**Component Diagrams**

StartUML

<https://docs.staruml.io/working-with-uml-diagrams/component-diagram>

<https://staruml.readthedocs.io/en/latest/modeling-with-uml/working-with-component-diagram.html>

1. *Component-based development* is an approach where a *system* is broken down into *subsystems*, which are broken down further into *components*. Sometimes the term *module* is used synonymously for either subsystem or component.
2. Example – UML Component Diagram:



Source: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/>

*Another Component* provides an API of services, the *Provided Interface*. The *Provided Interface* must be implemented by a class in *Another Component*. Sometimes, it can be a [*façade*](https://en.wikipedia.org/wiki/Facade_pattern#:~:text=The%20facade%20pattern%20(also%20spelled,complex%20underlying%20or%20structural%20code.)*,* which is simply a class that simplifies access to a set of other classes.

1. Interfaces can be expressed in 3 ways:





Source: <http://www.technologyuk.net/software-development/uml/component-diagrams.shtml>

1. A white-box view of a component diagram shows the classes that realize the component’s provided interfaces and can be expressed in several ways:

|  |  |
| --- | --- |
| List the classifiers that realise the component's provided interfaces | A structure diagram showing component classifiers and dependencies |



Source: <http://www.technologyuk.net/software-development/uml/component-diagrams.shtml>

1. Example – A Customer Repository component provides a Customer Lookup interface which is required by an Order System component. An Inventory System component provides a Product Accessor interface which is also required by the Order System component. Draw the corresponding component diagram.



Source: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/>

1. Example – Components can be nested. Sometimes the outer component is called a *subsystem.*



Source: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/>

1. UML Component Diagram:

“…the component diagram allows a senior developer to group classes together based on common purpose so that the developer and others can look at a software development project at a high level.”

Source: <https://www.lucidchart.com/pages/uml-component-diagram>

“…use UML 2 component diagrams as an architecture-level artifact, either to model the business software architecture, the technical software architecture, or more often than not both of these architectural aspects.

“Component diagrams are particularly useful with larger teams. Your initial architectural modeling efforts during cycle 0 should focus on identifying the initial architectural landscape for your system. UML component diagrams are great for doing this as they enable you to model the high-level software components, and more importantly the interfaces to those components. Once the interfaces are defined, and agreed to by your team, it makes it much easier to organize the development effort between subteams. You will discover the need to evolve the interfaces to reflect new requirements or changes to your design as your project progresses, changes that need to be negotiated between the subteams and then implemented appropriately.”

Source: <http://agilemodeling.com/artifacts/componentDiagram.htm>

1. What exactly are component diagrams? The UML specification are somewhat vague (by design, I understand) and so you will see slightly different usages. We will consider a component as a grouping of classes (and interfaces) into packages.

What exactly are interfaces? In a general sense it is the public API that the component exposes. More specifically, they are Java interfaces and classes that define the public API.

1. Example – [3 Layer/Tier Architecture](https://en.wikipedia.org/wiki/Multitier_architecture#Three-tier_architecture): the presentation layer, the business logic layer, and the data layer. Usually you design the layers so that higher layers require services from the immediately lower layer; conversely, lower levels don’t require services of higher layers.



Source: <http://agilemodeling.com/artifacts/componentDiagram.htm>

1. Example – JUnit 5 Dependency diagram:



Source: <https://junit.org/junit5/docs/current/user-guide/#dependency-diagram>

1. Example – Insurance Policy Administration:

<https://upload.wikimedia.org/wikipedia/commons/b/b8/Policy_Admin_Component_Diagram.PNG>

1. In large systems, components are organized into subsystems, which are really just components themselves. In the example below there are three subsystems:



Source: <http://www.uml-diagrams.org/examples/online-shopping-uml-component-diagram-example.html?context=cmp-examples>

**Homework**

1. Draw a component diagram for this situation: A hotel reservation system is being architected. It is decided that there will be a reservation component that provides a reservation services interface. There is also an accounting component that provides an invoice interface which is required by the reservation component. There is also a customer component that provides two interfaces: access customers and manage customers. The access customers interface is required by the reservation component.