lab-soft
software engineering laboratory

Identifying Design Problems in the Source Code

A Grounded Theory

Leonardo Sousa[1], Anderson Oliveira[1], Marcos Kalinowski[1], Rafael de Mello[1], Willian Oizumi[1], Simone Barbosa[1], Alessandro Garcia[1], Roberto Oliveira[1], Carlos Lucena[1], Rodrigo Paes[1], Baldoino Fonseca[2], Jaejoon Lee[3]

[1] PUC-Rio, [2] UFAL, [3] Lancaster University

ICSE '18

Agenda

- Introduction
- Concepts and Motivation
- Research Design
- Grounded Theory
- Related Work
- Threats to Validity
- Conclusion
- Future Work

lab-soft
software engineering laboratory

Introduction

Introduction

- A *Design problem* is the result of one or more inappropriate design decisions
- It impacts the non-functional requirements of a software system
- May cause the redesign or even the discontinuation of software systems

Introduction

- Identifying *Design problems* can itself quickly turn into a very complex task
  - Lack of Documentation
  - Symptoms scattered through program elements

- Many studies proposed solutions for assisting developers in identifying *Design Problems*

Introduction

- However, they oversimplified the process of identification

- RQ: How do developers identify *Design Problems* in source code?

- Build a **Grounded Theory** to explain

lab-soft
software engineering laboratory

Concepts and Motivation

Concepts and Motivation

- Design Problems
- Design Problems Identification
- Grounded Theory

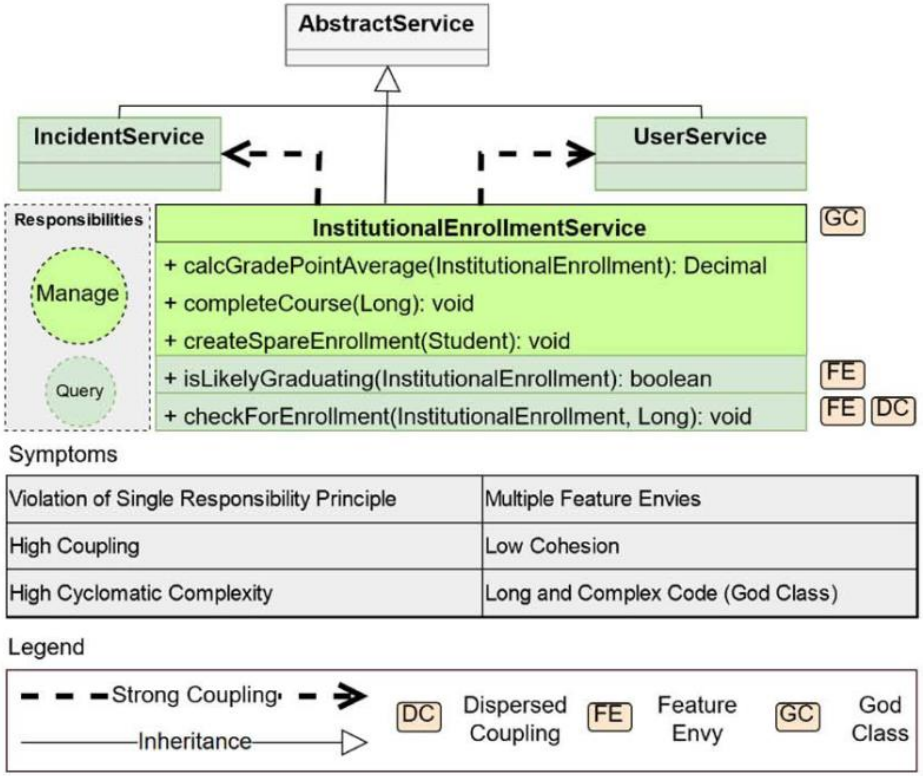
Concepts and Motivation

- Design Problems

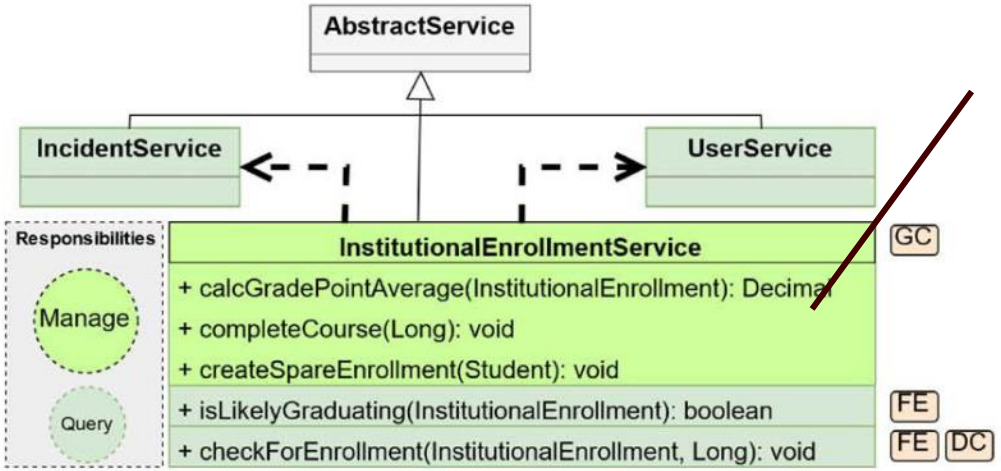
"A design problem arises from one or more design decisions which, when implemented in source code, negatively affect software quality requirements."

Concepts and Motivation

- Design Problems Identification



Concepts and Motivation

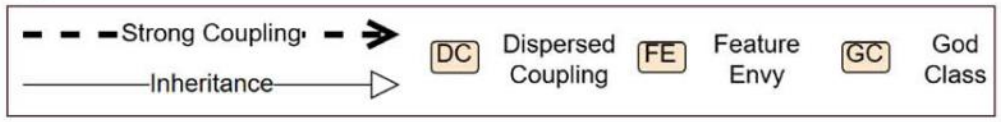


1º - Developer found Feature Envy in Methods

Symptoms

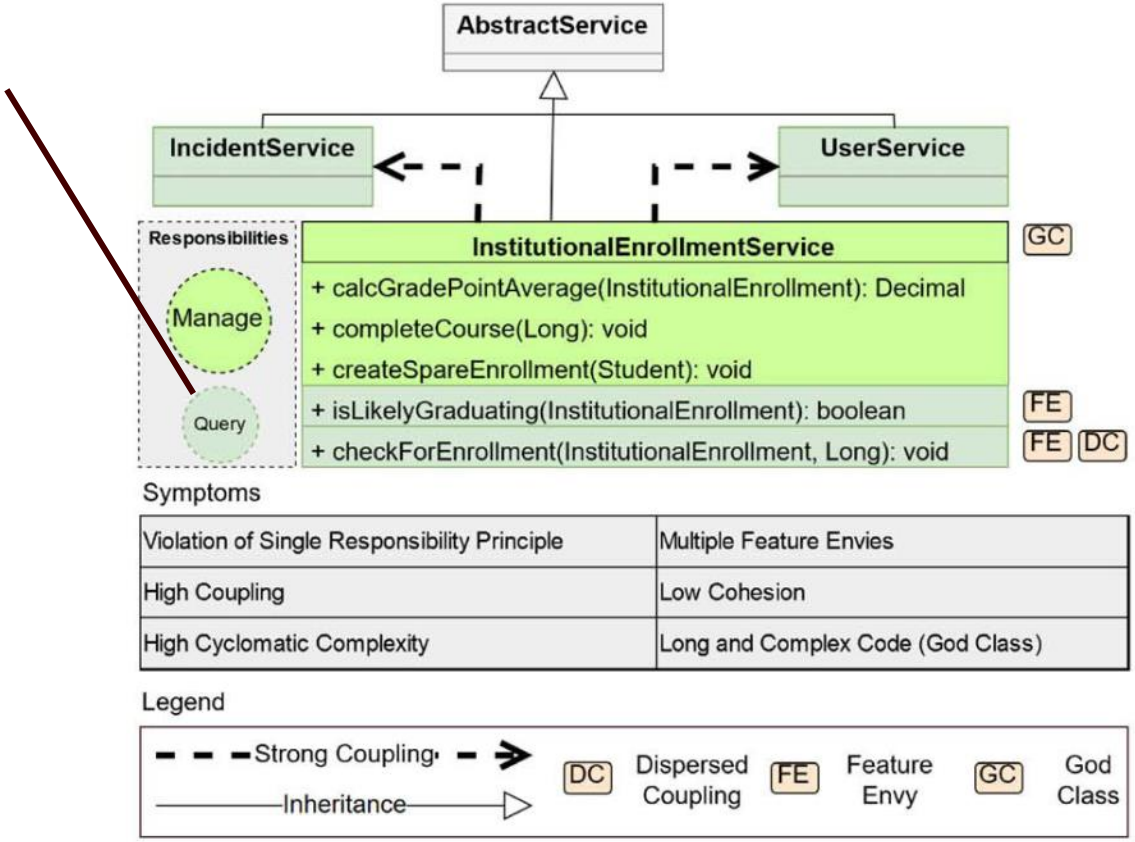
Violation of Single Responsibility Principle	Multiple Feature Envies
High Coupling	Low Cohesion
High Cyclomatic Complexity	Long and Complex Code (God Class)

Legend

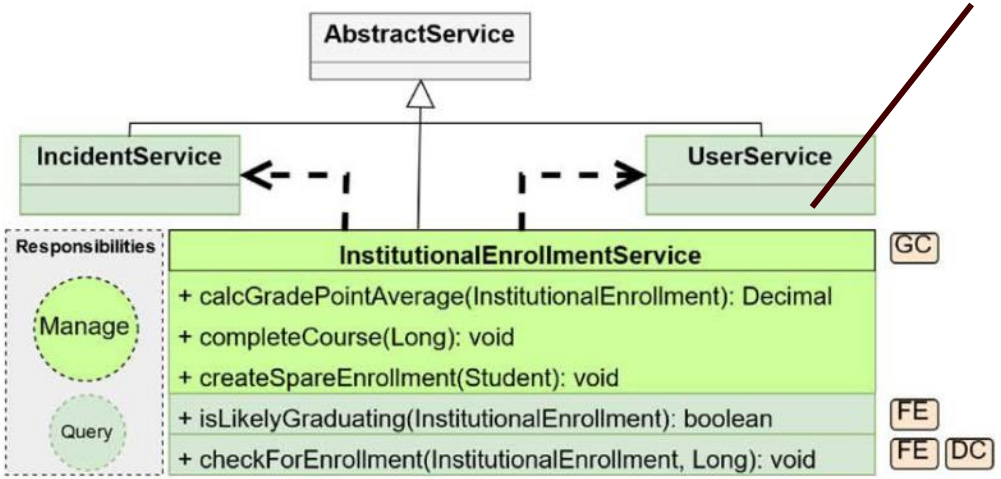


Concepts and Motivation

2º - Based on Feature Envies findings, developer identifies two distinct responsibilities implemented by the class



