SWA – Architectural Design Methods: Part II

Lecture 04

BIL428 Software Architectures

Asst.Prof.Dr. Mustafa Sert
msert@baskent.edu.tr

Department of Computer Engineering, Başkent University
Ankara 06810 TURKEY

Contents

- Classification of Architecture Design Methods
- Architectural Requirement Analysis
- Domain Analysis

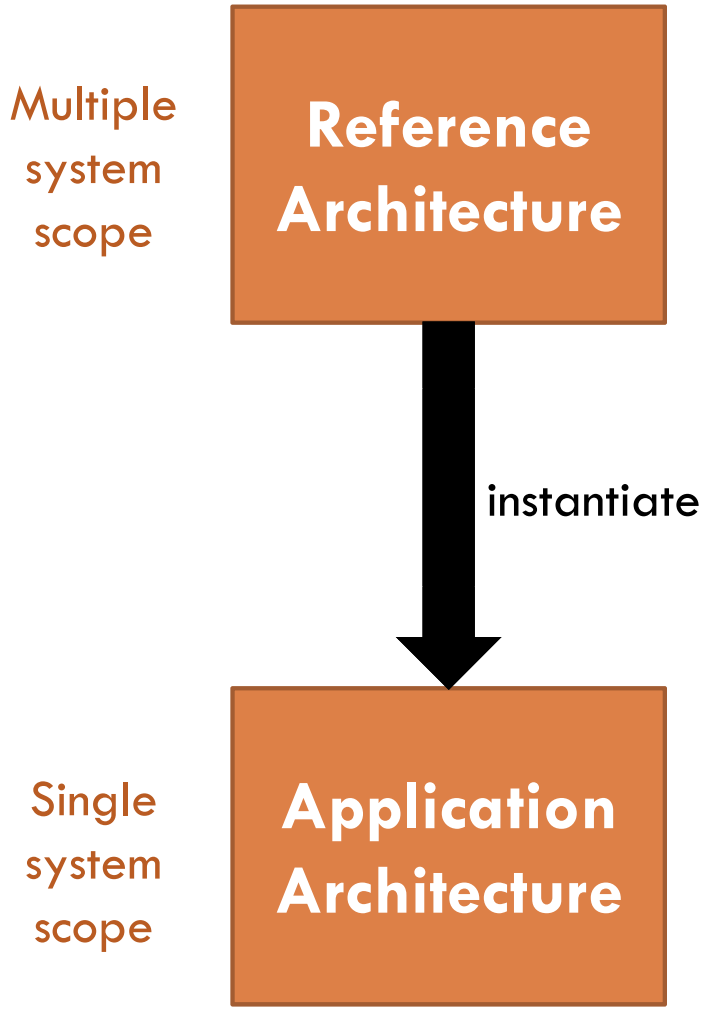
Classification of SWA Methods

- **Single-system scope**
  - Focuses on developing architecture for a single system/product, which is also the scope of this course
- **Multiple-system scope**
  - SW product-line architectures – Focuses on developing architecture for a set of relational SW products

Reference Architecture vs. Application Architecture

- The reference architecture is a core architecture that captures the high-level design for the applications of the software product line

- The application architecture is a specialization of the reference architecture for a concrete product



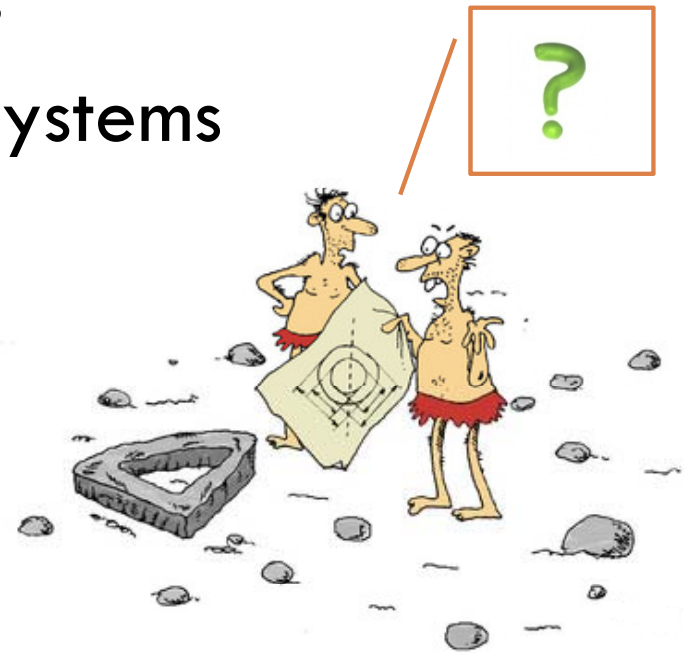
Classification of SWA Methods - Source

Which one should I select? And Why?

- **Artifact driven**
  - Derives abstractions from requirements using artifacts in the method
- **Use-case driven**
  - Derives abstractions from uses cases
- **Pattern-driven**
  - Derives abstractions from patterns
- **Domain Driven**
  - Derive abstractions from solution domains a domain analysis process
- The **focus of this course** is to use domain driven methods, which also applies the patterns..

Artifact Driven

- Start from textual requirements
- Look at artifact types in the method and try to identify artifacts from requirements specification
- Group the relational artifacts in **subsystems,** these are the architectural components
- Define the relations btw the subsystems



Artifact Driven – Example PC Factory

Requirements Specification: PC FACTORY
Date: 16-October 2003

A software system for a computer company, which consists of two departments, a factory and sales and marketing department. The factory assembles desktop PCs and tower PCs. All the components of a PC are delivered by different external suppliers. A PC consists of one monitor, a cabinet, and a keyboard. The cabinet includes a chassis. A chassis on its turn is composed of a bus, floppy disk drive, an optional CD-ROM drive, a memory unit, CPU, and power supply. A bus may incorporate a network card. A memory unit includes many RAM chips. The sales and marketing department administers properties of each PC, like the type, weight, make, price, amortization, power consumption etc. A client uses a purchase order to order PCs or set of computer components from the company.

