UML
Component Diagram

→ **componente**:
- →“scatola nera” che offre interfacce, comportamento interno non noto
- → rimpiazzabili e componibili
- → **entità logica** implementata tramite artefatti

→ sviluppo basato sui componenti: selezione, composizione componenti (*glue code*)

→ modellazione architettura software (*high level design*) di un sistema

→ **notazione**:
- → **componente**: rettangolo con parola chiave `<<component>>` e nome
- → **interfaccia**: lollipop se fornita; socket se richiesta
  - → assemblamento interfaccia nel punto di collegamento
- → **dipendenza**: freccina tratteggiata (“`global`” se componente usata da tutti)

Deployment Diagram
→ strettamente collegato al CD ma vista molto più implementativa
→ modellazione di relazioni tra parti hardware e software di un sistema

→ **notazione**:
- → **nodi**: rettangoli 2.5D
  - → tipo di risorsa computazionale: periferica fisica (*device*), ambiente d’esecuzione software (browser, VM, docker, …), …
- → **connessioni**: linee
  - → canali di comunicazione per informazioni (TCP/IP, …)
- → **artefatti**: rettangoli
  - → entità concrete (file, script, pagine HTML, …)
  - → deployate sui nodi
  - → **dipendenza** tra artefatti: freccia tratteggiata
- → **relazione** `<<manifest>>`: rappresentazioni fisiche delle componenti nei CD

Package Diagram
→ raggruppamento di elementi UML
→ package: definisce un namespace
→ descrizione packages e dipendenze
- → **dipendenza**: all’interno del package esiste (almeno) una classe che dipende da una classe in un altro package

→ suddivisione in packages molto complessa (principî di buona programmazione)