Documenting Software Architecture

Explanation of the software structure

The architectural principles adopted and constraints in force

Development and deployment

technologies and platforms

A justification of how the

architecture

satisfies the

requirements

There are many different

stakeholders

Development Team

Business Sponsors

QA Team

Database Administrators

System

Support Staff

Other Teams

Security Team

Compliance Audit Team

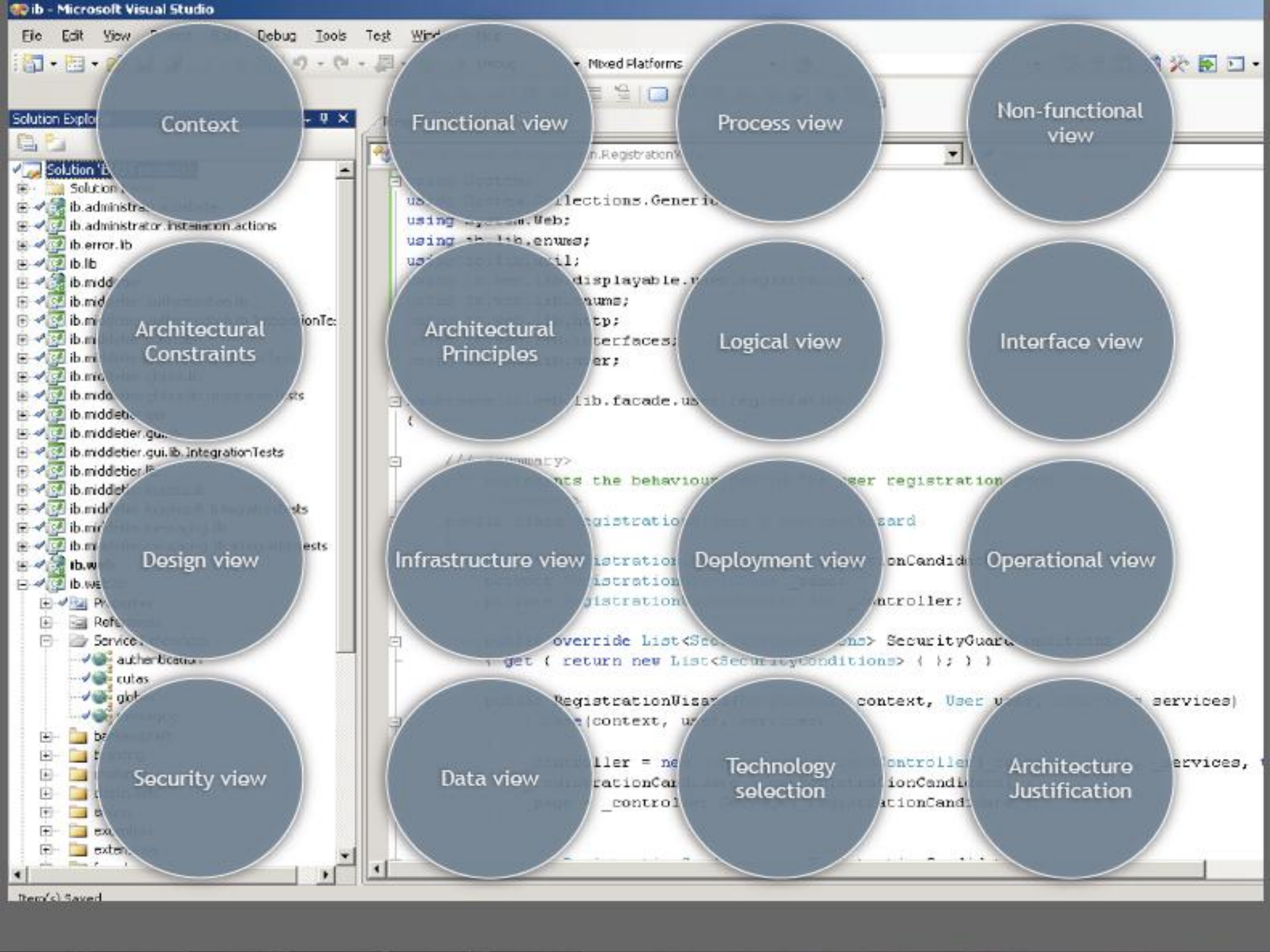
Each stakeholder has

different needs

of your software architecture

The description of your software architecture needs to

take these views into account

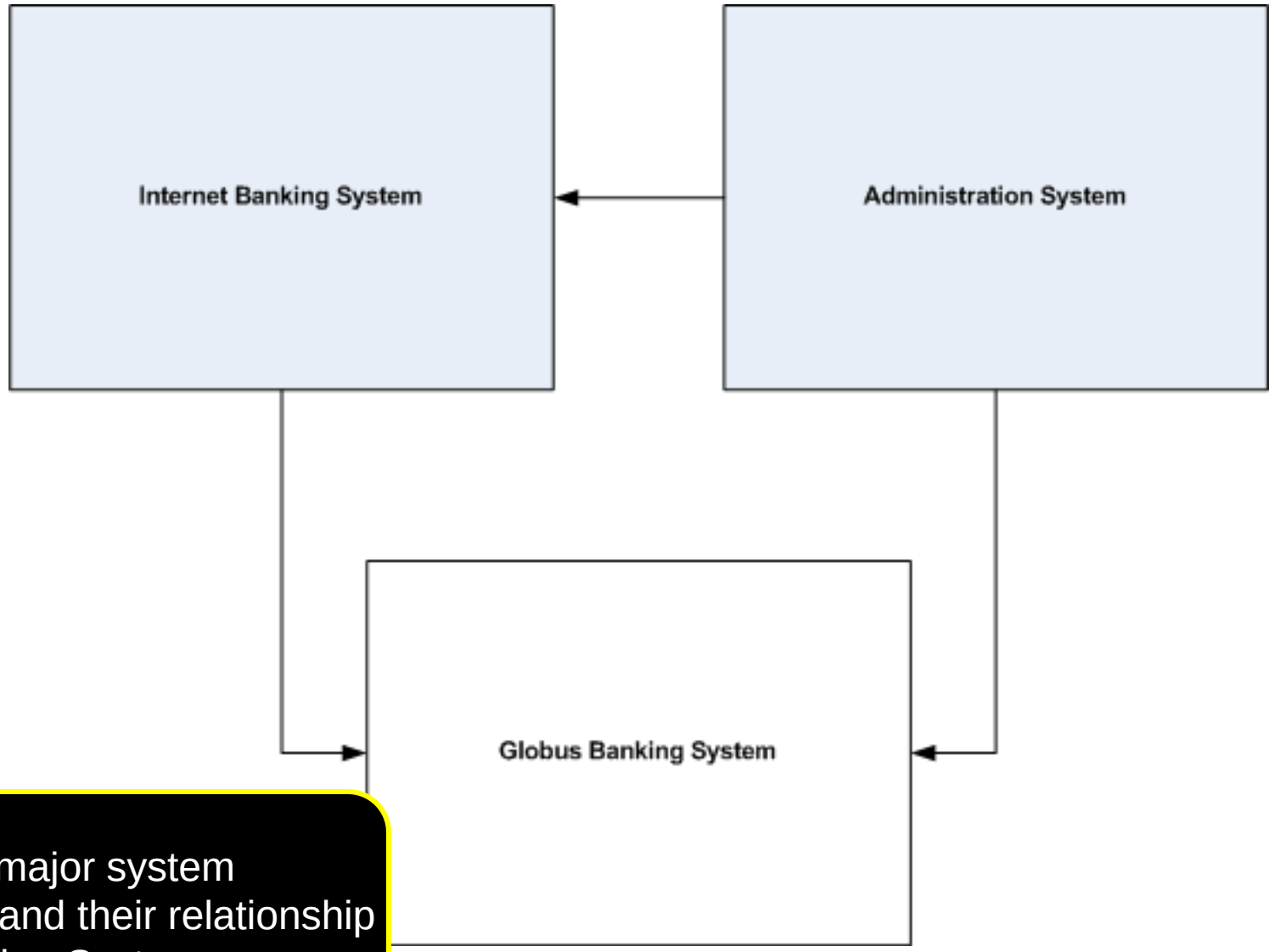

Context
Functional view
Process view
Non-functional view
Architectural Constraints
Architectural Principles
Logical view
Interface view
Design view
Infrastructure view
Deployment view
Operational view
Security view
Data view
Technology selection
Architecture Justification

Start your software architecture document by setting the scene

Context

Context diagrams allow you to

set the scene



It shows the major system components and their relationship with the existing System.

