Constructing EA

Activity — convert design to code

① achieve construction Readiness
② Construct sol layers - app framework and appli comp
③ Code Review & static code analy
④ compilation & creating deploy packages
⑤ unit testing
⑥ code profiling and code coverage analysis.

Construction Readiness

before construction

① Define constru plan

- seq and schedule of tasks
- inter dependencies
- tech risk.
- resources Req

Objective: complet constru in short time

ii to Coding: test case prepa

- setting up test case env
- planning Review activities
- planning version control
- releasing planning.

② Defining package Structure

Code units - have to structured in a meaning ful & logical way

Based on: loosely coupling tight cohesion, reusability use of third party lib-

③ setting up a Config management plan

- versions has to be tracked to ensure integrity and consistency of artifacts.
This is called config. manag
- Artifacts that are managed are config.obj

plan provides

- Specify the tool to manage. config obj
- repository structure
- access rights - diff us

④ Setting up Dev Env

- Provide facility to develop, build, review, analyze verify the code - during development

elements need to setup

① H/w incl. OS
② server ③ integrated dev enviroment ④ third party lib.
⑤ building tools, unit testing tool, profiling tool, static code analy tool, - Licenses

Installation & config servers
file server, db server, appli server.

Setting up IDE (Intg, Deve, Env)

IDE is configured with relevant lib, plug-ins, dlls and sever config.

Setting up frameworks
- build on top of EA

appli framework
import & config the lib of framework.

2. S/w Construction map

- developer get design, UML seq diag, class diag to understand design

↓

translate to code

Developer while coding may include

① How - overall pict
② What - Relationship blw components
③ How - components interact outside layer
④ How - view flow of function

need to represent design elements in diff form

↓

better understanding of components, interations visualize overall solution

(S/w Construction Map

Construction Map

↓

Coupled with existing design

Provide

① Visualizing big pict of solution,
② inter and intra relationship among various components across layers.

Construction Solution Layers

- not ~~for~~ from scratch
- reusable assets repository and frameworks

Reusable components

① Appln framework → provide code across all layers
② repository of tech components
③ repository of domain specific components ; like

- check credit rating component
- quote generation component

workflow component
rules engine
generic reporting components

S/w Construction Map

↓

Coupled with existing design

Provide

(1) Visualizing big pict of solution,
(2) inter and intra relationship among various components across layers.

3. Construction Solution Layers

- not ~~for~~ from scratch
- reusable assets repository and frameworks

Reusable Components

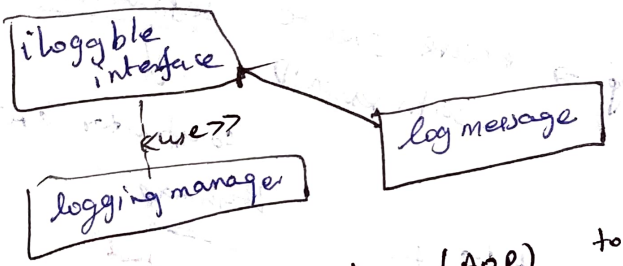
(1) Appli framework → provide code across all layers
(2) repository of tech components
(3) repository of domain specific components ; like

- check credit rating component
- quote generation component
- workflow component
- rules engine
- generic reporting components

Infrastructure Service Layer Components

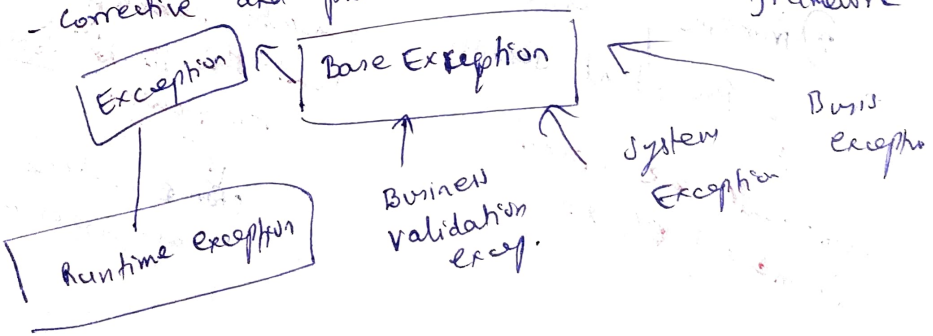
- logging, session management, security, exception handling, auditing, caching
  - constructed as part of application framework
  - all in utility folder