Not precise,
Ambiguous,
Redundant,
Difficult to understand,
Not complete…

Artifact Driven – Example PC Factory

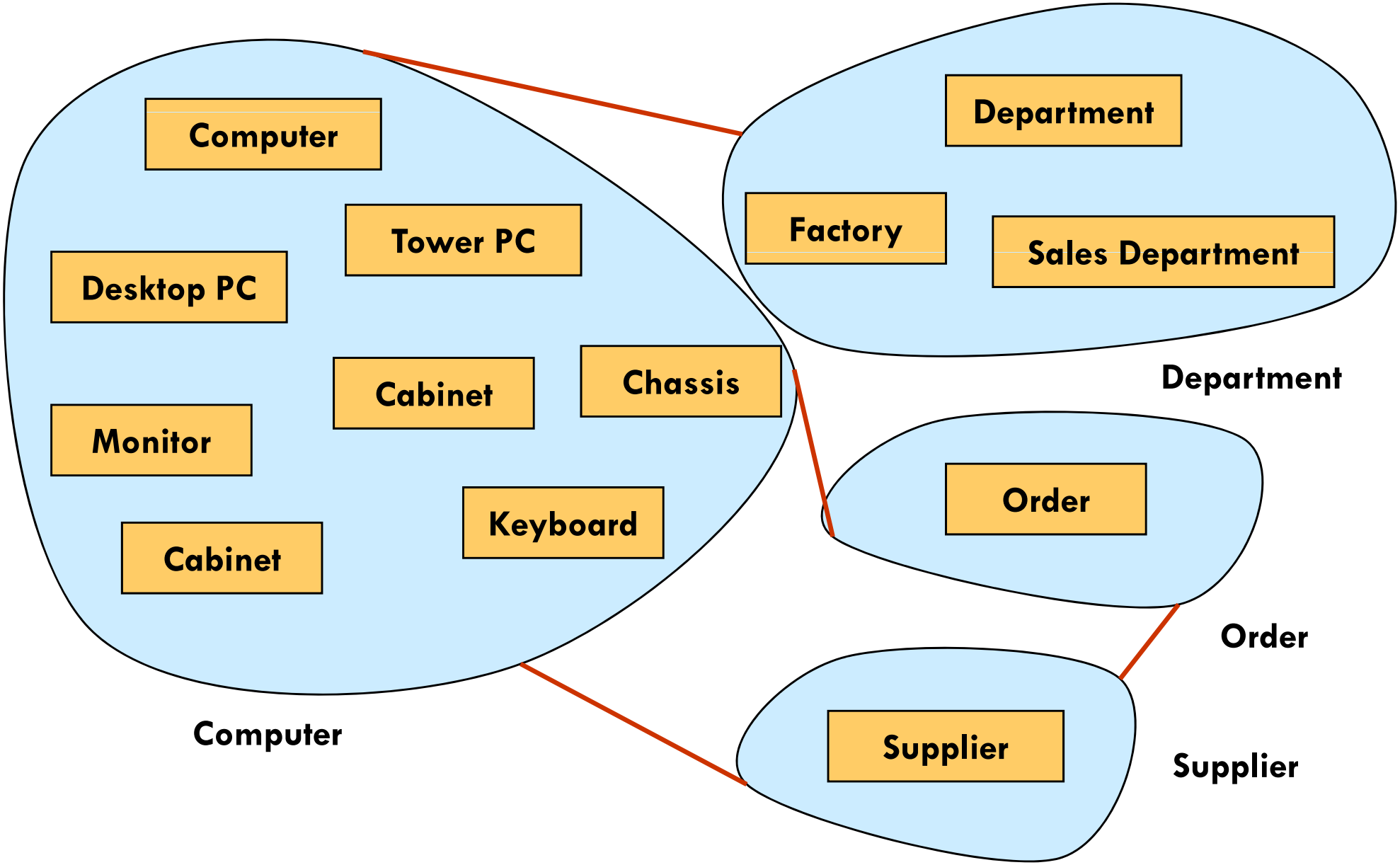
Requirement Specification

Requirements Specification
PC FACTORY
Date: 16-October 2002

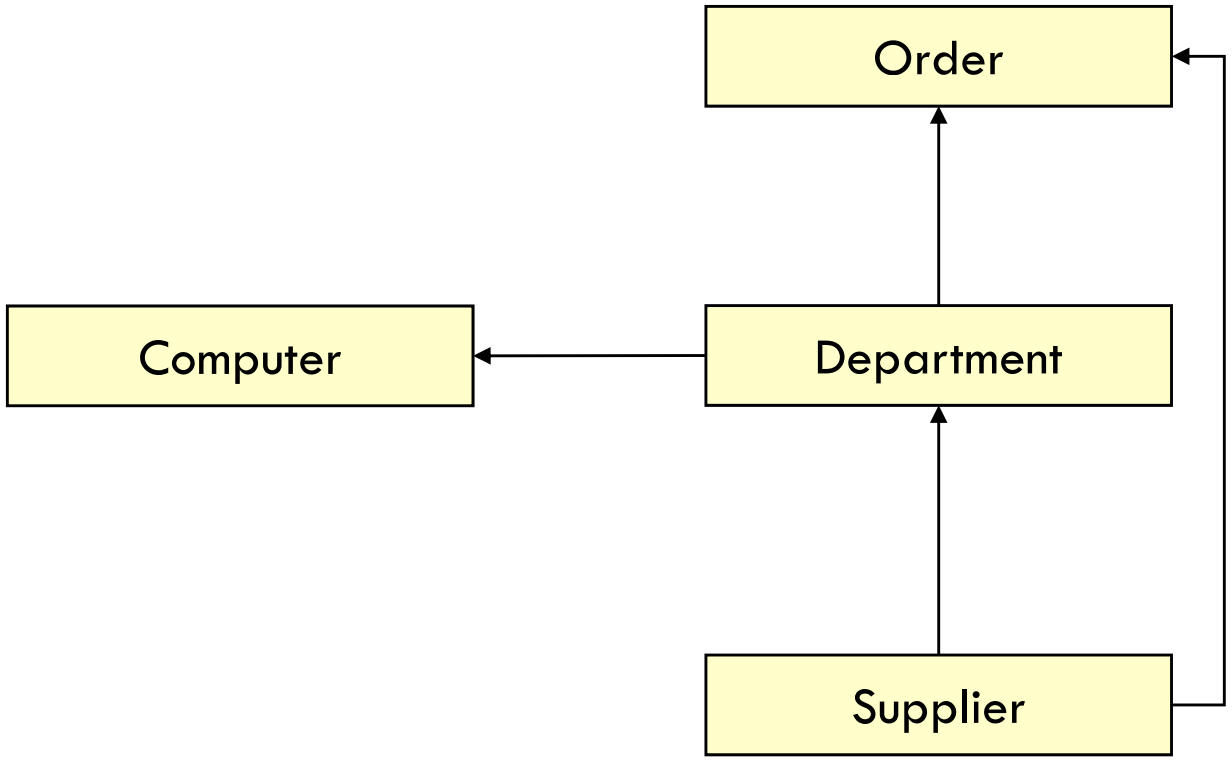
A software system for a computer company, which consists of two departments, a factory and sales and marketing department.
The factory assembles desktop PCs and tower PCs. All the components of a PC are delivered by different external suppliers. A PC consists of one monitor, a cabinet, and a keyboard. The cabinet includes a chassis. A chassis on its turn is composed of a bus, floppy disk drive, an optional CD-ROM drive, a memory unit, CPU, and power supply. A bus may incorporate a network card. A memory unit includes many RAM chips.
The sales and marketing department administers properties of each PC, like the type, weight, make, price, amortization, power consumption etc. A client uses a purchase order to order PCs or set of computer components from the company.

Ok, this is a class, this isn't.
This might also be a class,
And this one also…

Identified Classes/SubSystems



Identified Architecture

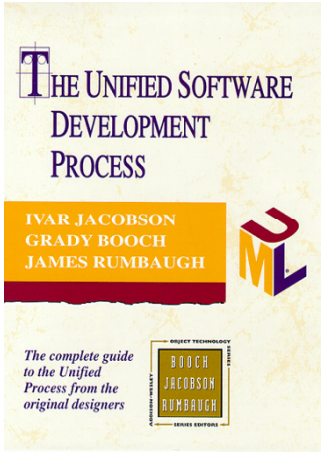


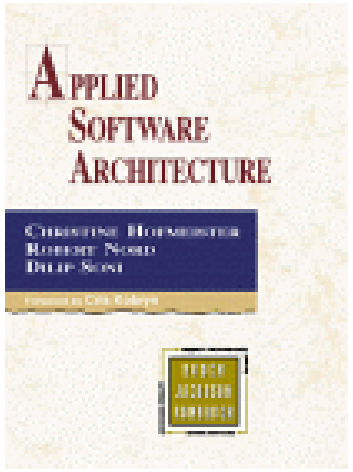
Obstacles of Artifact-driven Approach

- Textual requirements are imprecise and are less useful as a source for deriving architectural abstractions
- Subsystems have poor semantics to serve as architectural components
- Composition of subsystems is not well-supported.

Use case driven

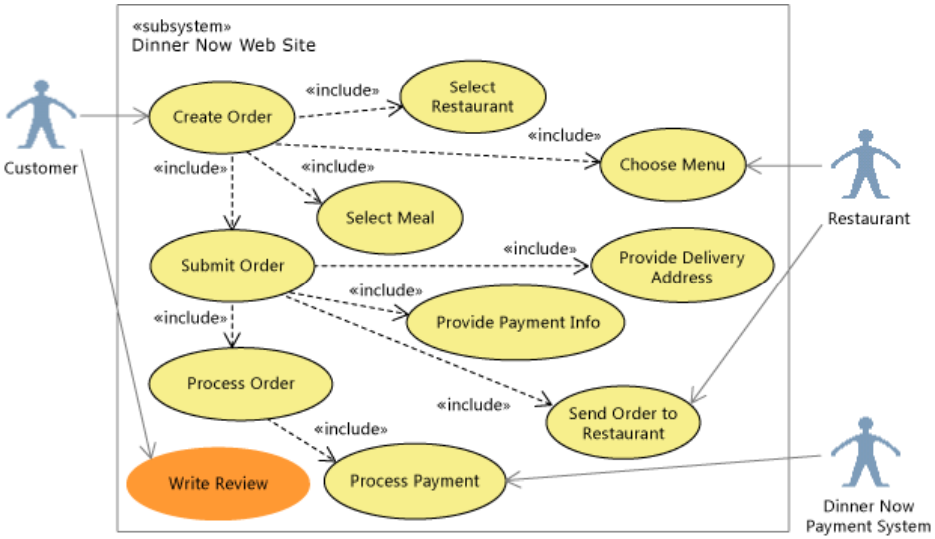
- Extract use cases
- Identify fundamental classes from use cases.
- Group these classes in *packages*, these are the architectural components.
- Define the relations between packages.





