

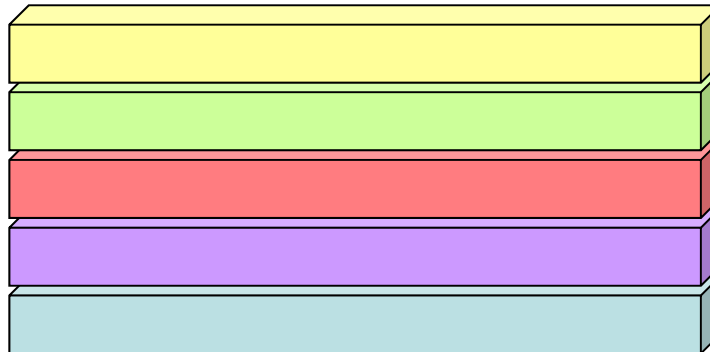
Layered Architecture

Layered Architecture

- Context
- Problem
- Solution
- Architecture Definition Considerations
- Pros and Cons
- Application to Software Systems
- Layered architecture in UML
- Layered Architecture Example
- Layered Architecture Design Principles
- References

Context

- A complex system that is composed of a large number of components across multiple levels of abstraction.



Problem

- How do you structure a system to support such requirements as maintainability, reusability, portability, scalability, robustness, and security?



Forces to balance



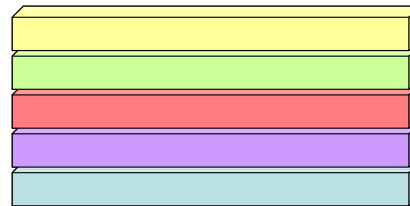
- Similar responsibilities shall be grouped
- Concerns shall be separated among components
- Changes shall be localized to one part of the solution



It shall be possible to reuse and substitute system components for alternative implementations



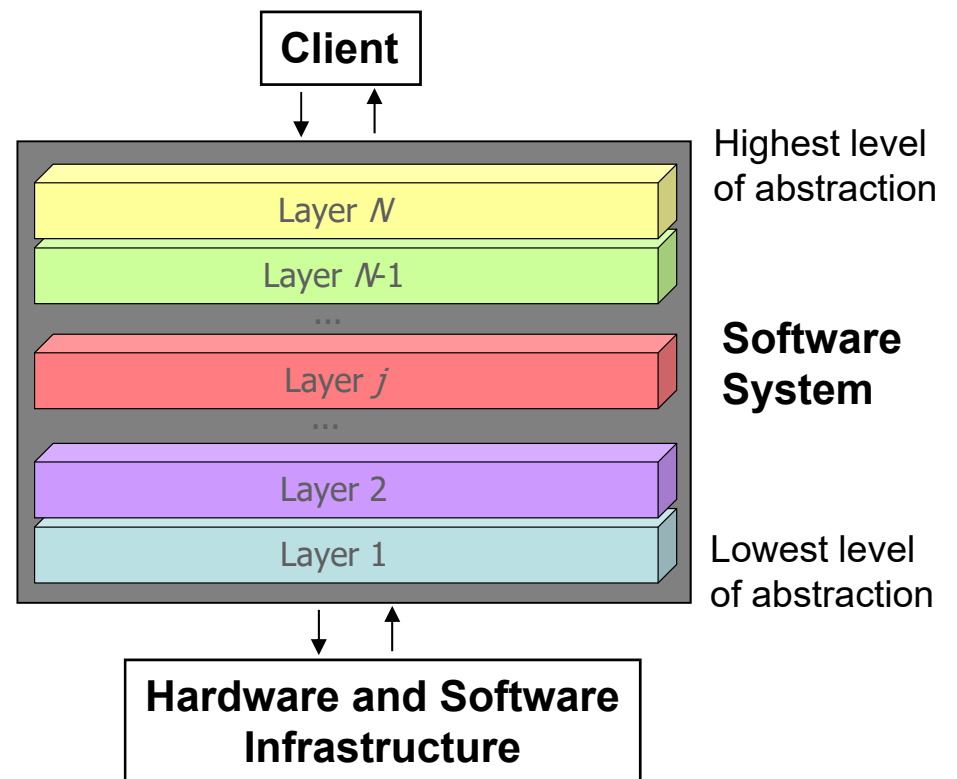
The system shall be portable to other platforms



Intermediate services may provoke longer procedure call chains

Solution

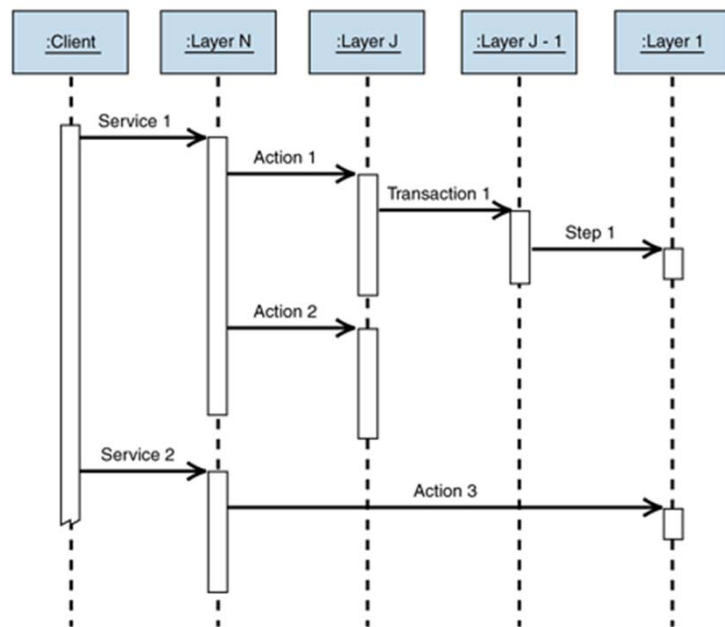
- The system is structured in an appropriate number of layers.
- All the layers are set vertically.
- All components in the same layer must work at the same level of abstraction.
- The services provided by layer j use the services provided by layer $j - 1$. Simultaneously, layer j services can depend on other services included in same layer.