logging
- used all solution layers.
- use third party package Log4



use aspect oriented programing (AOP) to achieve better modularity

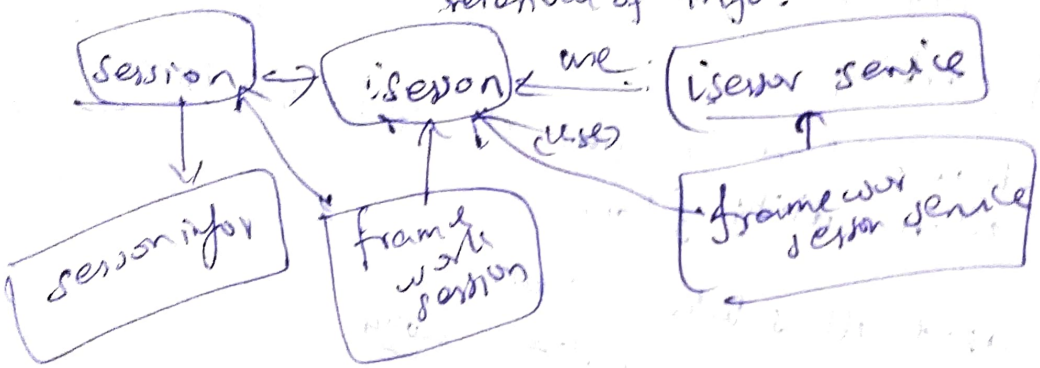
Exception handling
- corrective and preventive measure, part of appln framework



Session management

inmemory storage tech →
session ID, HTTPSession.

session class → creation of session, add/Remov the info, detail from session, retarival of info.



session obj ⇒ ensure create with sessionID ; sessionfo obj

constructor create

Hashmap

(session related)

session management feature

① session affinity → all req sent to particular node of cluster env. - use caching

② session tracking

③ inmemory session & Presistent session → use cookies, appli server generate sessionID and sent this ID in a cookies to brower

session attr are stored in memory (or) db

store cookies with all session activity

caching

tree cache, each node as a map for storing key-value pair

ehcache ↔ JBosscacheserver

ehcache methods: it have get, put and remove obj from cache

lazy loading
caching modes { locking level, trans. iso. level.

used to declare caching is req

Cache

- clientside – N/w bandwidth, same data locally or across the cluster.
- serverside
  - multiuser – same data.

Presentation Layer Components

- Framework & ted
- Package stru
- Appli framework & componet, Appli com

- using JSP, servelet, POJO (Plain ~~obj~~ old Java obj)
- developer use Frameworks → Struts, JSF, Tapestry, stripes
  - ↓ select
    1. right design patter
    2. right look, input validation, max. reuse, session management, internationalization, and caching of data.

Frame work and technologies

selecting right framework → ensure performance, maintainability, extensibility, bring std. and struct

during ~~design~~ archi phase. select technical phase laying down arch.

Selecting framework

(1) internationalization support
(5) navigation support, req processing
(2) data validation, testing support

other factors
- framework documentation
maturity of framework
usability
QoS consi., avail workforce etc

Features	Struts	JSF
Acceptance	mature & popular not part of JEE specification	part of JEE
working	action driven model	event driven model
support	mature & stable many IDEs	new, but gain maturity rapidly
user interface	Does not std UI model	Based on Std UI model
client supp	tags generate only HTML	XML based client also, non html
development	based on custom tag lib. does not support reusable components	based on std UI comp reusable Comp.

② Package structure

- Appli specific component are created on top of framework componets.

