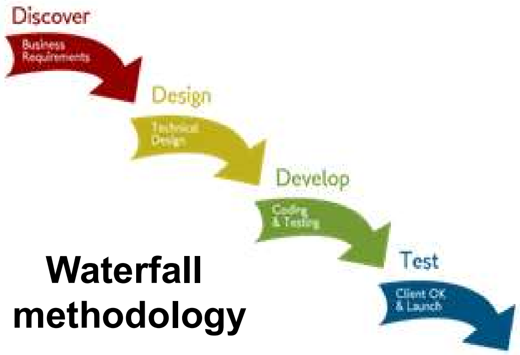
Introduction to Software Design using Traditional Methodologies

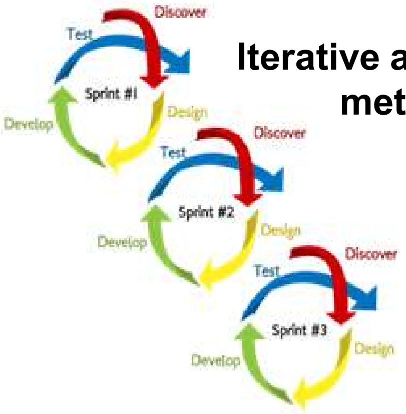
Introduction to Software Design using Traditional Methodologies

- Software Design and Architecture using Traditional Methodologies
- Software Architecture Views
- Pattern-based Design
- References

