UNIVERSITY of GUELPH

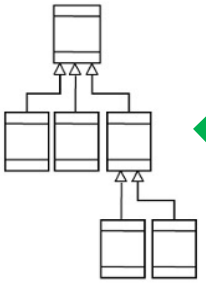
ENGINEERING

Package and Component Diagrams

Fattane Zarrinkalam

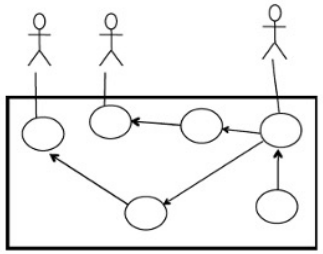
Fowler: UML Distilled: A Brief Guide to the Standard Object Modeling Lang. 3rd Ed. (Chapter 7 &14)
K. Hamilton and R. Miles “Learning UML 2.0. O'Reilly”, ISBN-10: 0- 596-00982-8, 2006. (Chapter 12 & 13)

UML Diagrams



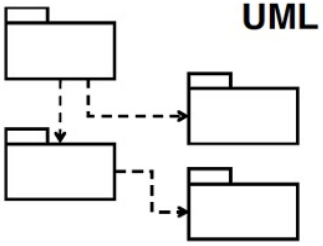
UML Class Diagrams

- information structure
- relationships between data items
- modular structure for the system



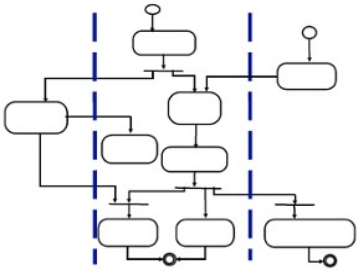
Use Cases

- user's view
- Lists functions
- visual overview of the main requirements



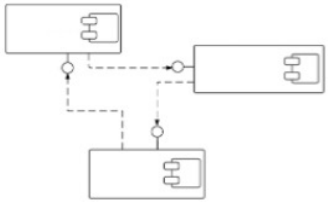
UML Package Diagrams

- Overall architecture
- Dependencies between components



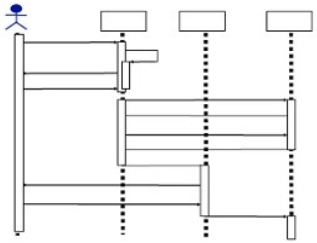
Activity diagrams

- business processes;
- concurrency and synchronization;
- dependencies between tasks;



UML Component Diagrams

- static implementation view of a system
- Interfaces between components



UML Sequence Diagrams

- individual scenario
- interactions between users and system
- Sequence of messages

Structural Diagram

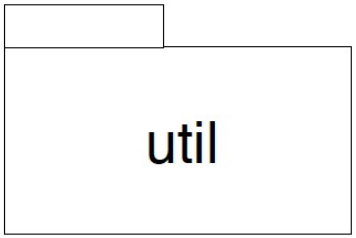
- Class diagram
- Package diagram
- Component diagram

Package Diagram

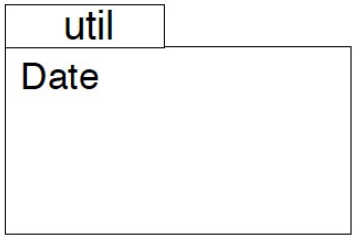
UML Packages

- UML elements can be grouped together in packages
- Elements of a package may be:
  - classes;
  - models (e.g., use case models, sequence diagrams, etc.)
  - other packages (representing subsystems or modules);
  - a package can contain both other packages and classes.
  - each element of a UML model is owned by a single package