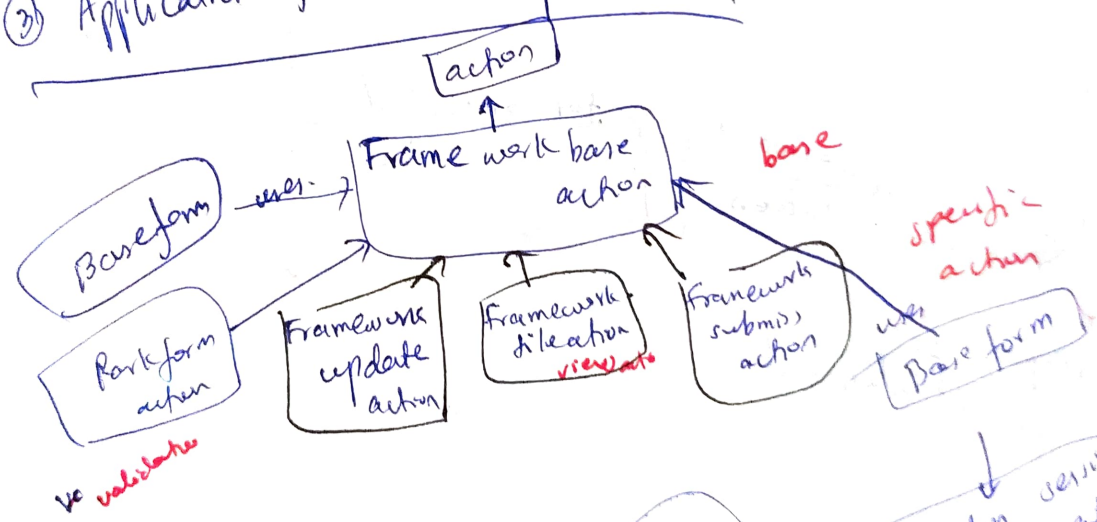
reusable

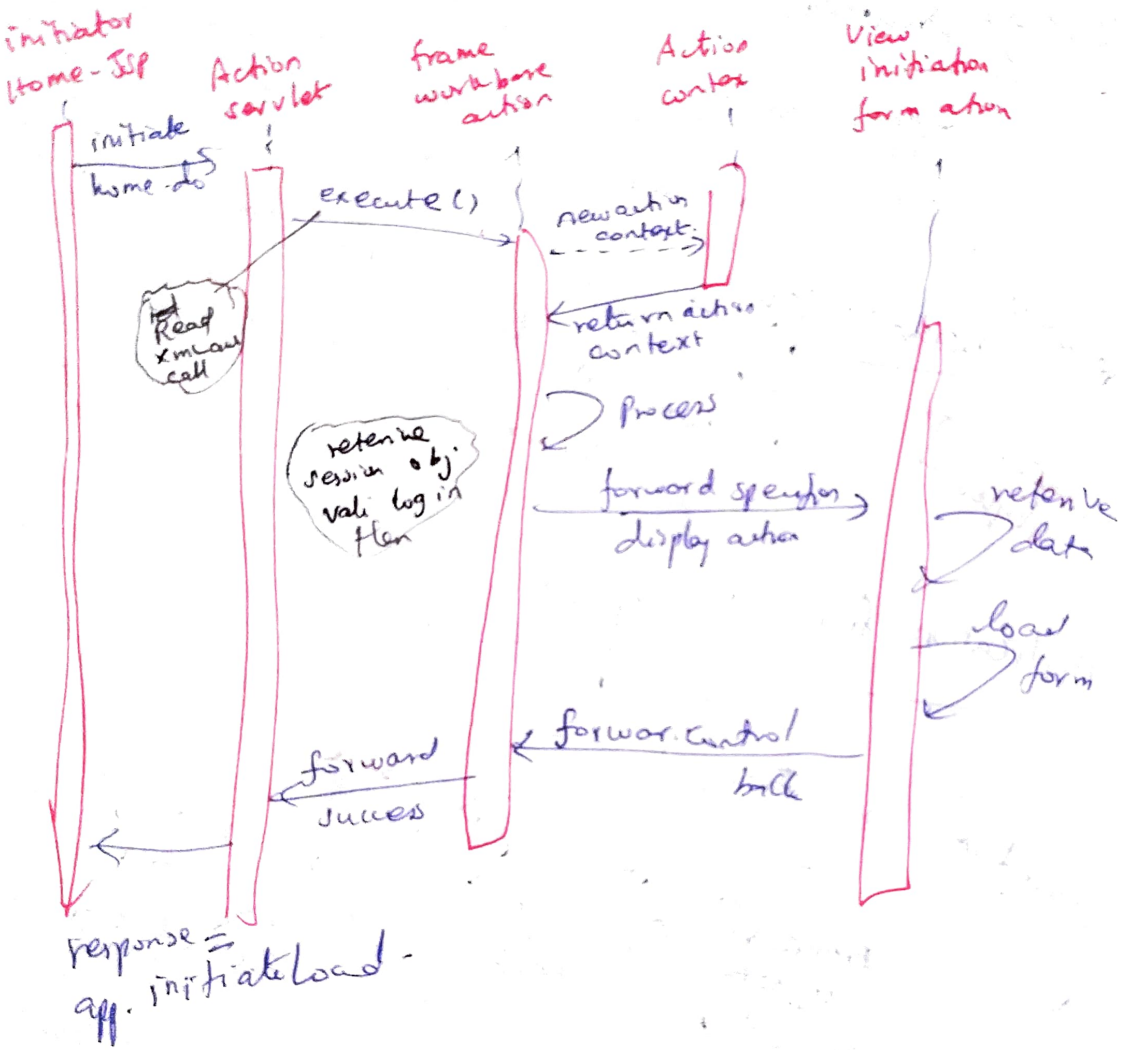
Presentation layer
- web content
- source code for presentation layer

static images
stylesheet

JSP → related to Appli framework

③ Application framework components





Softwar Construction Map ⟹ ① Validate SSN

② interation session & caching

③ logger

Business Layer Components

using POJO, extend readily available frameworks

design team consider

1. apply right design pattern
2. design optimized transc management
3. Concurrency control manage
4. business workflow, rule repositories

1. framework and tech.

- easy to use, out of box capabilities.