Solution

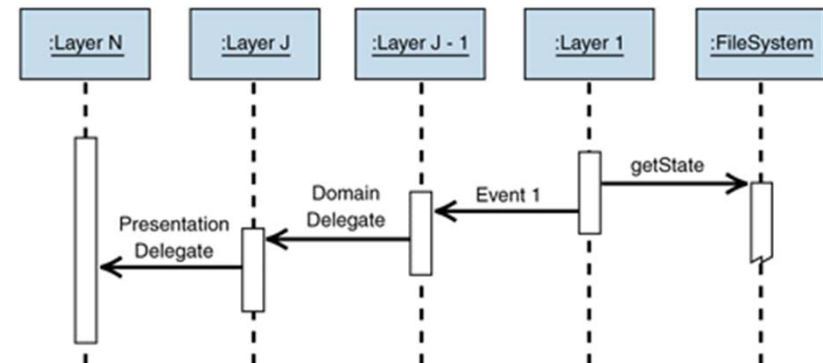
Top-down communication

An external entity interacts with the highest layer of the stack. The highest layer uses one or more services of the lower layers



Bottom-up communication

Layer 1 is monitoring the state of some external entity and notifies it to the higher layers



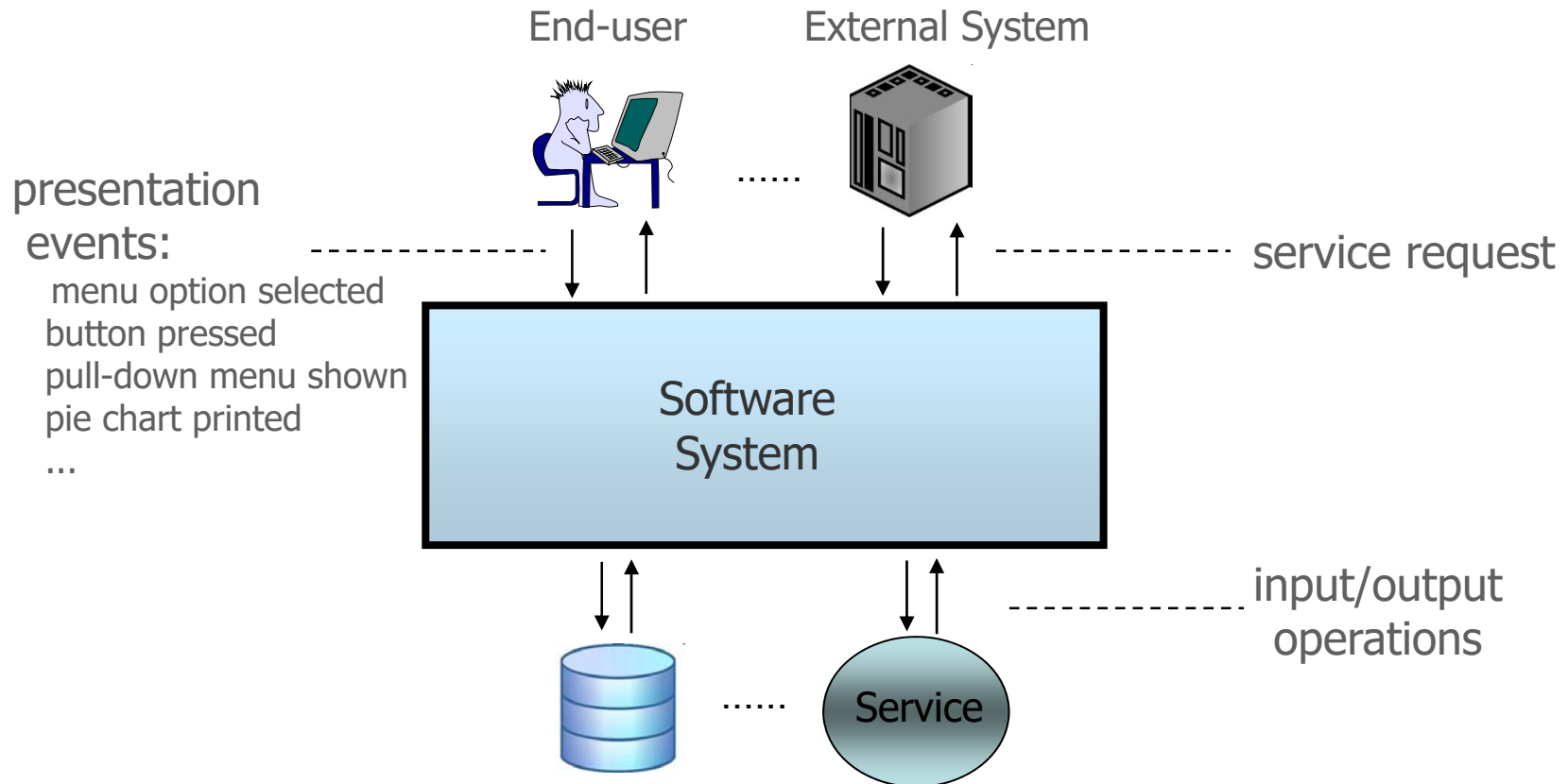
Architecture Definition Considerations

- Define the abstraction criteria and the number of layers
- Determine the functionalities in each layer:
 - The highest layer defines the services exposed to the outside.
- Specify the contracts for each layer:
 - The interface defines the services exposed to the immediately upper layer.
- Structure the layers individually
 - A component cannot be distributed into two or more layers.
- Apply design principles
 - Both inter- and intra-layer
- Design a strategy to manage errors
 - Handle the errors in the layer where they are detected or propagate to the immediately higher layer.