Software Design and Architecture using Traditional Methodologies



Waterfall methodology

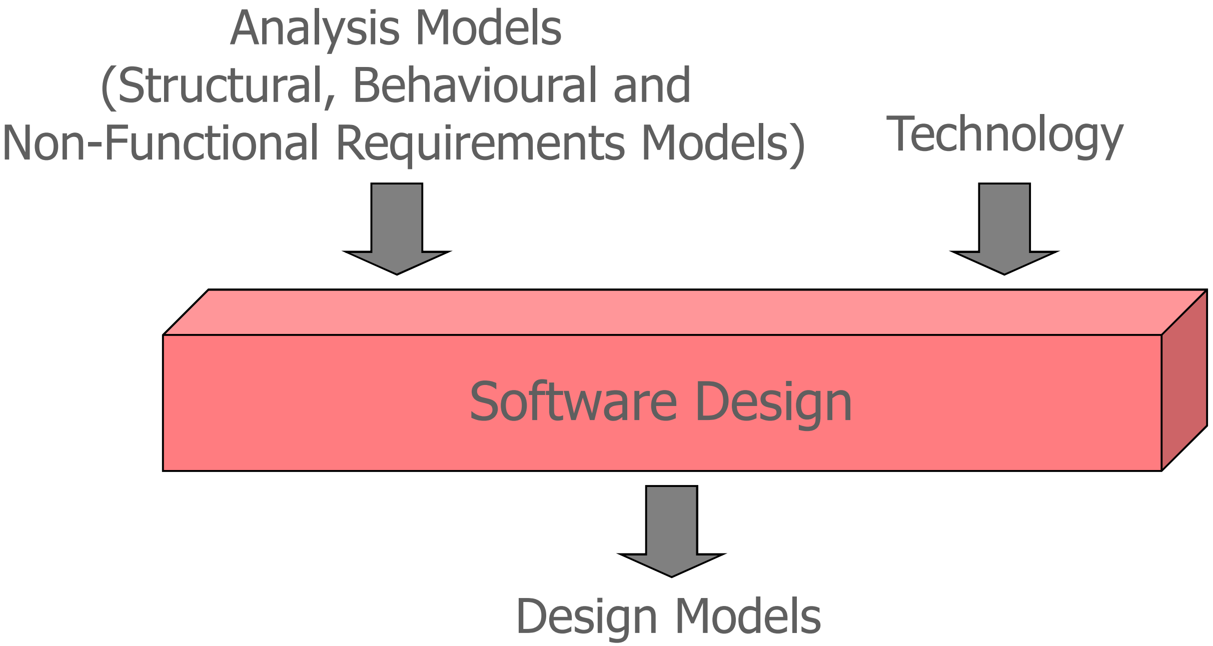


Iterative and incremental methodology

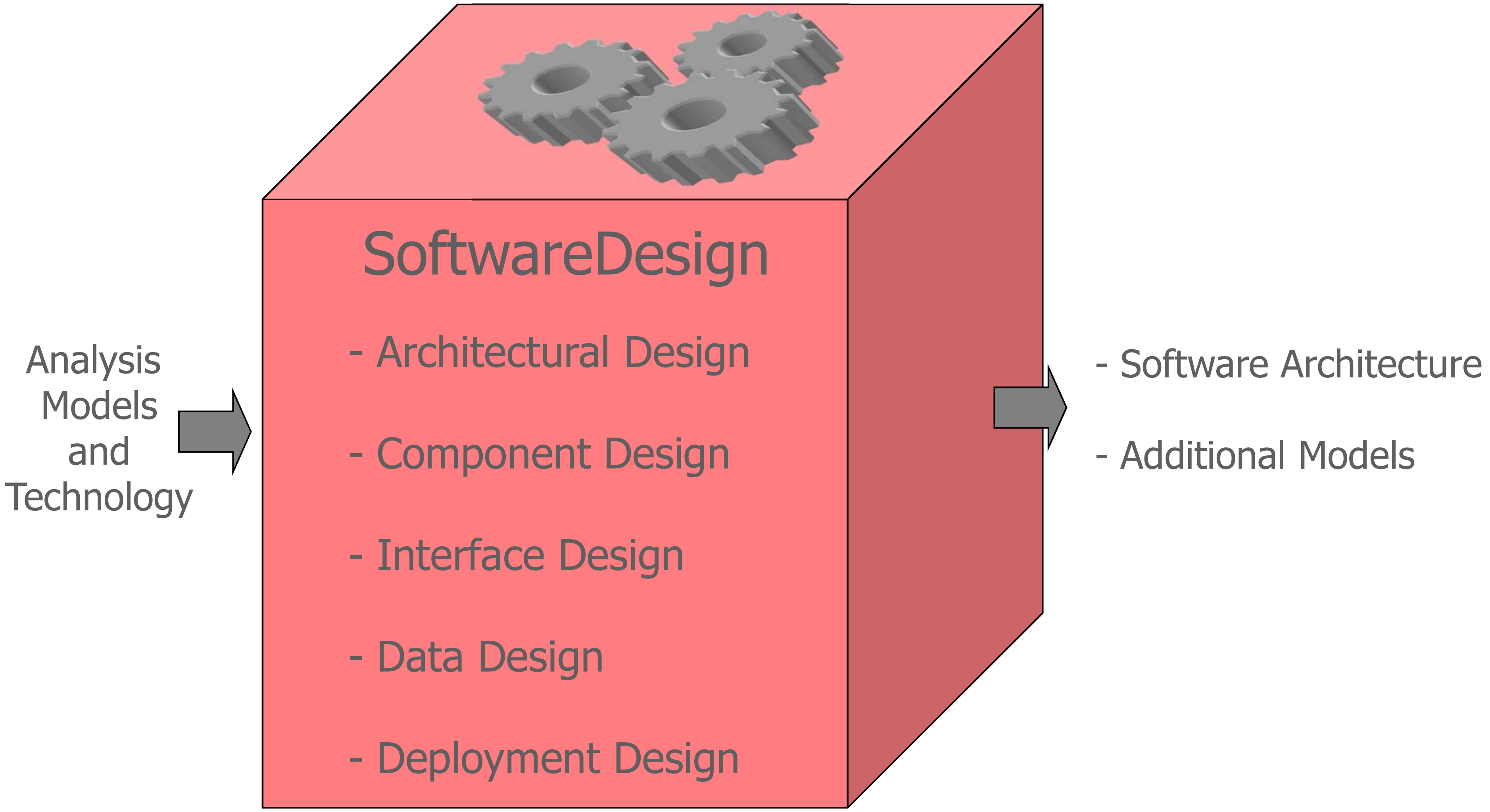
- Software design and Architecture in waterfall and in some of iterative methodologies starts when the discover phase is finished or almost finished.
- Software design and Architecture is very well documented and completed before coding starts.
- Software design is focused on completing modules of the architecture.
- Software design is a heavy process.
- Software design requires architects and designers.
- Architects have infrequent interactions with business people.