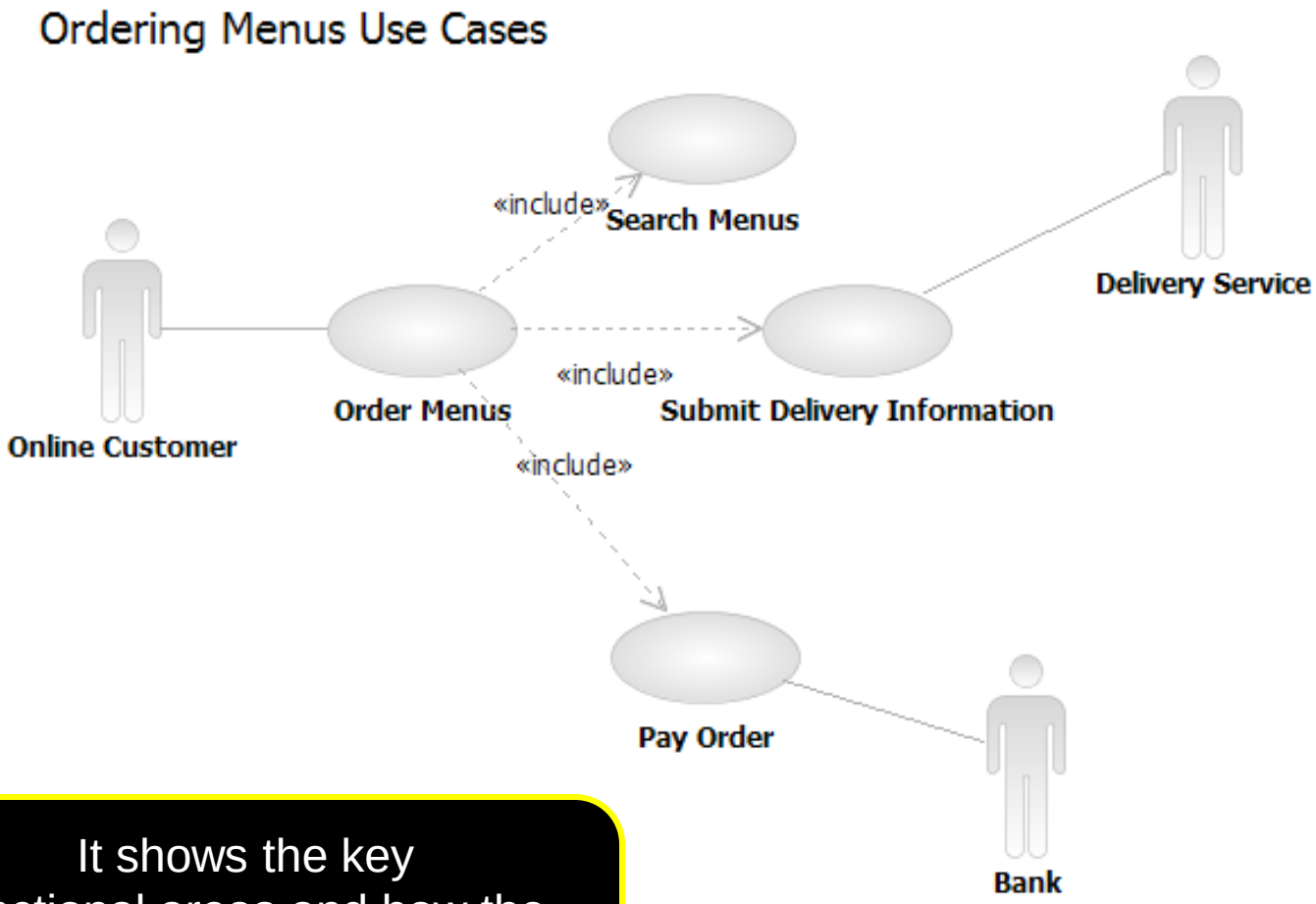
Functional View

Summarizes key

functional areas

Identifies

key users



It shows the key functional areas and how the functionality is utilized by key types of users

Helps you to identify the

architecturally significant

use cases

Include a summary to highlight

why are they

architecturally significant.