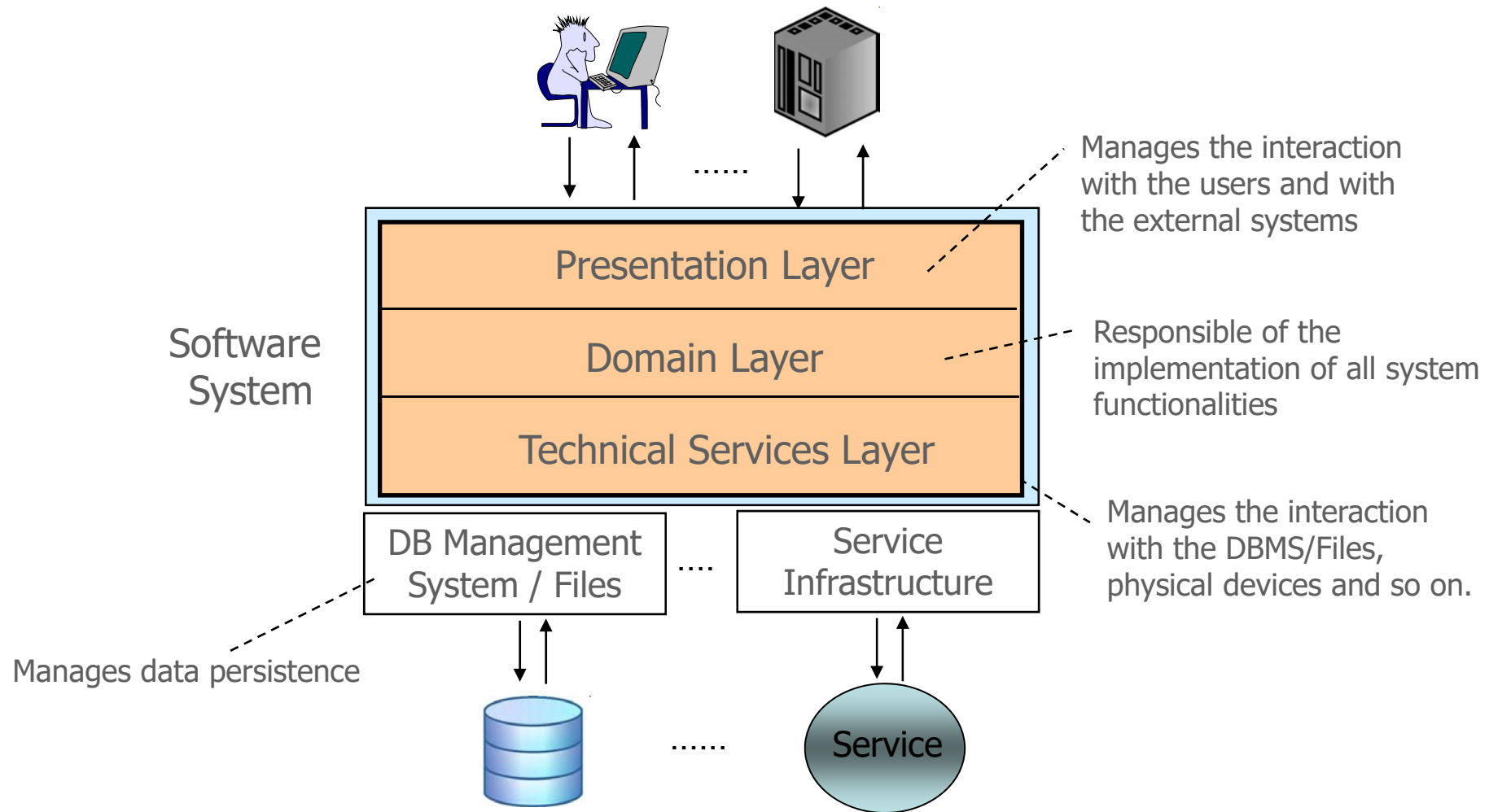
Pros and Cons

- **Pros:**
 - Maintainable, Reusable, Portable, Testable.
- **Cons:**
 - Efficiency
 - Unnecessary or redundant work
 - Difficulty to establish the granularity and determine the number of layers.

