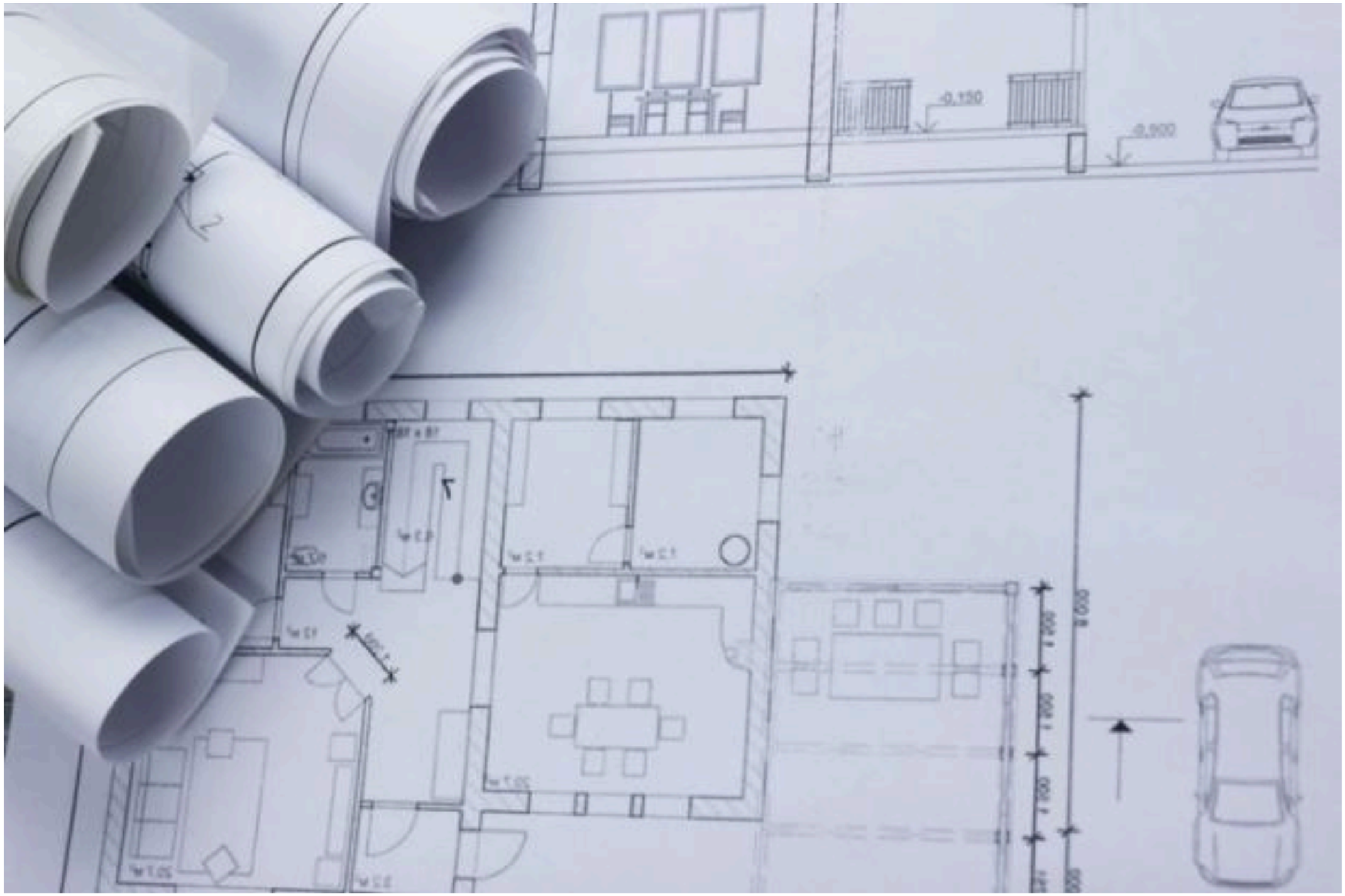


Software Architecture

About me

- skan.ai
- ai.robotics
- Welldoc

Agenda

- Application Architecture Scope/ Role
- Arch Requirements (sei)
- Arch Design (views)
- Arch Doc (views)
- Arch Eval (sei)