Concepts and Motivation

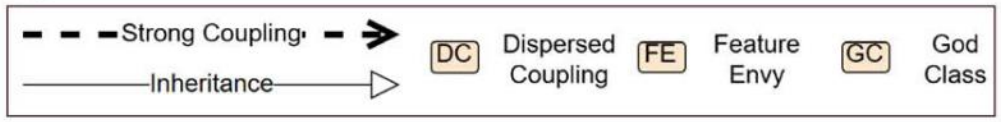


3º - Developer found high dependency with other classes

Symptoms

Violation of Single Responsibility Principle	Multiple Feature Envies
High Coupling	Low Cohesion
High Cyclomatic Complexity	Long and Complex Code (God Class)

Legend



Concepts and Motivation

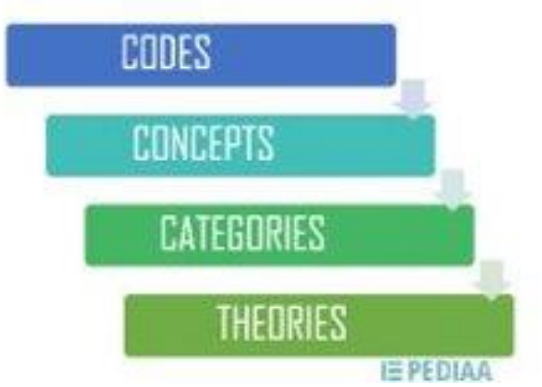
- Design Problems Identification
  - Developer concludes the class has a design: problem:
    i. Implementation of two unrelated responsibilities
    ii. Strong coupling with other classes
    iii. Overly complex method
  - Had to analyze at least three other classes and rely on three types of symptoms
- Several tools focus only on one type of symptom

Concepts and Motivation

- Grounded Theory (GT)
  - Understanding a phenomenon like the Identification of Design Problem requires more explanation than a couple of examples
  - Which can be achieved by a theory

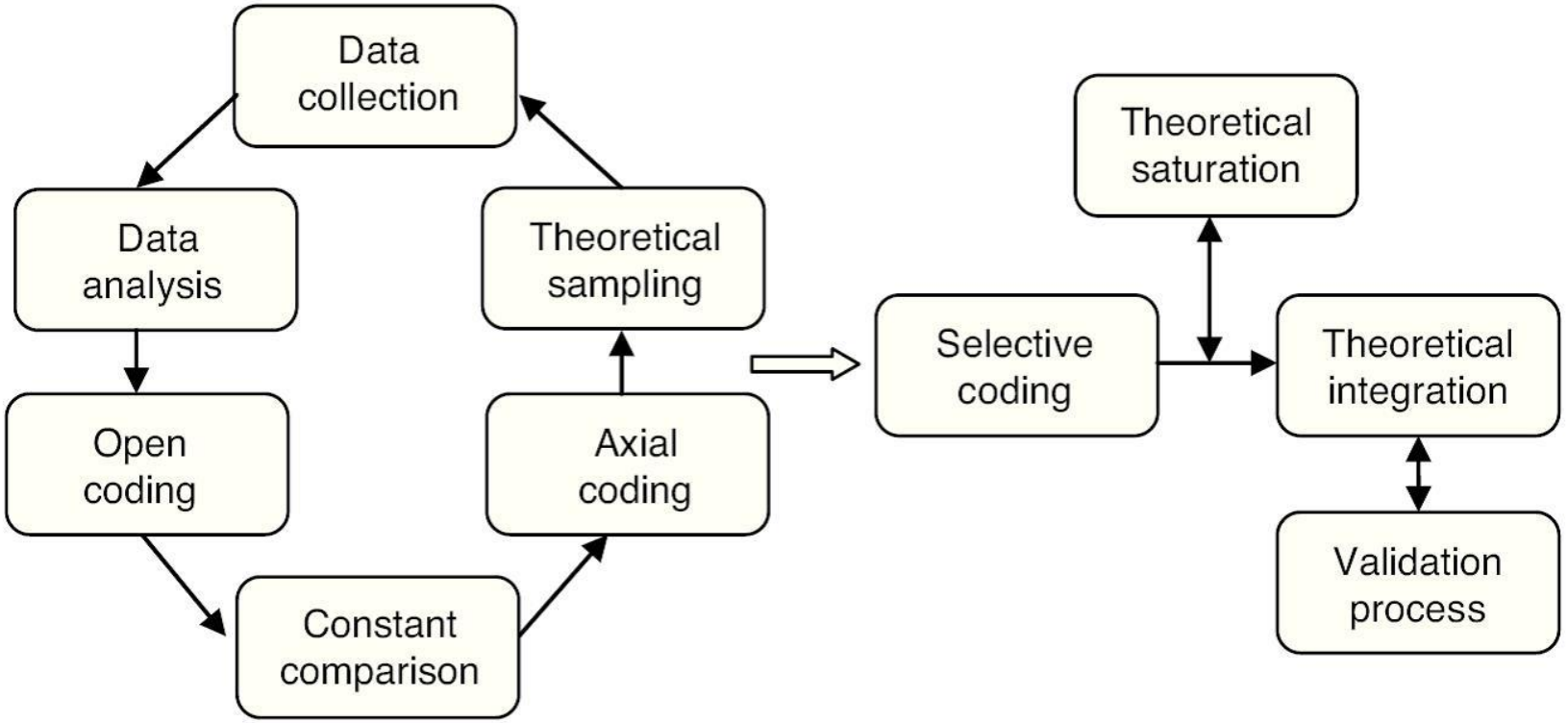
Concepts and Motivation

- Grounded Theory (GT)
  - What is GT?
    - A Qualitative research method that uses a systematic set of procedures to inductively develop a theory about a phenomenon
  - Adopted *Strauss and Corbin's* version



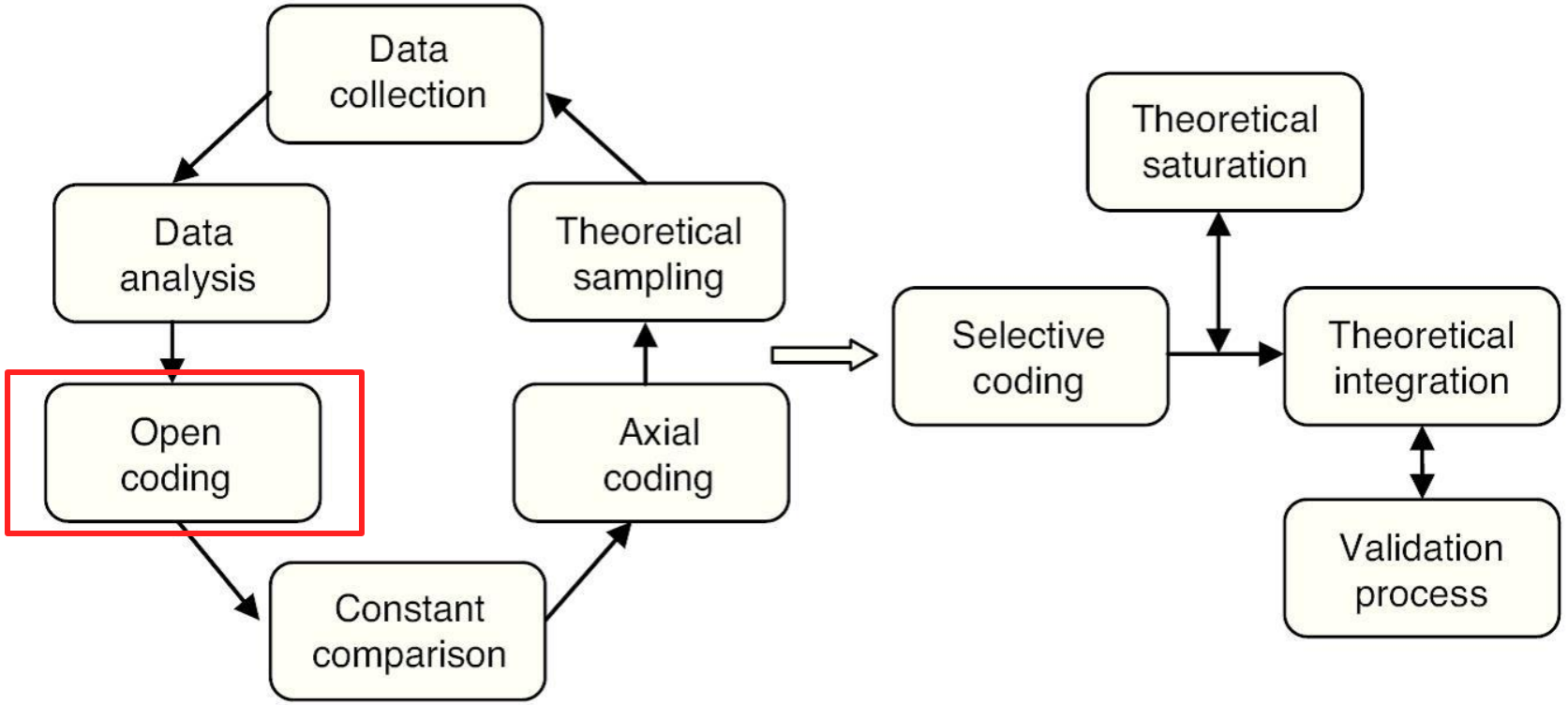
Concepts and Motivation

- Grounded Theory (GT)