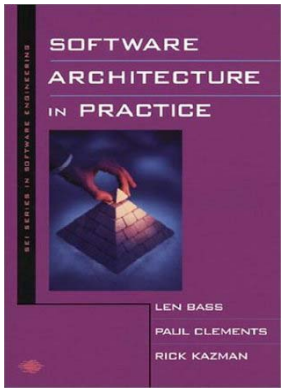
Obstacles

- Selecting architecturally relevant use cases is not systematically supported.
- Use cases do not provide a solid basis for architectural abstractions
- Package construct has poor semantics to serve as architectural abstractions.



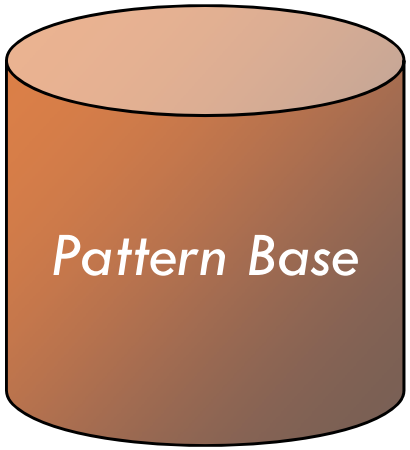
Pattern driven

- Start with requirement specification
- Select appropriate patterns from a pattern base.
- Compose these patterns.



Pattern

- Pattern is a generic and reusable design solution for recurring problems in a given context.
- Each pattern describes a solution, problem and the context.
- Patterns can be used to construct software architectures.
- Examples:
  - Layers, Blackboard, Pipes and Filters, etc.



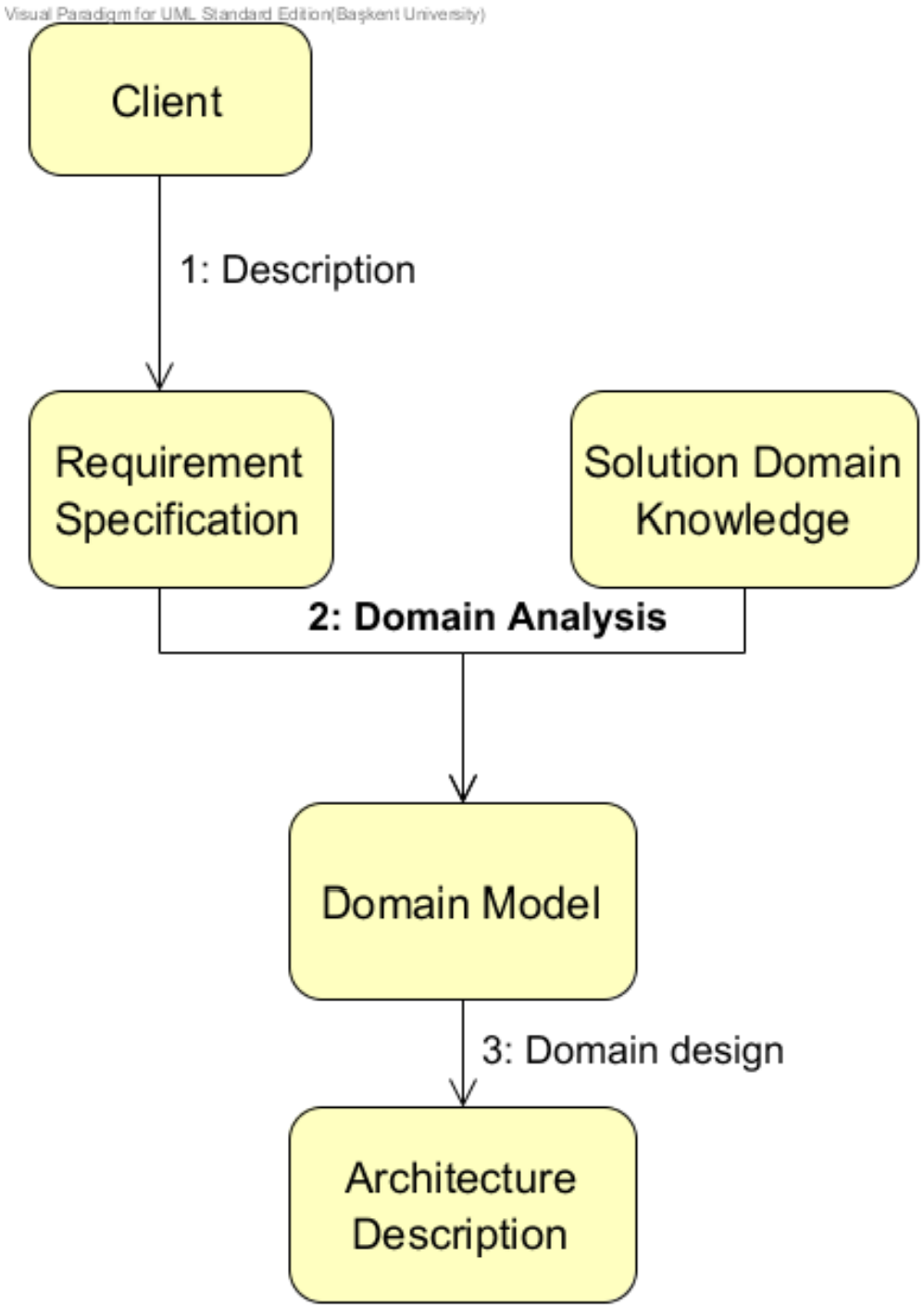
Obstacles of Pattern-Driven Approaches

- Patterns only might not be sufficient for deriving architectural abstractions.
- Selection of patterns is not well supported and depends on experience of software engineer.
- Applying patterns is not straightforward and requires thorough analysis of the problem.
- Composing patterns is not well supported.

