

Large-scale structural modeling

Modeling the physical structure of a system

Justin Templemore - Object-oriented system design with UML

Large-scale structuring

- A class represents a single logical entity of a system.
- Classes are not usually deployed individually, but in **packages** of closely-related classes.
- In addition, the system may depend on other **physical** components, such as a database, third-party libraries, or physical devices.
- These may be centralised, or more likely, distributed on multiple processing **nodes** connected by communication **channels**.

Justin Templemore - Object-oriented system design with UML

Large-scale structuring

- Large-scale structuring is a design-time activity which shows how the logical classes will be deployed.
- UML provides us with three notation:
 - packages
 - component diagrams
 - deployment diagrams

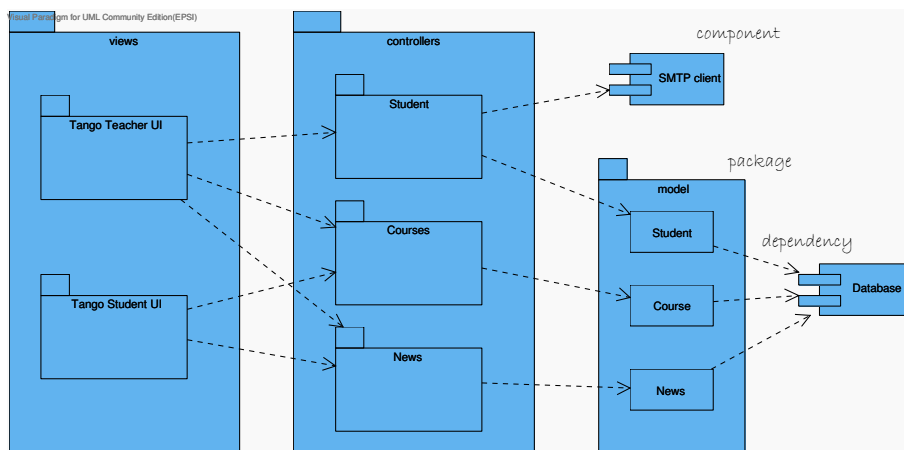
Justin Templemore - Object-oriented system design with UML

Packages

- **Packages** are used to organise the classes in a model.
- It is equivalent to a **folder** or an archive and contains classes and other packages
- There are no formal package organisation rules; a package usually represents an executable program or sub-system, or a library of related classes.
- In Java, a **JAR** (java archive) is the most common package type. This is a convenient mechanism for distributing programs made up of many classes in the compressed GZIP format.

Justin Templemore - Object-oriented system design with UML

Packages example



Justin Templemore - Object-oriented system design with UML

Component diagram

- A component diagram shows the various physical components of a system and their dependencies.
- A component represents a physical block of deployable code, such as a library or executables.
- This is often a single package, but may also be a group of packages distributed together.
- Other hardware and software components of a system such as physical devices, databases and files can also be represented on the diagram.

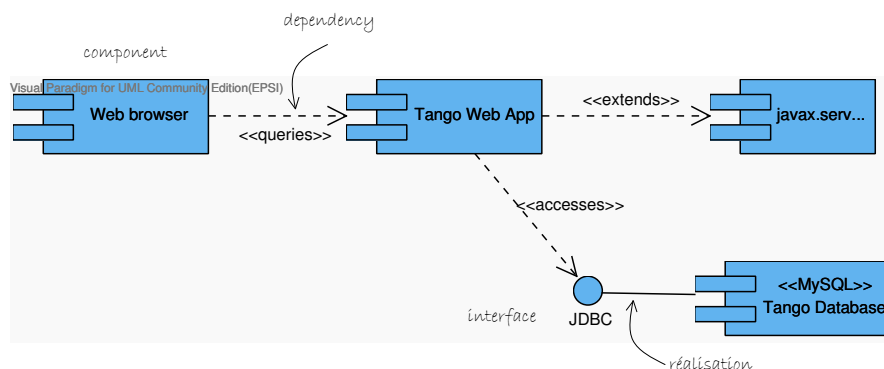
Justin Templemore - Object-oriented system design with UML

Component diagram

- **Dependencies** amongst components show how changes to one component may require other components to change.
- These may be stereotyped to have more meaning:
 - <<compilation>>
 - <<access>>
 - <<communication>>
 - <<requires>>
- **Interfaces** show abstract service interfaces which one component provides, and another depends on.

Justin Templemore - Object-oriented system design with UML

Component diagram example



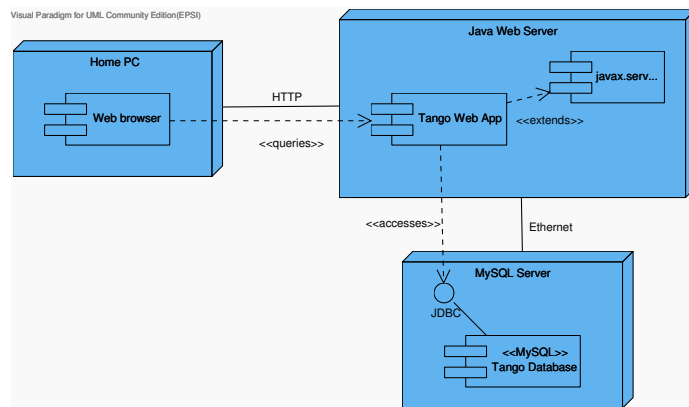
Justin Templemore - Object-oriented system design with UML

Deployment diagram

- A deployment diagram shows system hardware configuration consisting of
 - *nodes* : devices with computing capabilities
 - *communication channels* between nodes
- Very useful for showing the physical architecture of multi-processor and distributed systems.
- Can be combined with component diagrams to show the distribution of components to different nodes.

Justin Templemore - Object-oriented system design with UML

Combined component and deployment diagram example



Justin Templemore - Object-oriented system design with UML

Another example