Spring framework
- opensource
- several sub frameworks
- provide services across layer

1. Provide rich UI support.
2. Programatic and declarative trans. management
3. Suppor all data access framewrk - JDO, JPA, Toplink
4. Support batch processing & remoting

use spring batch

RMI, RMI-IIOP, SOAP.

Two feature

① IoC (Invention of Control
(application framework call application code

② AOP (Aspect oriented prog.)

Dependency injection

↓ config file ⇒ Contain dependencies among obj

Spring framework IoC ⇒ container use this and injects the ref obj on demand

AOP : provide Cross cutting Concern of EA.

EJB 3.0

- Core API of JE
- provide scalable, secure, life cycle mang

Feature	EJB 3.0	Spring framework
Acceptance	Java EE std	opensource
Trans support	use EJB Container	using Spring AOP
state manag	Stateful session EJB	non-singleton (Prototype beans)
dependency injection	primitive support dep inj	exhaustive support several type of dep inj

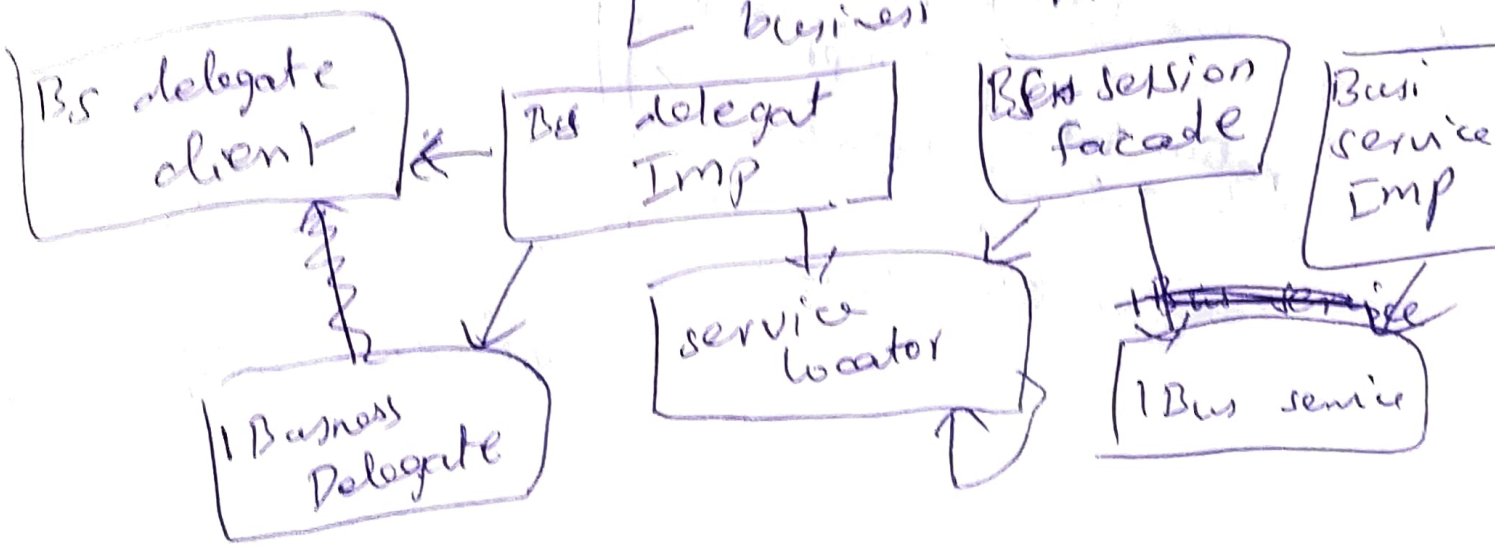
Package structure

- Cust app framework
- foundation of busin layer components
- Appli components create on top of these

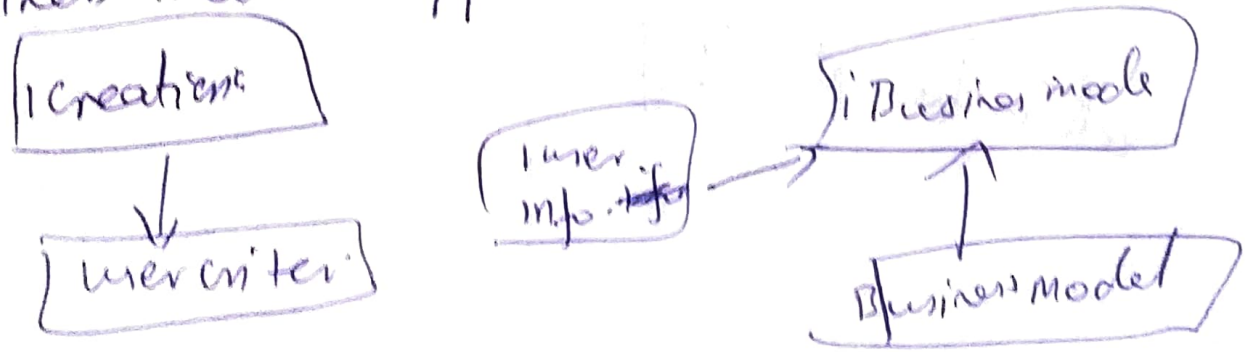
Application framework components:

business layer framework

- Business delegate compon
- session facade "
- business service "
- business model "



business model app framework components.



