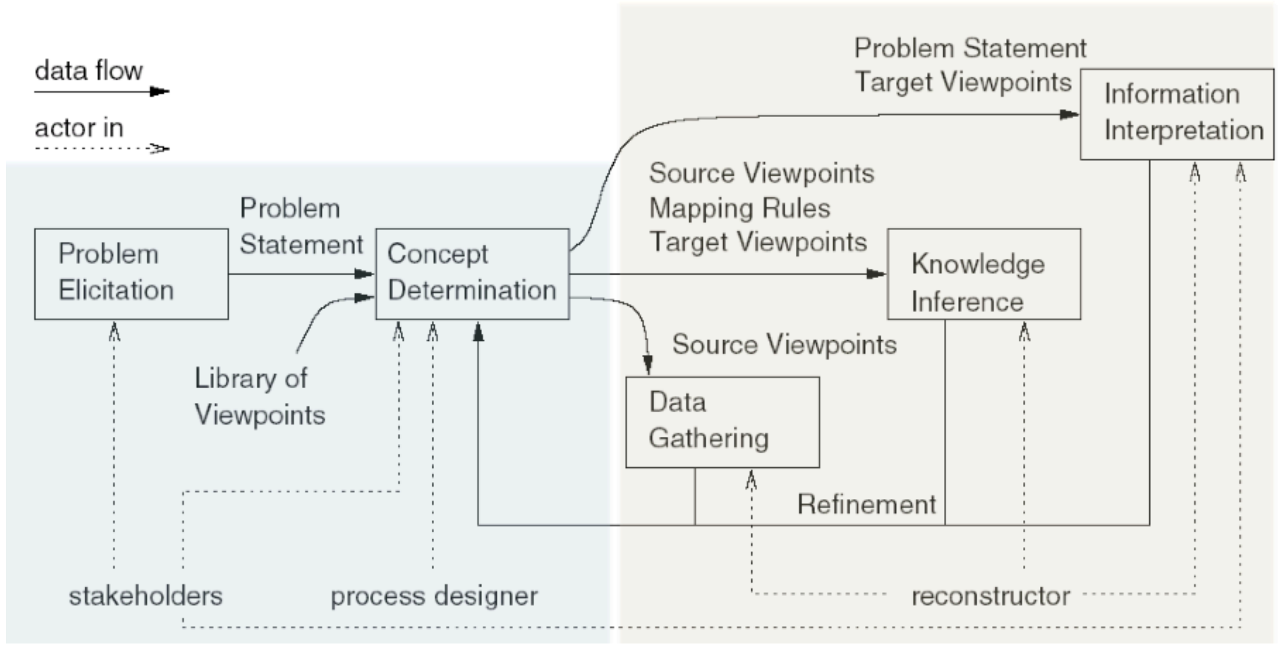
## Execution: Knowledge inference

- Going from source to target views
- Abstracting low-level information



## Execution: Information interpretation

- Visual representation
- Analysis, creating new documentation



## Data Gathering

Example: Google Collab with Basic Data Gathering

Or, *why source viewpoints are not necessarily architectural?*

## Individual Assignment

### Goal is to

- **Recover the architecture of an existing system**
- Document the outcome in an **individual report**
    - brief (not more than 3 -- 5 pages)
    - do not explain to us what Symphony does in the report
    - focus on your results

## Individual Assignment (contd.)

### Case-Study Systems

1. The Zeeguu Project - **default**
   - Online Deployment (invite code: zeeguu-beta)
   - Code:
     - Python Backend: Zeeguu-API
     - React Frontend: Zeeguu-Web
   - A paper about the system
2. Another system that you know
   - if it has comparable complexity (>200 files)
   - you confirm with me about the appropriateness of the system

# Individual Assignment (contd.)

## Viewpoints to Recover

1. Module Viewpoint
   - we will write example code snippets in collab to support this
   - makes the most sense for the Zeeguu system
2. Other Viewpoints
   - some of your colleagues looked at the docker-compose.yml to figure out deployment
   - might make more sense for another system - the Zeeguu one is too simple (could be done together with the module)

## Individual Assignment (contd.)

### Tools

- Are important for recovery
- **If you can program**, then this is your chance to be coding **analysis tools** over the upcoming lectures
    - you can still code as a team! (you only have to write the analysis on your own)
- **If you can't program**, then you'll have to find third party tools (the time the programming ones spend on programming, you'll be spending on finding third party tools)

## For Next Week

### Reading

- Symphony: View-Driven Software Architecture Reconstruction
- Demeyer et al., Object Oriented Reengineering Patterns (Chapter 1.2)

### Practice & Think About

- Google Collab with Basic Data Gathering
  - Understand the code
  - Think about techniques for "abstracting" this information
- Can you find equivalent off-the shelf tools?

### Questions & Feedback

- Use the anonymous form
- Or the forum if it's of general interest