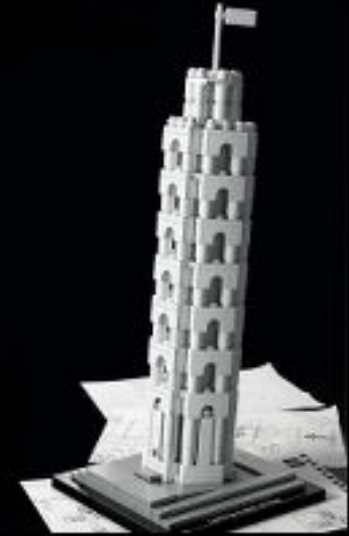


# Architecture



## Software Architecture and Software Architect

T-Systems RUS.  
JavaSchool

Saint-Petersburg,  
July'15

Sergey N Lukin



LIFE IS FOR SHARING.

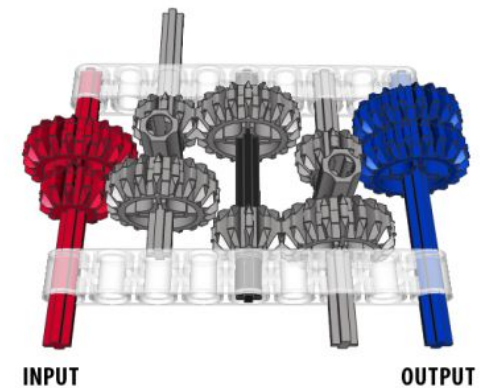
# Intro

- 0 Introduction
- 1 Questions
- 2 Terms
- 3 Simplification

# Agenda

## Content

- 0 Intro
- 1 Basic design principles
- 2 Design example
- 3 Multilayered architecture
- 4 Architect role in PLC



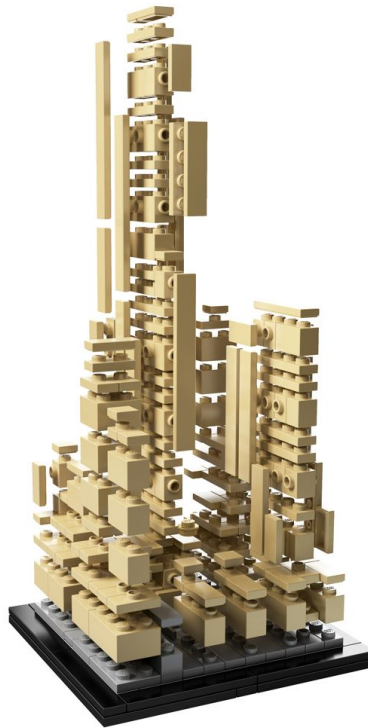
# Why we need Architects?

I know Java and SQL,  
why I need Architect for  
our product?



# Why we need Architects? Short Answer.

I know Java and SQL, why I need Architect for our product?



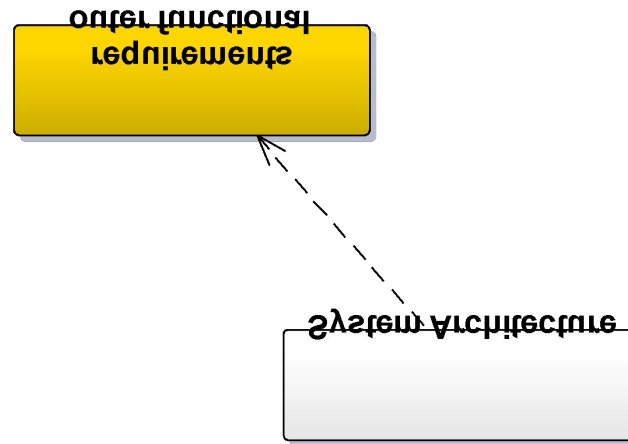
Because your product has **Architecture**