Process View

If your system implements a business process, include a summary of it

A process view can be a useful
way to summarize the

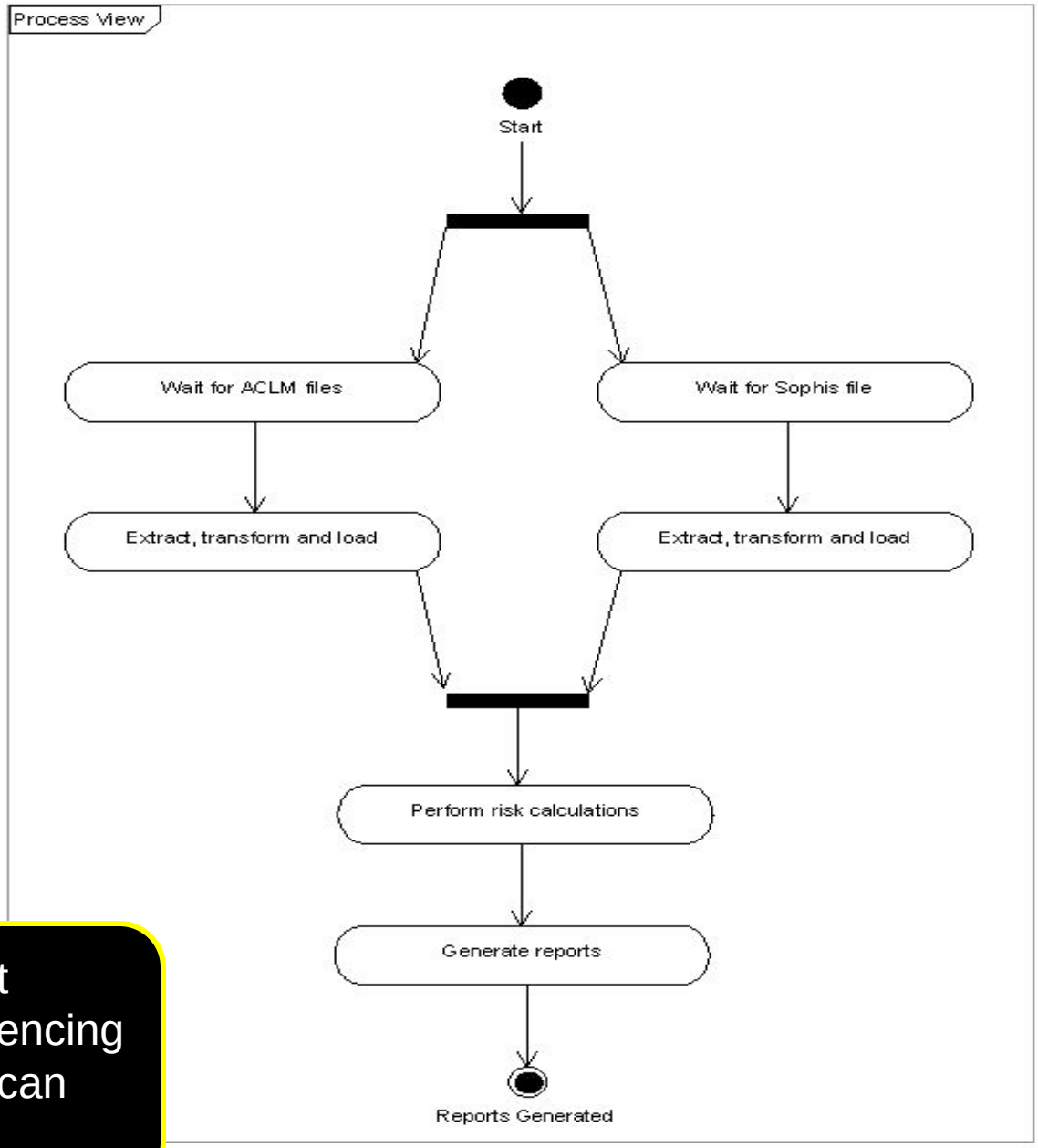
overall process

being implemented by the system

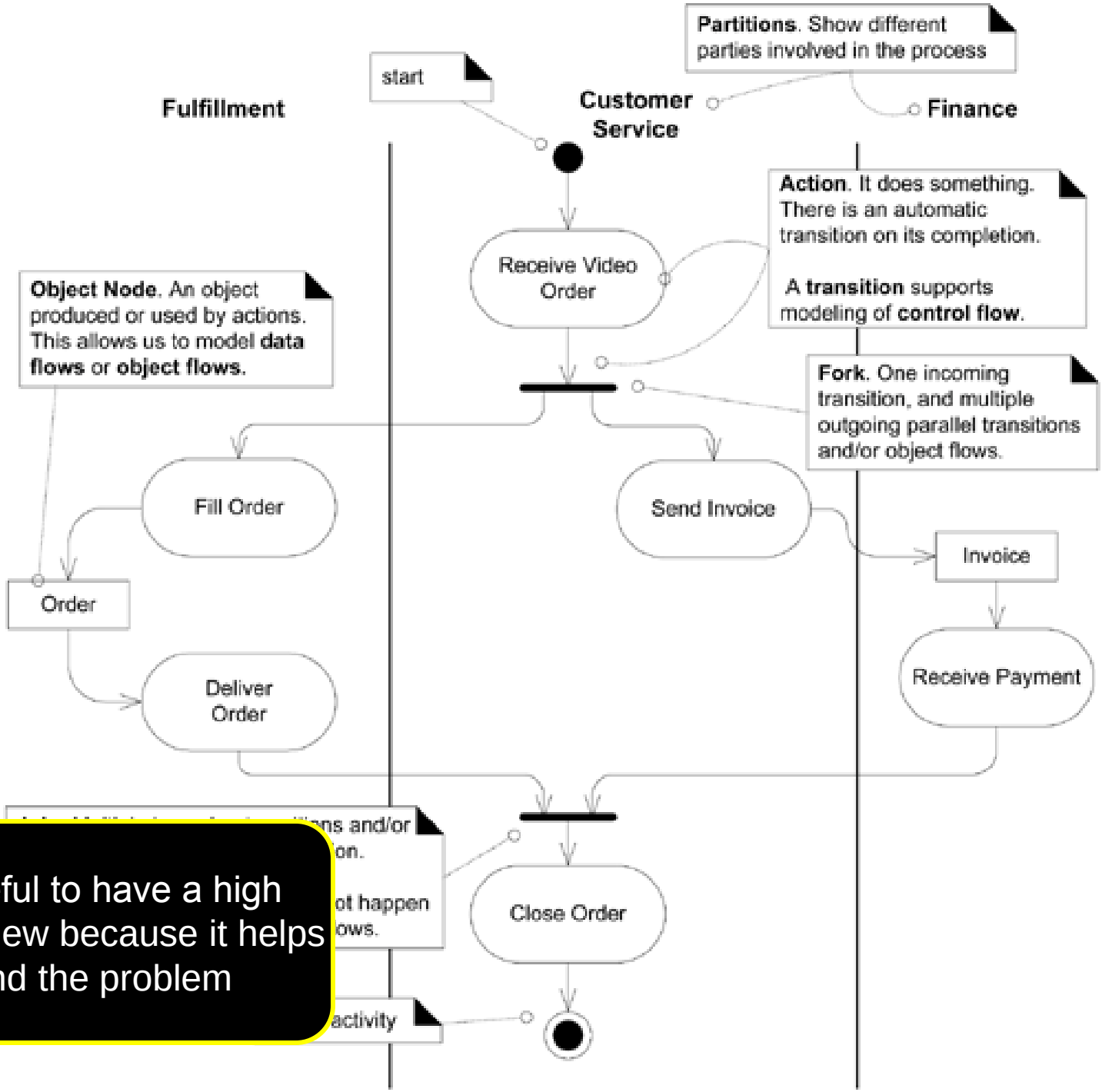
steps, their order,

flow of information,

concurrency and parallelism



shows the important steps along with the sequencing and where parallelism can occur.



It's often useful to have a high level process view because it helps understand the problem

Non Functional View

Does your solution
need to be very

fast?

Does your solution
need to be very

scalable?

Does your solution
need to be highly

available?

Does your solution
need to be very

secure?

Does your solution
need a guaranteed

audit trail?

- Performance
- Availability
- Usability
- Security

End User's view

- Time To Market
- Cost and Benefits
- Projected life time
- Targeted Market
- Rollout Schedule
- Integration with Legacy System

Business Community view

- Maintainability
- Portability
- Reusability
- Testability
- Interoperability

Developer's view

Summarize

the key non-functional areas

Include a summary to highlight

Why are the

architecturally significant.

Architectural Constraints

Software lives within
the context of the

real world

The real world has

Constraints

Approved

technology lists

Local and Public

Standards

Common

protocols and
message formats

Tactical or strategic?

What are the constraints?

Why are they being imposed?

How do they affect the architecture?

How are you working with them?

It can be useful to document
the constraints and their

influence

(positive and negative)