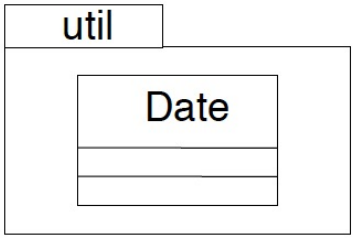
Package Notations



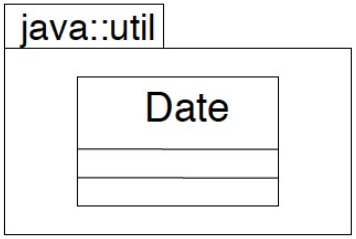
named package



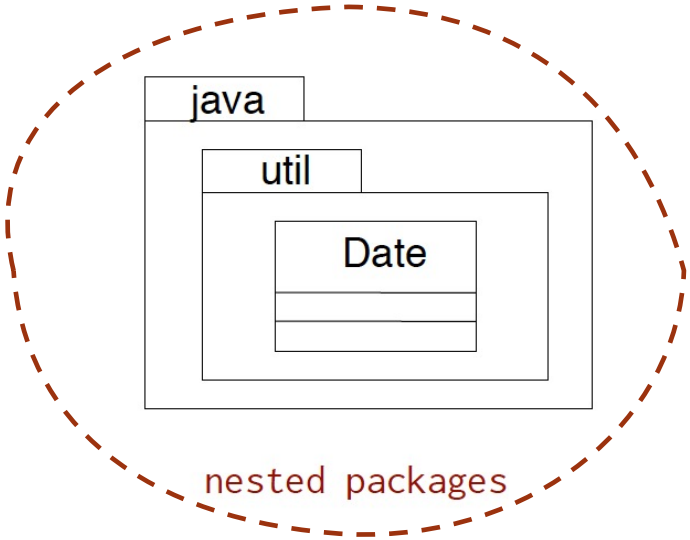
package with list
of contained classes



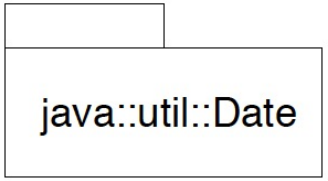
package containing
a class diagram



fully qualified
package name



nested packages

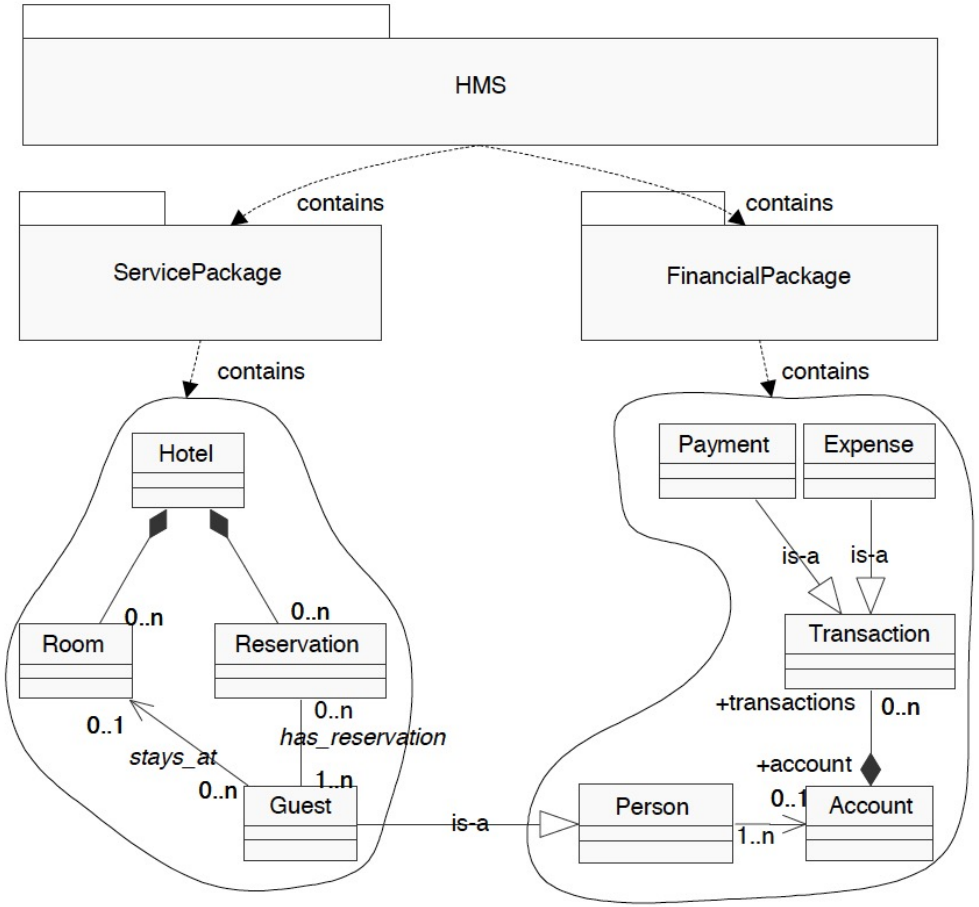


fully qualified
class name

UML Packages

- Criteria for decomposing a system into packages:
  - Different owners
    - who is responsible for working on which diagrams?
  - Different applications
    - each problem has its own obvious partitions;
  - Clusters of classes with strong cohesion
    - e.g., course, course description, instructor, student,…
  - Or: use an architectural pattern to help find a suitable decomposition…