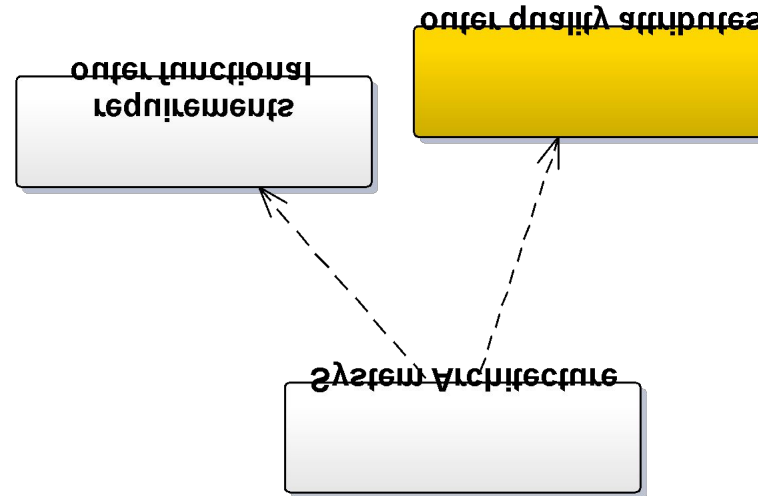
# Different architectures



# Outer Function requirements

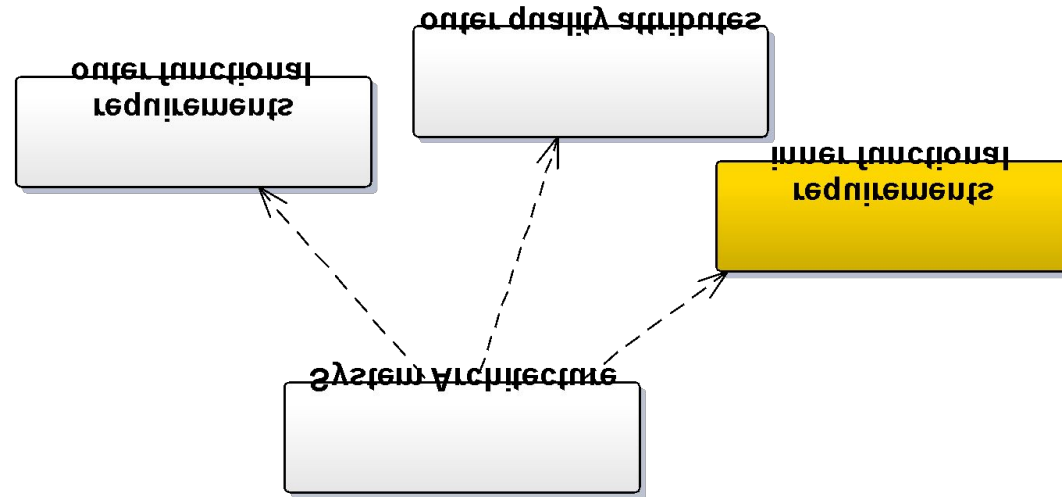


# Outer quality attributes

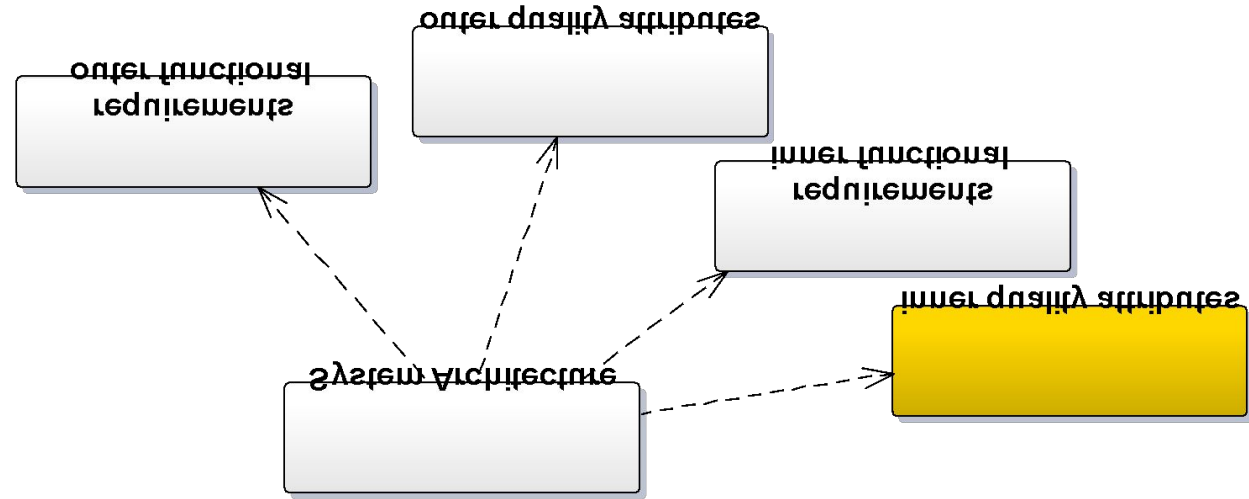




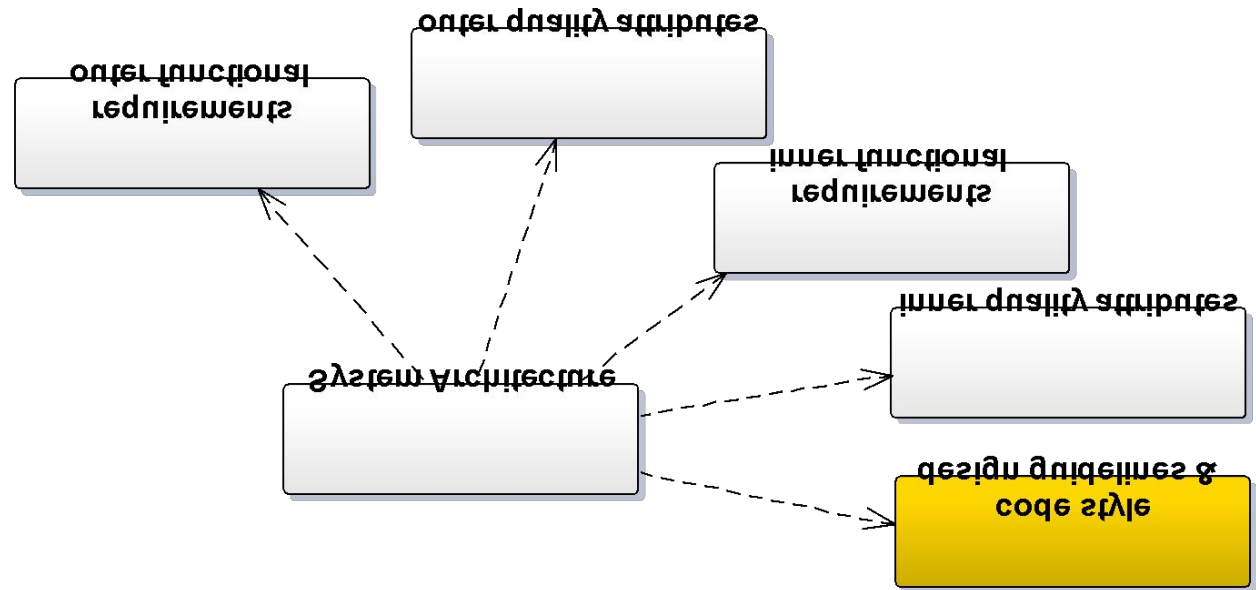
# Our internal functional requirements



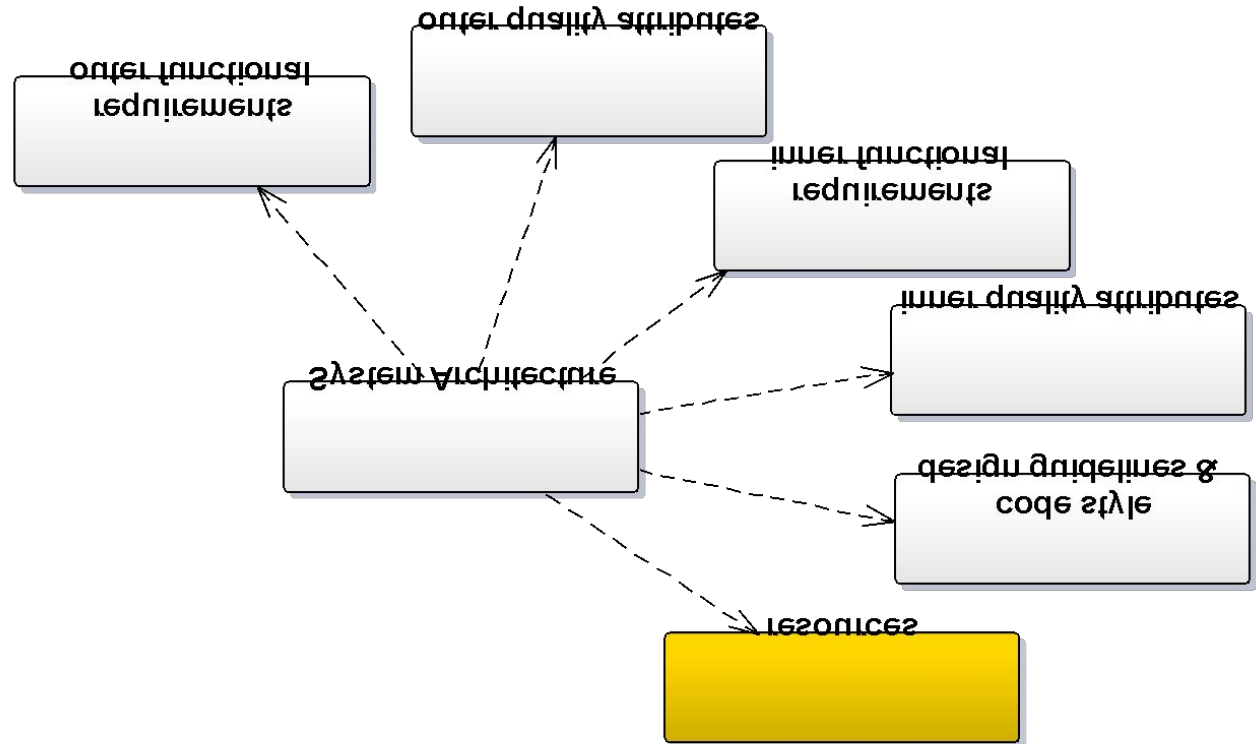