Domain driven

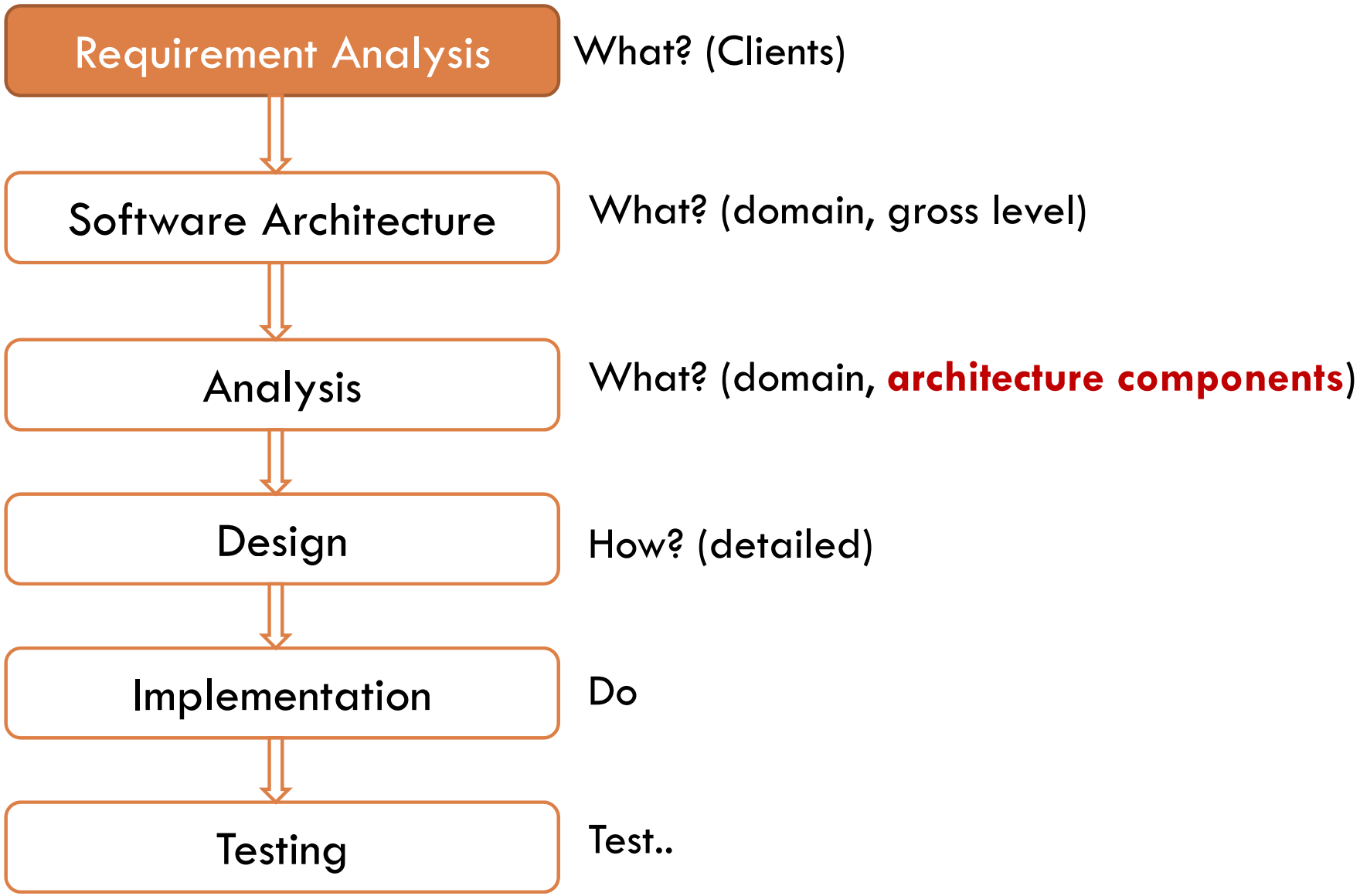
- Define requirements
- Apply domain analysis to provide domain model
- **Extract architectural abstractions from domain model** concepts
- Define architectural relations based on domain model

Conceptual Model for Domain driven approaches



Architectural Requirement Analysis

Requirement Analysis in Life Cycle



Definition

- Requirements are the statements that identify the essential needs of a system in order for it to satisfy customer needs
- A customer's need might be to solve a problem, achieve and objective or satisfy contract, standard or specification
- What is missing/incomplete in this definition ?
- the phrases **customer needs** must be replaced with **stakeholder's need** in both bullets, above..

How to define stakeholders?

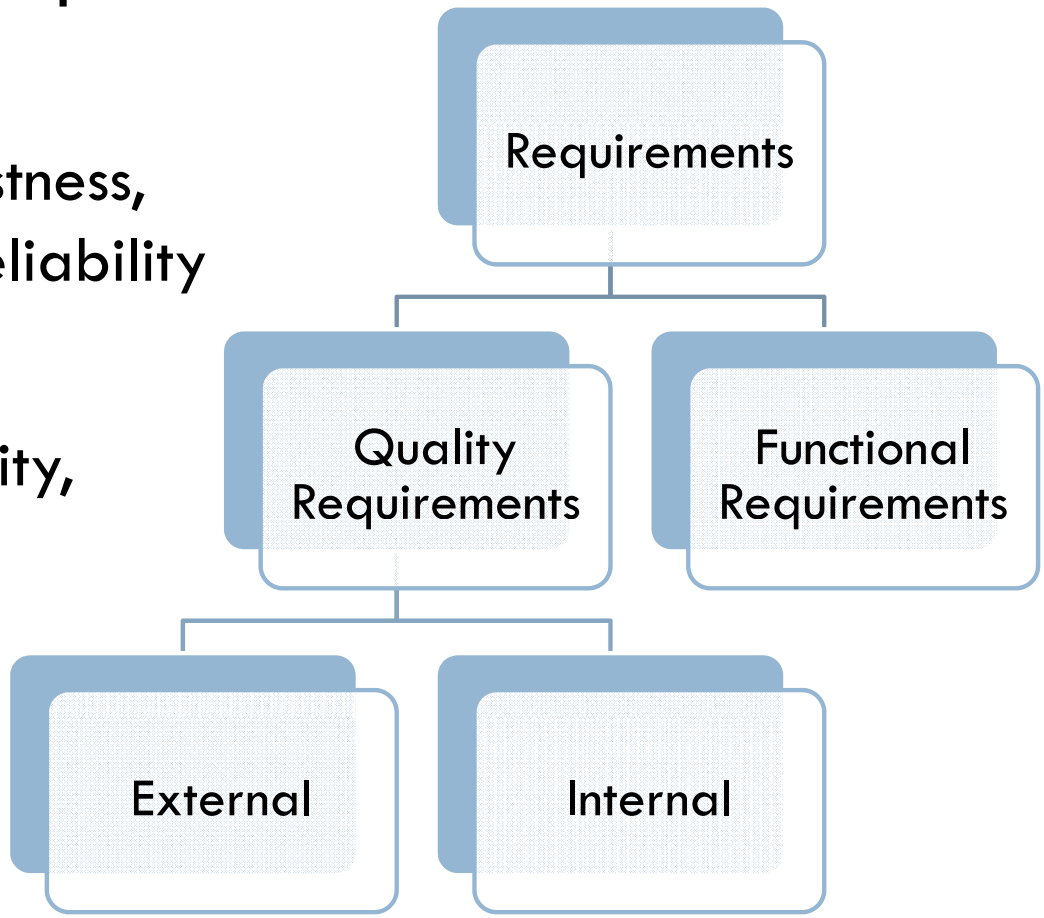
- IEEE 1471 recommended standard on architectural description states the following minimal set of stakeholders:
  - Users of the system
  - Customers of the system
  - Developers of the system
  - Maintainers of the system

Example: ATM Stakeholders

- Bank customers
- Representatives of other banks
- Bank managers
- Counter staff
- Database administrators
- Security managers
- Marketing department
- Banking regulators
- HW and SW maintenance engineers
- …

Types of Requirements

- Functional vs. Quality Requirements
  - External
    - Understandability, Robustness, Correctness, Usability, Reliability
  - Internal
    - Reusability, Maintainability, Portability, Testability, Adaptability



Domain Requirements

- Come from the application domain of the system and that reflect characteristics of that system
- Maybe new (non-) functional requirements, constraints on existing requirements
- Extracted through a **domain analysis** process

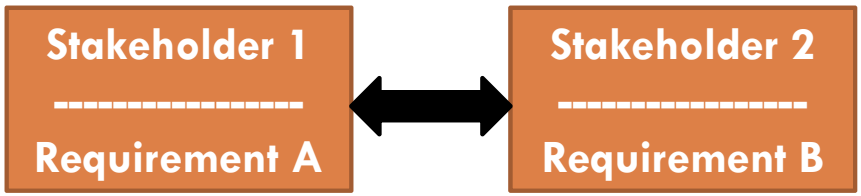
Prioritization of Requirements

- Different prioritization mechanism can be applied:
  - **Essential**
    - The application must support that requirement → the architectural drivers
  - **Conditional**
    - This requirement will need to be supported at some stage, but NOT necessarily in the first release
  - **Optional**
    - Desired, but they are NOT the drivers of the design..

Conflicting Requirements

Three dependencies btw architectural requirements

- **No Dependency**
  - The inclusion of one requirement is not directly dependent on the other
- **Required**
  - The inclusion of one requirement requires the inclusion of another requirement
- **Conflicting**
  - Requirement A conflicts with requirement B
    - Re-use vs. time-to-market
    - Adaptability vs. time-performance



Conflict Resolution

- Rule 1: there is NO simple solution to conflicting requirement ☺
- Discussion with the relevant stakeholders is necessary
- To come up with possible solution scenarios
- Consider appropriate trade-offs.