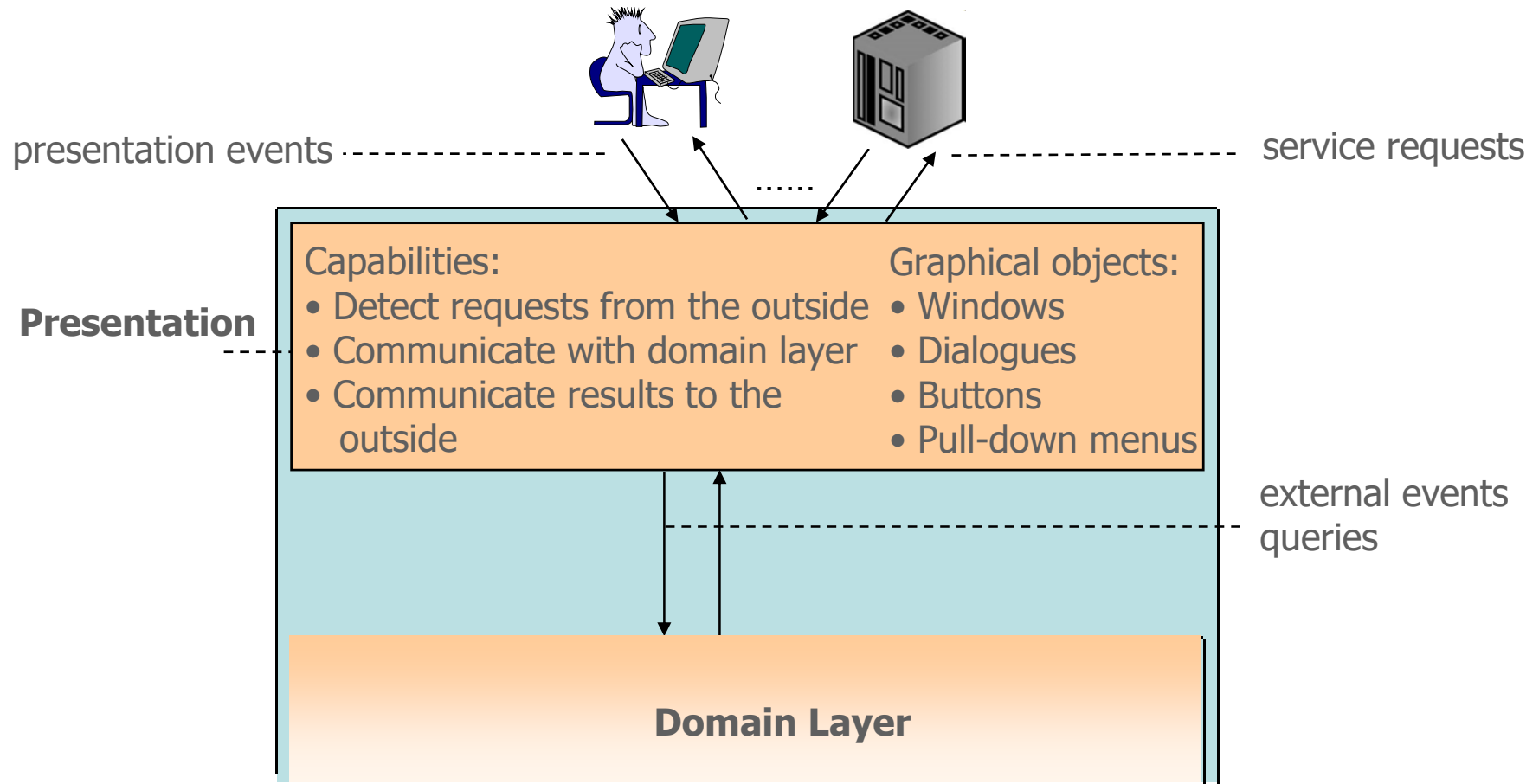
Application to Software Systems



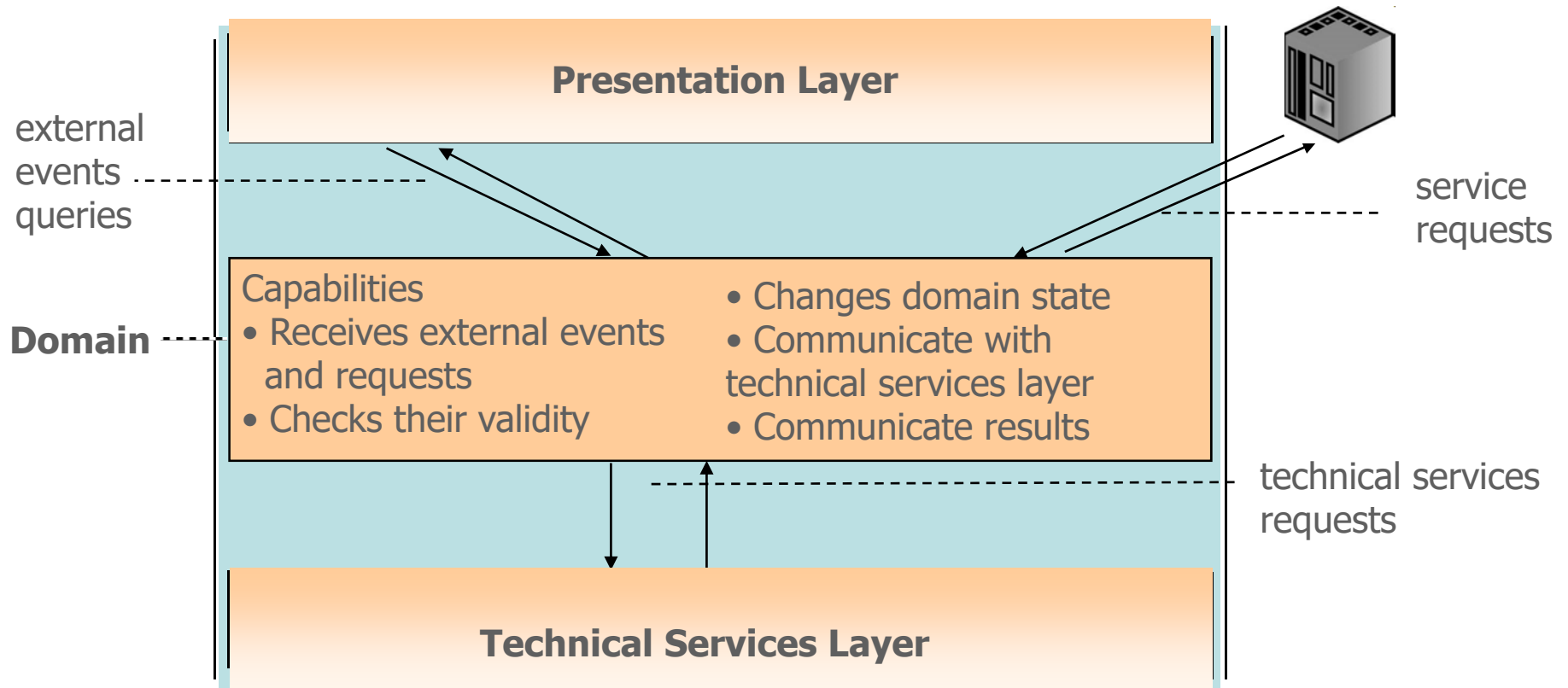
Application to Software Systems



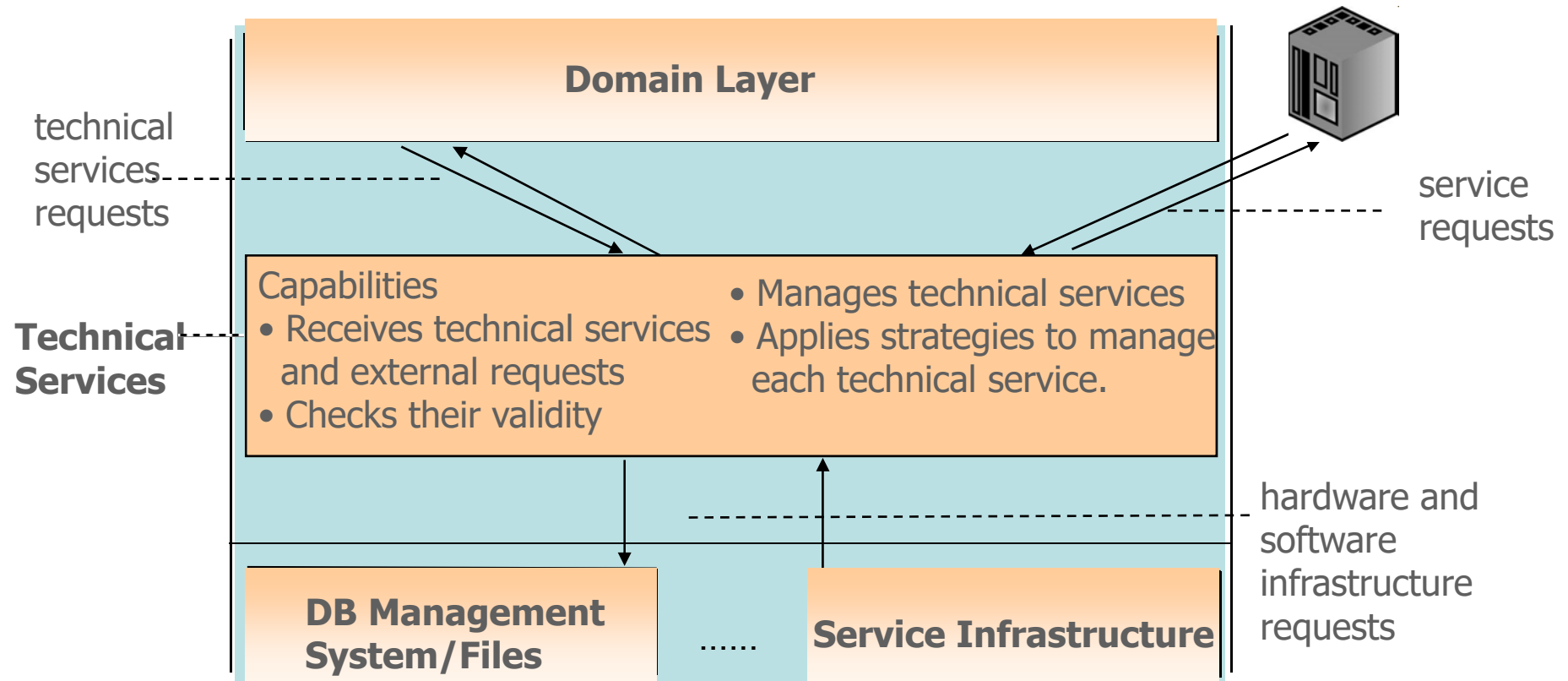
Application to Software Systems



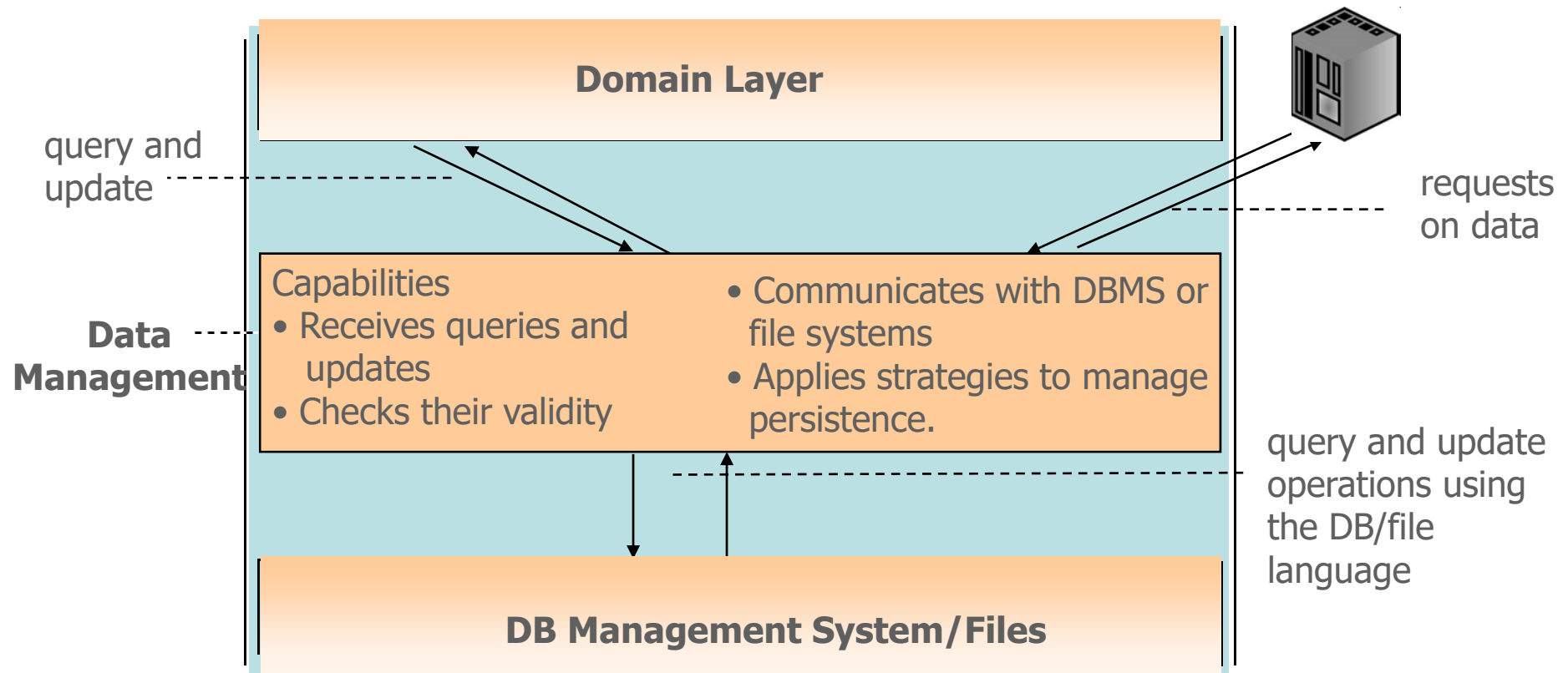
Application to Software Systems



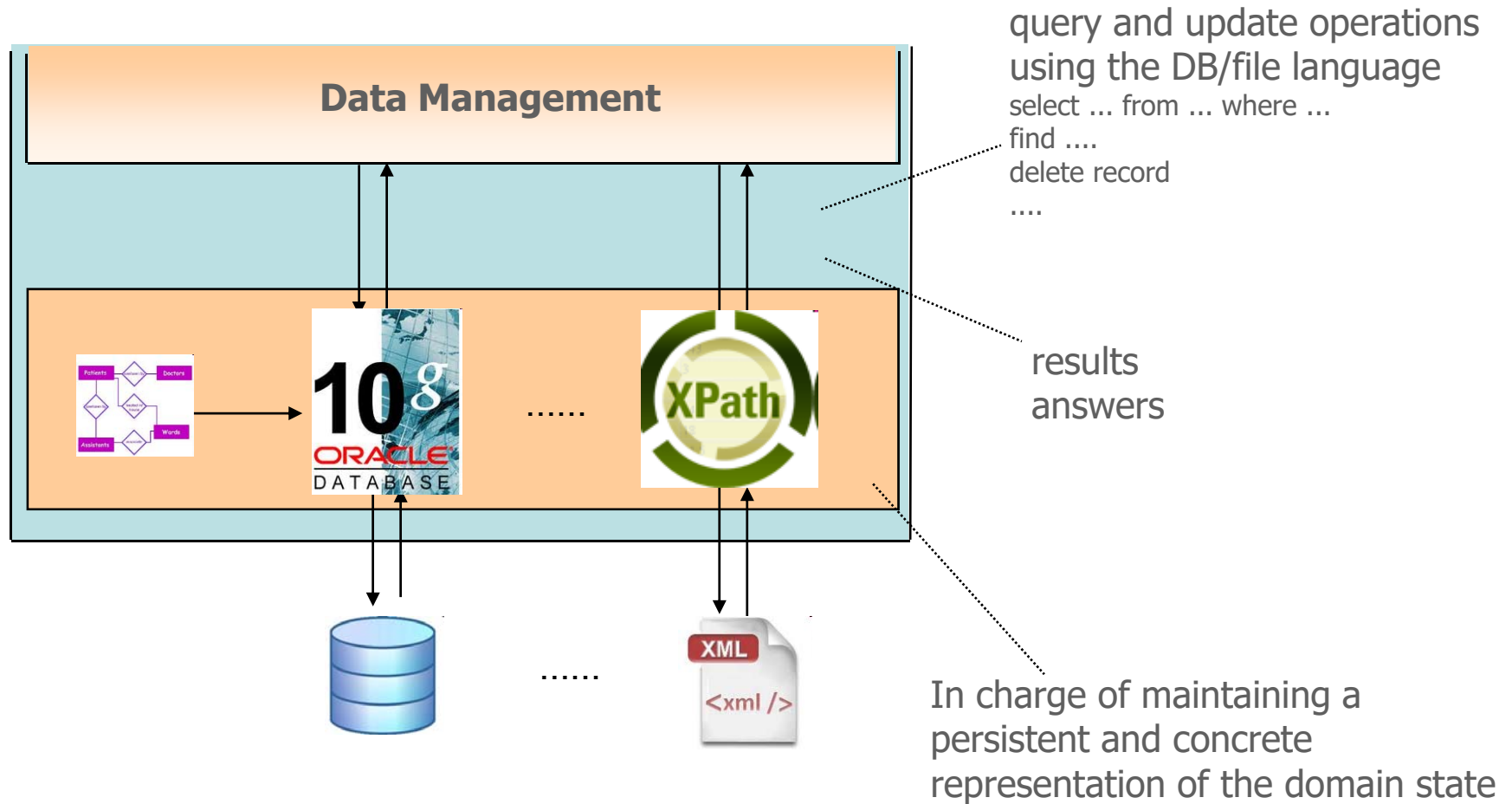
Application to Software Systems



Application to Software Systems



Application to Software Systems



Layered architecture in UML

- UML package diagrams are often used to illustrate the layered architecture of a system. A layer can be modeled as a UML package.