Package Decomposition

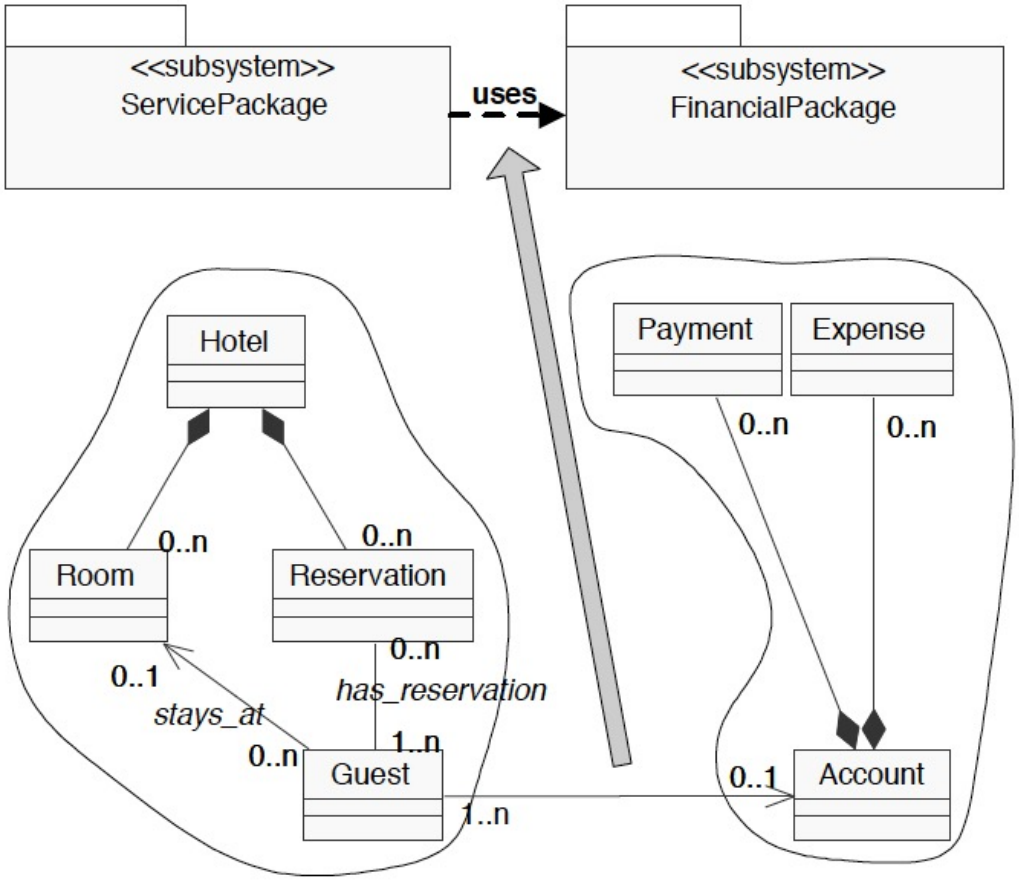


Source: “Automated Abstraction of Class Diagrams, TSE 2002

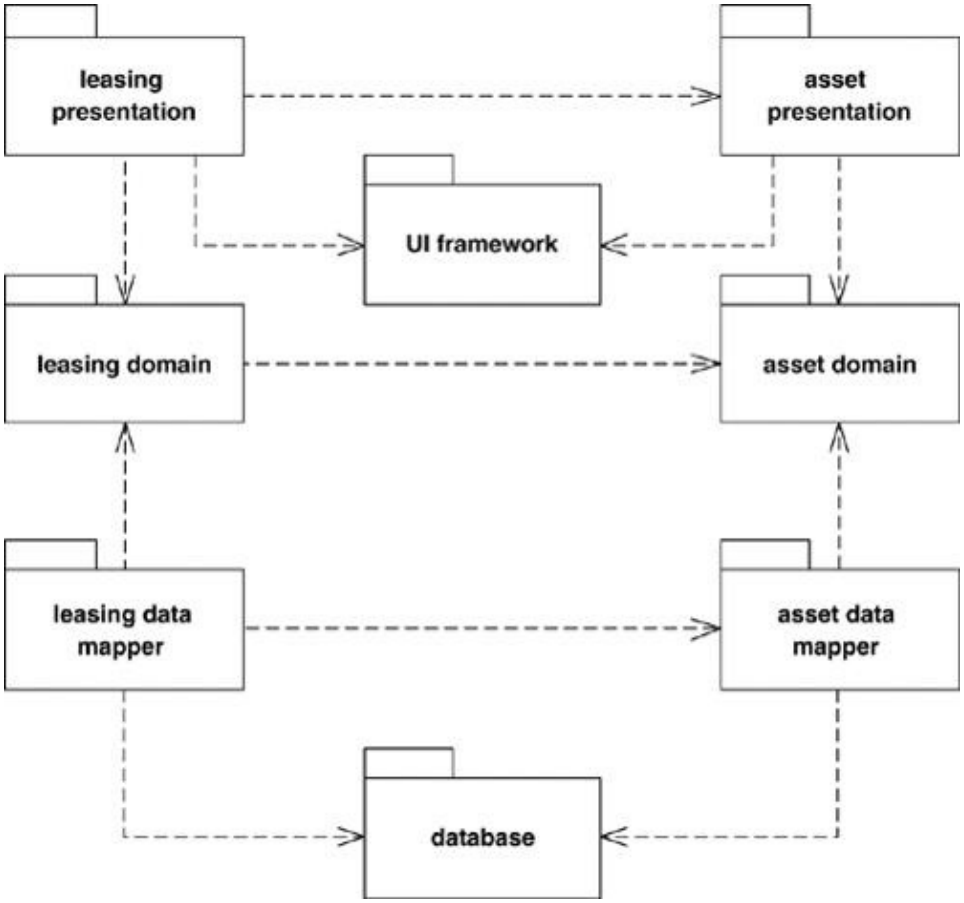
Package Diagram

- A package diagram shows packages and their dependencies
- Represents the system at a higher abstraction level