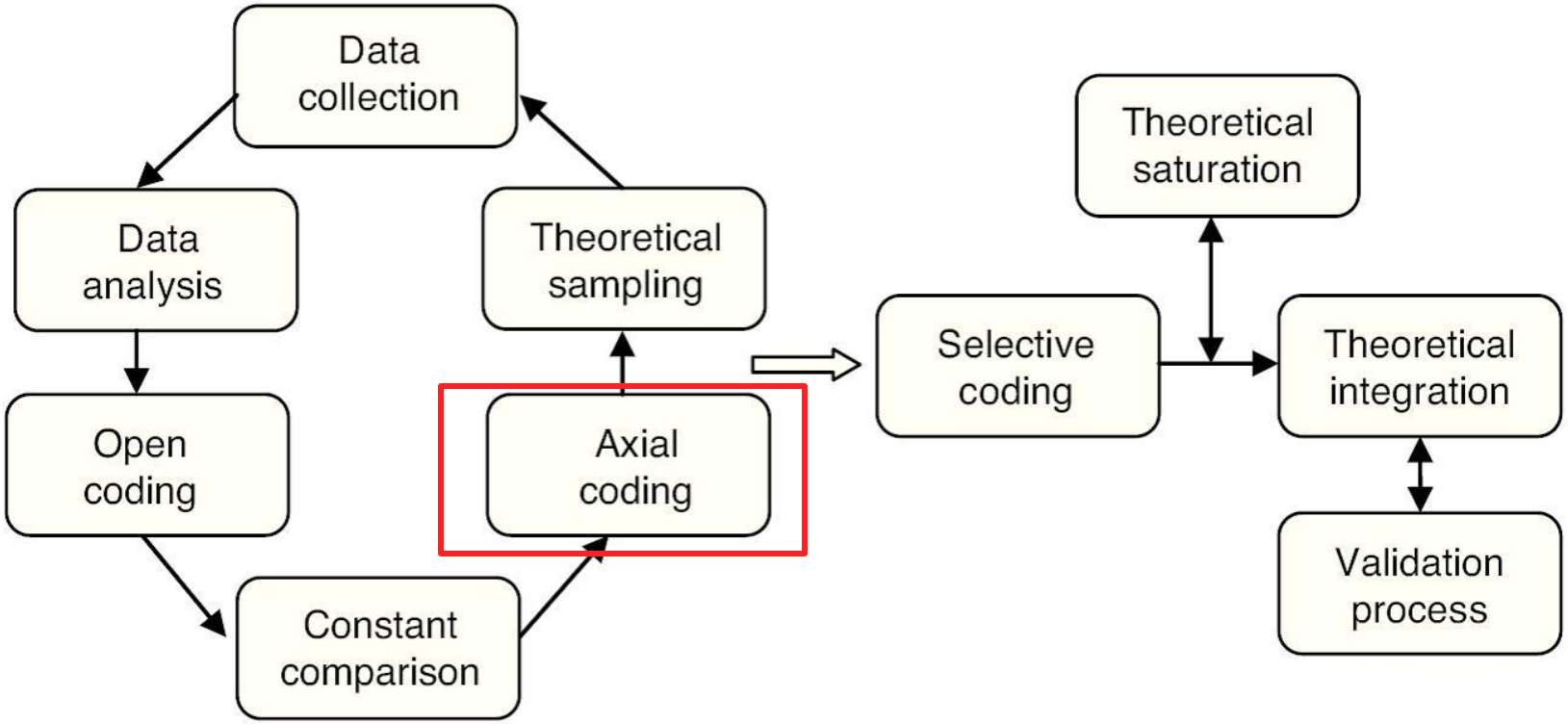
Concepts and Motivation

- Grounded Theory (GT)



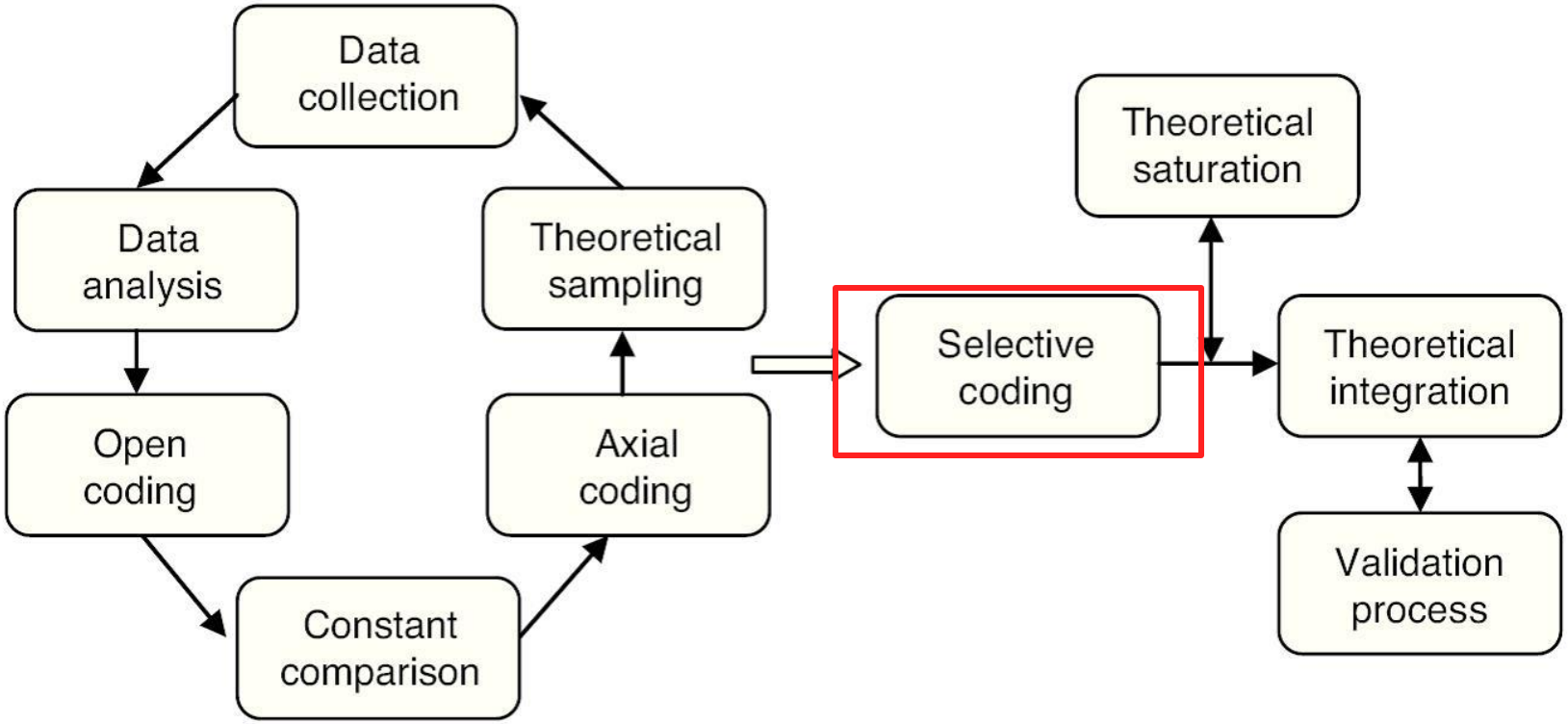
Concepts and Motivation

- Grounded Theory (GT)



Concepts and Motivation

- Grounded Theory (GT)



lab-soft
software engineering laboratory

Research Design

Research Design

- Multi-trial controlled experiment in different software companies

- Analyzed collected data and derived a theory (GT) that's provides an **overview**, an **explanation** and **understanding** on how developers identify design problems on source code

Software Systems & Developers' Selection

- Criteria to select the companies
  - Experience of their developers
  - Size in terms of number of developers in a project
  - Application domain of their projects
  - Development Process
- Main Goal
  - Heterogeneity of companies and different from author's collaboration context

Software Systems & Developers' Selection

- Five Software companies were selected

- From North and Northeast of Brazil

Characteristic	Company 1	Company 2	Company 3	Company 4	Company 5
Company Type	Government Company	Private Company	Private Company	Private Company	Government Company
Domain	University Administration	Software Factory	Software Factory	Industrial Automation	Government Administration
Programming Language	Java	Java	Java	Java, Android, IOS	Java
Selected Software Systems	S1	S2	S3	S4, S5, S6	S7, S8

Software Systems & Developers' Selection

- Systems that met the characteristics:
  1. Systems in different stages of design degradation
  2. Systems from different domains and with different sizes with respect to the number of modules and developers
  3. Systems that were not in the initial versions
  4. Systems developed in Java

Software Systems & Developers' Selection

- Developers were the subjects of experiment
  - Pairs, except by T10 (asked by manager)

Team	ID	Experience (years)	System	Company
T1	D1	3	S1	1
	D2	5		
T2	D3	13	S1	1
	D4	14		
T3	D5	14	S1	1
	D6	6		
T4	D7	7	S2	2
	D8	2		
T5	D9	4	S2	2
	D10	4		
T6	D11	10	S3	3
	D12	8		
T7	D13	12	S4	4
	D14	13		
T8	D15	4	S5	4
	D16	8		
T9	D17	4	S6	4
	D18	10		
T10	D19	7	S7	5
	D20	7		
	D21	9		
T11	D22	12	S8	5
	D23	9		

Experimental Tasks

- Four Activities:
  1. Subjects characterization
     - Education level, Professional Experience with software development, Java Programming, Knowledge about design problems
  2. Training
     - About software design and design problems
     - 40 minutes
     - Examples of problems: Component Overload, Fat Interface, Cyclic Dependency

Experimental Tasks

3. Problem identification
   - Identify Design Problems on systems
   - 90 minutes
   - Think-aloud method
   - Task recorded on video

4. Follow-up
   - Questionnaire about their perception of the task
   - And whether each symptom was useful to identify a design problem

Provided Data

- Make Activity 3 (Problem identification) more realistic
- Simulating use of tools
- Detected symptoms combining the output of methods and tools:
  - SonarQube, DECOR, Organic