- A UML package diagram provides a way to group elements as classes, other packages and so on.
- Packages may have dependency relationships to other packages. The UML dependency line is used for this.
- A dependency indicates a situation in which a change to the supplier element may require a change in the client element.

Layered Architecture Example

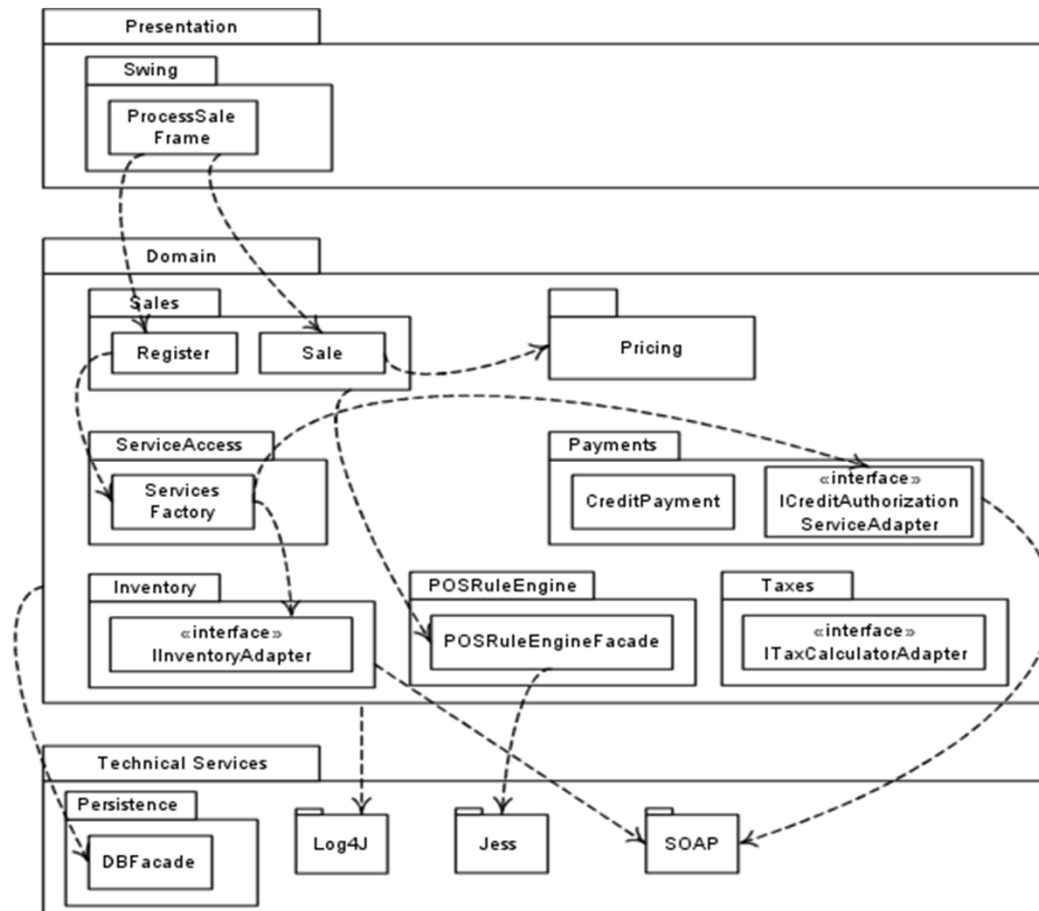
- NextGen is a point-of-sale system (POS) used to record sales and handle payments.
- It is typically used in a retail store. It includes hardware components as a computer and bar code scanner; and software to run the system.
- It interfaces to various service applications, such as a third-party tax calculator and inventory control. These systems must be relatively fault-tolerance; that is even if remote services are temporarily unavailable (such as the inventory system), they must still be capable of capturing sales and handling at least cash payments.
- A POS system increasingly must support multiple and varied client-side terminals and interfaces.