Software Design and Architecture using Traditional Methodologies

- Inputs and outputs of software design


Analysis Models
(Structural, Behavioural and
Non-Functional Requirements Models)
Technology
Software Design
Design Models

Software Design and Architecture using Traditional Methodologies



Analysis Models and Technology

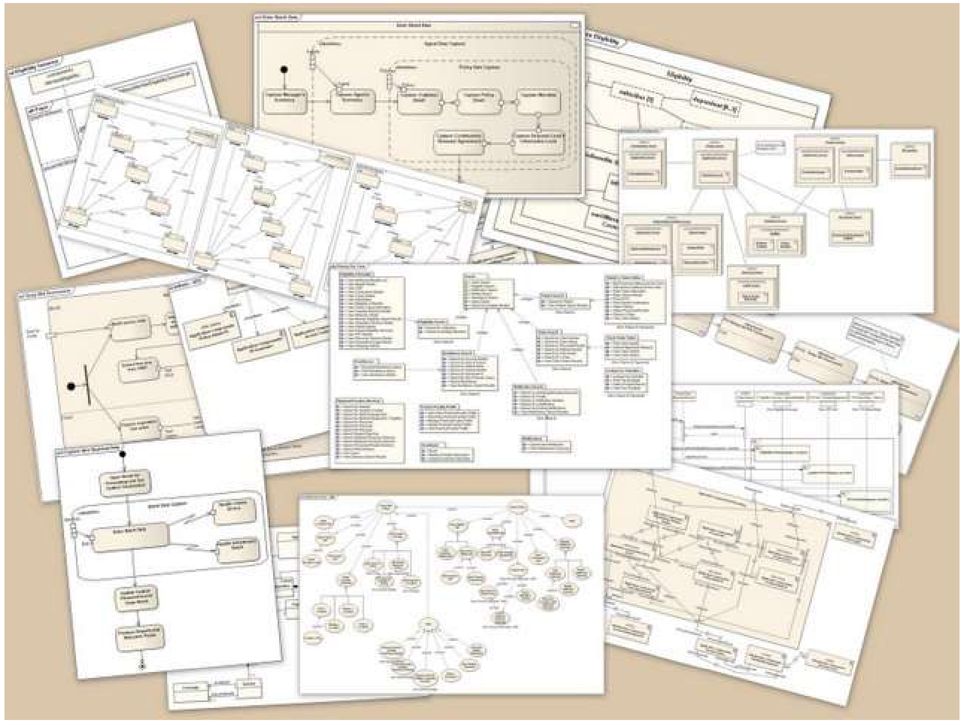
SoftwareDesign

- Architectural Design
- Component Design
- Interface Design
- Data Design
- Deployment Design

- Software Architecture
- Additional Models

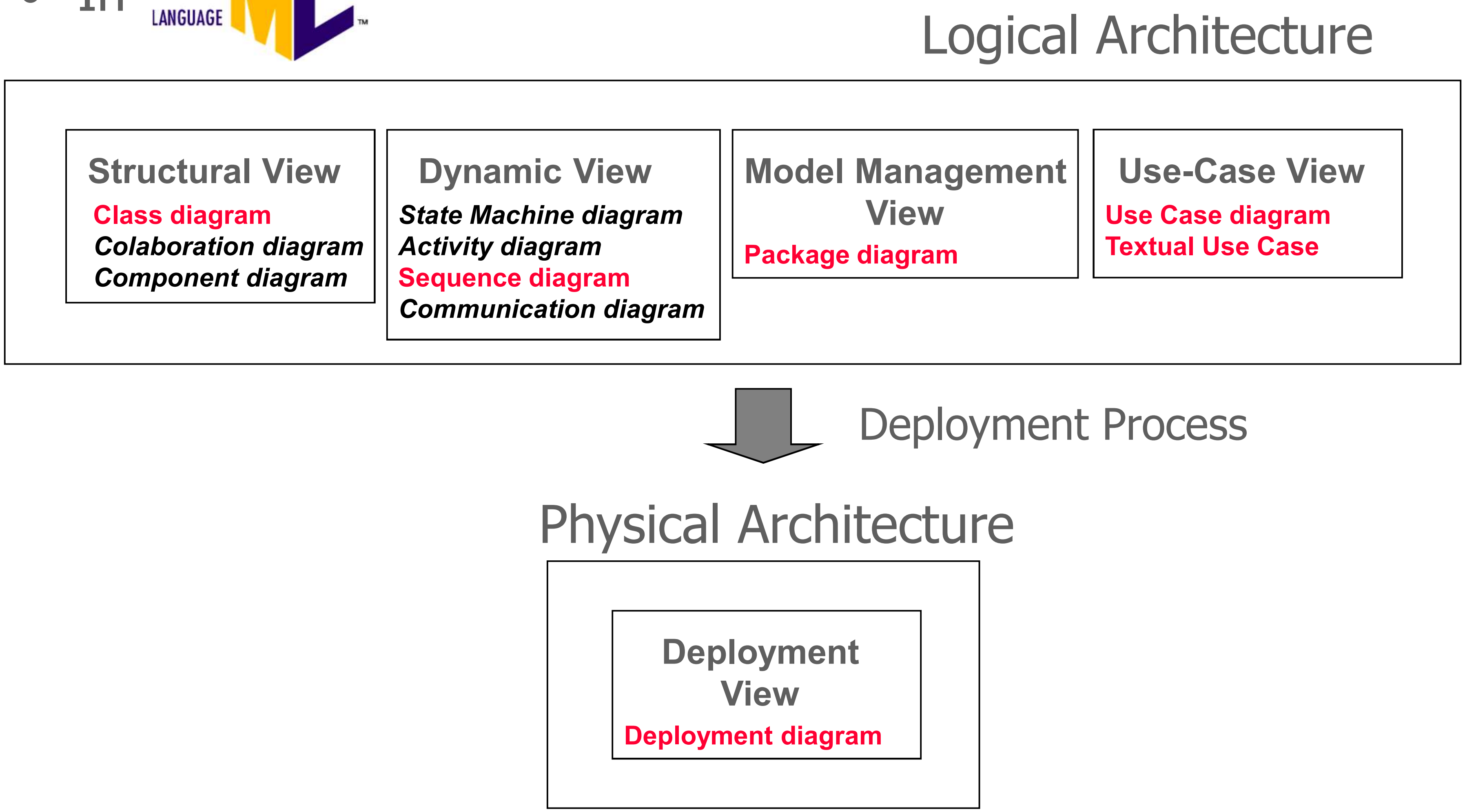
Software Architecture Views

- The use of different views to represent the architecture of software systems in traditional methodologies allows us to address separately the concerns of various stakeholders of the architecture (end-users, developers, project managers, etc…)



Software Architecture Views

- In UNIFIED MODELING LANGUAGE UML™



Logical Architecture

Structural View

Class diagram
Colaboration diagram
Component diagram

Dynamic View

State Machine diagram
Activity diagram
Sequence diagram
Communication diagram

Model Management View

Package diagram

Use-Case View

Use Case diagram
Textual Use Case

Deployment Process

Physical Architecture

Deployment View

Deployment diagram

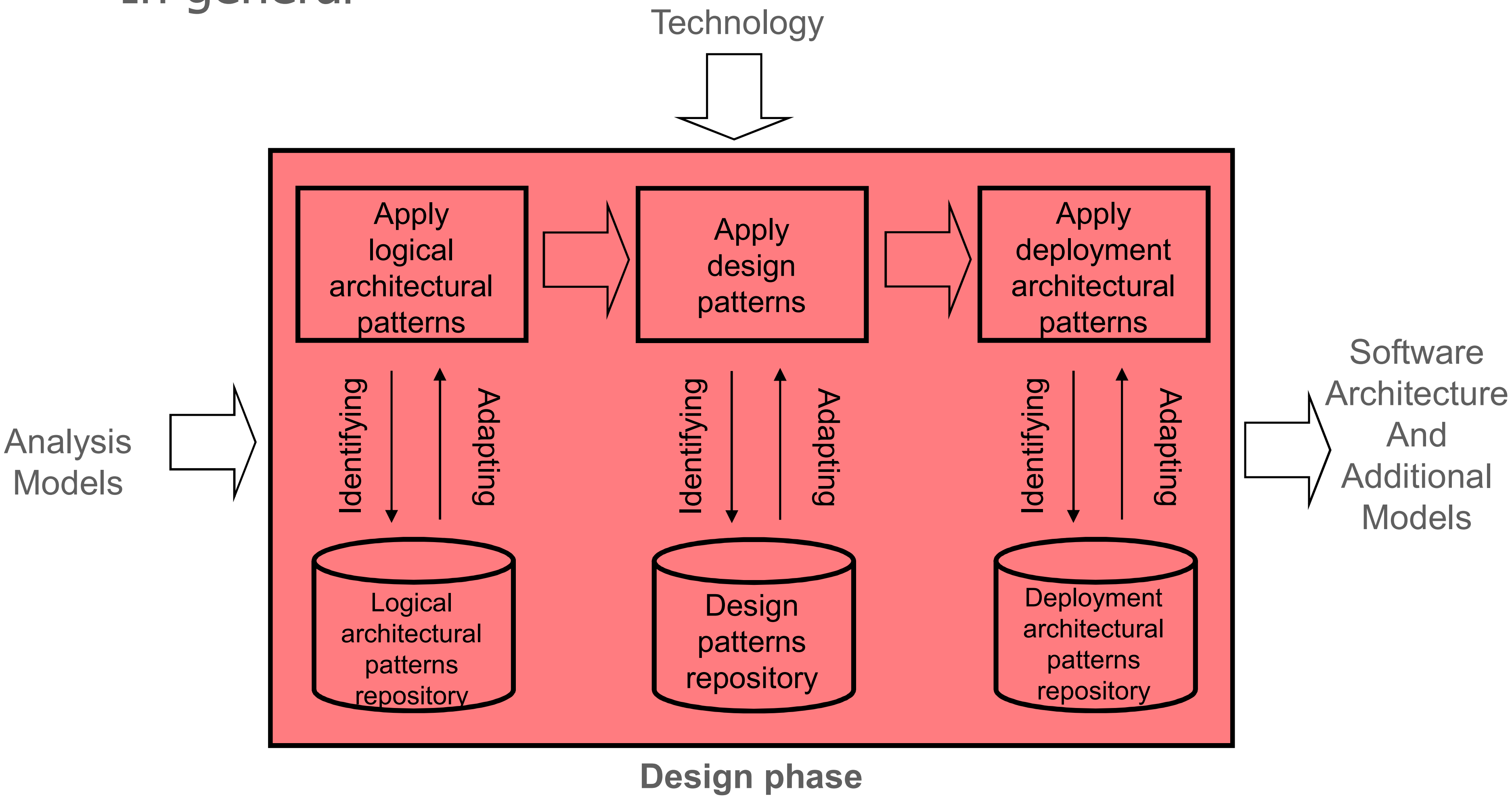
Pattern-based Design

- **Pattern-based design** creates a new application by finding a set of proven solutions to a clearly delineated set of problems. Each problem and its solution is described by a design pattern that has been catalogued and vetted by other software engineers.

- Two types of patterns used at the design phase:
  - Architectural patterns
  - Design patterns