- Years of experience
- Technology stack
- Business Domain
- Expectations

Architecture and Design

Quality attribute ?	Approach (tactic/style) ?	Measure ?
• Maintainability	• Modularization • Health monitoring • Config • Documentation • Styling • Automated tested • CI/CD • …	• Cyclomatic complexity • Code coverage • Low Coupling

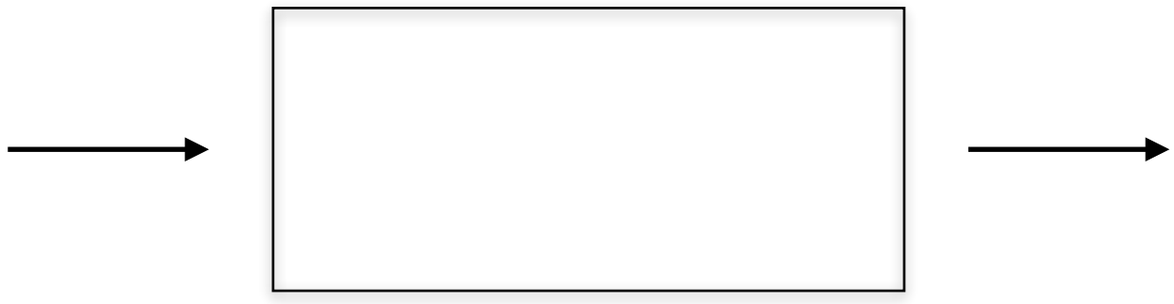
(What) Quality attribute ?	(How) Approach (tactic/style) ?	Measure ?
• Performance	• Caching • Compression • Parallel • Object pooling •	• Latency • Throughput • Response time •

(What) Quality attribute ?	(How) Approach (tactic/style) ?	Measure ?
• Correctness • Maintainability • Performance(cpu, memory, disk, network, …) • Scalability (volume of resource: cpu, memory, disk, network, …) • Availability • Security (trust) • Reliability (trust) • Robustness (rugud) • Usability • Interoprability • Portability	• Caching • Compression • Parallel • Object pooling • Reusability • Authorization • Input validation • Throttling (max) • ACID (transaction) • Testability	• Latency • Throughput • Response time • % of down time • % of uptime • tps • Probability • Mtf, mtbf, pf, … • No of clicks •

Quality Models(McCall model, the Boehm's model, the IEEE, SEI)

(What) Quality attribute ?

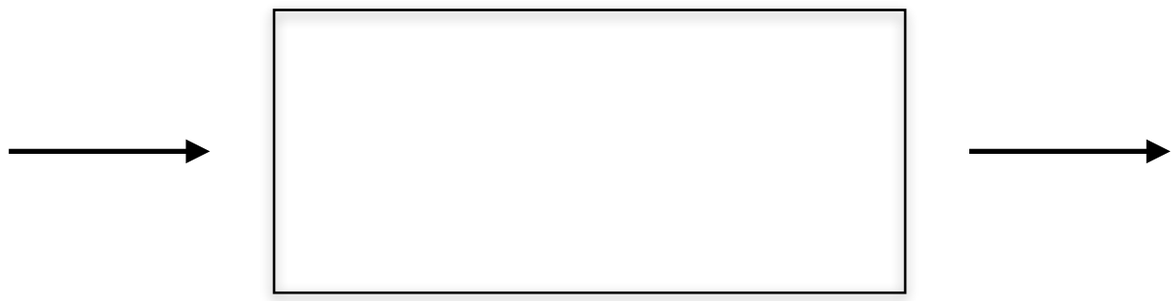
Measure ?



(How) Approach (tactic/style) ?

