- Software architecture is the set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both.
- Software architecture → software design
- Architecture details is the details that you include in your architecture model to reason failures
- Design details is the full design of the system includes other details necessary to build the system, but not needed to reason failures
- Architecture is design, but not all design is architecture
- Architecture consists of architecture design decisions, and all others are non-architectural
- Architects pay more attention to qualities that arise from architecture choices
- Because <quality attribute 1> is more important <quality attribute 2> for this system, we choose <technical (design/architecture) option>, accepting <drawback>
- problem → System design → existing system
- System analysis (requirements model):

(1) Non functional requirements and constraints
(2) Use case model (verbs)
(3) object model (nouns)
(4) Sequence diagram

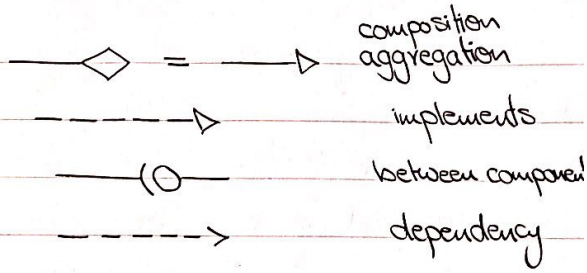
- System design:

(1) Design goals (derived from non-function requirements)
(2) Software architecture
(3) boundary use case

- System design (eight issues):
  1. Identify design goals [non functional requirments]
  2. Subsystem decomposition [functional model] – coherence & coupling
  3. Identify concurrency [Dynamic model / sequence diagram] – parallelism
  4. Hardware/software mapping [object model / class diagram] – Buy vs Build
  5. Persistent data management [object model / class diagram] – install the exe on how many
  6. Global resource handling [Dynamic model / sequence diagram] – access control
  7. Software control [Dynamic model / sequence diagram]
  8. Boundary conditions [functional model]
- Stakeholders have different design goals:
  - Enduser – developer : portability & good documentation
  - Enduser – client (customer): Runtime & efficiency
  - Enduser – client (customer) – developer : Reliability

Scanned with CamScanner

- Typical design trade off:
  - → functionality ↑ usability ↓
  - → cost ↑ robustness ↑
  - → efficiency ↑ portability ↓
  - → Rapid development ↑ functionality ↓



- System design: analysis model → design model
- Subsystem provide services to other subsystems
- object / classes → subsystems (module) → component
- functional requirements → use case → services
- During system design, subsystem is defined in terms of services
- During object design, subsystem interface is defined in terms of operations
- APIs are defined during implementation
- coherence measures dependency among classes
- coupling measures dependency among subsystems
  - ↳ high coupling: changes to one subsystem will have high impact on the other subsystem
  - low coupling: a change in one subsystem does not affect any other subsystem

⟹ high coherence & low coupling

Scanned with CamScanner