- Inter package dependencies summarize the dependencies between their contents.

Finding Dependencies

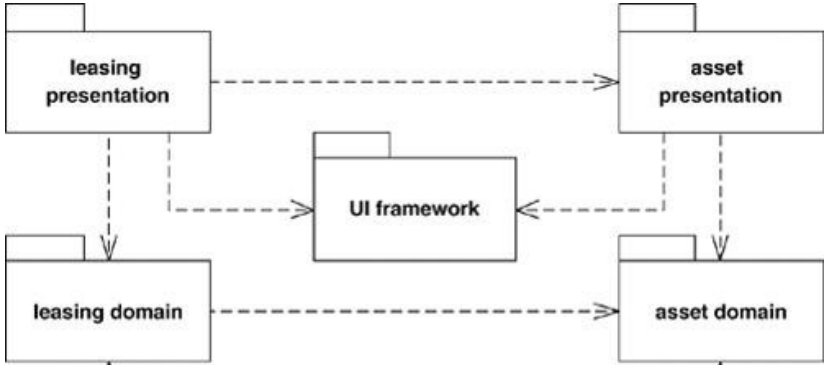


Package diagram for an enterprise application



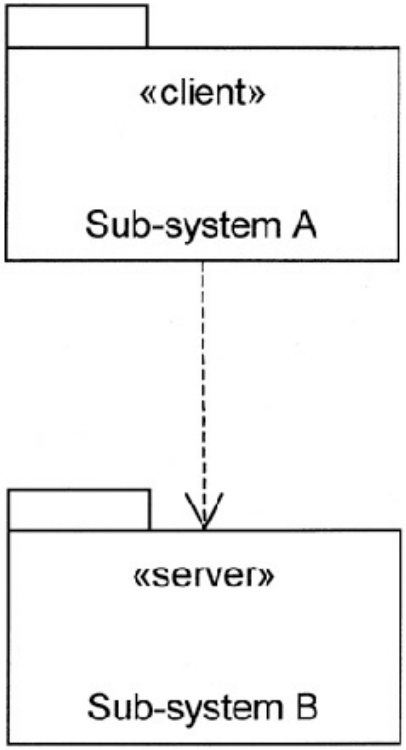
Packages and Dependencies

- The dependency relationships are not transitive.