Knowledge

“Quality requirements”



Architecture

“Approaches”

Implicit Architecture	Explicit Architecture
Performance tuning **(after)**	**Performance engineering** **(before)**
Hacking **(after)**	**Threat Modeling** **(before)**

“functional requirements”

Understand

“Quality requirements”

Collect

Quality engineering

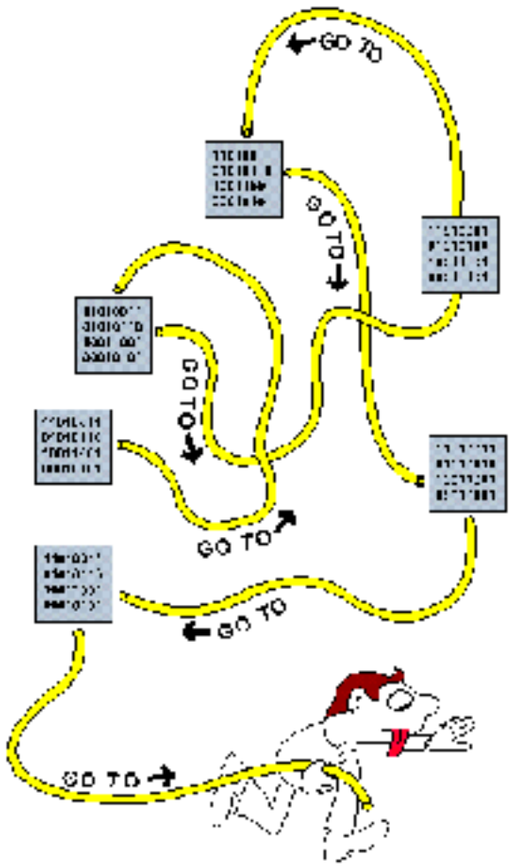
Decide

- Architecture
- High Level Design
- System Design
- Arch Design

“Approaches”

Knows

Styles, arch patterns, tactics,
Arch anti patterns.



“guidelines”

Follow

“fun requirement”

Understand

Manage
Code
Complexity

Create

- Detail Design
- Implementation Design
- Code design
- Low level design

“Skeleton for Code”

Knows

OO patterns, fun pattern,
lang idioms

Togaf, dodaf, zachman fwk

Enterprise Architect

(align)

Product/Solution Architect
(Quality of the product)

Application Architect

(Quality of the application)

Domain Architect

(Quality of the process)