Provided Data

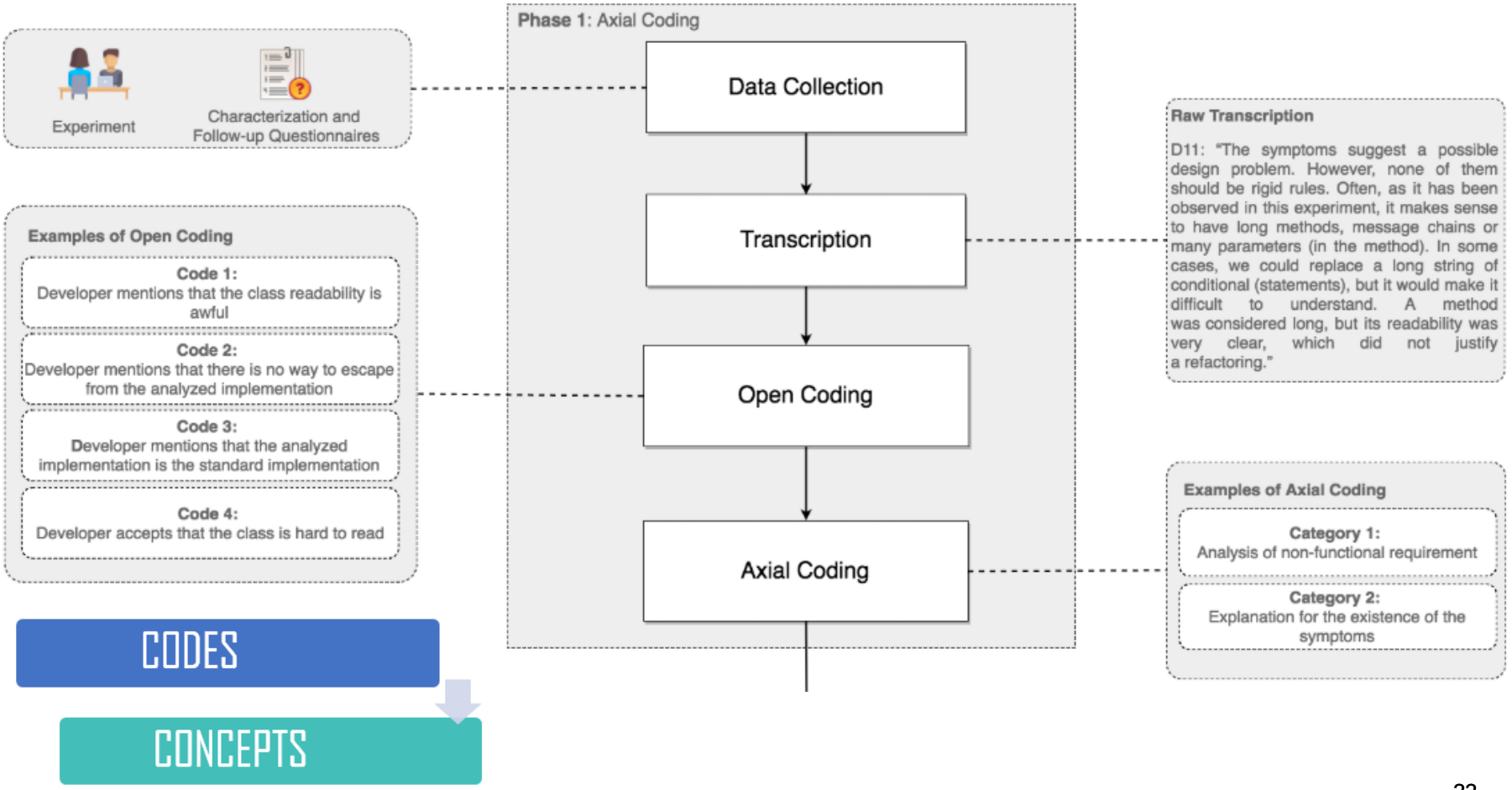
- Types of Symptoms
  - Violation of Non-functional Requirements
  - Code Smells (15 types from fowler's catalog)
  - Design Pattern and Object-Oriented Violation
  - Quality Requirements
  - Visual representation of software
- Developers were not limited to this outputs
- Presented the symptoms on a web page