Layered Architecture Example



Layered Architecture Example

- Layered architecture of POS system and UML Notation



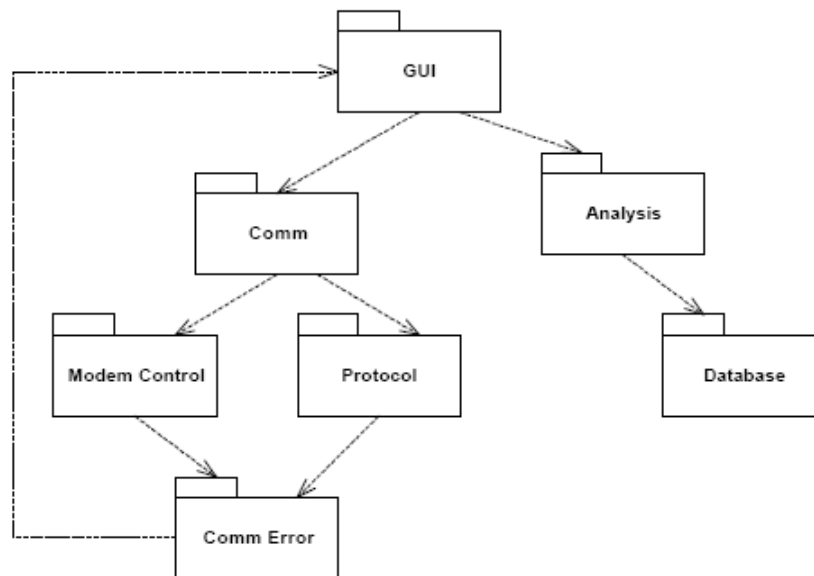
Layered Architecture Design Principles

- Package Cohesion Principles:
 - The Release Reuse Equivalence Principle (REP): The granule of reuse is the granule of release (Reusability).
 - The Common Closure Principle (CCP): Classes that change together, belong together (Maintenance).
 - The Common Reuse Principle (CRP): Classes that are not reused together should not be grouped together (Reusability).

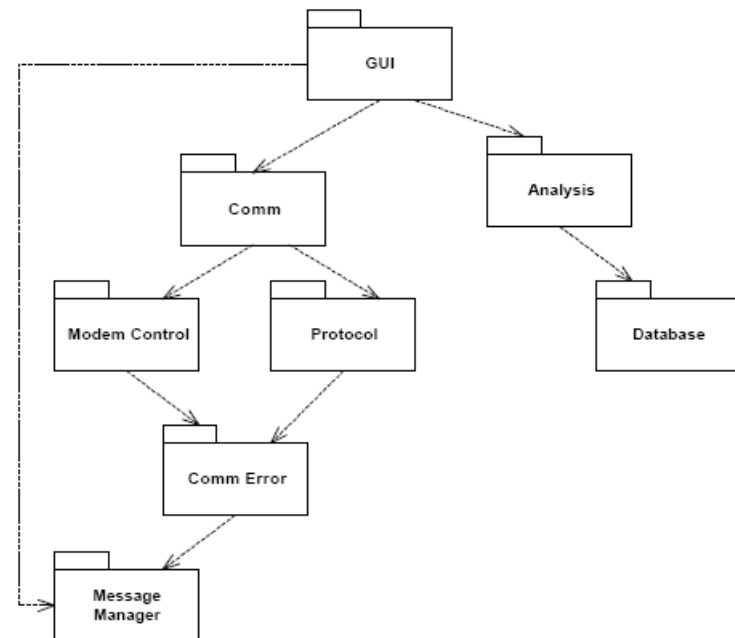
Layered Architecture Design Principles

- Package Coupling Principles:
 - The Acyclic Dependencies Principle (ADP): The dependencies between packages must not form cycles.