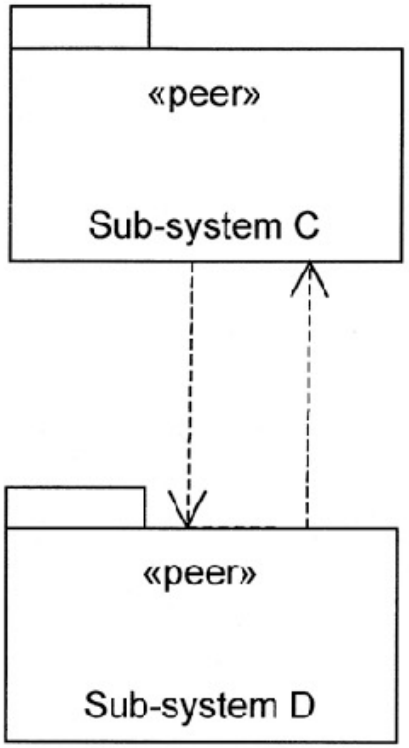


- The more dependencies coming into a package, the more stable the package's interface needs to be.
- Some packages are used in so many places that it would be a mess to draw all the dependency lines to them.
  - In this case, a convention is to use a keyword, such as «global», on the package.

Dependency cycles (to be avoided)



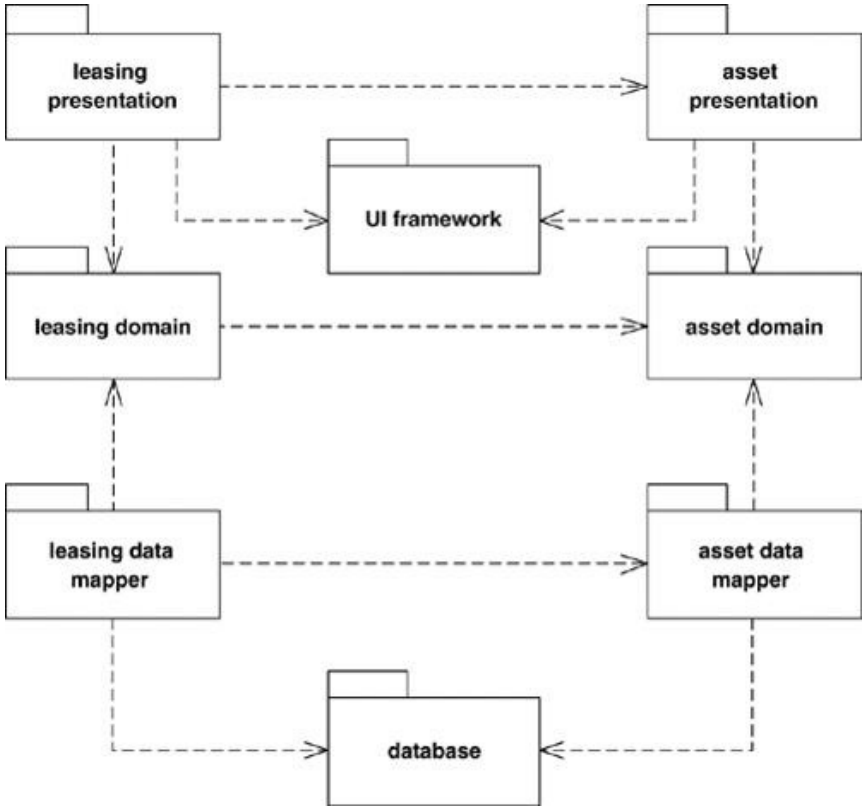
The server sub-system does not depend on the client sub-system and is not affected by changes to the client's interface.



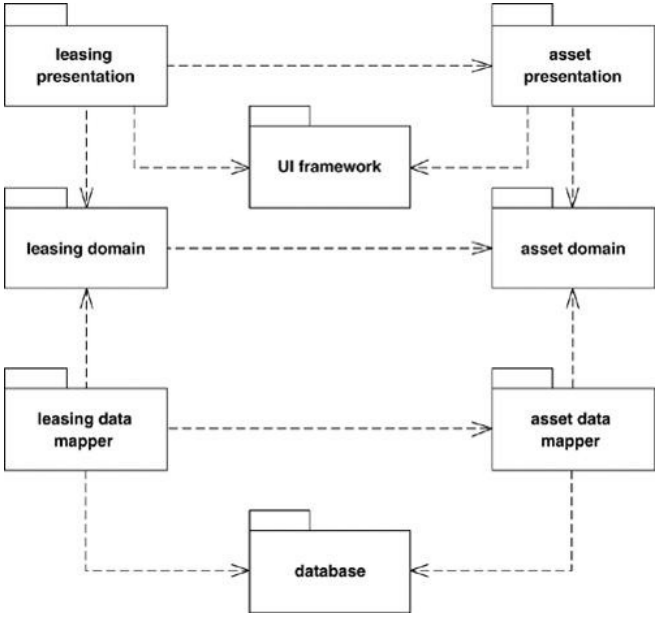
Each peer sub-system depends on the other and each is affected by changes in the other's interface.