# Our internal quality requirements



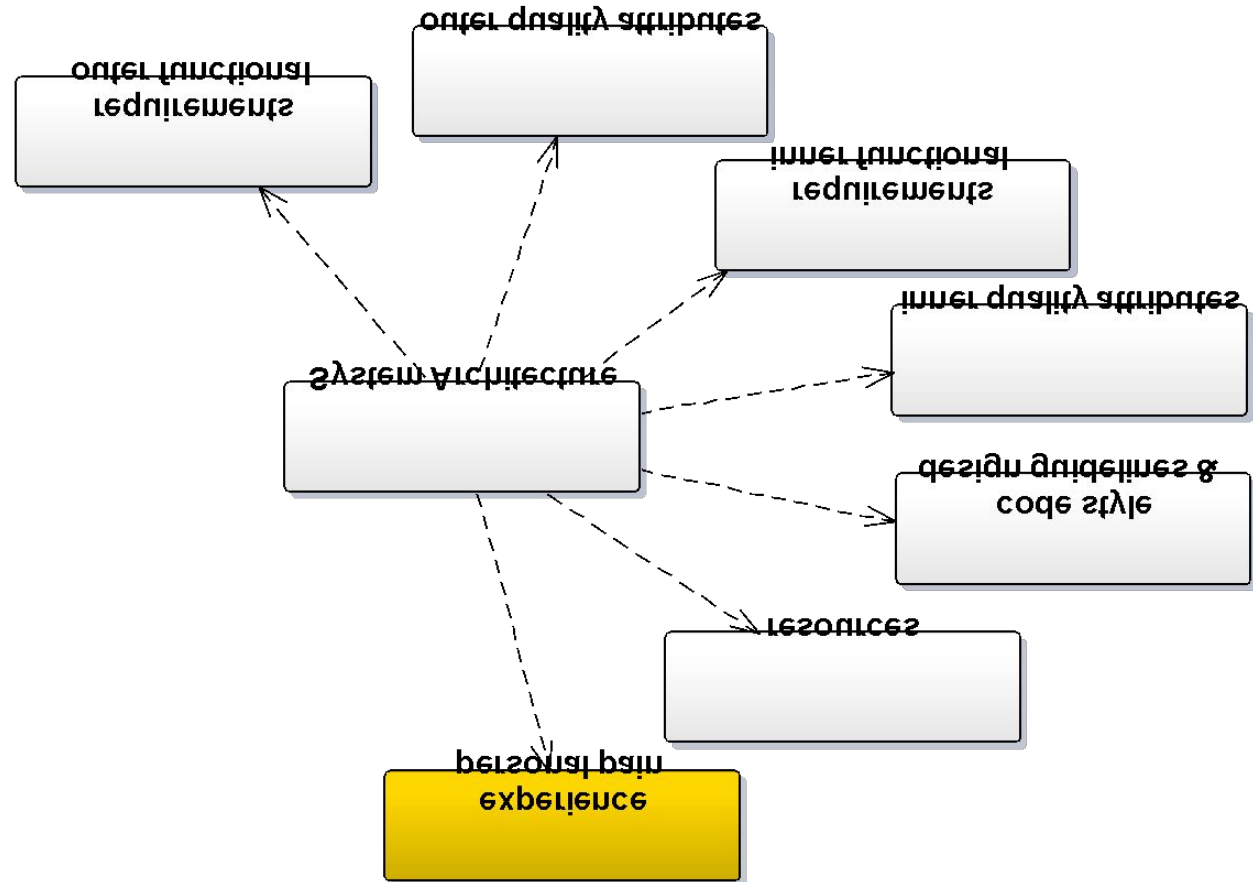
# Design Guidelines



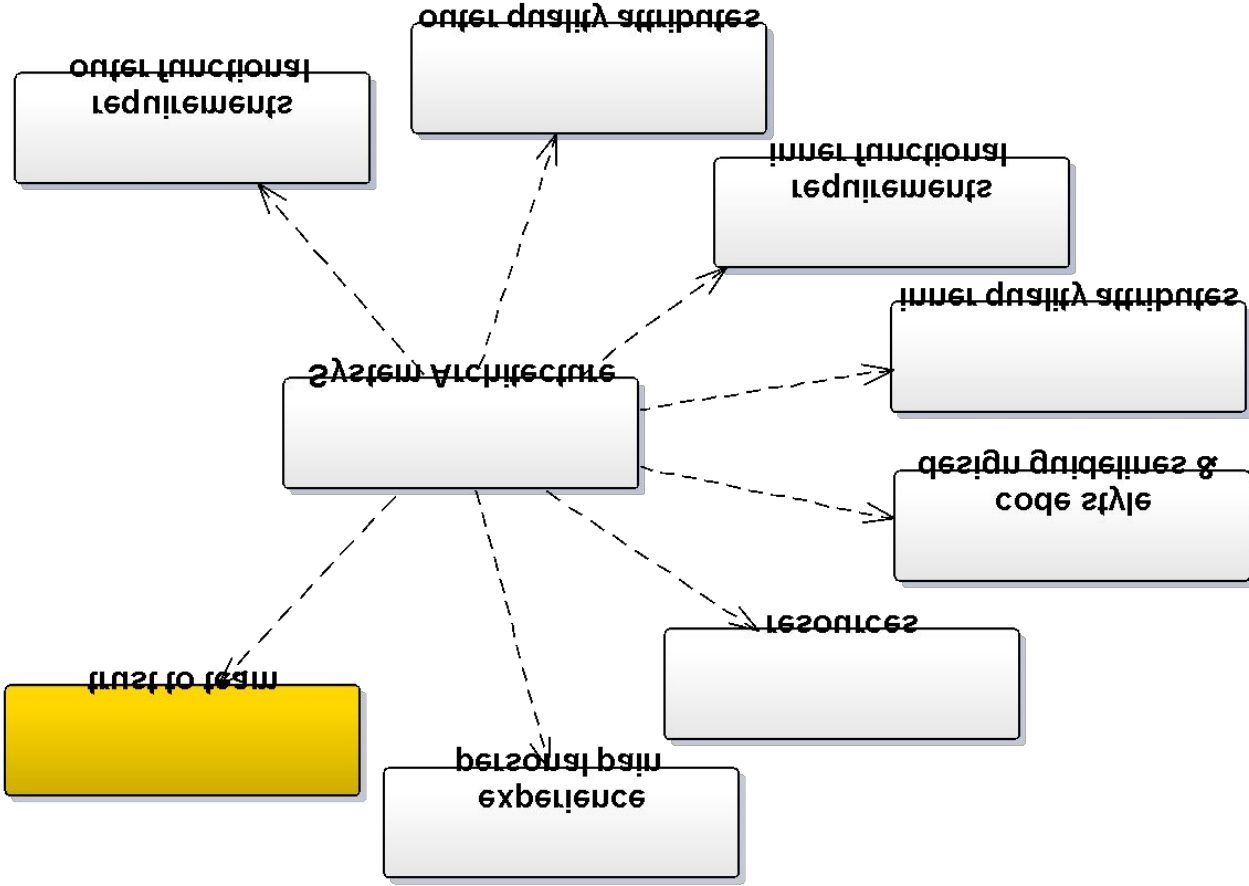
# Resources



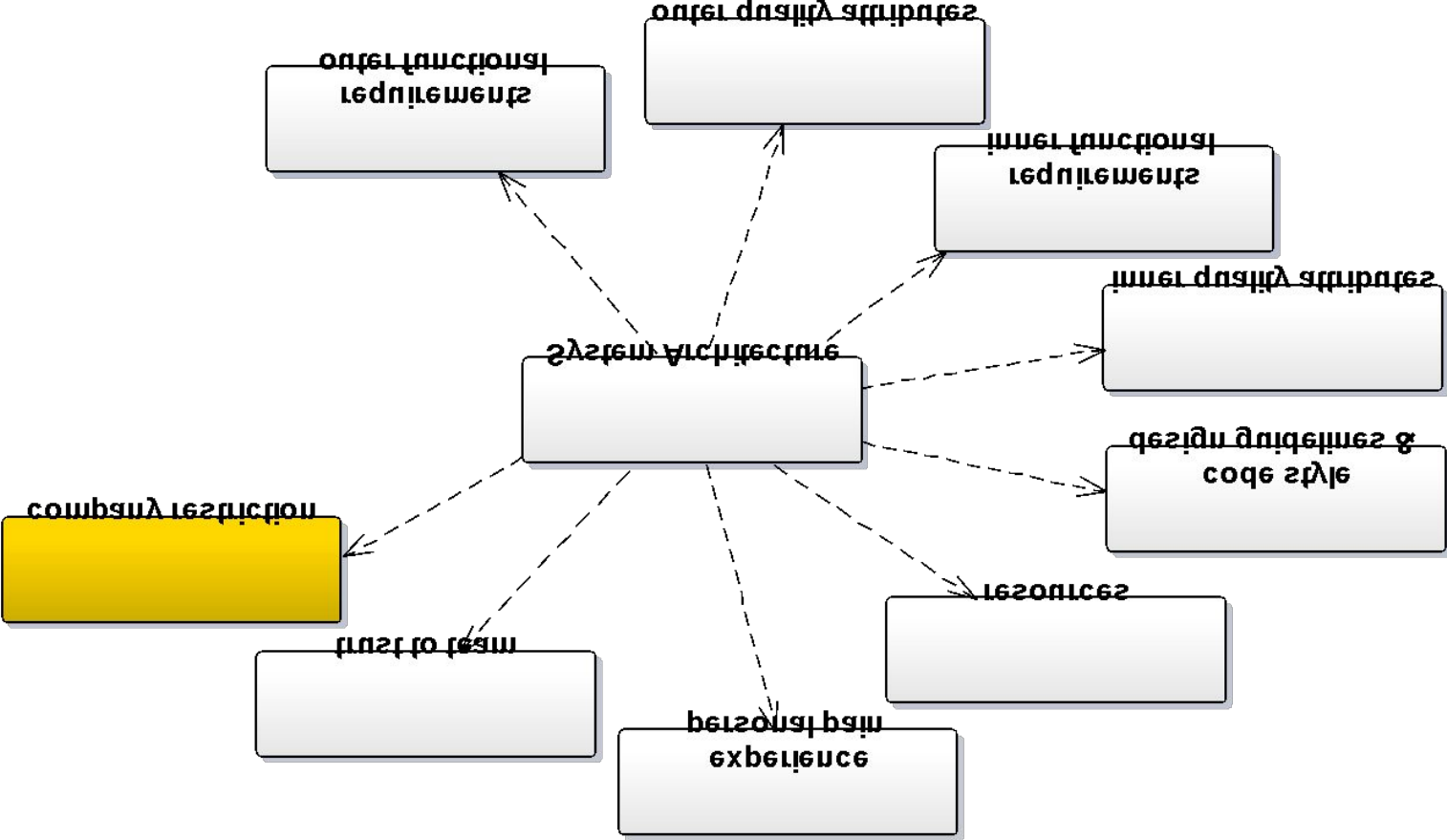
# Personal pain experience



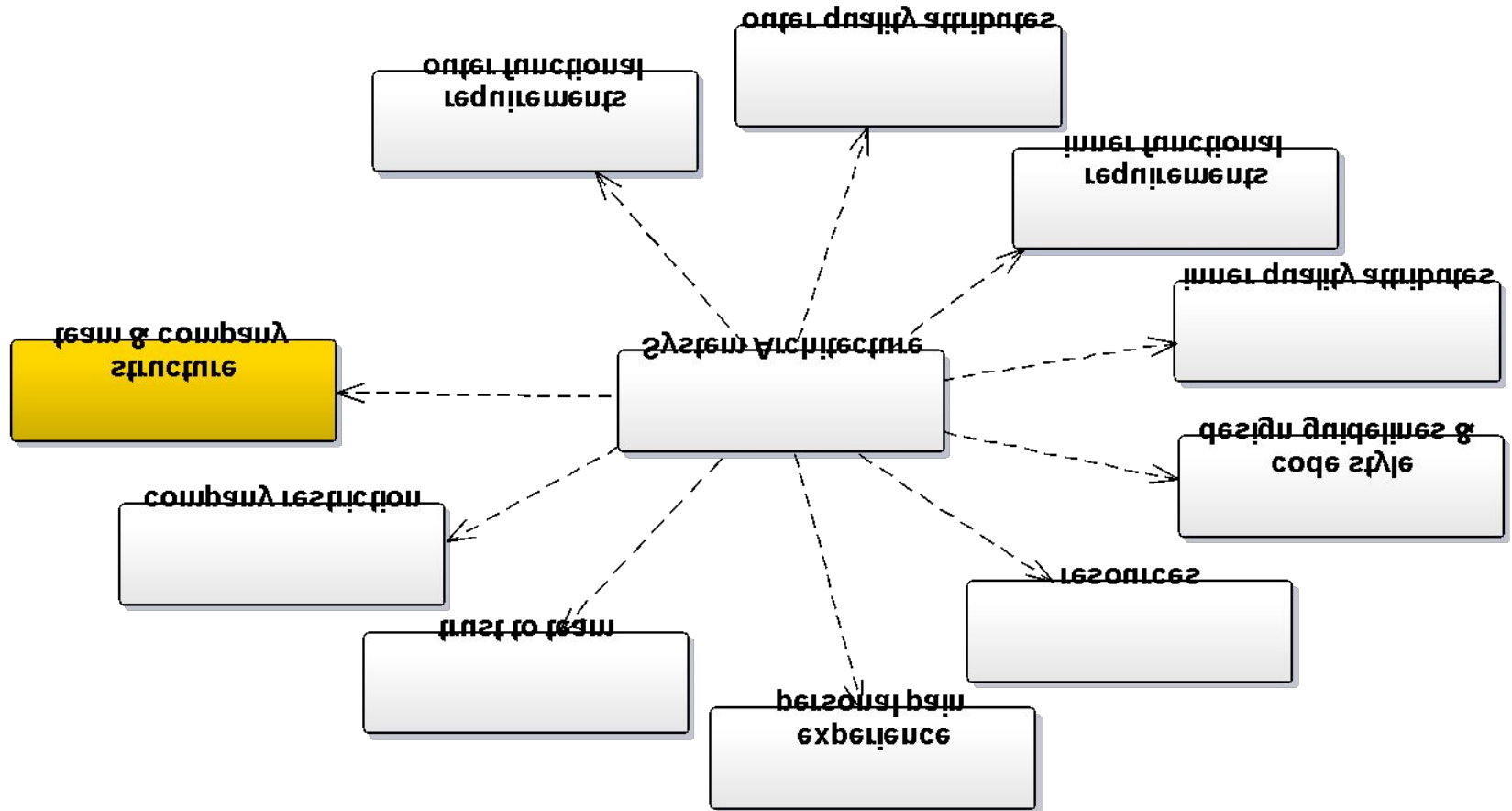