ADP Violation



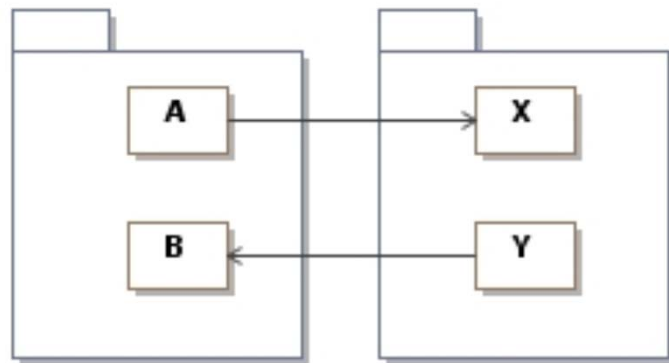
ADP Satisfaction



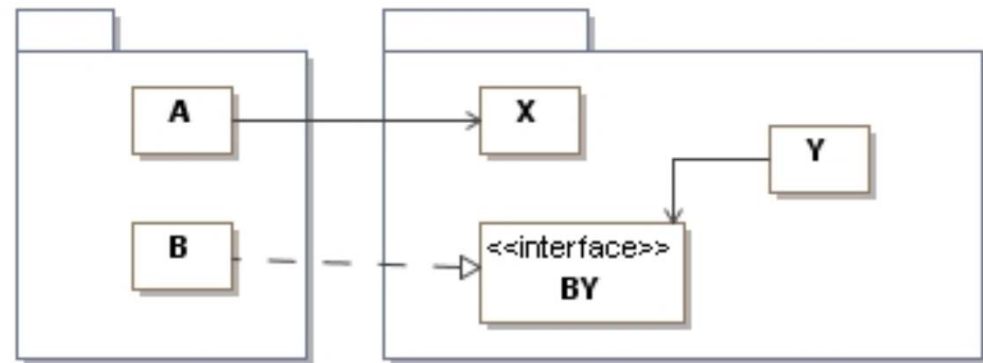
Layered Architecture Design Principles

- Another way for breaking cycles

ADP Violation



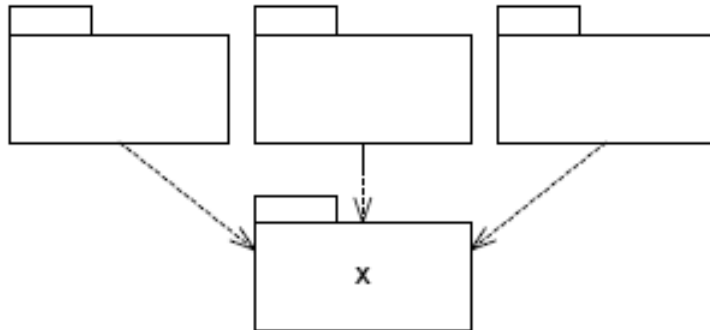
ADP Satisfaction



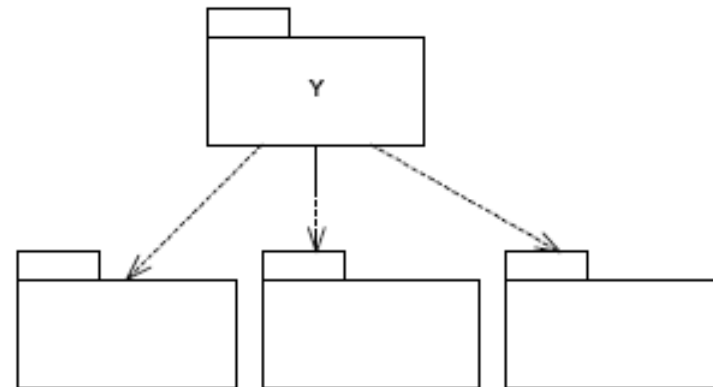
Layered Architecture Design Principles

- Package Coupling Principles:
 - The Stable Dependencies Principle (SDP): Depend in the direction of stability.

X must be a stable package

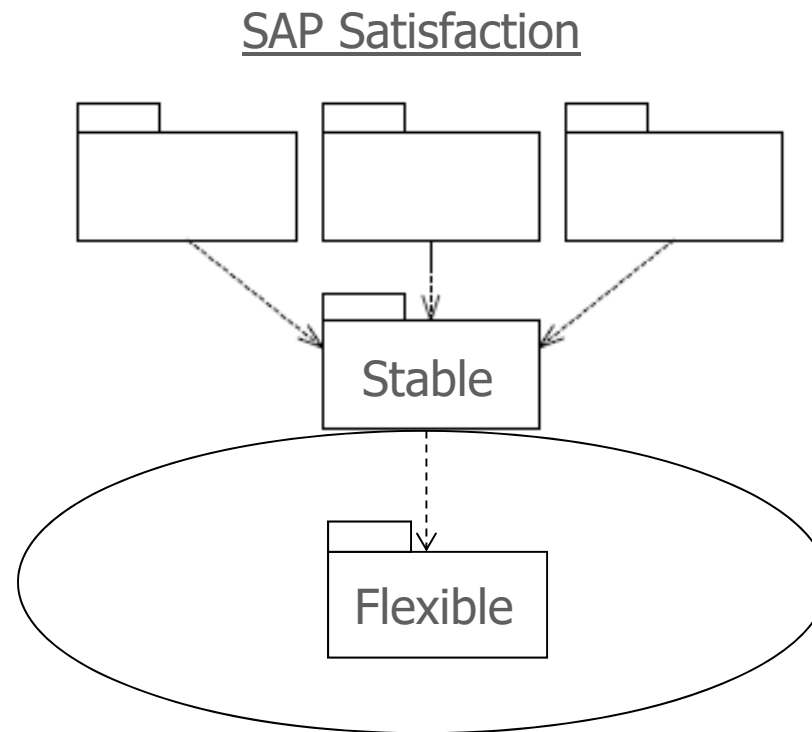


Y is an instable package



Layered Architecture Design Principles

- Package Coupling Principles:
 - The Stable Abstractions Principle (SAP): Stable packages should be abstract packages.



References

- *Pattern-oriented Software Architecture. A System of patterns*
F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal
John Wiley & Sons, 1996. Pages 31-51
- *Applying UML and Patterns*
C. Larman
Prentice Hall, 2005 (Third edition), ch. 13 and 34
- *Agile Software Development: Principles, Patterns and Practices*
R.C. Martin
Prentice Hall, 2003
- Layered application
Microsoft
<http://msdn.microsoft.com/en-us/library/ff650258.aspx>
- Design principles and design patterns
R.C. Martin
[http://www.objectmentor.com/resources/articles/Principles and Patterns.pdf](http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf)