Chapter 33-34 – Architecture

Why architectural analysis is important ?

[1] Reduce the risk of missing sth centrally important in the design of the systems
[2] Avoid applying extensive effort to low priority issues
[3] Help align the product with business goals

When do we start architectural analysis?

In UP, architectural analysis should start even before the first development iteration (i.e. should start in Elaboration phase), as an architectural issues need to be identified and resolved in early development work. Architectural analysis is a high req. priority and failure to do so is a high risk.

Architectural analysis→ is specialization of reqs analysis. It is concerned with the identification & resolution of the sys non-functional requirements (e.g. security), in the context of the functional requirements (e.g. process sales). It includes identifying variation points and the most probable evolution points.

Need to address 2 kinds of protected variations

[1] Variation point: existing current system or requirements
[2] Evolution point: potential and speculative points that may arise in the future

What reqs strongly influence architecture?

[1] Security?
[2] 3rd party systems?
[3] Volatile business rules?

Arch. Analysis cont.→ the term in UP encompasses both architectural investigation (identification) and architectural design (resolution). Examples:

[1] How do reliability & Fault-tolerance failover affect the design?
[2] How does Licensing costs vs performance/ supportability of commercial SW vs. open-source affects profitability?
[3] How do adaptability/configurability affect the deign?
[4] How does brand name and branding affect the architecture?

Steps in arch. Analysis→

[1] ID **architectural drivers or factors** in SS or UCs: Could be any functional or non-functional req. They may not be purely technical in nature
  - Legal -> comply with FURPA
  - Marketing -> Consistent brand image
  - Project resources -> not enough time to code from scratch

[2] Evaluate alternative designs. These are **architectural decisions**. Formulate quality scenarios that define measurable/observable metrics

---

Prioritizing arch. reqs →

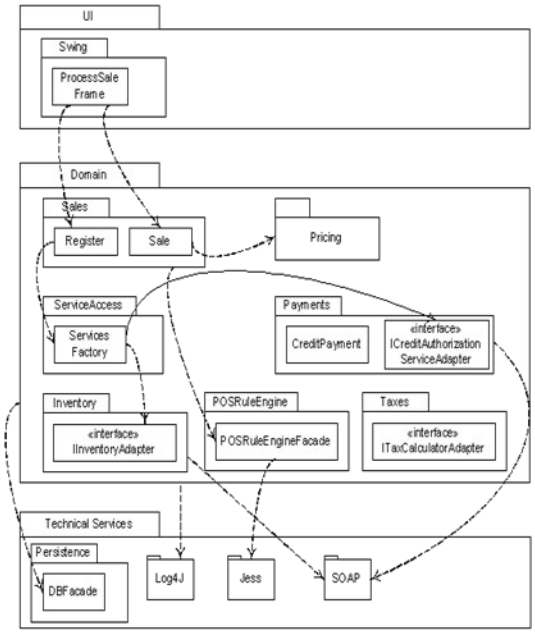
- Keep good records of analysis, alternative designs, motivations, constraints, etc.
- Record 'why' & 'why not'! (people forget)
- Estimate probability of future evolution
- Hierarchy of arch. goals
- Legal: must be done
- Business goals
- All other

Arch. design principles→ Basic GRASP patterns

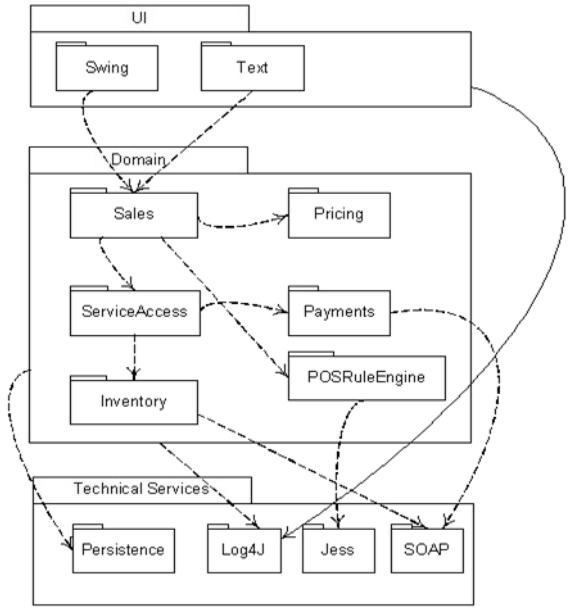
[1] Low coupling
[2] High cohesion
[3] Protected variations

NexGen arch. Model→

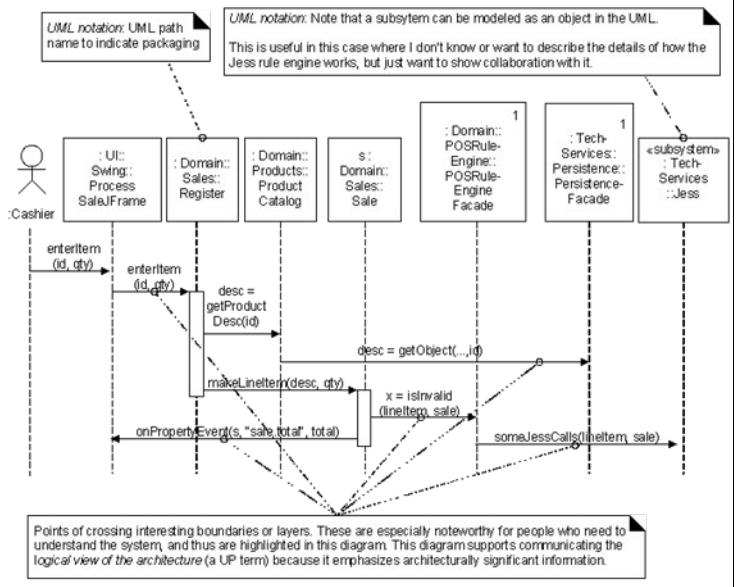
[1] Only key packages/types shown
[2] Basic Java foundation classes not shown
[3] Application layer deferred to later iteration as complexity dictates



NexGen architecture w/ dependencies indicating coupling between packages



Simplified view of coupling



Dynamic view of package coupling using sequence diagram

Layers Pattern

- What are the layers
- How are they connected
- Uses Façade, Controller & Observer patterns for connections between layers and packages
- Façade used to connect subsystems
- Exposes few coarse-grained services
- Consolidates, mediates
- Reduces # of network calls
- Use for 'downward' communication