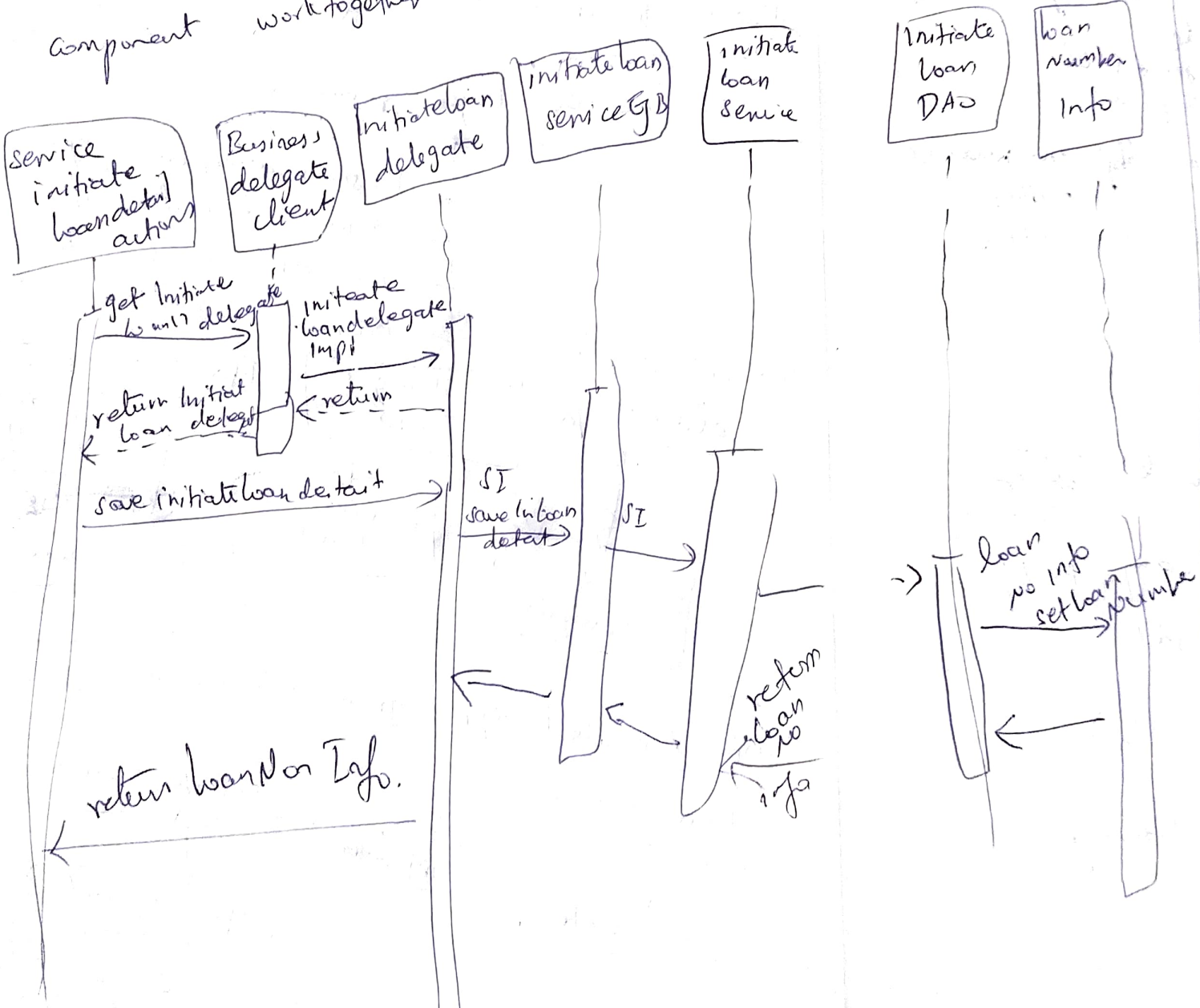
ep
inject

Application component

⇒ bis layer ⇒ core of ~~app~~ appln

Initiate loan

how business layer application component work together



Data Access layer Component

* decoupling @ relational table (or other sources)

Frameworks and technologies

JDBC => SQ queries directly embedded in Java code

need coupling b/w database & obj

* ORM (Object Relation mapping)

~~tightly coupled Java obj and relational data store~~

To avoid that

ORM

- bridge / mapping } b/w Java obj & data store

↓ through xml config (or) annotations

using Hibernate → framework

- open source ORM framework (using object concept)

it support Hibernat Query Lang (HQL)

↓ extension of SQL

~~JPA (Java persistent API)~~ (JPAL)

Part of JEE 5

Java persistent Query Language

difference b/w JPA and other framework.