Requirement Specification Techniques

- Writing the textual requirement specification
- Developing use-case diagrams
- Defining scenarios
- Constructing a (UI) prototype
- Defining state diagrams
- …

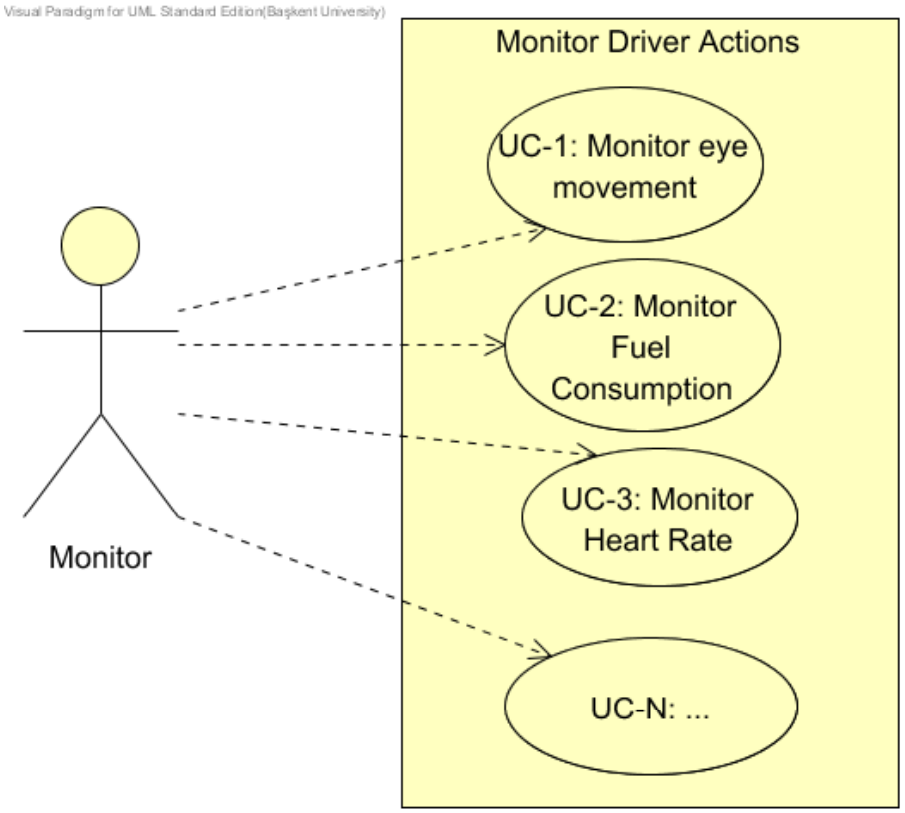
Textual Requirements

Requirements Specification
Driver Monitor System
Author: …

DESCRIPTION
The system will be an on-board automotive diagnostics system that will diagnose the driver and the car performance and provide the driver with appropriate feedback. The system will monitor both the human and the car. The human can be diagnosed based on the physiological behavior such as heart rate, blood pressure, eye movements, etc. The car can be monitored against its performance regarding fuel consumption, acceleration, oil state of the breaks, etc. The feedback will be given on a display on the dashboard. These will be based on RED/GREEN zone displays or multifunctional displays..

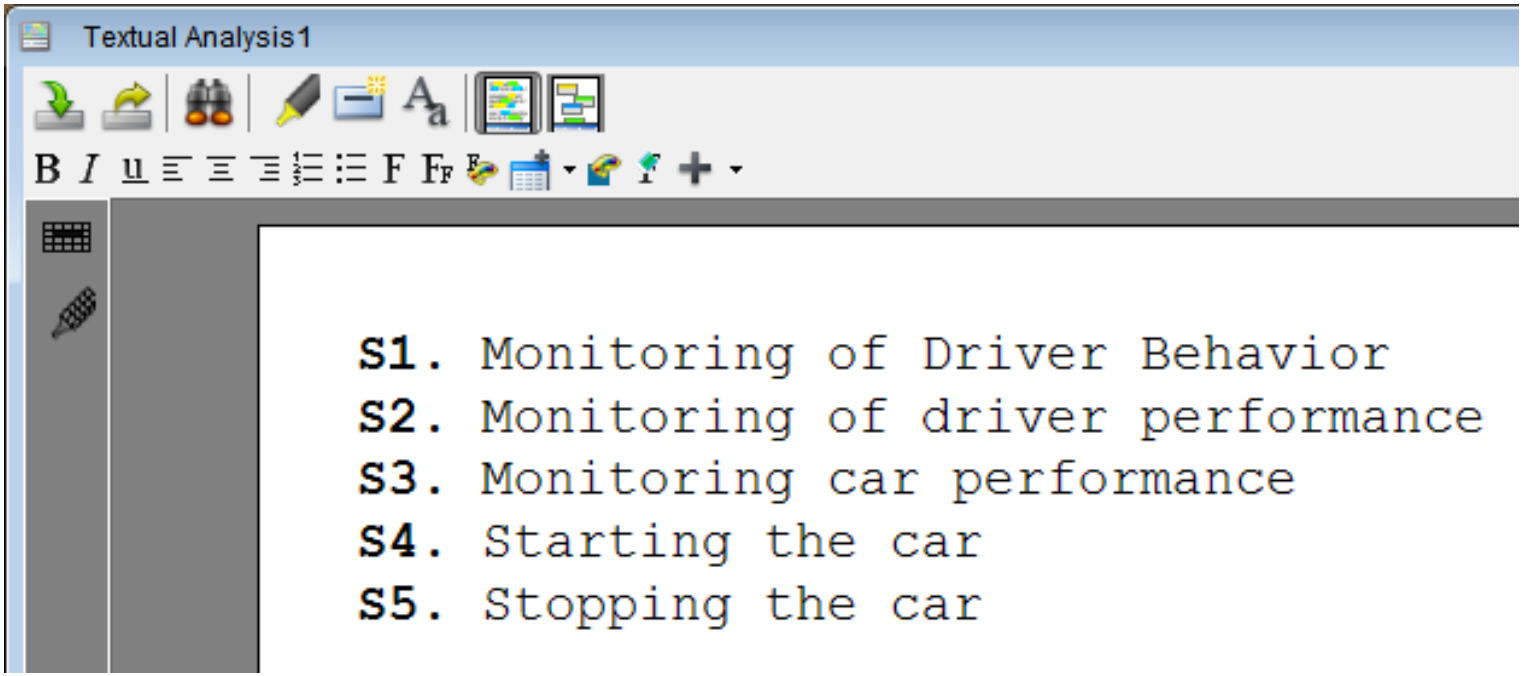
Use-case Diagram

- Shows the system's actors, use cases along with their relationships.
- Models the context of the system
  - Actors outside the system
    - Primary actors are on the left
    - Secondary (supporting) actors on the right
- Models the functional requirements of the systems
  - As use-cases



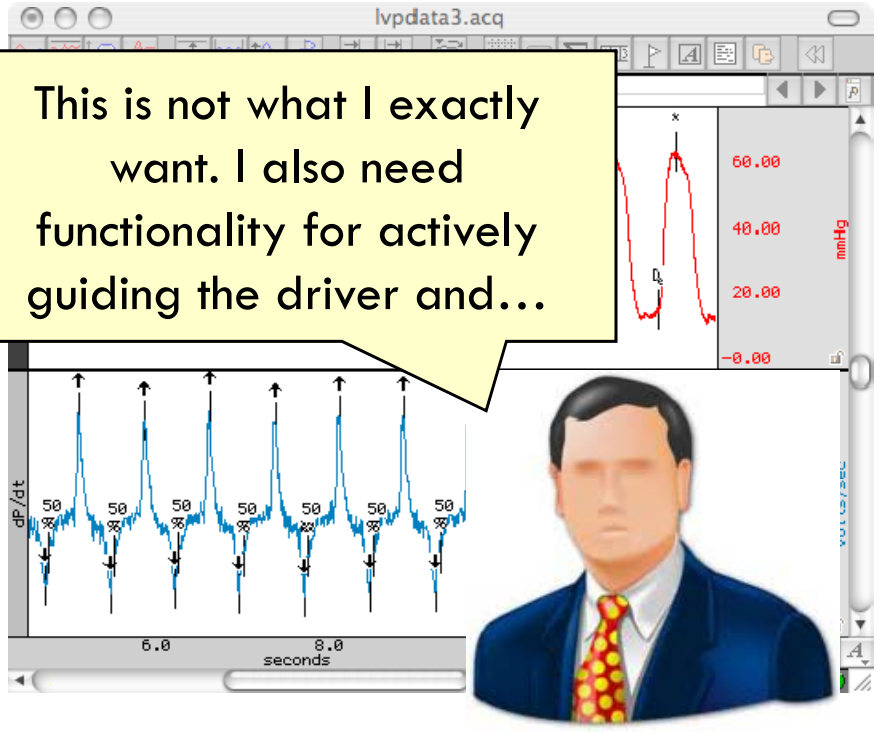
Scenario

- Brief description of desired and/or anticipated use of system
- Represented as textual sentences
- Widely used during requirements elicitation