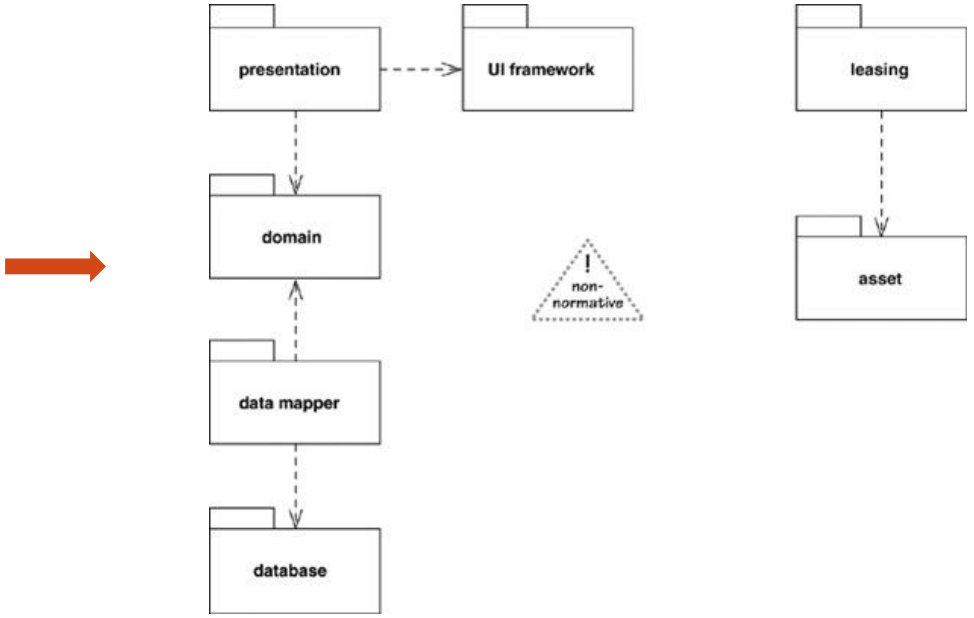
Package Aspects

- Two kinds of structures:
  - A structure of layers in the application:
    - presentation, domain, data mapper, and database.
  - A structure of subject areas:
    - leasing and assets.
- You can separate the two aspects