	JPA	Other
Acceptance	Part of JEES platfor	Hibernate, iBatis and oracel toplink.
Control on queries	limited, generate automatically	strong control, map Java obj and queries.
Applicability	JPA is prefered in appli Control on queries not needed	Framework ~~iBatis~~ are used in db ~~base~~ centric appli. tight control on queies.

Best pratices on data access layer

identifying call back —
- each entity do some manipulation ~~in~~ during life cycle of EA
- ex ~~tat~~ feetch phone No Relace - by [ ], ~~do~~ this is done by call back

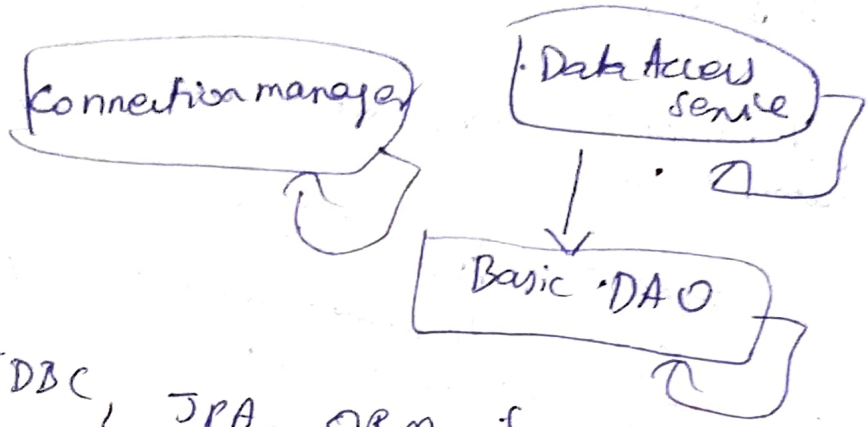
identifying embedded object
- is to improve reusability
addr. => comm, addres, loan gurantor's addr.
cust permanet adders.

Packagestru

DAO (spring DAO) used to design framework & appli componet.

Application framework components

In .Lmos use use POJO



use JDBC, JPA, ORM framework

=> encapsulate a service that integrates with another appli to fetch data

use RMI

Integration layer Components

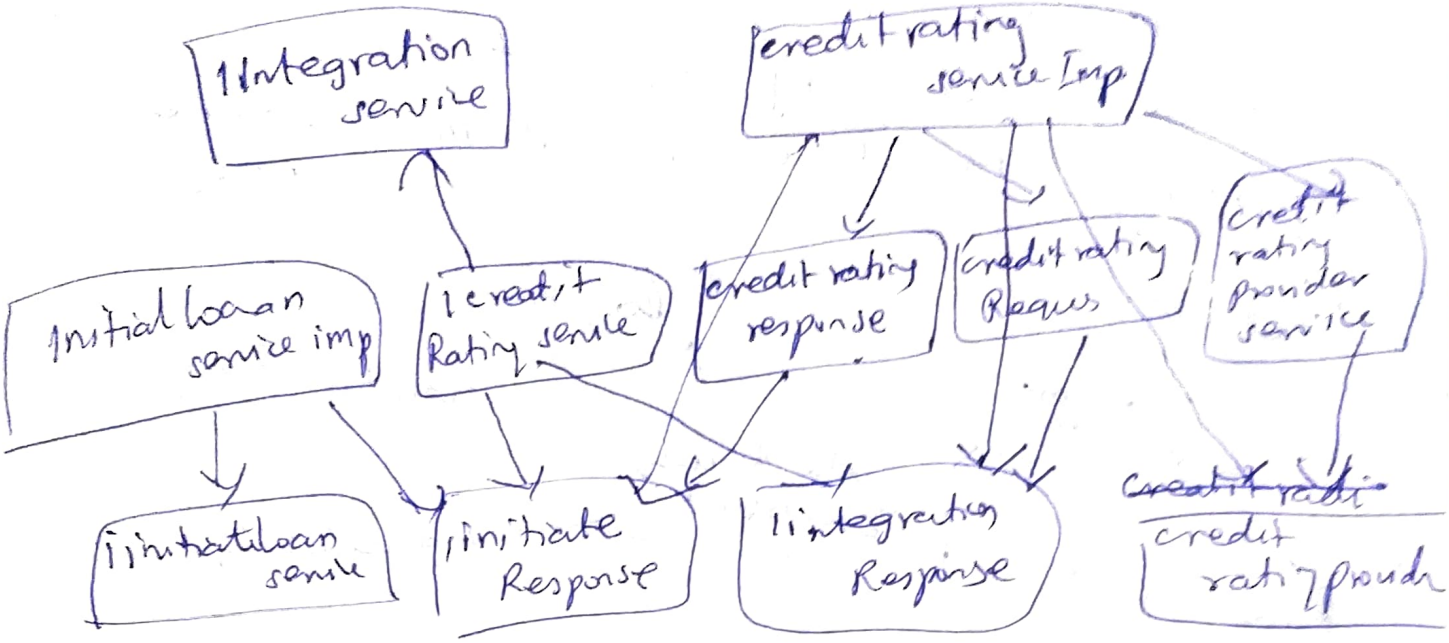
Loms use ~~ASSA~~ Jax-WS API to implement SOAP based web services integration b/w Loms and credit rating system.