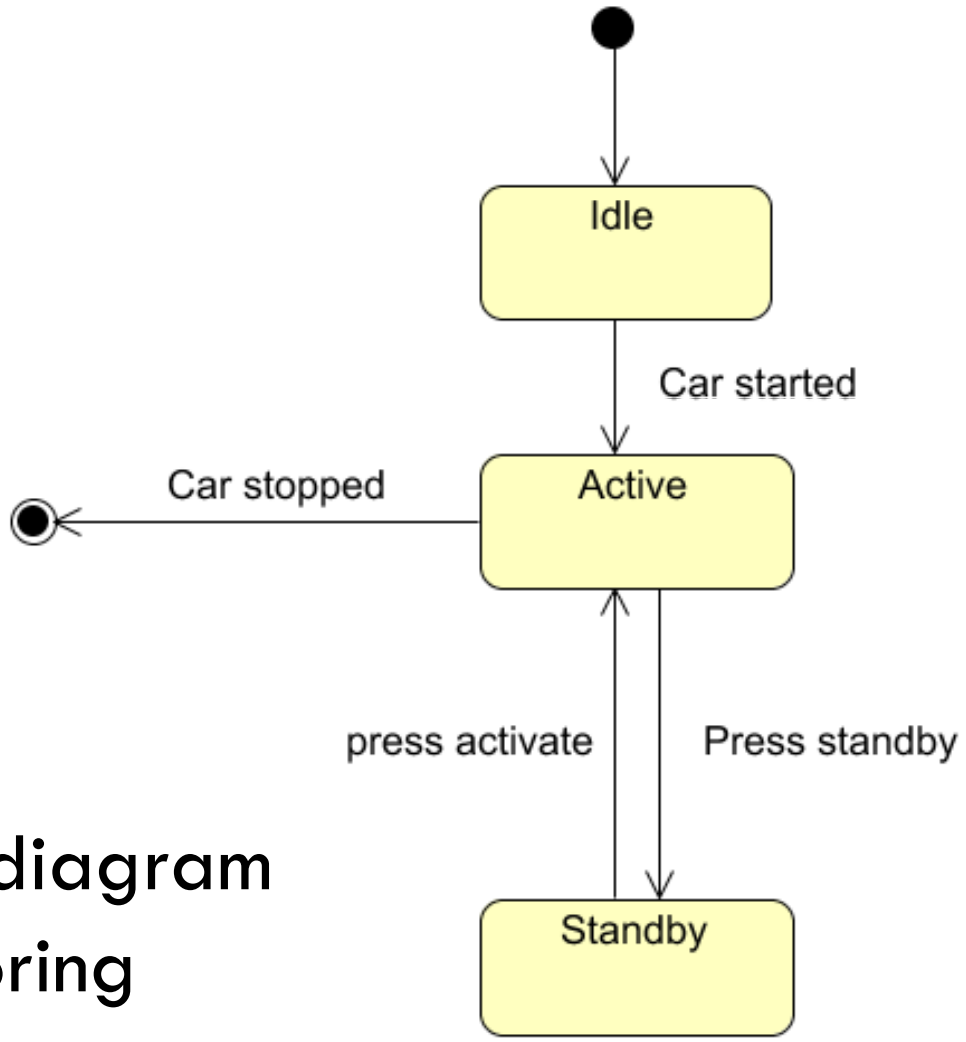
Prototypes

- GOAL
  - Requirement specification & verification
- **Throw-away** vs. **evolutionary prototype**
- Often user interface (UI) prototype



State Diagrams

- Defines the dynamic behavior of the system
- Crucial for Safety Critical Systems



State transition diagram for monitoring

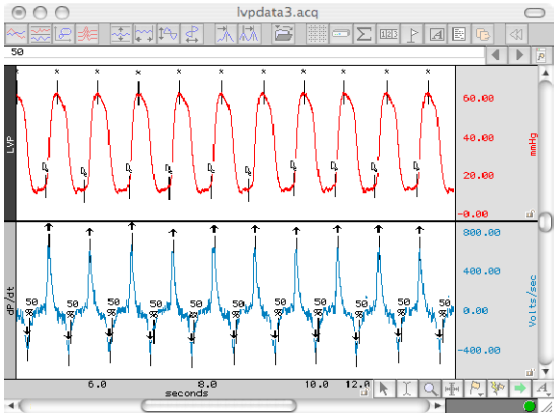
Domain Analysis

Requirements to Solution..

Requirements Specification
Driver Monitor System
Author: …

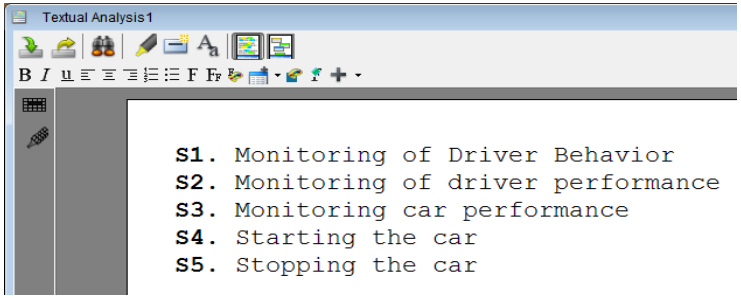
DESCRIPTION
The system will be an on-board automotive diagnostics system that will diagnose the driver and the car performance and provide the driver with appropriate feedback. The system will monitor both the human and the car. The human can be diagnosed based on the physiological behavior such as heart rate, blood pressure, eye movements, etc. The car can be monitored against its performance regarding fuel consumption, acceleration, oil state of the breaks, etc. The feedback will be given on a display on the dashboard. These will be based on RED/GREEN zone displays or multifunctional displays..

Textual Requirements

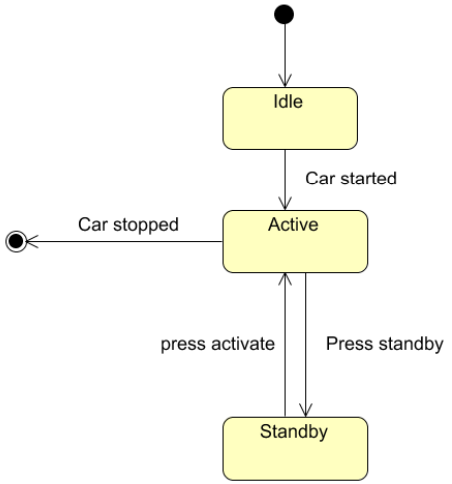


Prototypes

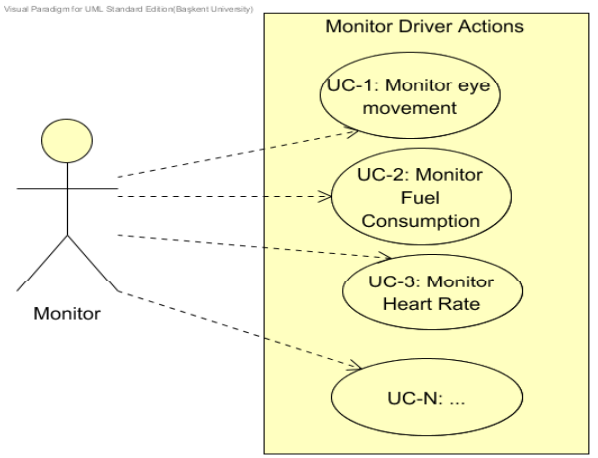
I know the requirements… BUT, how can I derive my solution?