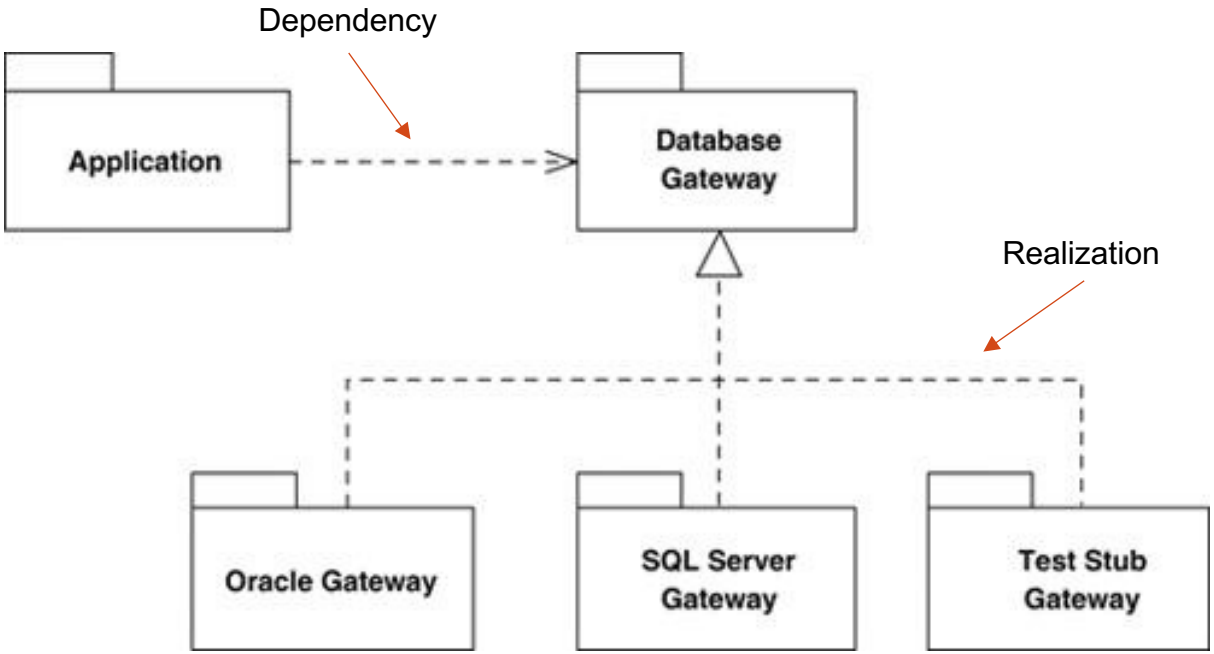
Package Aspects



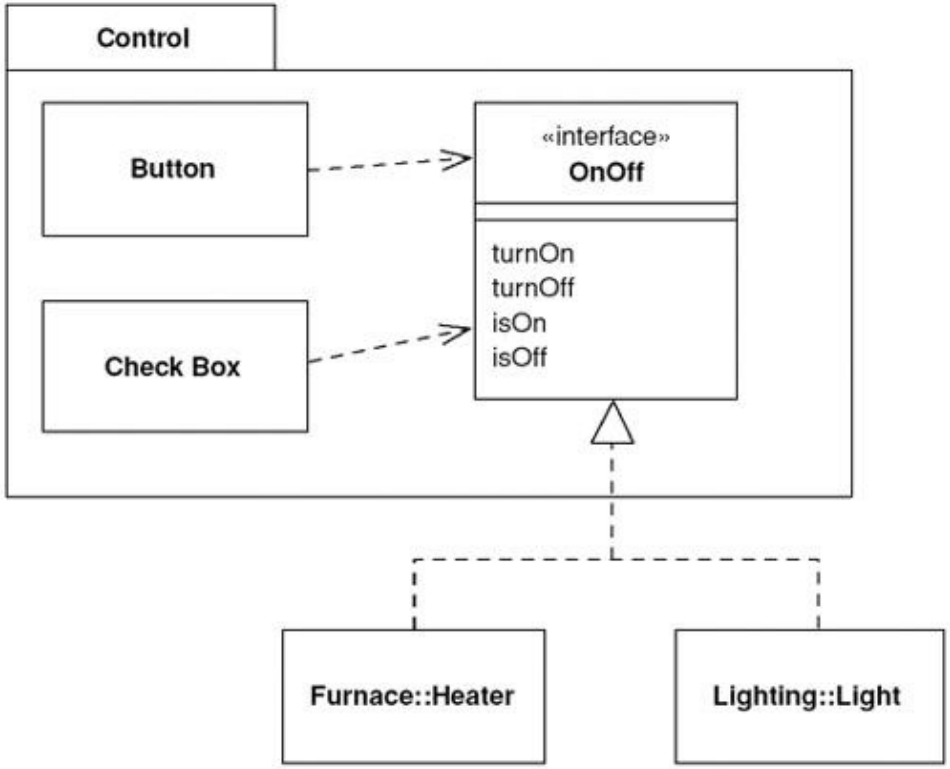


Implementing Packages

- It's quite common for an interface and its implementation to be in separate packages



Implementing Packages



When to Use Package Diagrams

- Larger-scale systems to get a picture of the dependencies between major elements of a system.
- Keep an application's dependencies under control.
- Correspond well to common programming structures.
- Represent a compile-time grouping mechanism.

Component Diagram

Component Diagram

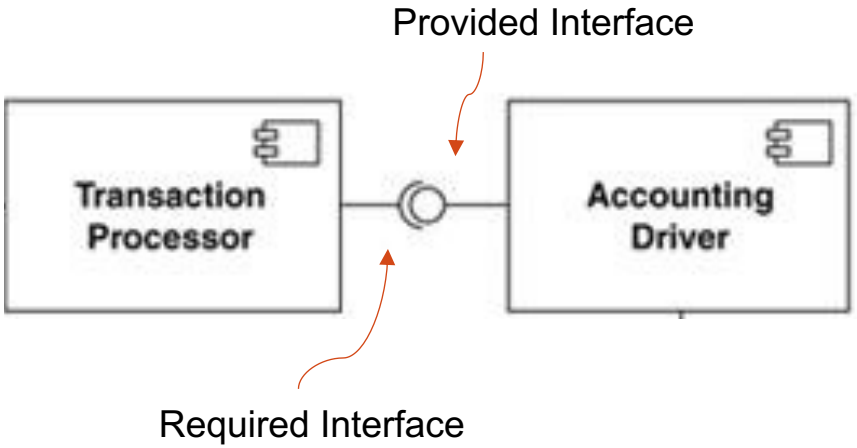
- Components are subsystems: replaceable parts of the system with well-defined interfaces
- Components are connected through implemented and required interfaces
- They represent pieces that are independently purchasable and upgradeable
  - Dividing a system into components is as much a marketing decision as it is a technical decision