Architectural Principles

Constraints are typically

Imposed

by some one else

Principles are what you

want to adopt

Architectural principles can improve

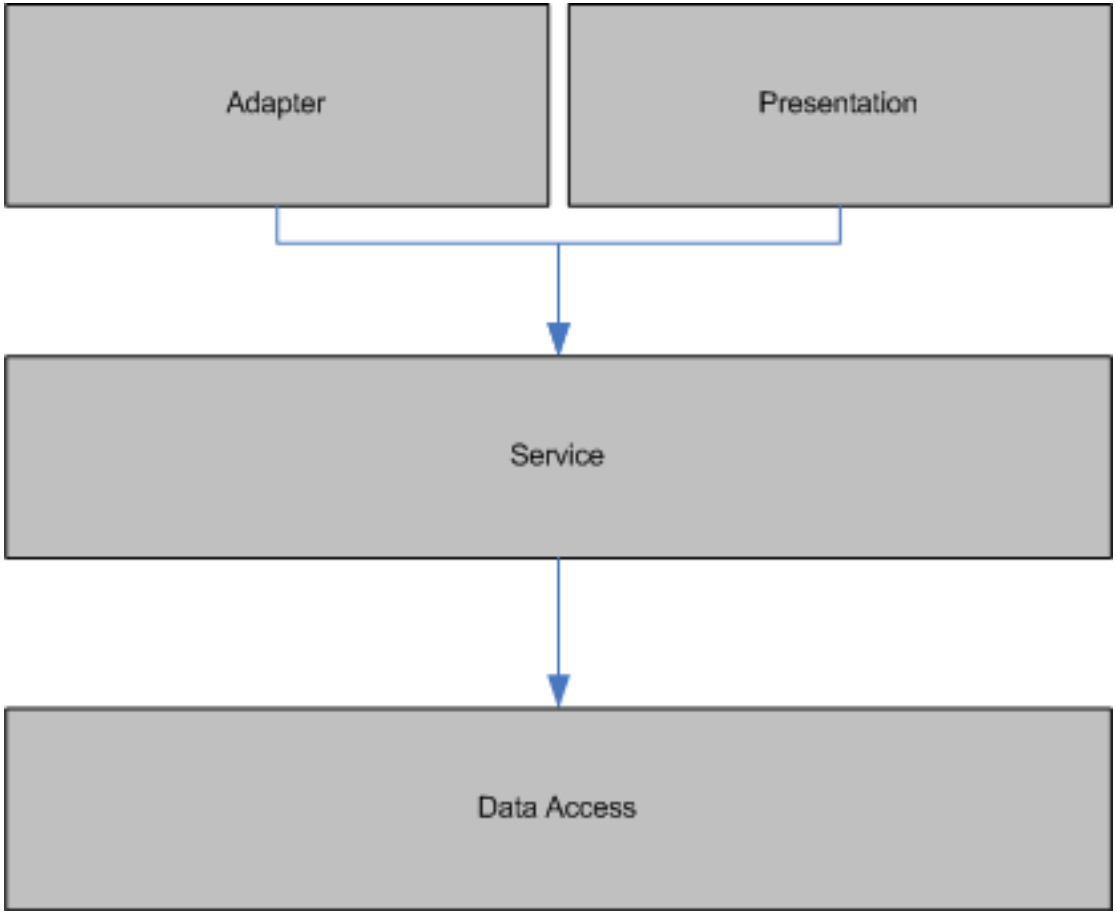
consistency

all web pages should follow the

following MVC approach

use the HTTP session

object in the following way...



Shows an architectural Layering Approach. The key layers and how the flow of logic moves through them.

Logical View

Logical views of a system
are probably the most

widely used

A diagram showing the

major components

and their **interaction**

(including technology choices)

Structure

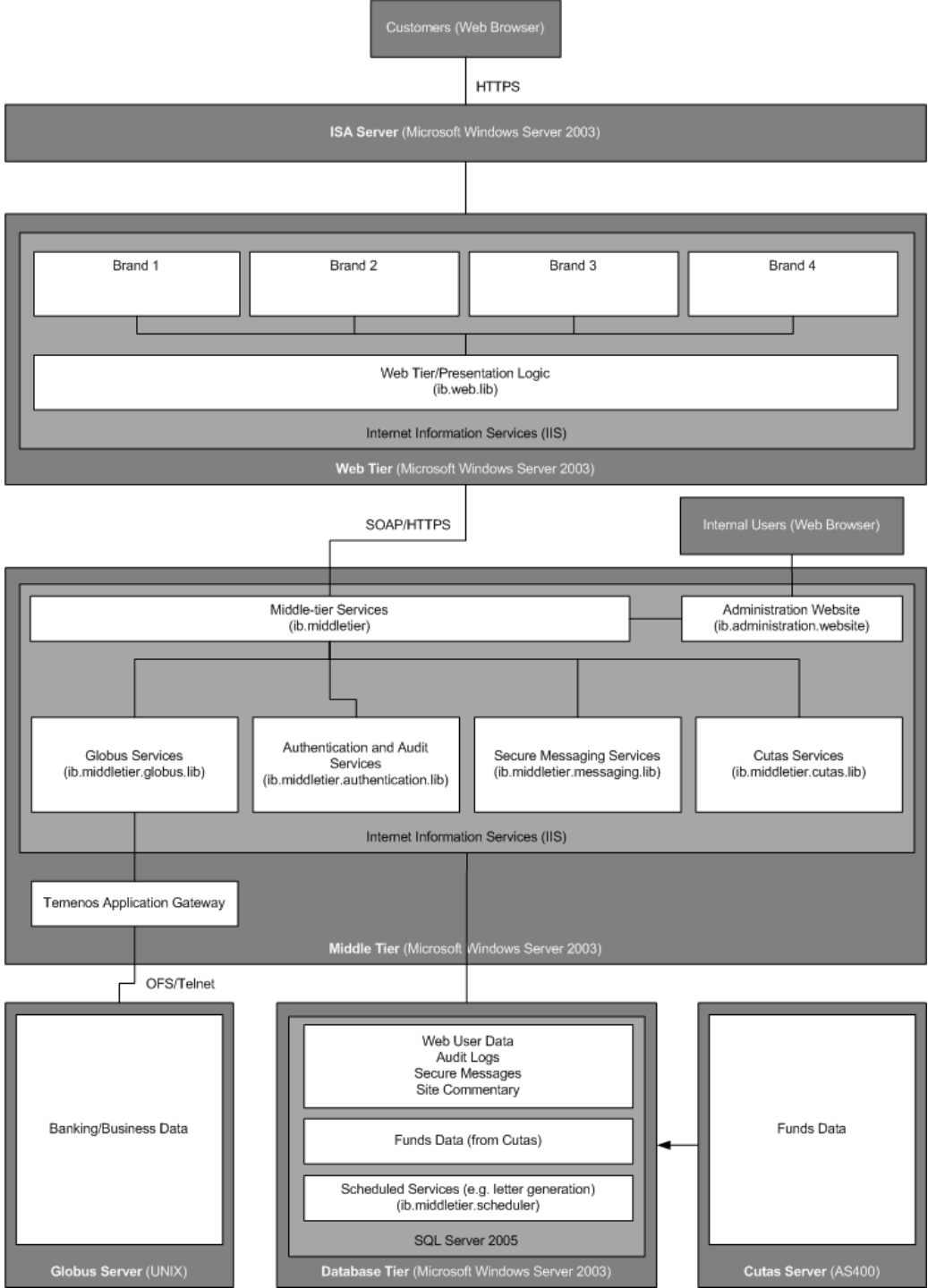
of the system

It’s your **big**

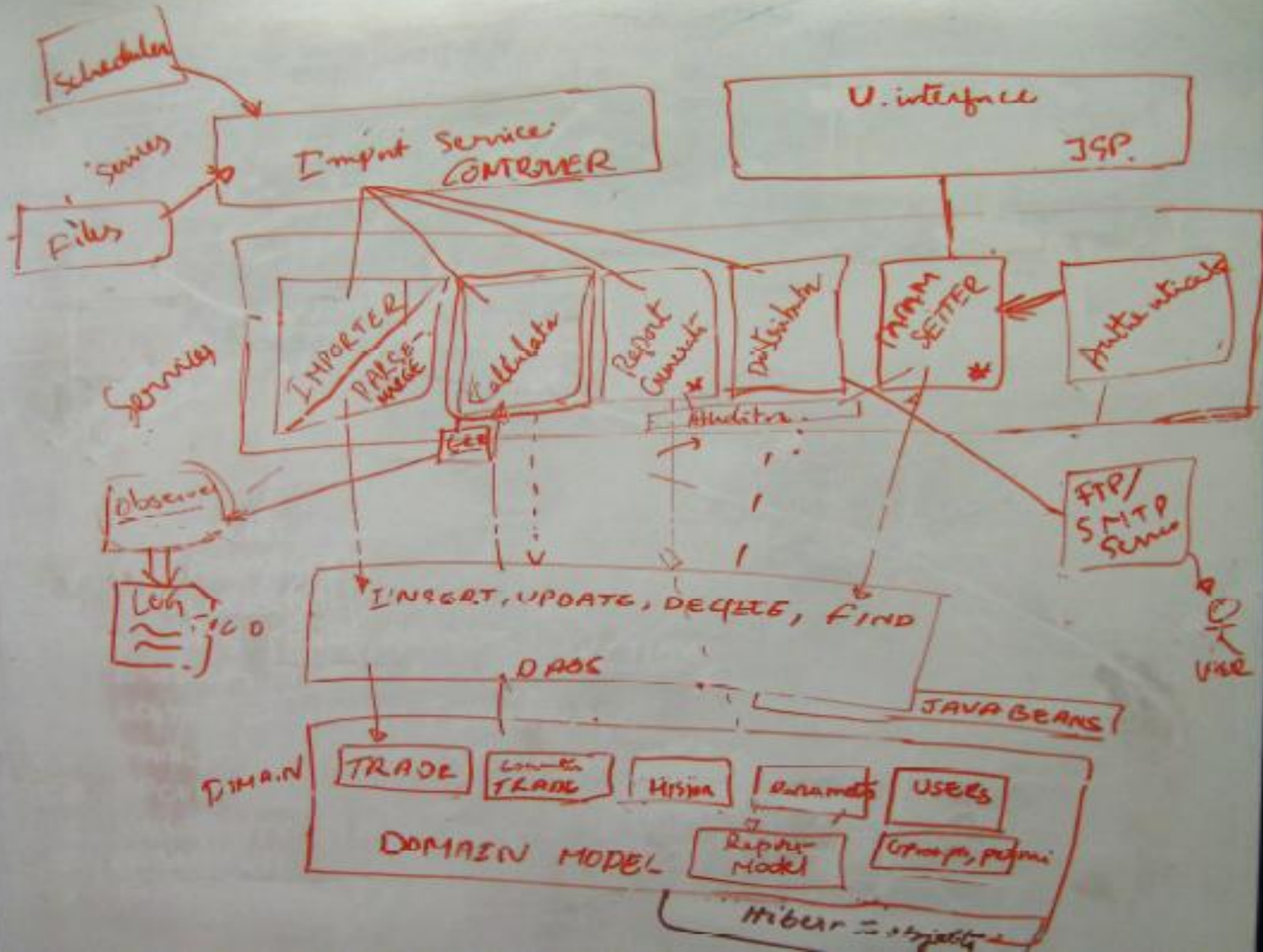
picture

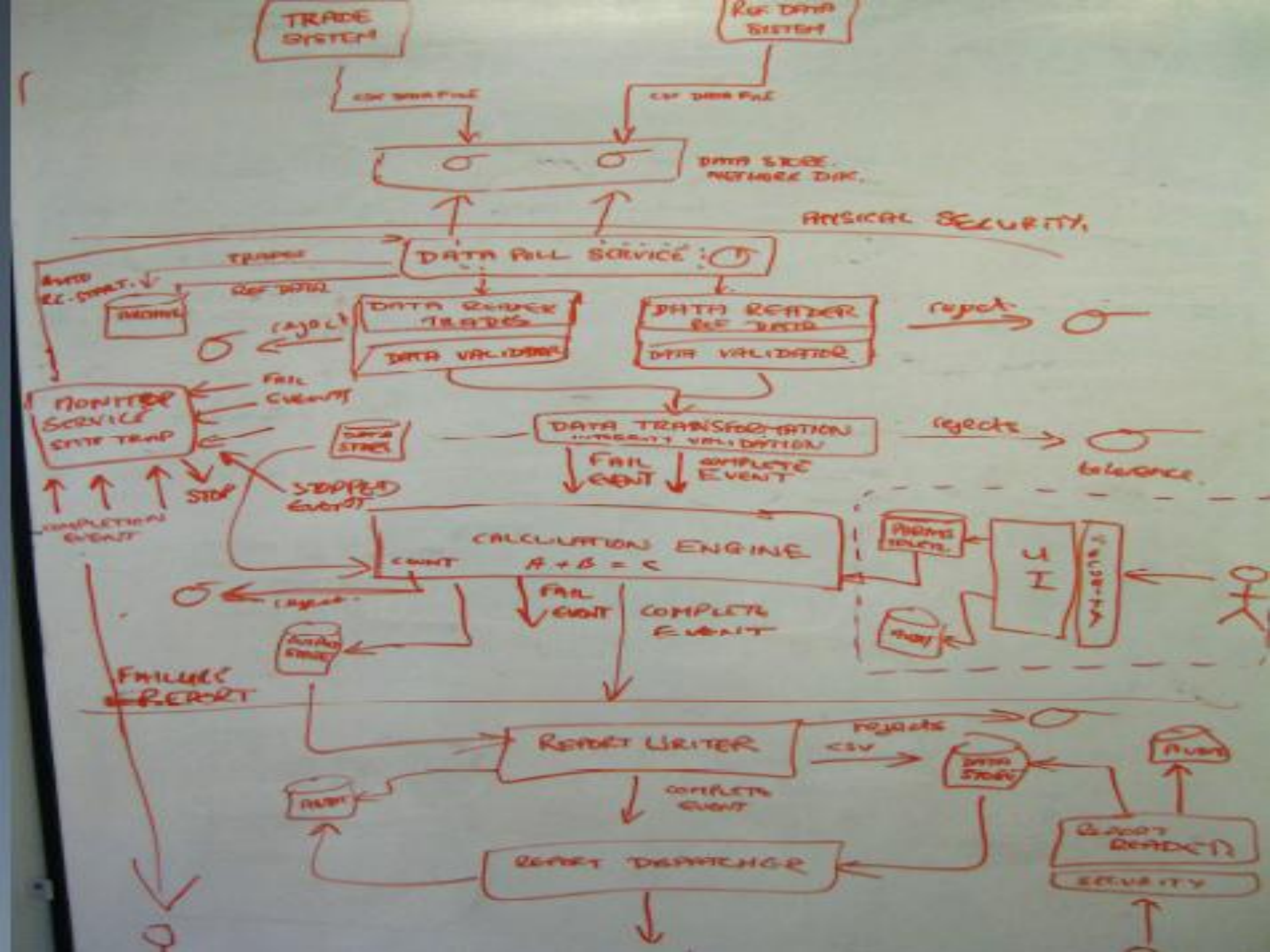
Often useful to
include the links to

external systems

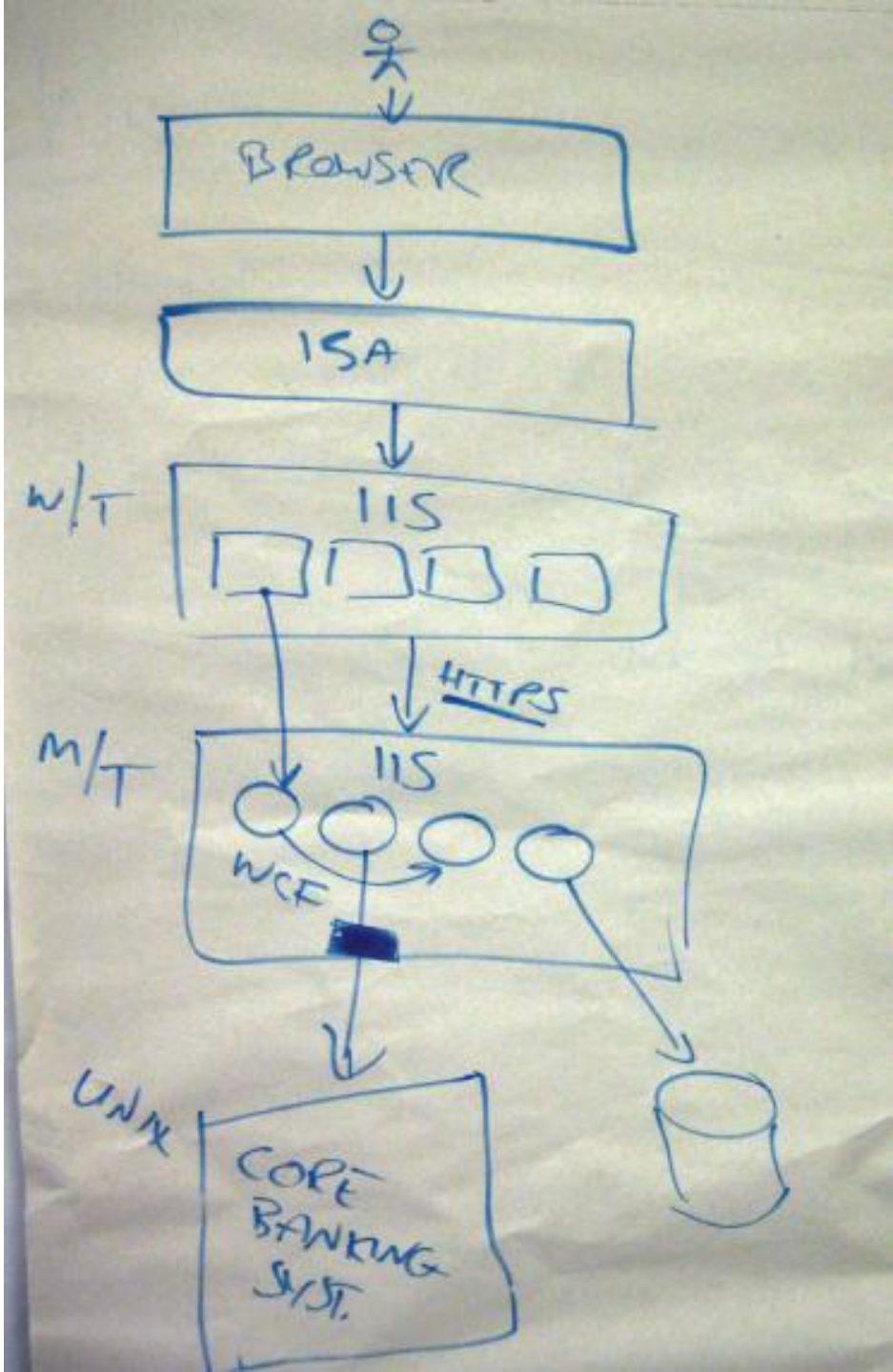


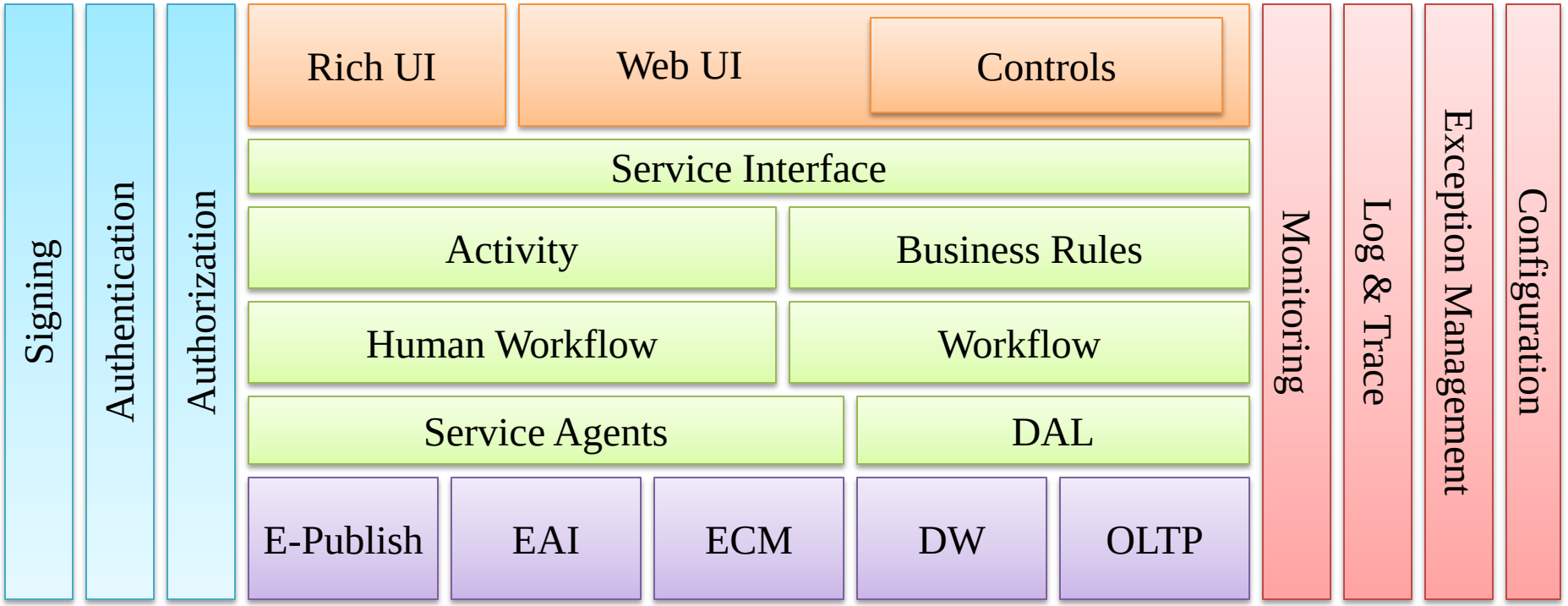
It's the sort of diagram that many people would use to start explaining how a system is designed and how it works, forming the basis for further diagrams with more detail.


Scheduler
Import Service CONTAINER
U. interface
JSP
Files
Importer
Calculator
Report Generator
Distributor
Transaction Setter
Authentication
Observer
Log
I/O
INSERT, UPDATE, DELETE, FIND
DAOS
JAVA BEANS
DOMAIN
TRADE
USERS
DOMAIN MODEL
Report Model
FTP/ SMTP Server
User


TRADE SYSTEM
REF DATA SYSTEM
DATA STORE
PHYSICAL SECURITY
DATA ALL SERVICE
TRADES
REF DATA
DATA READER TRADES
DATA READER REF DATA
DATA VALIDATOR
reject
MONITOR SERVICE
FAIL EVENT
DATA TRANSFORMATION
INTEGRITY VALIDATION
FAIL EVENT
COMPLETE EVENT
reject
tolerance
STOP
STOPPED EVENT
COMPLETION EVENT
CALCULATION ENGINE
A + B = C
COUNT
FAIL EVENT
COMPLETE EVENT
UI
FAILURE
RESTART
RESTART WRITER
reject
csv
COMPLETE EVENT
RESTART DISPATCHER
SECURITY

Logical view is typically diagrams with commentary for each major component and interaction




Signing
Authentication
Authorization
Rich UI
Web UI
Controls
Service Interface
Activity
Business Rules
Human Workflow
Workflow
Service Agents
DAL
E-Publish
EAI
ECM
DW
OLTP
Monitoring
Log & Trace
Exception Management
Configuration

Interface View

Highlights the major interactions

with external systems
and the nature of those
interactions

What is the system?