Scenarios



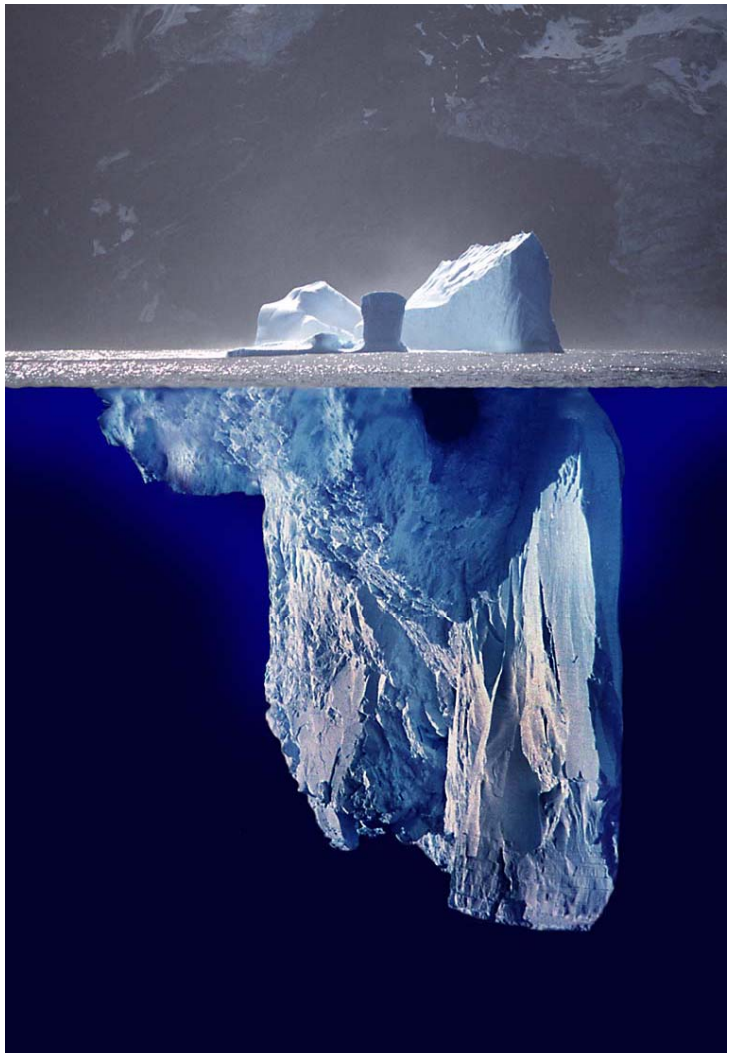
State Diagrams



Use-Case Model

Requirements are Incomplete

- How to identify architectural abstractions from the requirements?
- Are these are right abstractions?
- How do we know?
- Requirements might NOT be optimally specified
  - Over/under specified, wrong..



Domain Understanding

- Domain
  - An area of knowledge or activity..
- We should understand the domain for which we will build the architecture..

Example Domains

- Bank Systems
- Insurance Systems
- Health Care Systems
- Transaction Systems
- Image processing
- Information retrieval
- Multimedia Systems
- …

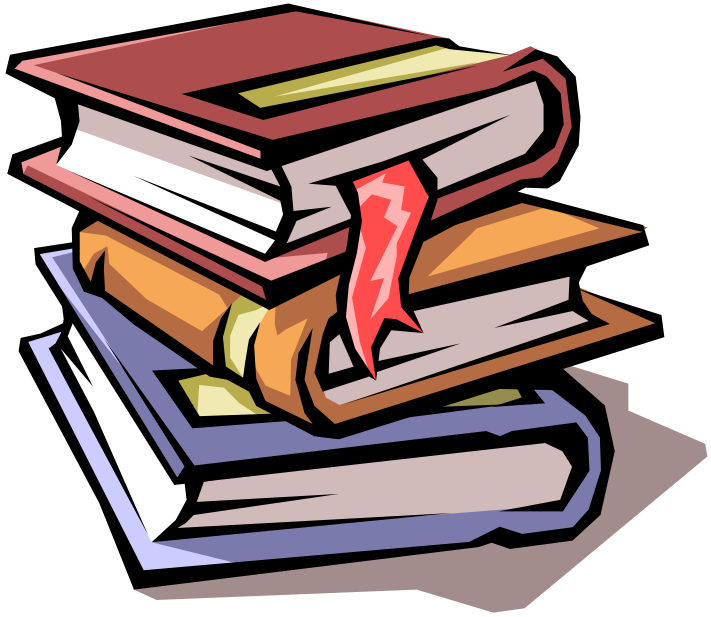
Identify the essence of domain, the fundamental and stable concepts..

Domains in Different Perspectives

- Problem domain (client's perspective)
- Business domain scenarios (business perspective)
- Solution domain (solution perspective)
- General knowledge (intuition and background)
- System/product knowledge (architecture)

Domain Analysis

- Is the systematic activity of
  - Collecting,
  - Organizing, and
  - Storing domain knowledge.



Domain Analysis Process

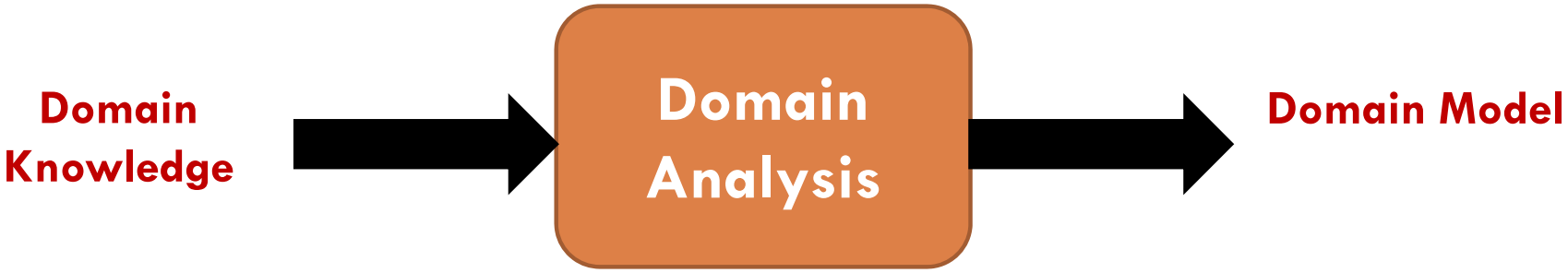
- **Domain Characterization**
  - Define boundaries/context of the domain
  - Identify knowledge sources
- **Data Collection**
  - Knowledge elicitation
- **Data Analysis**
  - Analyze commonality and variability
- **Data Classification**
  - Vocabulary construction
  - Abstraction, classification, generalization
- **Data Evaluation**
  - Evaluate domain model