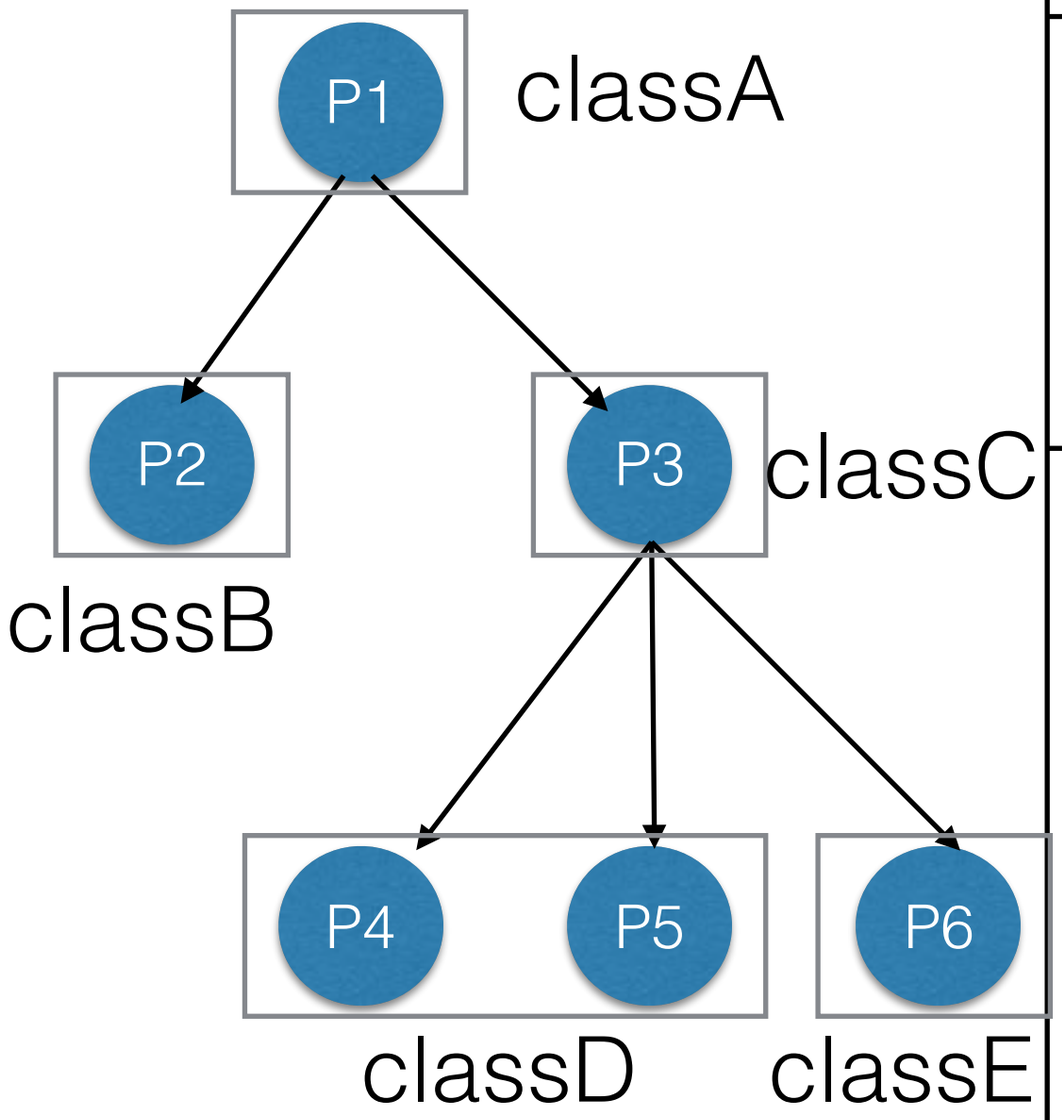
Vertical Architect

Security Architect
UX
Data
Infra
Cloud
Java
…

Proc vs OO vs fun

Procedural Prog

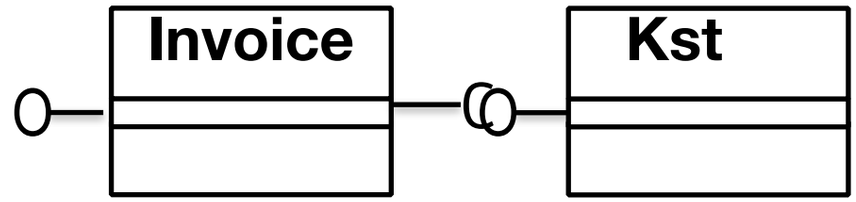
(tree)

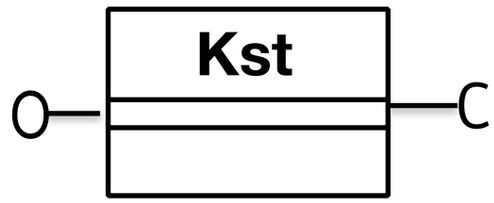


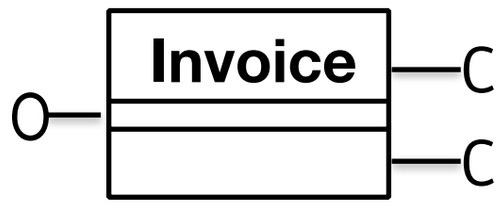
(top down)

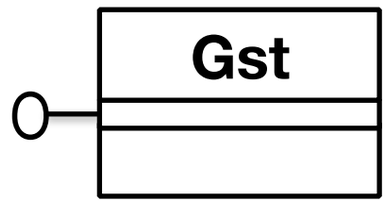
OO Prog

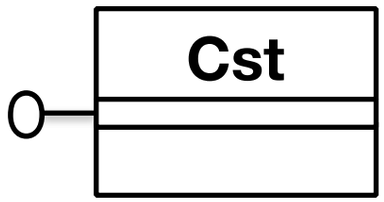
(Lego)