How do we

connect?

What is the

protocol?

What services

do you provide?

synchronous or asynchronous?

Is the interface always

available?

Do we need a

durable

message subscriber?

Can we receive messages

out of order?

Can we receive

duplicate

messages?

Is the interface

idempotent?

Who owns the interface?

How often does it

change?

If you upgrade,

do we need to make

modifications?

Include low level

technical details

too...

Details of the interface

(protocol, queue/topic names, connection strings, schema, required libraries, ...)

Include details about system interfaces to help you ensure everything has been thought about

Design View

A summary of how important

parts of your implementation

work

Templates

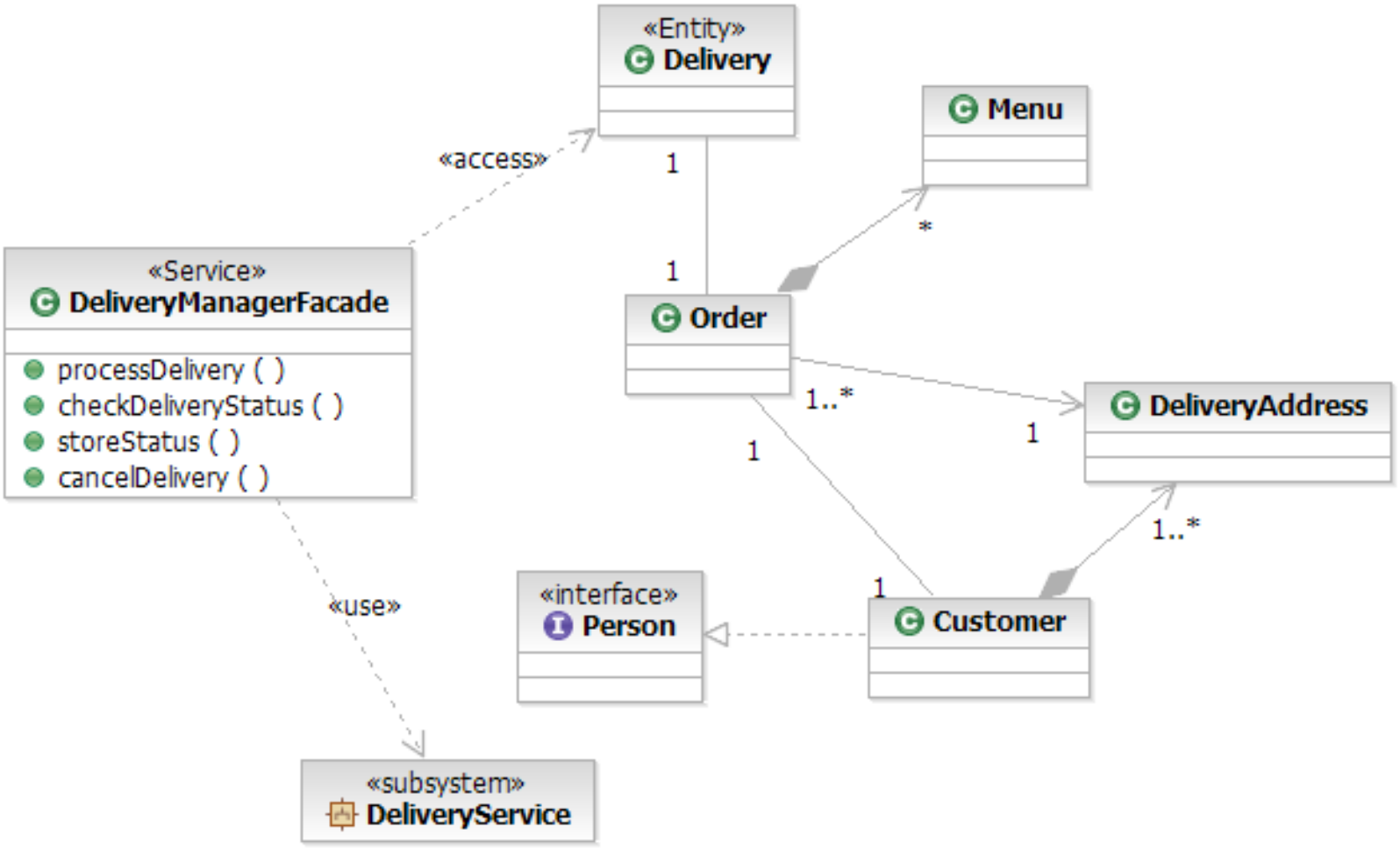
(e.g. an example service implementation)

Common

design patterns

used when building

features


«Entity»
Delivery
Menu
«access»
1
1
*
«Service»
DeliveryManagerFacade
processDelivery ( )
checkDeliveryStatus ( )
storeStatus ( )
cancelDelivery ( )
Order
1..*
1
DeliveryAddress
1
1..*
«interface»
Person
1
Customer
«use»
«subsystem»
DeliveryService

Infrastructure View

Infrastructure diagrams show the

physical hardware

and infrastructure on which

the software will be deployed

Does it support all of the components that make up your architecture?

Does it support

redundancy,

failover and disaster

recovery?

(if applicable)

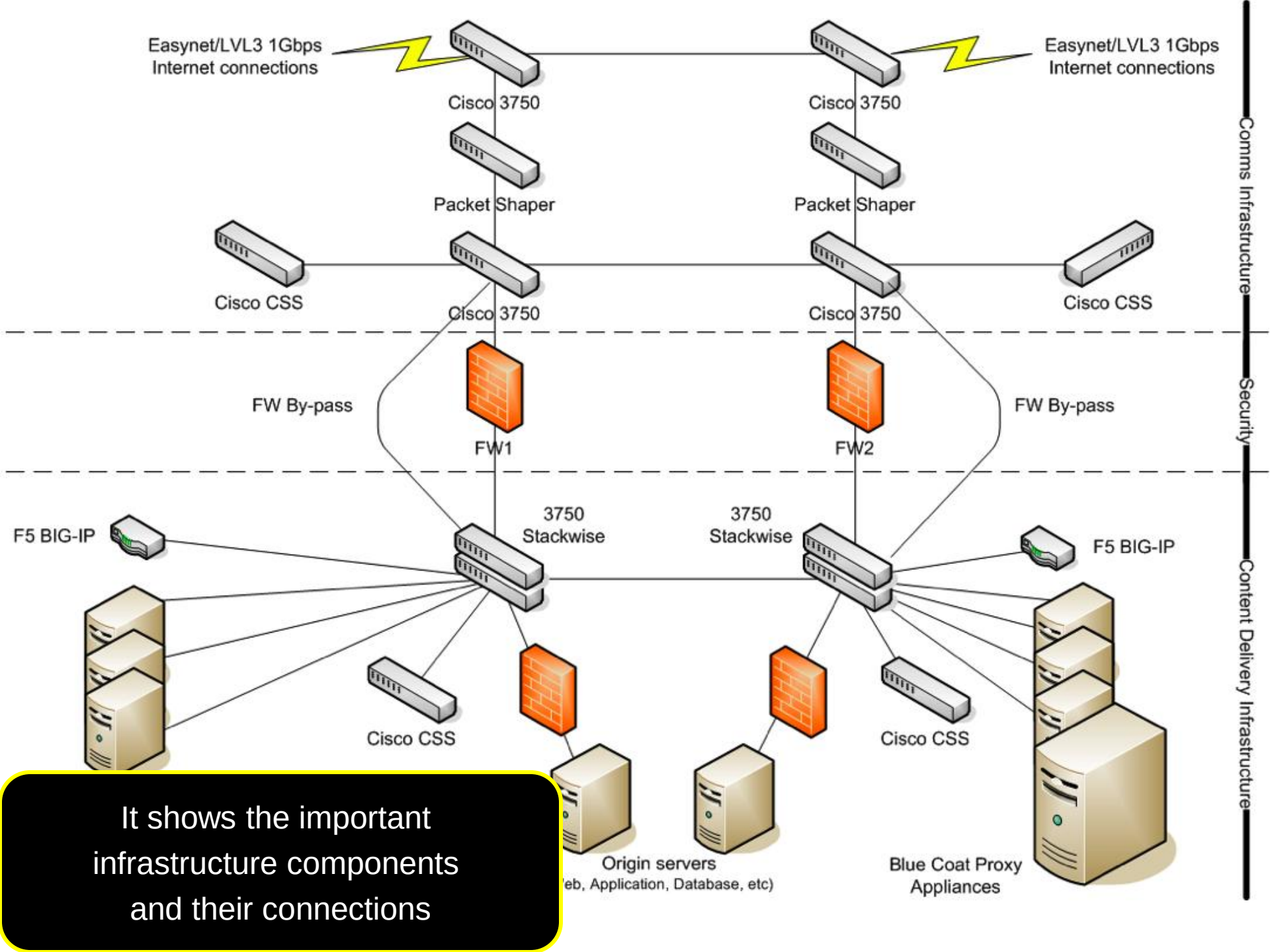
Is it appropriately

secure?

Is there sufficient

infrastructure for development, testing, pre-production, production, etc?

Who owns and looks after the infrastructure?



It shows the important infrastructure components and their connections

Diagrams may also include lots of additional information
(e.g. IP addresses, VLANs, host names, hardware model details, rack locations, etc)

As a “software” architect,
you may be out of your
comfort zone

Deployment View

How are the software components

deployed

onto the infrastructure?

This view represents how the

operational and

support staff

will see your system in production

How is data and code

replicated

between sites for

disaster recovery purposes?

Which tiers and components can be

scaled-out?

How are the components

installed

and configured?

(e.g. manual or automatic?)

Operational View

How will the application

be monitored?

How will the application

be managed?

How will problems be

diagnosed?

Include an operational view
to summaries how your
system works from a day-today
operational perspective

Security View

Authentication and authorization from a user perspective?

Confidentiality

of data in transit?

(internally and externally)

Certificates and keys?

Secure storage

of user credentials?

(e.g. away from the web and hashed)

Secure storage

of service account

credentials?

(e.g. encrypted in configuration files)

Runtime permissions?

(i.e. what users are your processes running as?)

Use of a

sandbox model?

Network

and infrastructure

security model?

(e.g. firewalls, DMZs, etc)

Data View

Are you managing

large

quantities of data?

Sometimes a data view is useful to

summarise

your architecture from
a data perspective

Data models

(e.g. entity relationship diagrams)

Information about
expected data

quantities

Database sizing

(data and logs)

Archiving

Back-ups

Data storage and replication

Audit logs and log files

Technology Selection

Include a technology justification if you are continually asked about it or are doing something different from the norm

Why did you choose technology X?

Why didn't you choose technology y?

Do the technologies
meet the architectural

constraints?

Do the technologies help satisfy the architectural

principles?

Reference

approved technology lists
or product evaluations
as appropriate

Architecture Justification

Include an architecture justification to prove to yourself that your architecture works!

Does your architecture

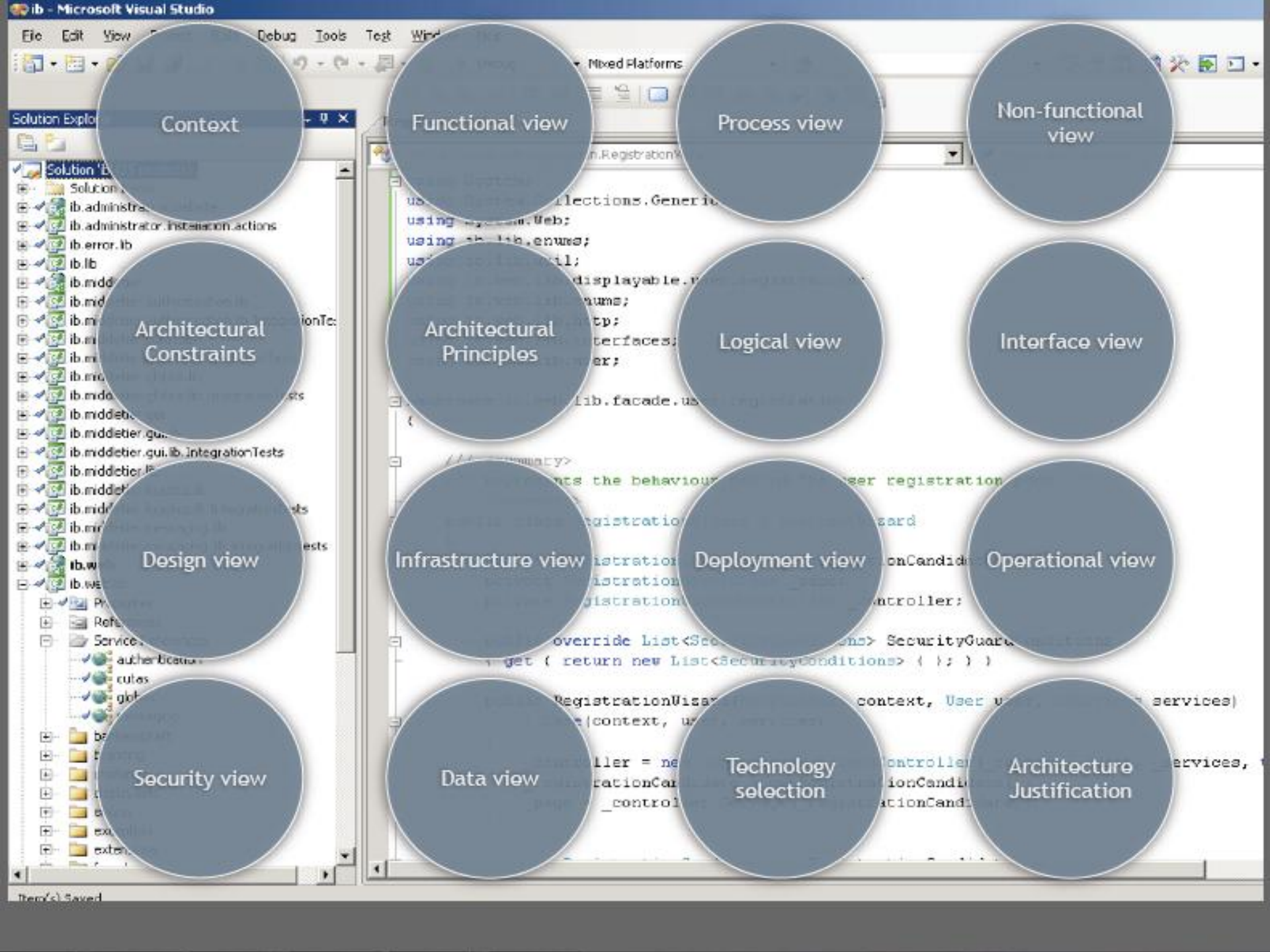
satisfy

the requirements?

Does your architecture
provide a

sufficient

platform for your solution?


Context
Functional view
Process view
Non-functional view
Architectural Constraints
Architectural Principles
Logical view
Interface view
Design view
Infrastructure view
Deployment view
Operational view
Security view
Data view
Technology selection
Architecture Justification