Pattern-based Design

- In general

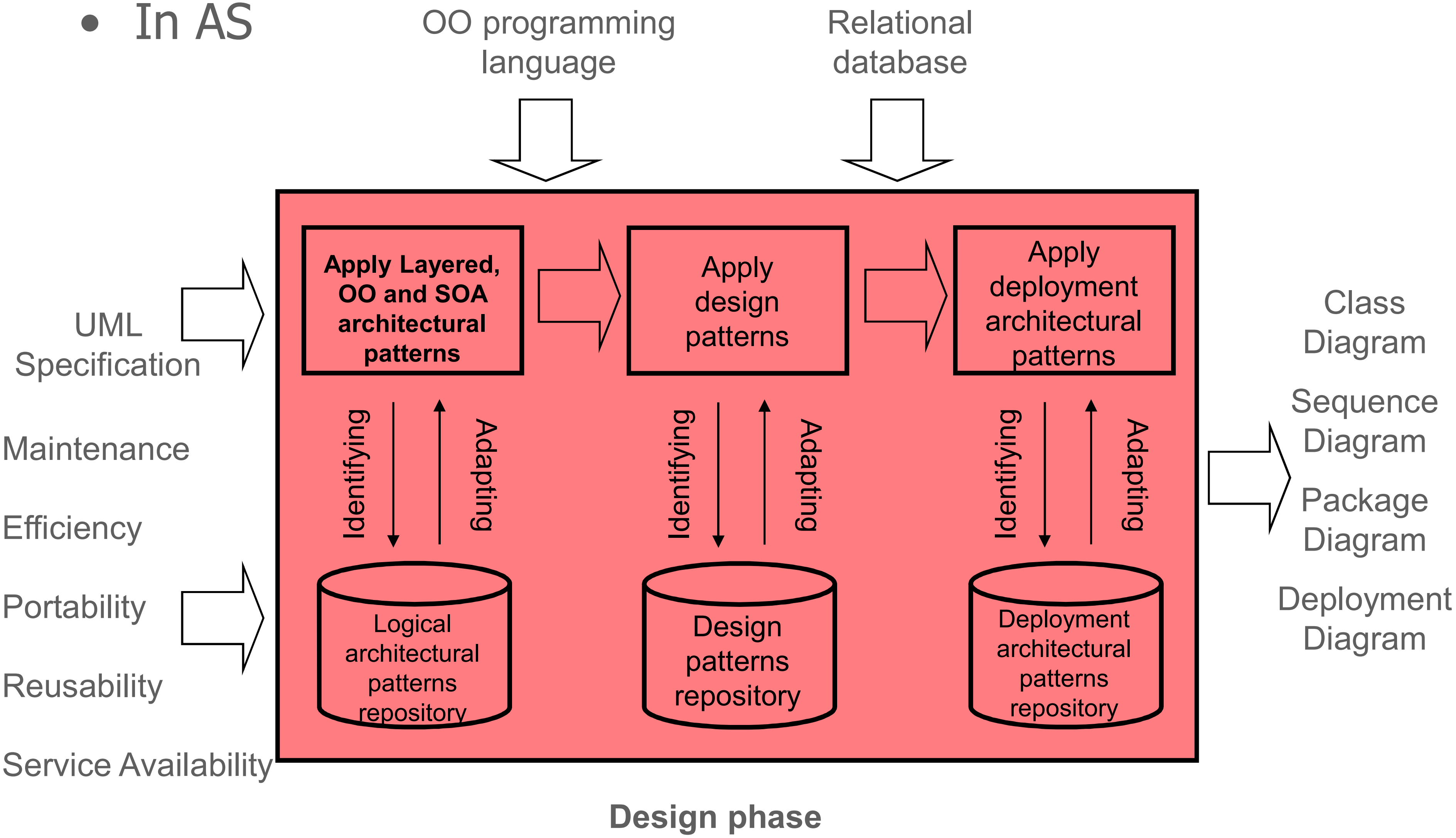

Technology
Analysis Models
Apply logical architectural patterns
Apply design patterns
Apply deployment architectural patterns
Identifying
Adapting
Identifying
Adapting
Identifying
Adapting
Logical architectural patterns repository
Design patterns repository
Deployment architectural patterns repository
Software Architecture And Additional Models
Design phase

Pattern-based Design

- In AS we will apply the pattern based design to software systems that:
  - Keep a consistent representation of the domain state.
  - Answer queries about the domain state.
  - Produce reactions when some predefined conditions are given.
  - Use external services

Pattern-based Design

- In AS



OO programming language

Relational database

UML Specification

Maintenance

Efficiency

Portability

Reusability

Service Availability

Apply Layered, OO and SOA architectural patterns

Apply design patterns

Apply deployment architectural patterns

Identifying

Adapting

Logical architectural patterns repository

Design patterns repository

Deployment architectural patterns repository

Class Diagram

Sequence Diagram

Package Diagram

Deployment Diagram

Design phase

References

- *Ingeniería del software. Un enfoque práctico*
  R.G. Pressman
  McGraw Hill, 2010 (Séptima edición), cap. 8, 9 and 10

- *Enginyeria del software: Especificació*
  D. Costal, X. Franch, M.R. Sancho, E. Teniente
  Edicions UPC, 2004

- *Applying UML and Patterns*
  C. Larman
  Prentice Hall, 2005 (3rd edition), ch. 33, 34 and 39

- *Software Engineering*
  I. Sommerville
  Pearson, 2011 (9th edition), ch. 6

- *The Unified Modeling Language Reference Manual*
  J. Rumbaugh, I. Jacobson, G. Booch
  Addison-Wesley, 2004 , ch. 3

- Microsoft Application Architecture Guide (2nd edition)
  Microsoft
  http://msdn.microsoft.com/en-us/library/ff650706.aspx, ch. 1,2 and 3