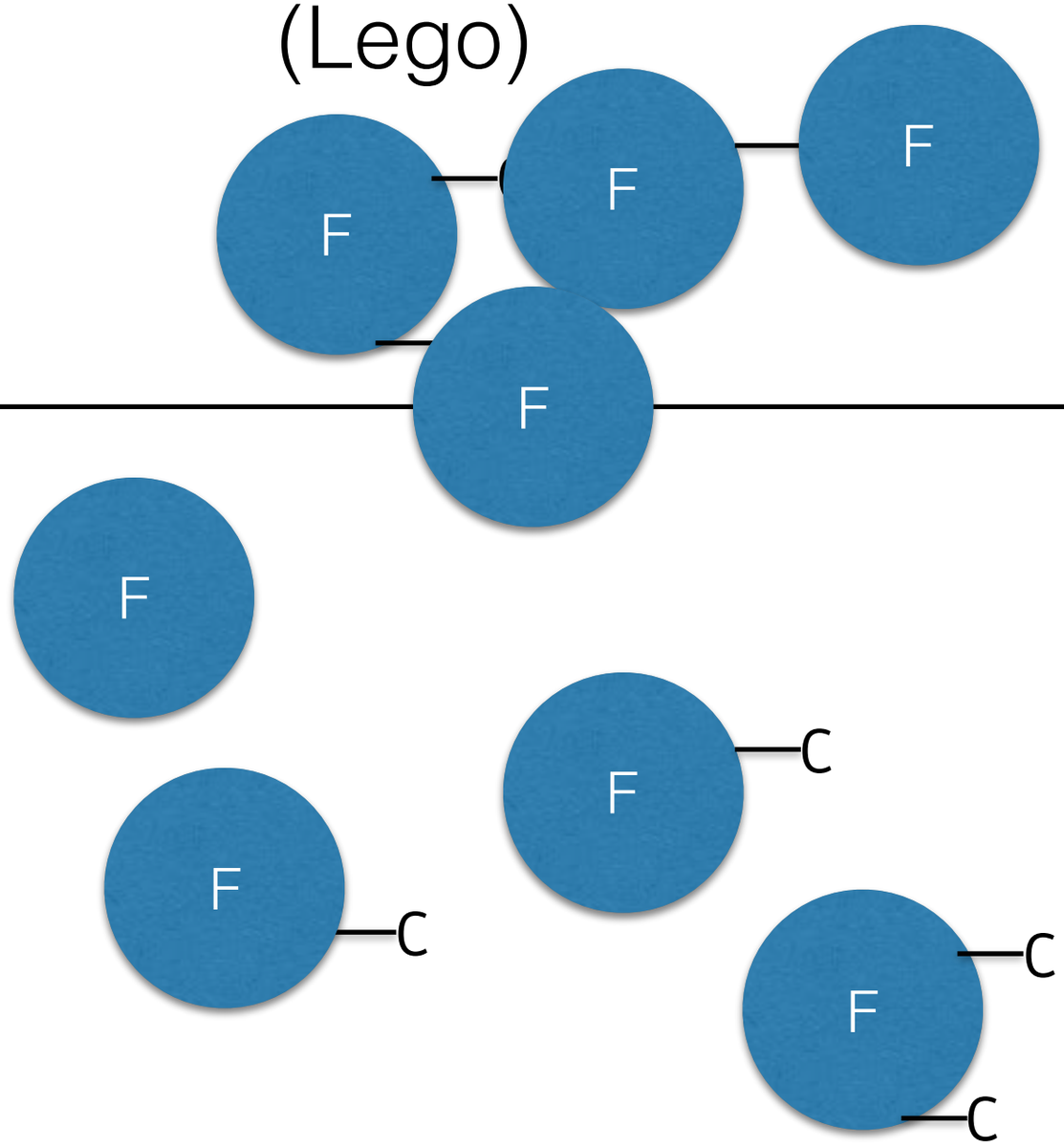




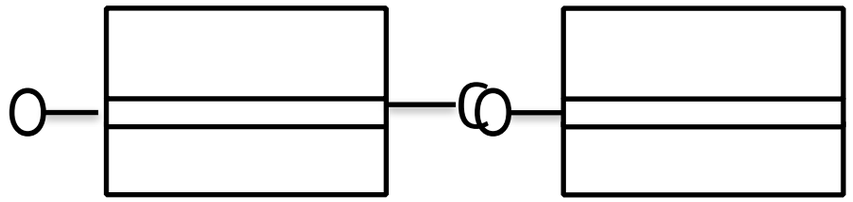


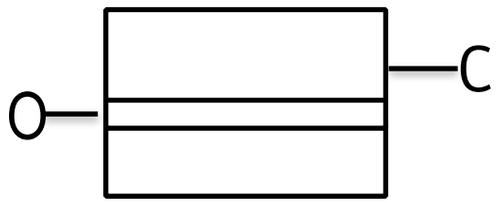
(bottom up)

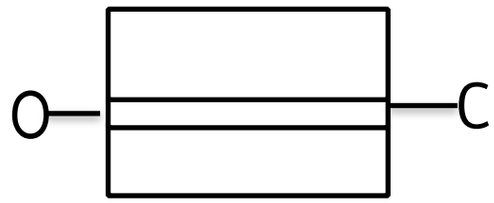
Functional Prog (Lego)

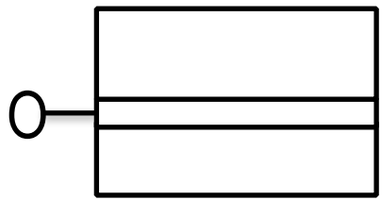


OO Prog (Lego)









	Proc (tree)	OO (lego)	Fun
Lang	C, py, java, C#, JS, c++	Java, C#, C++, py, js	py,js, J8,c#
Constructs	if/switch/goto/ Static methods	Polymorphism/ Exceptions	High order fun/ recursion/ closure
Performance	-	-	+ +
Security	-	-	-
Learning Curve	+ +	- -	-
Development Time	+ +	- -	+
Unit Test	- -	+	+ +
Code Maintainability/ Support Time	- - -	+ +	+

Todo

- 5 most important quality attributes for you domain
- At least 10 approach for each quality
- At least 3 measures for each quality
- Software Architecture in Practice -SEI practices
-

Anti patterns

- Alice in Wonderland

-

patterns

Case study

todo.com

- <todo.com>
- CRUD todo
- Web App
- Single user

GreatDeal.com

- <GreatDeal.com>
- Single product with n qty for a day
- Web App
- Collect payment if stock exist
- Send the product through delivery partner

bidder.com

- <bidder.com>
- Single product (1 qty) for a day
- Web App
- Collect payment from highest bidder
- Send the product through delivery partner

QAW process

- Quality attribute workshop
- Process to collect arch requirements
- Process to collect Quality Attribute scenario(NFR) | user story (FR)

1. Prepare seed Quality Attribute scenarios (NFR)
2. Get all stake holders in to a 1/2 day brainstorming session for NFR.
3. Collect Scenarios
4. Prioritise Scenarios

As a User I want to add a todo In the web portal when 100,000 users are using the portal. The portal displays a success message In < 3 sec time.

Source (who)	As a User
Stimulus (action)	I want to add a todo
Artifact (module)	In the web portal
Environment (context)	when 100,000 users are using the portal.
Response	The portal displays a success message
Measure	In < 3 sec time.

Source (who)	processor
Stimulus (action)	stops working
Artifact (module)	in the “central system”
Environment (context)	during peak traffic hours
Response (output)	start providing “degraded mode” service
Measure	The time spent in degraded mode should be no more than 5 minutes.