Component Diagram

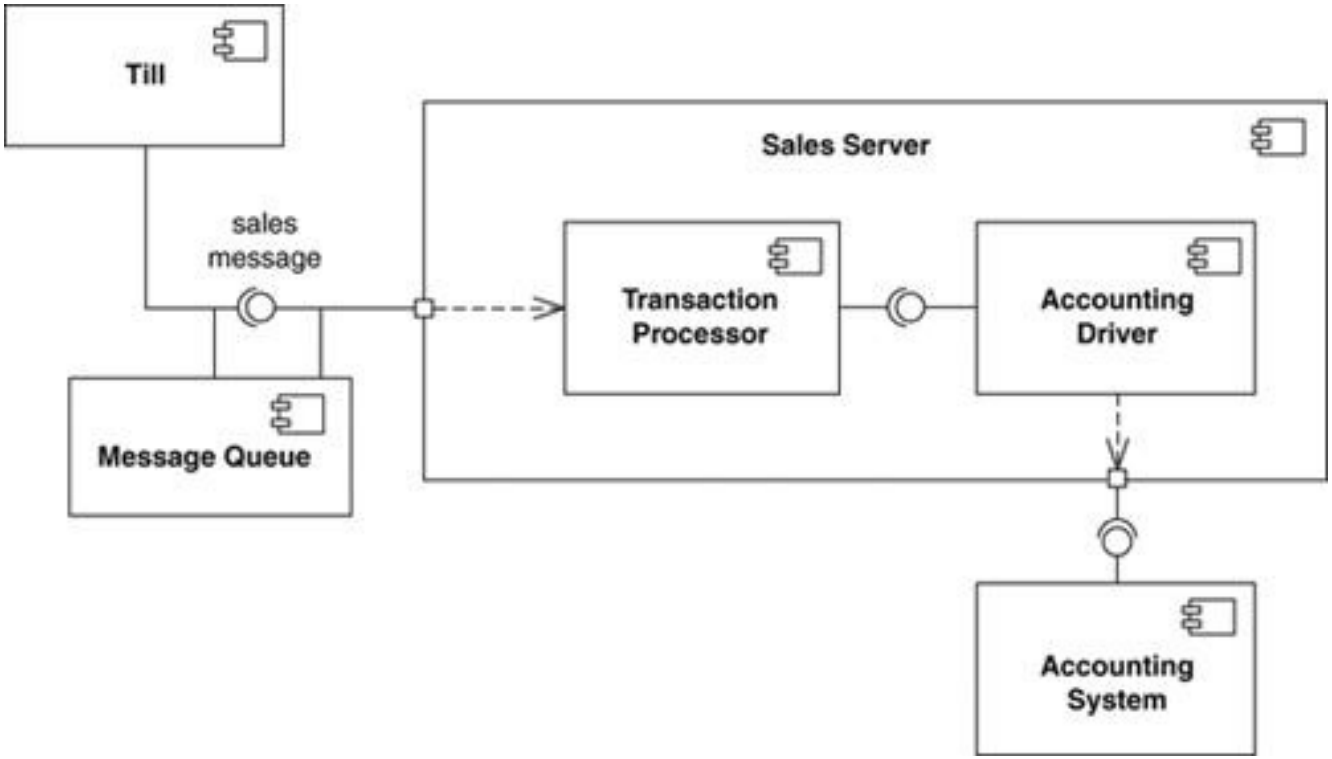
- Concerned with modeling the implementation of a system.
- Usually applied during design activities to determine how implementation activities will build the system;
- To determine the elements of the system on which implementation activities will focus.

Component

Assembly connector



Sample Component Diagram



When to Use Component Diagrams

- Dividing your large-scale system into components
- Show their interrelationships through interfaces or the breakdown of components into a lower-level structure

Summary

Summary

- Similarities of package and component diagram:
  - Structural diagrams that we can use to represent the architecture of our system in high-level units.
  - Define boundaries and are used to group elements into logical structures.

- Differences of package and component diagram :
  - Components are **groups of classes that are deployed together**
  - Packages are a general grouping device for model elements. Packages can group any model elements, even things like use cases, but in practice they usually group classes