Data Collection and Analysis

- Three main phases

- Used the framework proposed by *Sjøberg* to represent and describe the theory

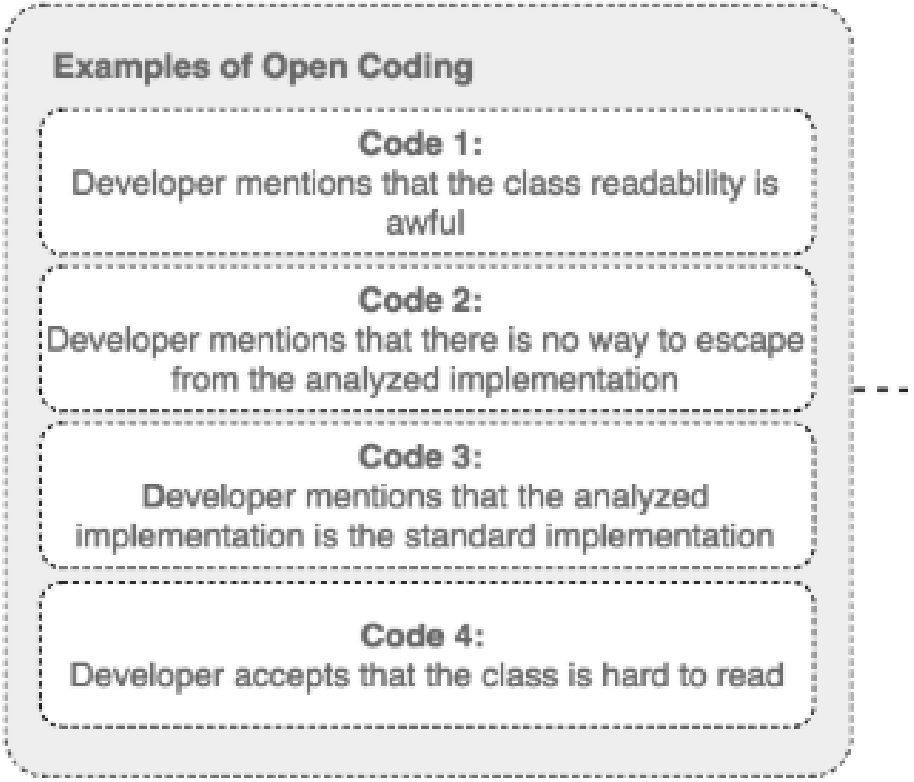
Data Collection and Analysis



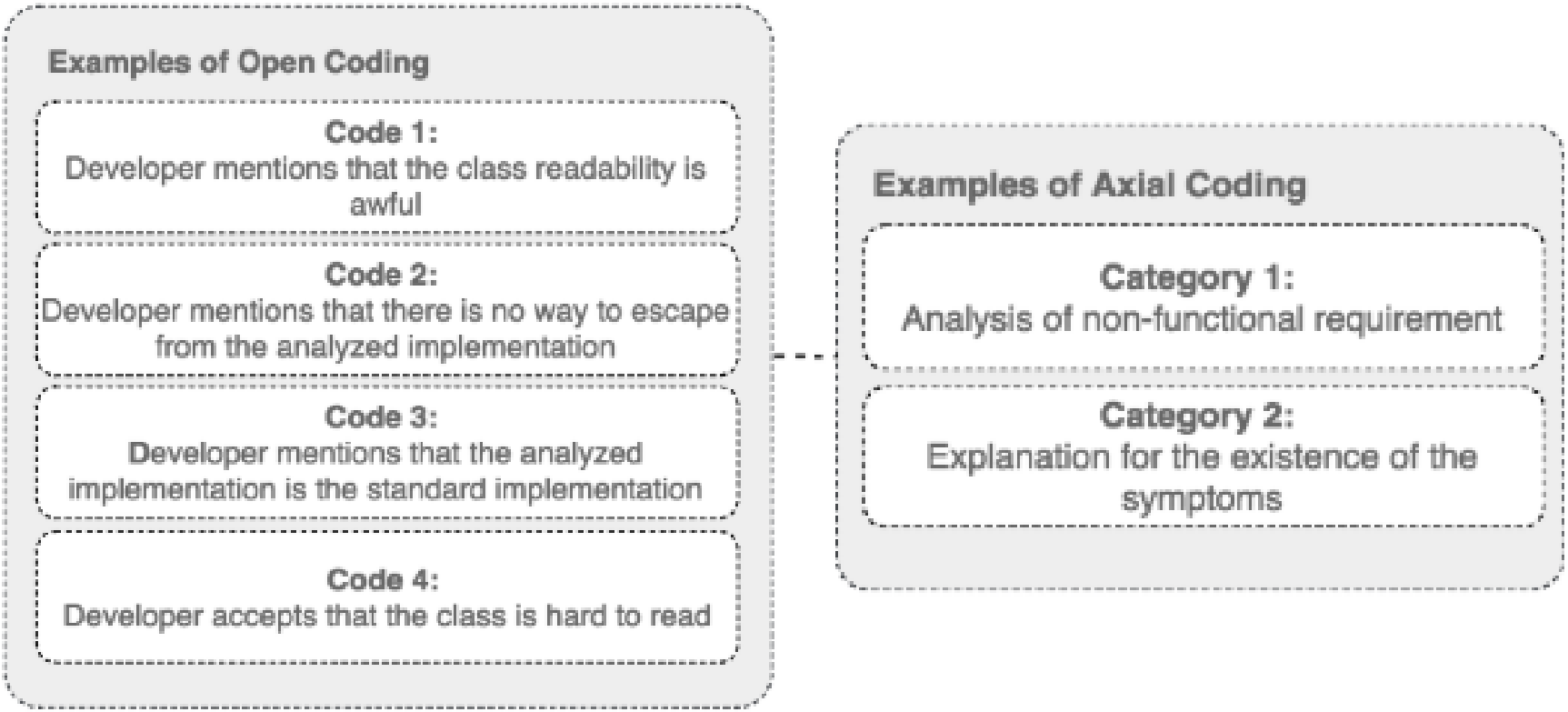
Data Collection and Analysis

TRANSCRIPTION

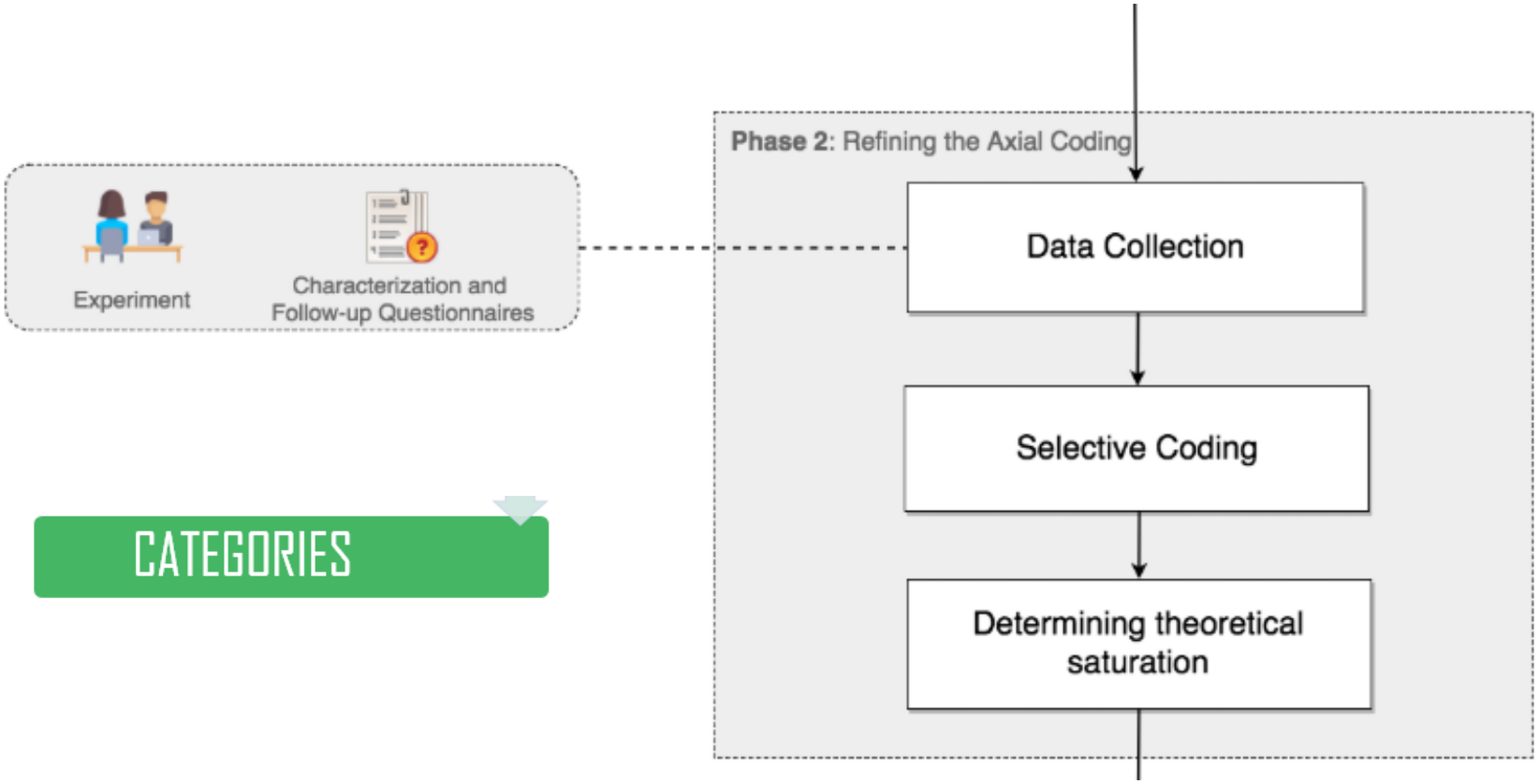
"D6: The readability here is awful, but there is no way to escape from this (implementation). That is the standard (implementation). (...) indeed, it (the class) is not easy to ready"



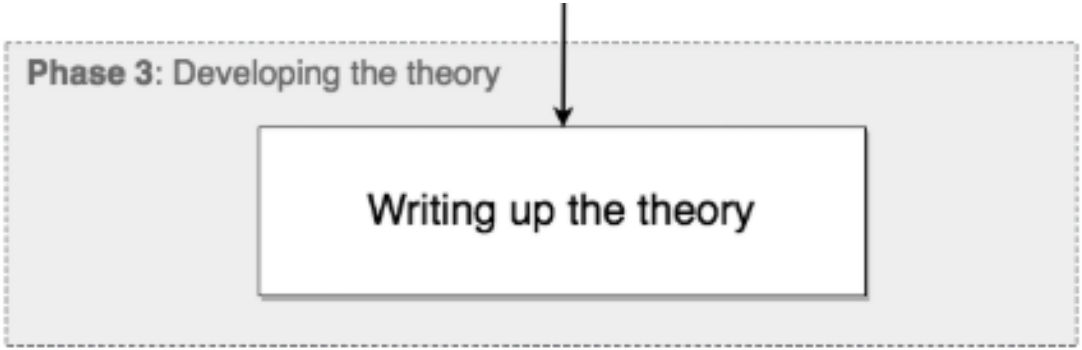
Data Collection and Analysis



Data Collection and Analysis



Data Collection and Analysis



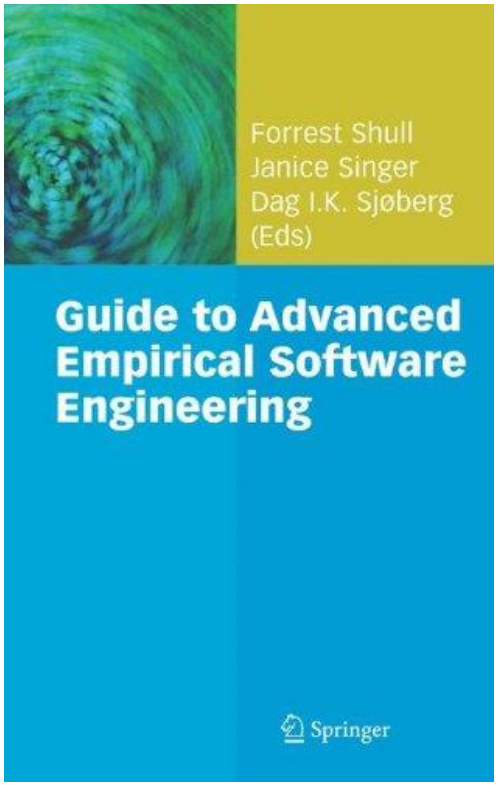
Construct	Proposition	Explanation	Scope
Basic particle that composes a theory	Interaction among constructs	Factors behind the propositions	Universe to which the theory is applicable

lab-soft
software engineering laboratory

Grounded Theory

Grounded Theory

- Framework proposed by *Sjøberg*
  - *Building Theories in Software Engineering*
  - Chapter 12 on *Guide to Advanced Empirical Software Engineering*



Grounded Theory

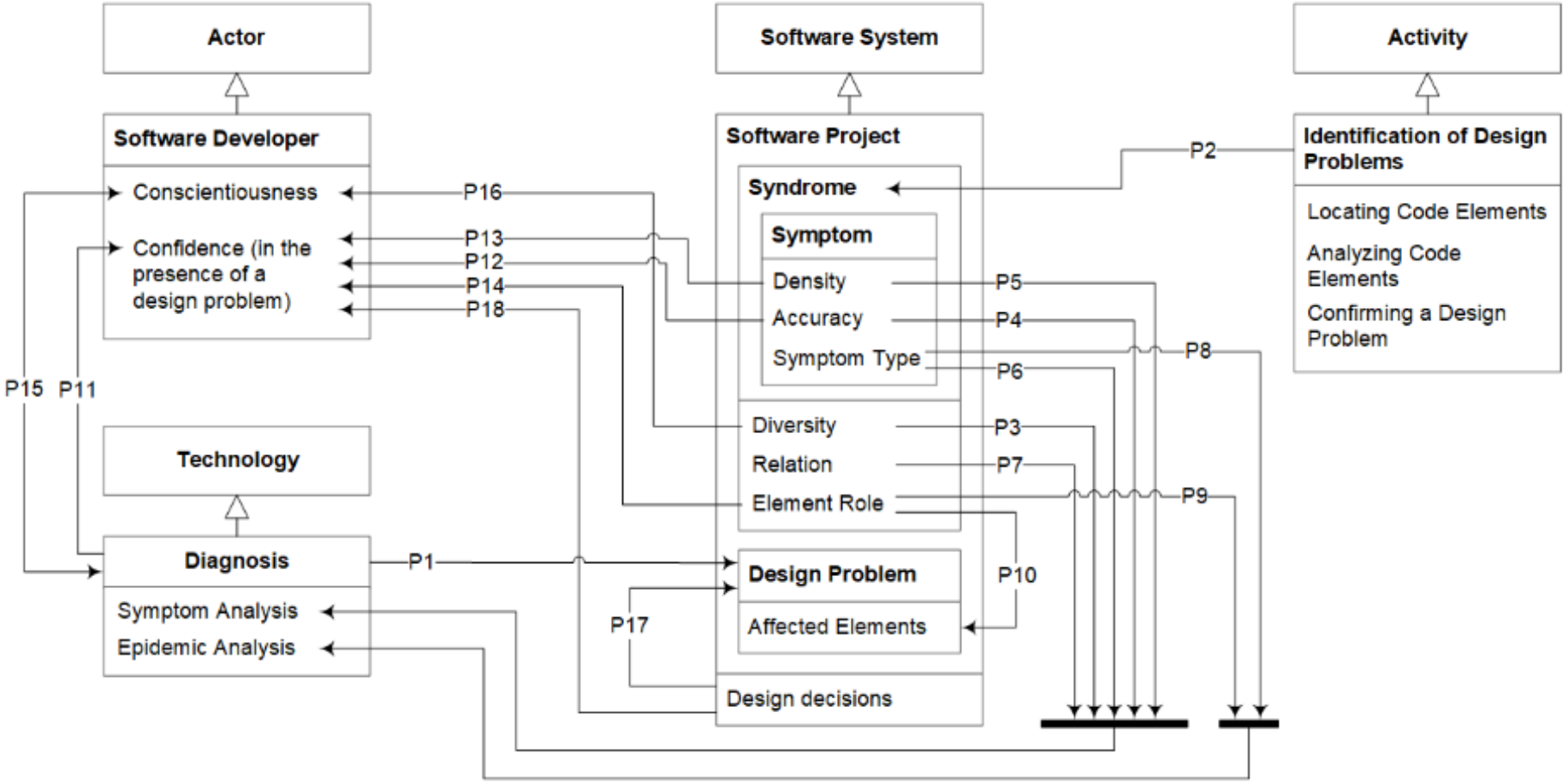
- Theory types
  - Analysis, Explanation, Prediction, Explanation and prediction, Design and Action