App

Piece1

Piece2

Piece3

Architecture

Collect Arch Requirements

1. Context view
2. Functional View
3. Quality View
4. Constraints

Build Arch

1. Logical View

Eval Arch

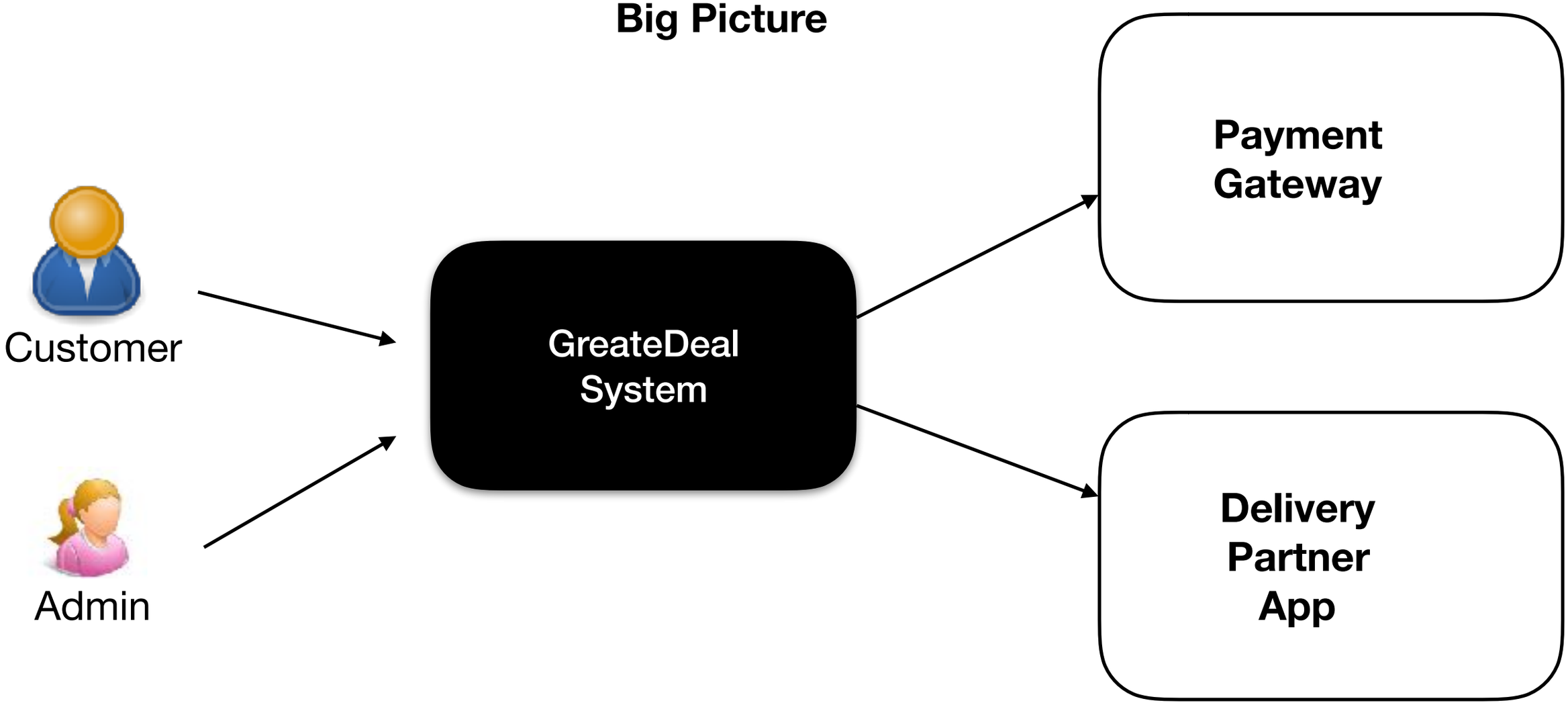
GreatDeal

Arch

Requirements Gathering

Context View

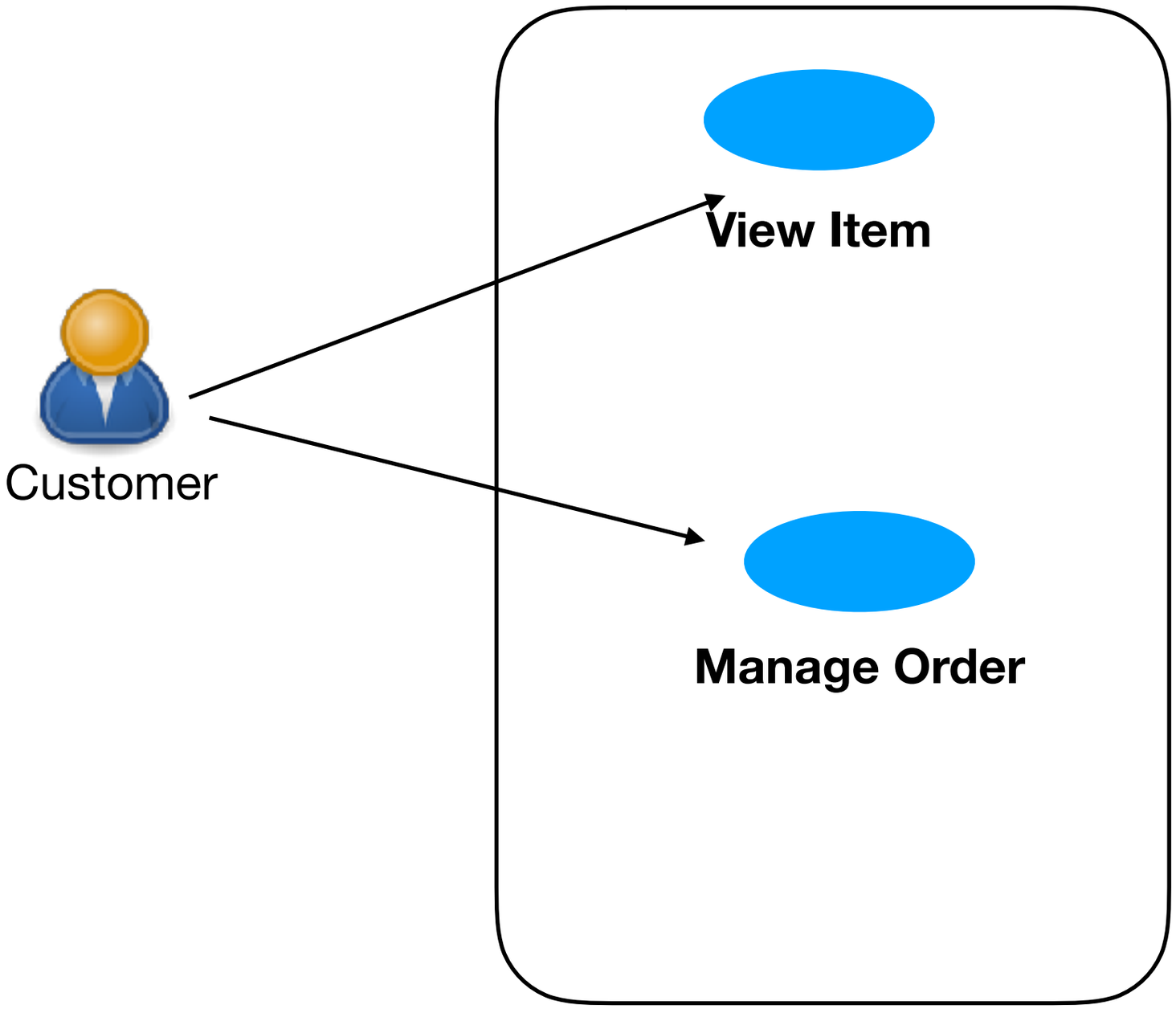
Black Box View
Big Picture



- Does it set the scene ?
- What is it that's being built?
- How does it fit into the surrounding environment ?
- Does it show relationship with the existing System ?

Functional View

Key functionality of the system



- 80:20 rule (20% is important)
- Does it Identifies key users ?
- Does it identify the architecturally significant use cases ?
  - **Business Critical**. The use case has a high usage level or is particularly important to users or other stakeholders when compared to other features, or it implies high risk.
  - **High Impact**. The use case intersects with both functionality and quality attributes, or represents a crosscutting concern that has an end-to-end impact across the layer and tiers of your application. An example might be a Create, Read, Update, Delete (CRUD) operation that is security-sensitive.
  - Include a summary to highlight why are they architecturally significant.

Quality View

- As a User I want to view the Deal of the Day when 100,000 users are using the portal. The portal displays the Item In < 1 sec time. (performance)

- When a user places an order, the payment fails in the server during peak hours and the order is cancelled and money is refunded within 2 hours. (reliability)

- When a user enters incorrect bidding value into the bidding Web App while product information is displayed. The system prints an error message for the respective user. User is able to bid again with correct value within 30 seconds. (robustness)

Constraints View

- Should support Internet Explorer 11
- Use open source stack
- API should be built using python

GreatDeal Architecture

Logical View

White Box View