use POJO interface

automaly generate soap request whenever require

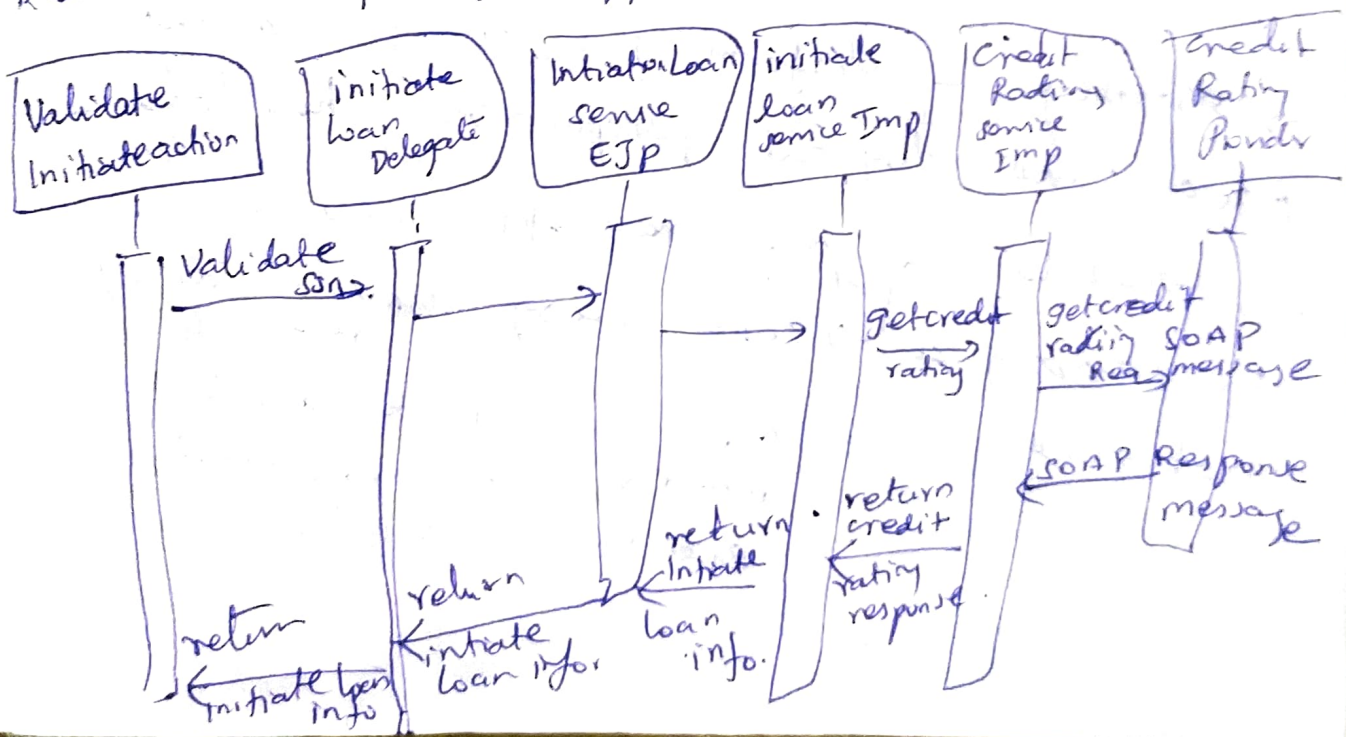
to communicate with provider

Application framework and components



Application Components

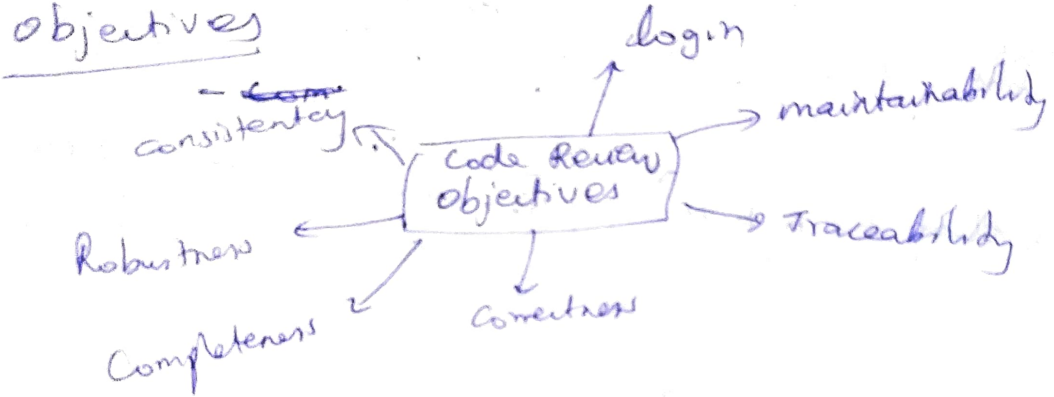
Integration ~~to~~ layer components are build on top of appli framework components.



Code Review

- measure degree of compliance with Specification
- verify quality of artifact
- Primary target of review in construction space

objectives



Completeness — check meet all key req
Correctness — avoiding hard coding, eliminate unused variables

Consistency — Uniformity in coding, comment

Review from logical perspective — how behave code

maintainability
Traceability ⟶ any req fun. missing
code easy understandable/ readable

Robustness ⟹ code is able to handle errors & unexpected events at runtime

Process

⇒ identify right set of reviewers

⇒ give time to perform

↓

use checklist & review guidence.

Reviewer checklist

1. naming conventions
2. Computational errors
3. Comparison errors
4. Control flow errors
5. ∞ (or) improperly termited loops
6. Interface errors
7. I/O errors
8. unused Variable
9. wrong initialization & default Value
10. Data ref error
11. Data declarative errors
12. unhandled exception (or) error condi:
13. error messages
14. help messages
15. hard coding
16. contention & deedlocks
17. inline document
18. Realtification (or) intentation of code.

Static code Analysis

=> analyse source code

Static code Analysis

focus on

- coding style
- logical bugs
- security Vulnerability
- code quality

- automated activity
- finding tools

coding style

- right type DS
- adopt control constructs
- readability

Report - inconsistencies on prefonfig rules for style specification

naming id
improper use of whitespace

- Rules related to style
- very begining of construction phase

easy (tool)

Logical bugs

- manual code review
- encoded rules verified

identify ⇒ not using stringbuffer (concatinating)
unhandled potential null ptr exception.

security Vulnerabilities

① Cross site scripting
- attacker pass malicious scripts.

② SQl injection
attacker cheng the logic of sql query.

③ improper session handing