- Type of proposed theory: Explanation
  - Identification of Design Problems
  - Design Problem Symptoms
  - Design Problem Diagnosis

Grounded Theory

- Scope: the theory is supposed to be applicable in systems in which developers intend to identify design problems by analyzing symptoms that manifest themselves in the source code

Grounded Theory



Identification of Design Problems

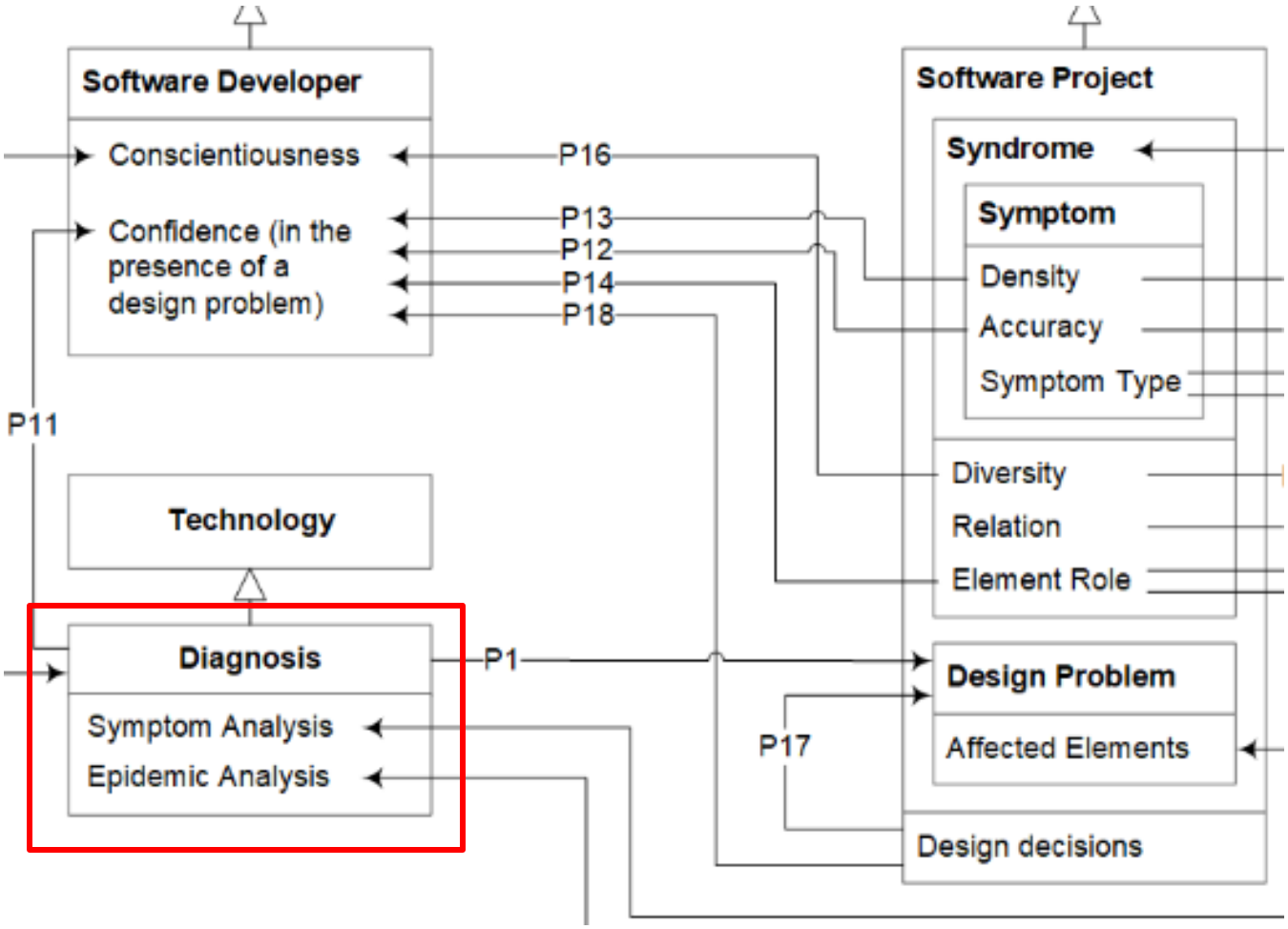
C1	Design Problem	A design decision that negatively impacts quality attributes
C2	Design Decisions	Decisions made during the software development process
C3	Symptom	An indication of the presence of a design problem

"A design problem (C1) arises in code elements due to one or more design decisions (C2), made intentionally or accidentally. In fact, a design problem may affect one or more elements in such a way that these elements manifest symptoms of its presence. A symptom (C3) is an indication of the presence of a design problem."

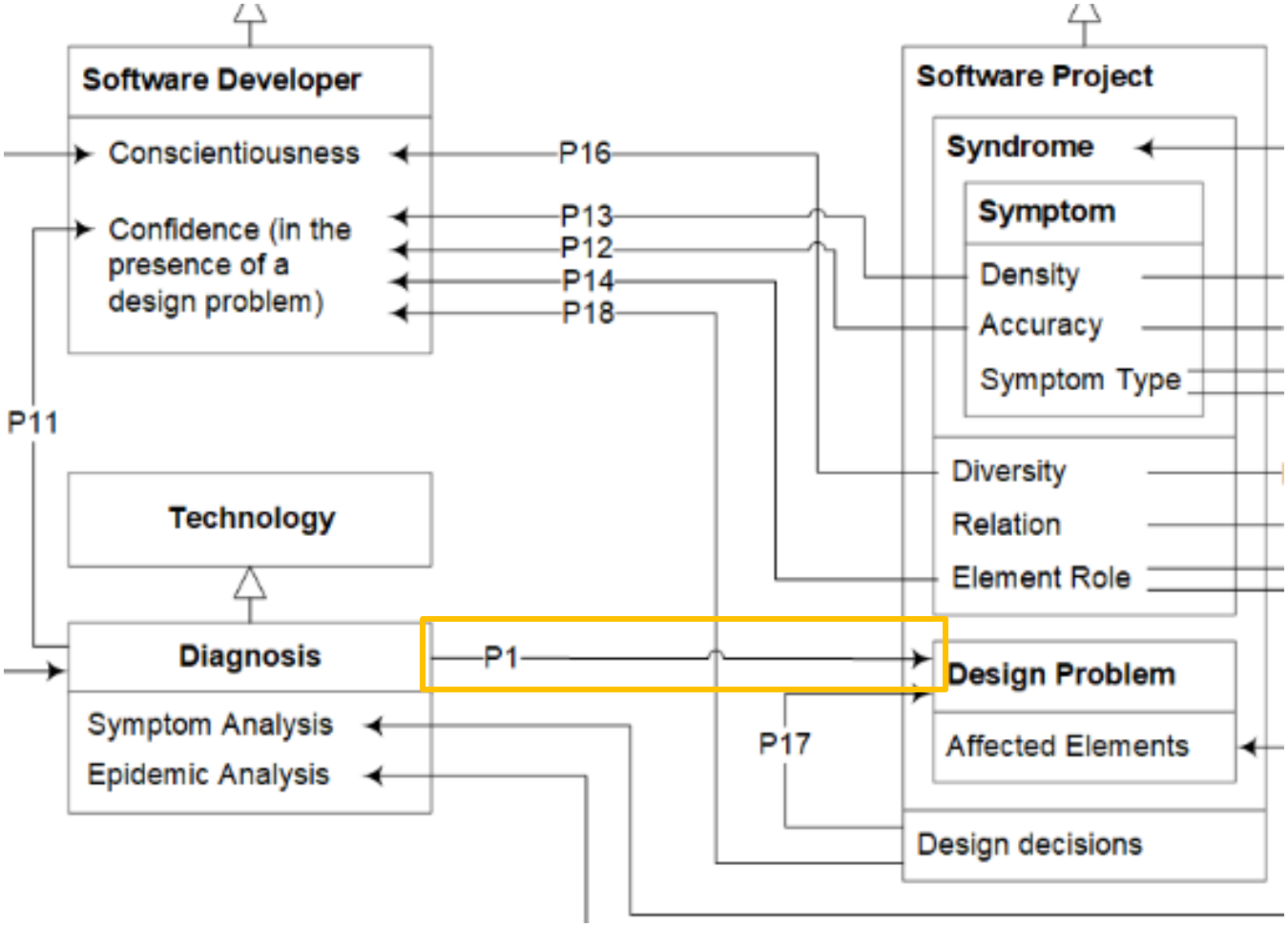
Identification of Design Problems

- Three Steps to Identify Design Problems Using Symptoms
- Propositions:
  - *P1: The diagnosis affects the identification of design problems*
  - *P2: Identification of design problem has three steps in which developers relies on design problems symptoms in all of them*

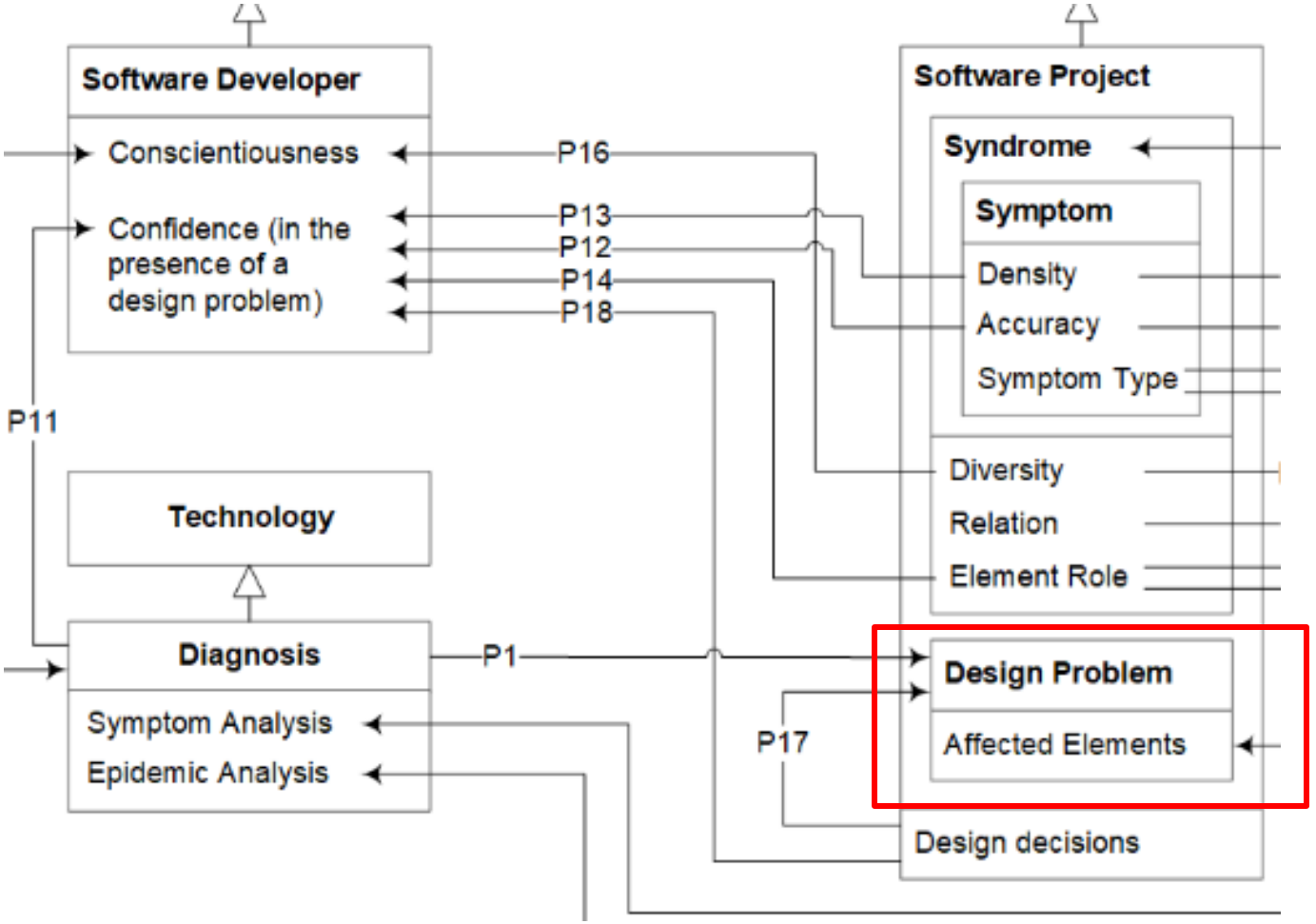
Identification of Design Problems



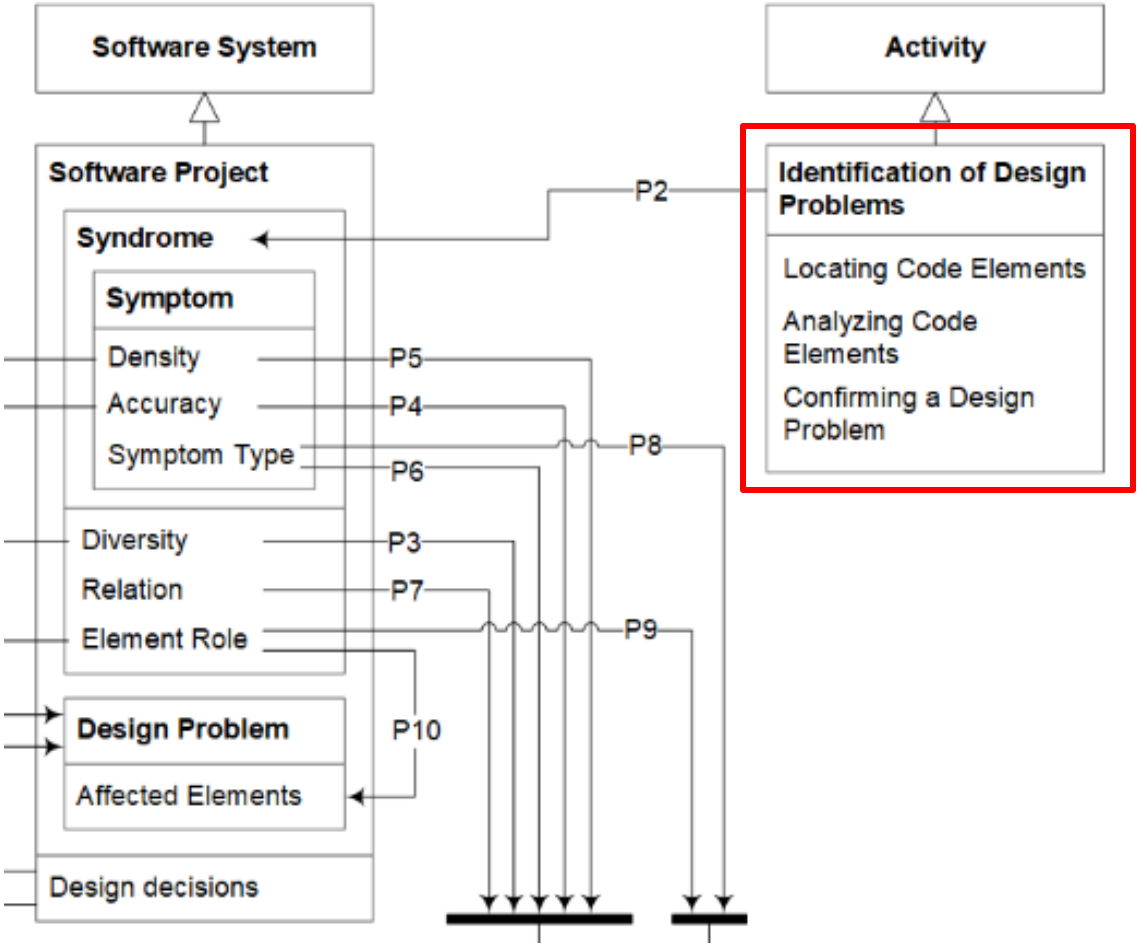
Identification of Design Problems



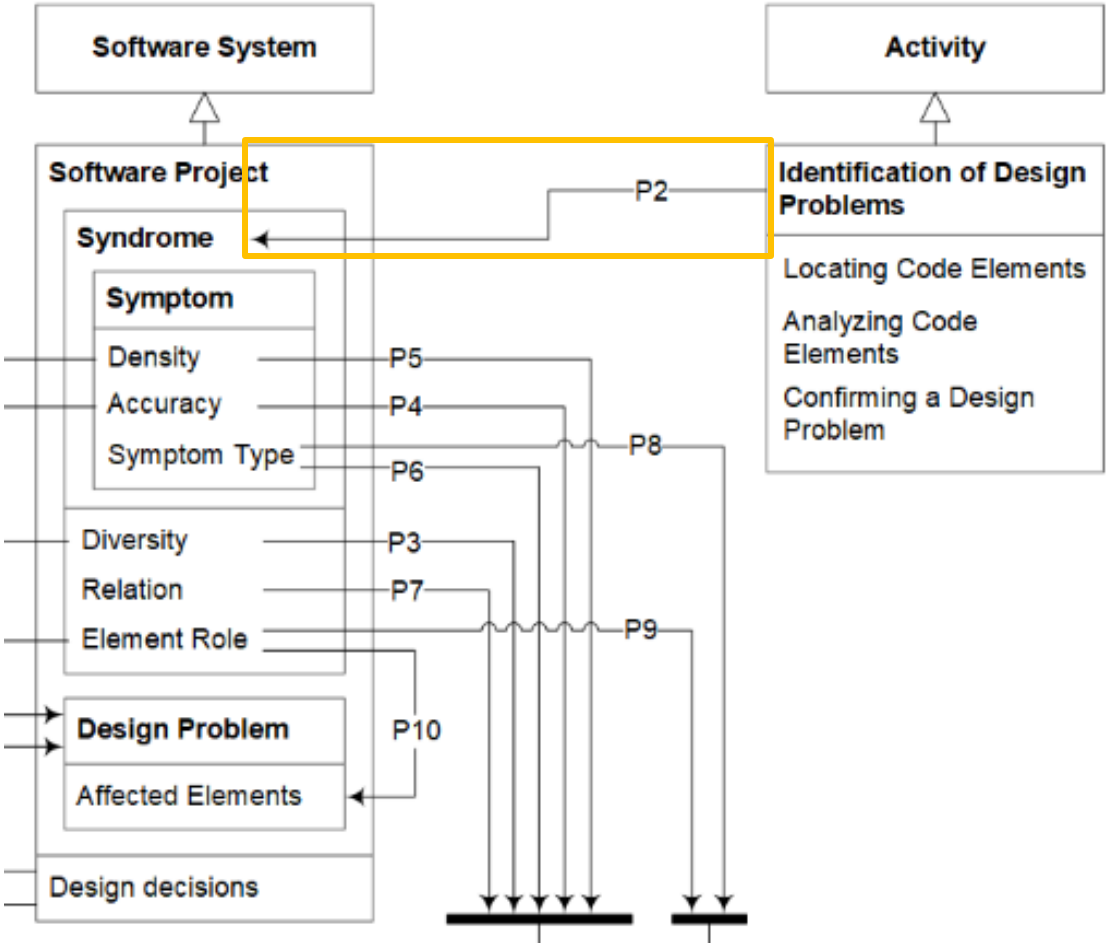
Identification of Design Problems



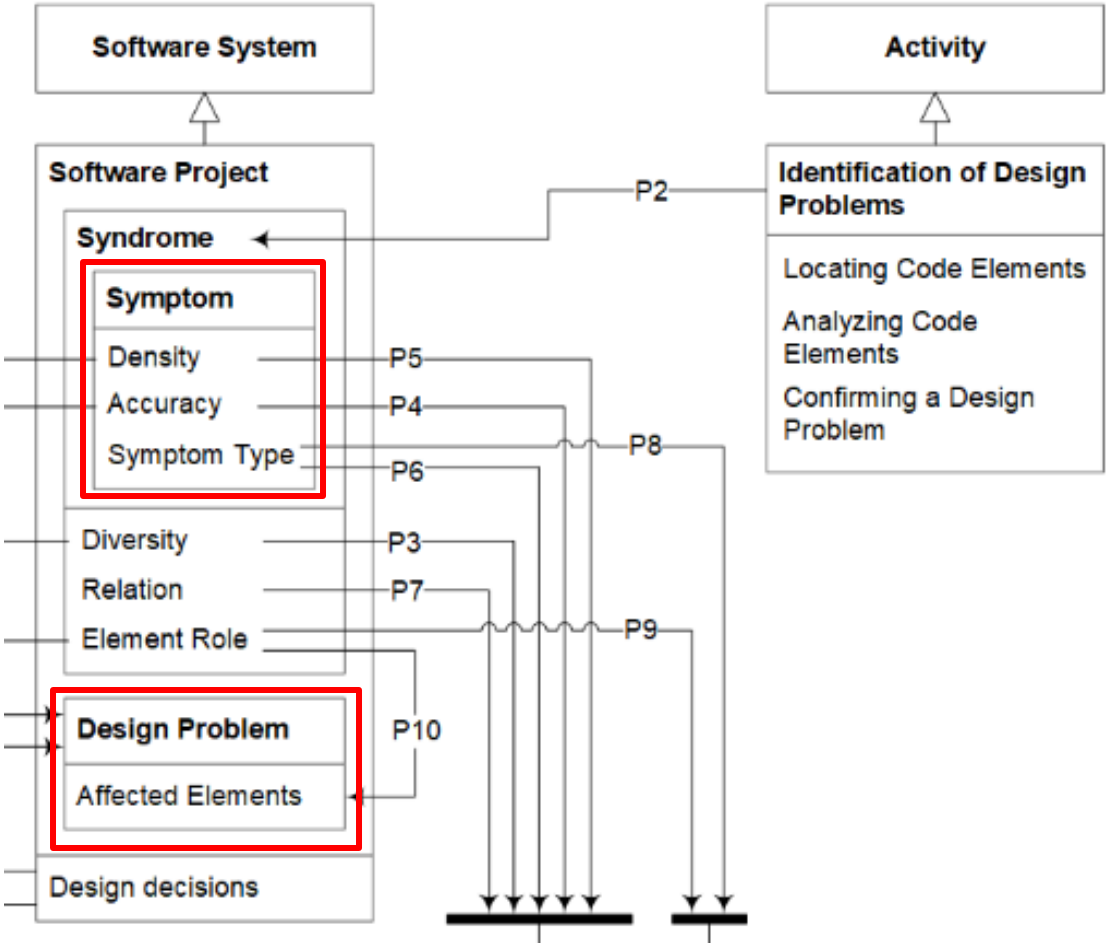
Identification of Design Problems



Identification of Design Problems



Identification of Design Problems



Design Problem Symptoms

- Developers do not always consider all the symptoms of a syndrome when identifying design problems

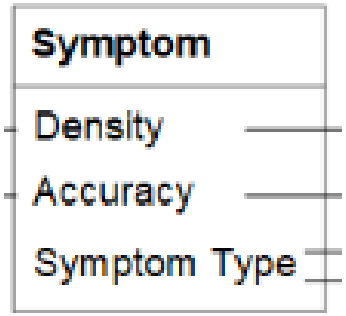
Symptom	Applied times	No. of contributions	Percentage of success
Design Pattern Violation	43	34	79.07%
Quality Requirements	43	31	72.09%
Violation of Non-functional Requirements	62	46	74.19%
Code Smells	37	17	45.95%
Violation of Object-oriented Principles	38	20	52.63%

Most Helpful

Less Helpful

Design Problem Symptoms

- Symptom attributes that drive developers to select what to analyze



C6	Symptom Type	A category to which a set of symptoms with common characteristics belongs
C7	Accuracy	The degree to which a symptom is correct in indicating a design problem
C8	Density	The number of symptoms instances in a syndrome

Design Problem Symptoms

- Diversity of a syndrome
  - Developers use the relation among the symptoms to discover other types of symptoms that can indicate a design problem

C9	Relation	How two or more symptoms are connected

  - Developers frequently located elements that manifested several different types of symptoms

P3	The diversity of symptoms influences which symptoms the developer will use during the diagnosis

Design Problem Diagnosis

- Diagnosis by two types of analyses

C11	Symptom Analysis	The process of analyzing a set of symptoms affecting a single element
C12	Epidemic Analysis	The process of analyzing elements affected by the same set of symptoms

- Incorrectly ignoring symptoms
  - Developers tend to not consider the same type of symptom in the element previously analyzed and with design problem ignored

Design Problem Symptoms

- Combining multiple related symptoms
  - Developers combine multiple symptoms in a single element to confirme design problem

Symptoms	Instances	Design Problems	Teams
1	16	11	T7, T8, T9
2	13	10	T1, T3, T7, T8, T9
3	14	11	T3, T4, T5, T6, T7, T8, T9
4	10	6	T2, T3, T5, T8, T9
5	3	1	T4

P7	The relation among the symptoms influences which symptoms the developer will use during the diagnosis

Grounded Theory

- Propositions concerning the developer
  - Confidence in the Presence of a Design Problem
  - Conscientiousness
  - Incapability of Providing an Alternative

Propositions concerning the developer

- Confidence in the Presence of a Design Problem

C14	Confidence	The degree to which they are convinced about the presence of a design problem

P11	The diagnosis affects the developer's confidence in the presence of a design problem
P12	The symptom accuracy affects the developer's confidence in the presence of a design problem
P13	The density of symptoms affects the developer's confidence regarding the presence of a design problem
P14	The role that the element plays on the system influences the developer's confidence

Propositions concerning the developer

- Conscientiousness

C15	Conscientiousness	A personality trait related to being careful, responsible, and persevering

P15	The consciousness affects the likelihood of a developer identify a design problem
P16	The diversity of symptoms affects the conscientiouness of the developers

Propositions concerning the developer

- Incapability of Providing an Alternative
  - Developers use the concept of design decision to justify why they do not consider the presence of a design problem (P17)

P17	The design decisions affects the confirmation of the presence of a design problem
P18	The design decisions affects the confidence developer's confidence in the presence of a design problem

Grounded Theory

- Towards Improving Design Problem Diagnosis
  - The theory presented can drive to solutions for supporting developers during design problem identification
  - For instance, some of these solutions
    - Supporting Multiple Symptoms
    - Prioritization of Similar Elements
    - Additional Support for the Developer

Grounded Theory

- Supporting Multiple Symptoms
  - Providing multiple design problem symptoms
  - Filtering relevant symptoms
  - Visualization support
- Prioritization of Similar Elements
  - Prioritizing epidemic elements
- Additional Support for the Developer
  - Providing an alternative implementation
  - Personalizing the detection of symptoms

lab-soft
software engineering laboratory

Related Work

Related Work

- None studies presents the diagnosis of design problem as a theory
- Nine studies that only focus on presenting the phenomenon rather than explaining it
- Sousa et al. Has proposed to explain how developers identify design problems, but not as a theory

lab-soft
software engineering laboratory

Threats to Validity

Threats to Validity

- Construct Validity
  - Symptoms were provided for developers
    - To avoid bias, symptoms were considered from literature
    - Also, managers told that developers were familiarized with them
  - The time allocated for the tasks
    - Pilot Study to adjust time

Threats to Validity

- Internal Validity
  - Difference between the developers' background knowledge
    - Diversity as an opportunity to strengthen the theory propositions
    - Also, training was provided

Threats to Validity

- External Validity
  - Number of subjects
  - All working on Brazil
    - Was Multi-company
    - Five different working environments
    - Eight different systems.
  - Covered only systems developed in Java

Threats to Validity

- Conclusion Validity
  - Participation of the author who followed the GT procedures
    - To mitigate it, the GT coding activities were shared with other researchers
    - The identification of the constructs and the depicting of propositions were performed separately by the first author and other researchers
    - Three authors conducted GT procedures independently

lab-soft
software engineering laboratory

Conclusion

Conclusion

- Conducted a multi-trial industrial experiment with developers from different companies, where they had to identify design problems in their systems

- Derived a theory describing the activities and factors that influence on how developers identify design problems

Conclusion

- The theory can serve to further understand, for example:
  - The theory reveals that developers rely on a heterogeneous set of symptoms, and they tend to combine symptoms
  - Also presents the characteristics of symptoms that developers consider helpful
- We discussed how the knowledge revealed by our theory can be used to advance the state-of-art

lab-soft
software engineering laboratory

Future Work

Future Work

- Execution of new empirical studies to assess in more depth the theory's propositions and explanations

- The goal of these studies is to use the theory to implement a novel family of solutions that are more effective than the current ones in helping developers identify design problems

lab-soft
software engineering laboratory

Identifying Design Problems in the Source Code

A Grounded Theory

Leonardo Sousa[1], Anderson Oliveira[1], Marcos Kalinowski[1], Rafael de Mello[1], Willian Oizumi[1], Simone Barbosa[1], Alessandro Garcia[1], Roberto Oliveira[1], Carlos Lucena[1], Rodrigo Paes[1], Baldoino Fonseca[2], Jaejoon Lee[3]

[1] PUC-Rio, [2] UFAL, [3] Lancaster University

ICSE '18

Software Engineering Lab (LabSoft)
http://labsoft.dcc.ufmg.br/