Scoping

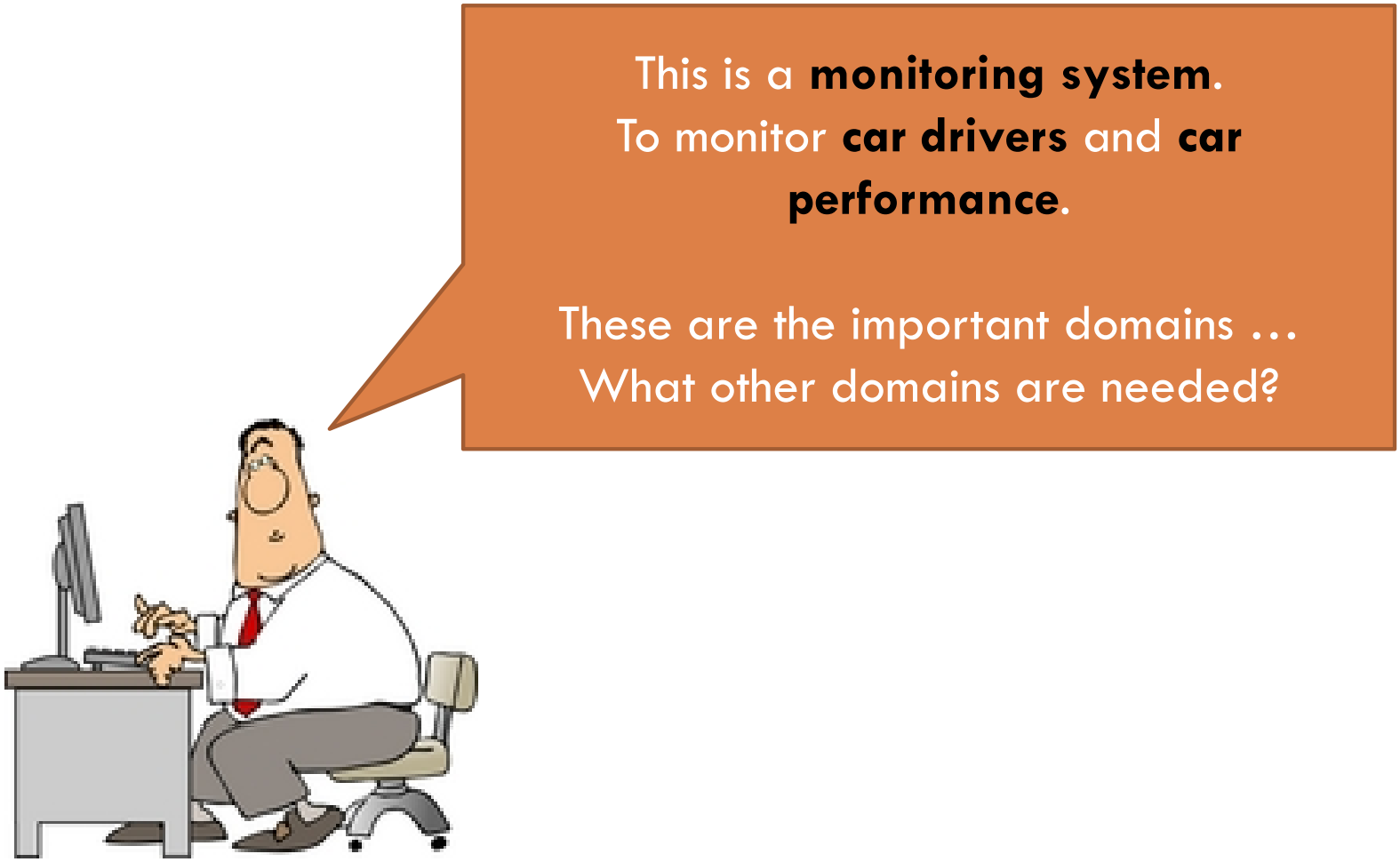
Modeling

Domain Analysis

- **Domain Scoping**
  - Define the domain of interest wrt stakeholders and business context
- **Domain Modeling**
  - Providing a domain model by data collection, data analysis, classification, and evaluation



Domain Scoping – Identify Domains



Identifying Domains

- How to find knowledge domains?
  - Categorize domains in **Taxonomies**
    - Structure/organize the body of knowledge that constitutes the field
    - Provide unifying constructs
    - May predict future development areas

EXAMPLE:

The 1998 ACM Computing Classification System (partial)

ACM Computing Classification System = Subject headings and shelfmarks Computer science			
Code heading	**Code sub-heading**	**Description heading/subheading**	**Aisle / book case**
A.		**General Literature**	
	A.0	General (Biographies/Autobiographies/General Literary Works)	43.17 - 43.18
	A.1	Introductory and Survey	43.18 - 43.19
	A.2	Reference (Dictionaries/Encyclopedias/Glossaries)	43.20
	A.3	Lecture notes in computer science	45.01 - 43.16
B.		**Hardware**	
	B.0	General	43.20
	B.1	Control Structures and Microprogramming	43.20
	B.2	Arithmetic and Logic Structures	43.20
	B.3	Memory Structures	43.20
	B.4	Input/Output and Data Communications	43.20
	B.5	Register-Transfer-Level Implementation	43.20
	B.6	Logic Design	43.20
	B.7	Integrated Circuits	43.20
	B.m	Miscellaneous	43.20
C.		**Computer Systems Organization**	
	C.0	General	43.00
	C.1	Processor Architectures	43.00
	C.2	Computer-Communication Networks	43.00
	C.3	Special-Purpose and Application-Based Systems	43.00
	C.4	Performance of Systems	43.00
	C.5	Computer System Implementation	43.00
	C.m	Miscellaneous	43.00
D.		**Software**	
	D.0	General	43.00 – 43.01
	D.1	Programming Techniques	43.01
	D.2	Software Engineering	43.01 – 43.02
	D.3	Programming Languages	43.02 – 43.03
	D.4	Operating Systems	43.03 – 43.04
	D.m	Miscellaneous	43.04
E.		**Data**	
	E.0	General	43.04
	E.1	Data Structures	43.04
	E.2	Data Storage Representations	43.04
	E.3	Data Encryption	43.04
	E.4	Coding and Information Theory	43.04
	E.5	Files	43.04

EXAMPLE:

The 1998 ACM Computing Classification System (partial)

ACM Computing Classification System = Subject headings and shelfmarks Computer science			
Code heading	**Code sub-heading**	**Description heading/subheading**	**Aisle / book case**
	G.2	Discrete Mathematics	42.17
	G.3	Probability and Statistics	42.17
	G.4	Mathematical Software	42.17
H.		**Information Systems**	
	H.0	General	42.17
	H.1	Models and Principles	42.17
	H.2	Database Management	42.17 – 42.18
	H.3	Information Storage and Retrieval	42.18
	H.4	Information Systems Applications	42.18
	H.5	Information Interfaces and Presentation	42.18
	H.m	Miscellaneous	42.18
I.		**Computing Methodologies**	
	I.0	General	42.18
	I.1	Symbolic and Algebraic Manipulation	42.18
	I.2	Artificial Intelligence	42.18 – 42.19
	I.3	Computer Graphics	42.19 – 42.20
	I.4	Image Processing and Computer Vision	42.20
	I.5	Pattern Recognition	42.20
	I.6	Simulation and Modeling	42.20
	I.7	Document and Text Processing	42.20
	I.m	Miscellaneous	42.20
J.		**Computer Applications**	
	J.0	General	42.20
	J.1	Administrative Data Processing	42.20
	J.2	Physical Sciences and Engineering	42.20
	J.3	Life and Medical Sciences	42.21
	J.4	Social and Behavioral Sciences	42.21
	J.5	Arts and Humanities	42.21
	J.6	Computer-Aided Engineering	42.21
	J.7	Computers in Other Systems	42.21
	J.m	Miscellaneous	42.21

Domain Scoping – Knowledge Sources

- Domain experts
  - E.g., experts on monitoring systems
- Technical literature
  - Textbooks, journals, etc.
- Existing Systems
  - E.g., monitoring systems

Domain Literature

Existing Systems

Domain Expert

Evaluation

- Has it been proven/experimented?
- Is it widely acknowledged by the experts in the domain?
- Is it out-dated?
- Is it stable?

Abstraction Quality

- We need to evaluate both
  - Objective quality
  - Relevance quality

```
Abstraction Quality (ks) = (Objectivity(ks), Relevance(ks))
```

- Here **abstraction quality()**, **objectivity()**, and **relevance()** represents functions that define the corresponding quality factors of the argument **ks,** where ks is the knowledge source.

Exercise

- Evaluate following knowledge sources for producing monitoring systems

Knowledge Source	Objectivity [0..1]	Relevance [0..1]	Abstraction Quality
Domain expert with 20 years of experience in developing monitoring systems/control systems			
MSc thesis on feedback control systems			
PhD thesis on physiological characteristics of the eye			
20 years old textbook on control systems			
Flight cabin monitoring systems			
Manager of company who builds control systems