- session timeout (check)

④ authentication / authorization.

⑤ inefficient exception management

6) sensitive data not persent in ~~code~~ code

Code quality

① class size ⇒ not more than 1000 lines of code

② Cyclomatic complexity
Check No. of linearly independent paths
↓
high ⇒ more testing
loop should be min, unnecessary condition avoided
nested loop - avoided.

③ comments to ~~to~~ code ration.
lower comment ⇒ difficult to understand

④ No. of attributes -

⑤ coubling b/w objects other classes coubled.

Build process & Unit testing

~~Build process~~

Unit test case. ⇒ ensure health of increment @ construction phase

Continuous integration & unit test case best practices

Build process

Job
- * accurate, maintainable, automated build process
- * Compilation of source code
- ~~* relative code unit from source code~~
- * Packaging appli components & lib
- * component specific compilation
- * deploying the application

* manual
  - confusion, error
  - take time

automatic
  - saving time
  - improve quality by standardization

Tool : Apache ANT
Maven

ANT

- using XML
- final deployment unit, JEE Appli
  use JAR, WAR, *

build scripts : dependencies among appli components
built process are taken care
* followed by testcases and deploying appli
* includes documentation activities

ANT => run automated unit test cases.
=> min. human intervention.

Unit testing

* test => unit of code
  ↳ module
  procedure
  classes.

*dev Resposibility

using JUnit tool

* easy find bugs easily.

unit testing presentation components.
Presence of all fields.
correct translation
localization.

business & integration
control flow
looping constructs.
total coverage

- test +ve and -ve cases
error and exception cases

3 primary steps =>

① Test planning

- end of req engineering phase
- create test plan (document test steps.)
  - what
  - who is responsi
  - what pre-req
  - assumption.
  - which criteria pass P/F fail test.

② Test cases and test data preparation

=> test case doc should be tested
- check control flow, appli logic

Description.

test case elements

test case description
input test case
expected result
actual result
pass / fail
iteration No.

* test data => selected carefully