# Development team



# Company restriction

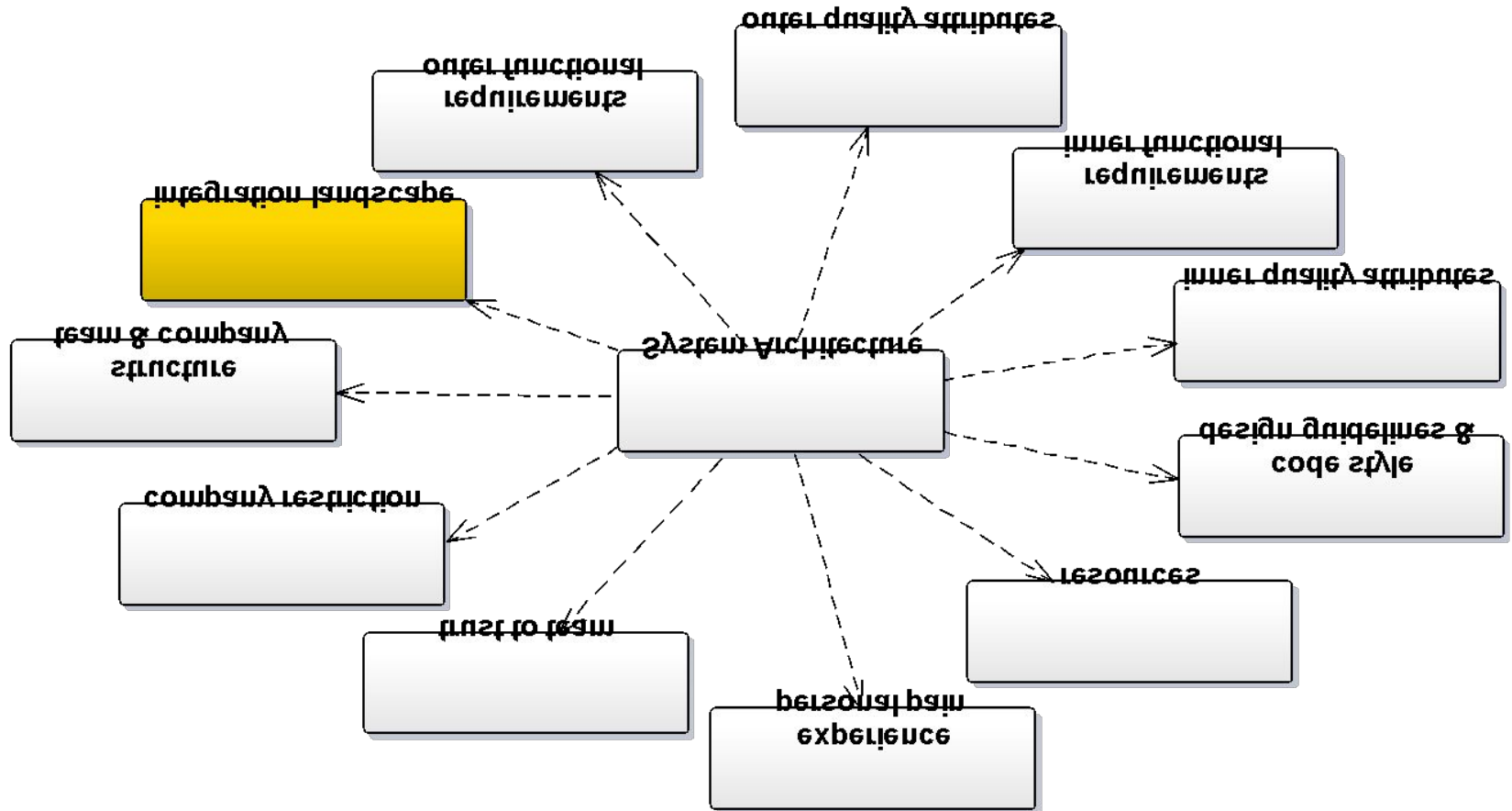


# Architecture depends on

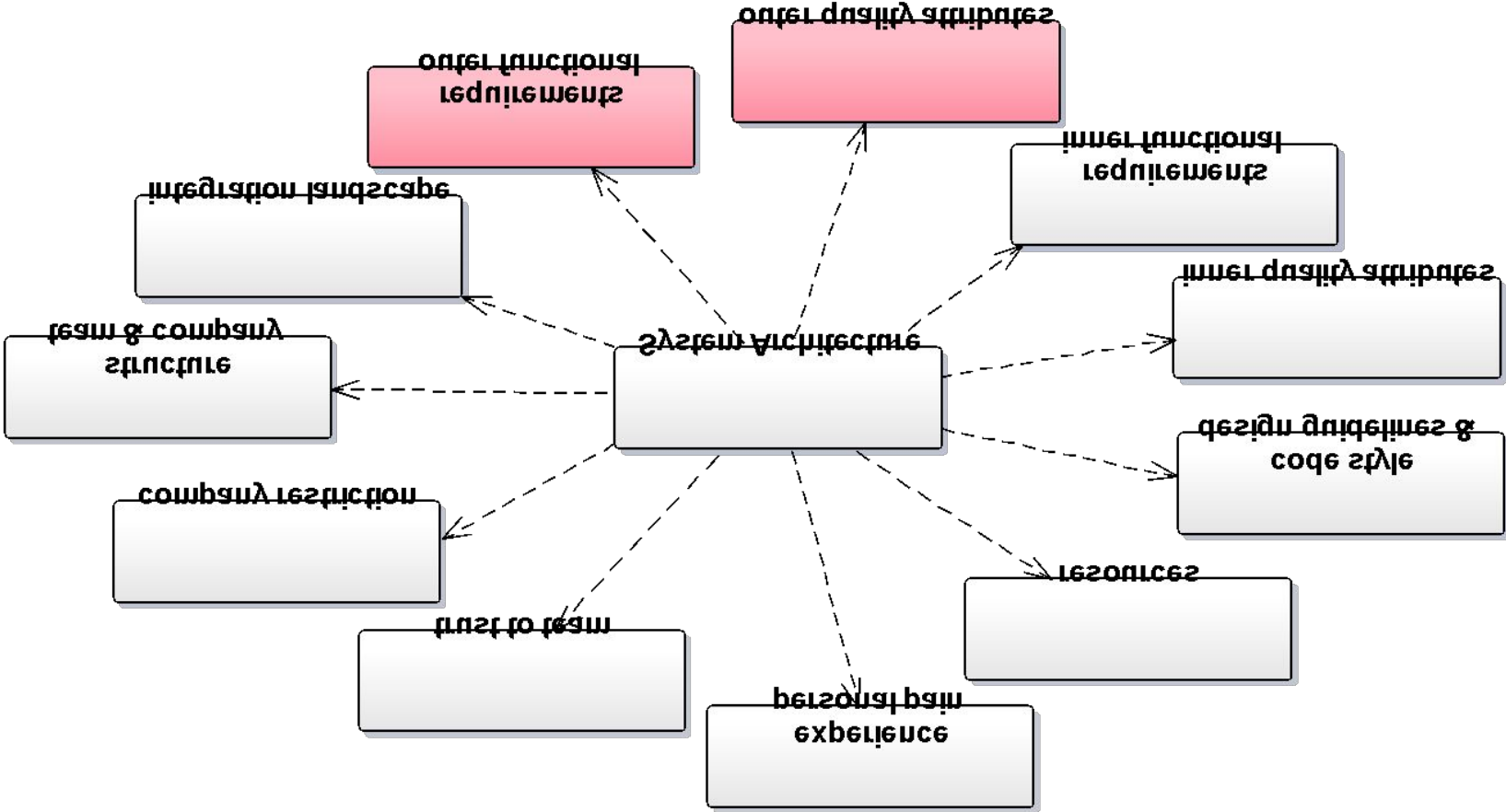




# Architecture depends on



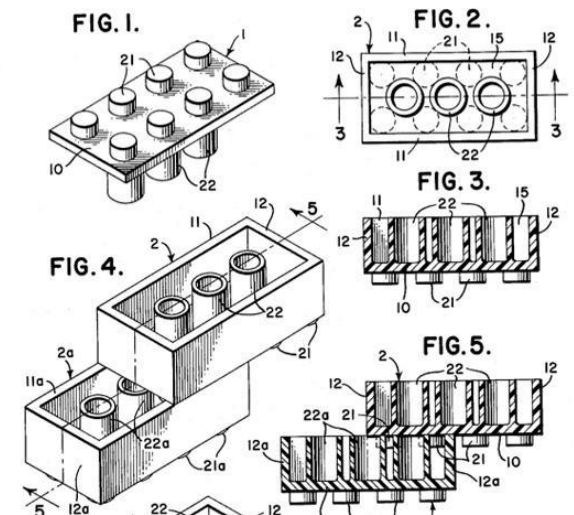
# But outer requirement is more important



# What means good architecture?

## Quality attributes

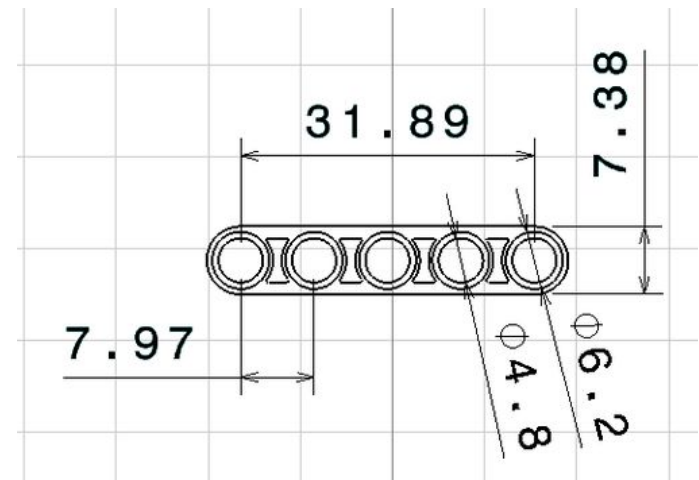
- Meet requirements
- Ready for change
- Ready for scaling and distribution
- Minimize cost
- Other NFR



# ISO 9126 Software quality

## Quality attributes

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability



# Key architecture principles and practice

## Principles

- Separation of concerns.
- Single Responsibility principle.
- Principle of Least Knowledge.
- Don't repeat yourself (DRY).

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication, managing sessions between applications
	Segments	4. Transport	Reliable delivery of packets between points on a network.
Media layers	Packet/Datagram	3. Network	Addressing, routing and (not necessarily reliable) delivery of datagrams between points on a network.
	Bit/Frame	2. Data link	A reliable direct point-to-point data connection.
	Bit	1. Physical	A (not necessarily reliable) direct point-to-point data connection.

# Key architecture principles and practices

## Common design practices

- Prefer composition to inheritance
- Separate the areas of concern between layers
- Be explicit about how layers communicate with each other.
- Define a clear contract for components.
- Keep design patterns consistent within each layer
- Do not mix different types of components in the same logical layer.
- Keep the data format consistent within a layer or component
- A component /object should not rely on internal details of other components/objects.
- Do not overload the functionality of a component.
- Keep crosscutting code abstracted from the application business logic as far as possible
- Establish a coding style and naming convention for development.
- Maintain system quality using automated QA techniques during

# Key architecture principles and practices

## Common design practices

- Prefer composition to inheritance
- **Separate the areas of concern between layers**
- Be explicit about how layers communicate with each other.
- **Define a clear contract for components.**
- Keep design patterns consistent within each layer
- Do not mix different types of components in the same logical layer.
- Keep the data format consistent within a layer or component
- **A component /object should not rely on internal details of other components/objects.**
- **Do not overload the functionality of a component.**
- Keep crosscutting code abstracted from the application business logic as far as possible
- Establish a coding style and naming convention for development.
- Maintain system quality using automated QA techniques during

- Consider the operation of your application

# Design example





# Design Example: Simple Enterprise Search (SES)

## Description:

SES is software of search information within an enterprise data.

## Requirements (plain style):

Users from our company should be able to search across office document. Initially documents can be stored on external storages (ftp, corporative web server, windows shared folder). Also it should be possible to upload document directly to system. Only managers and dedicated persons should be able to search across finance documentation. All our user have browser

# Trying to rewrite requirements

UR01. Users should be able to search accross documents.

UR02. Users can have different access right.

UR03. Document types: Word, XLS.

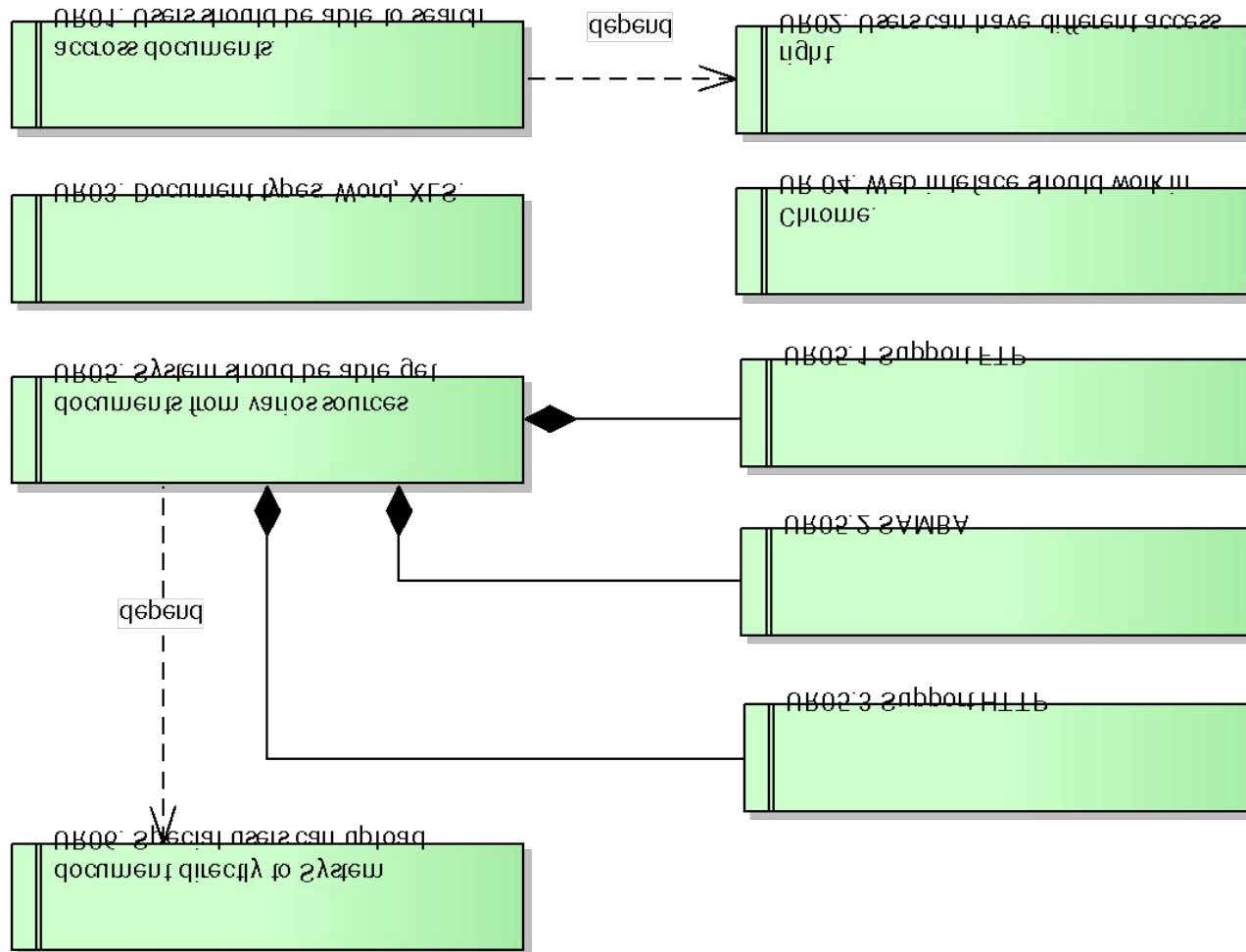
UR04. Web inteface should work in Chrome.

UR05. System should be able to get document via:

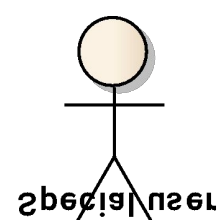
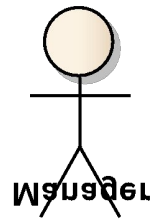
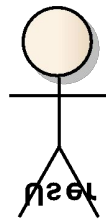
1. FTP
2. SAMBA
3. HTTP

UR 06. Special users can upload document directly to System

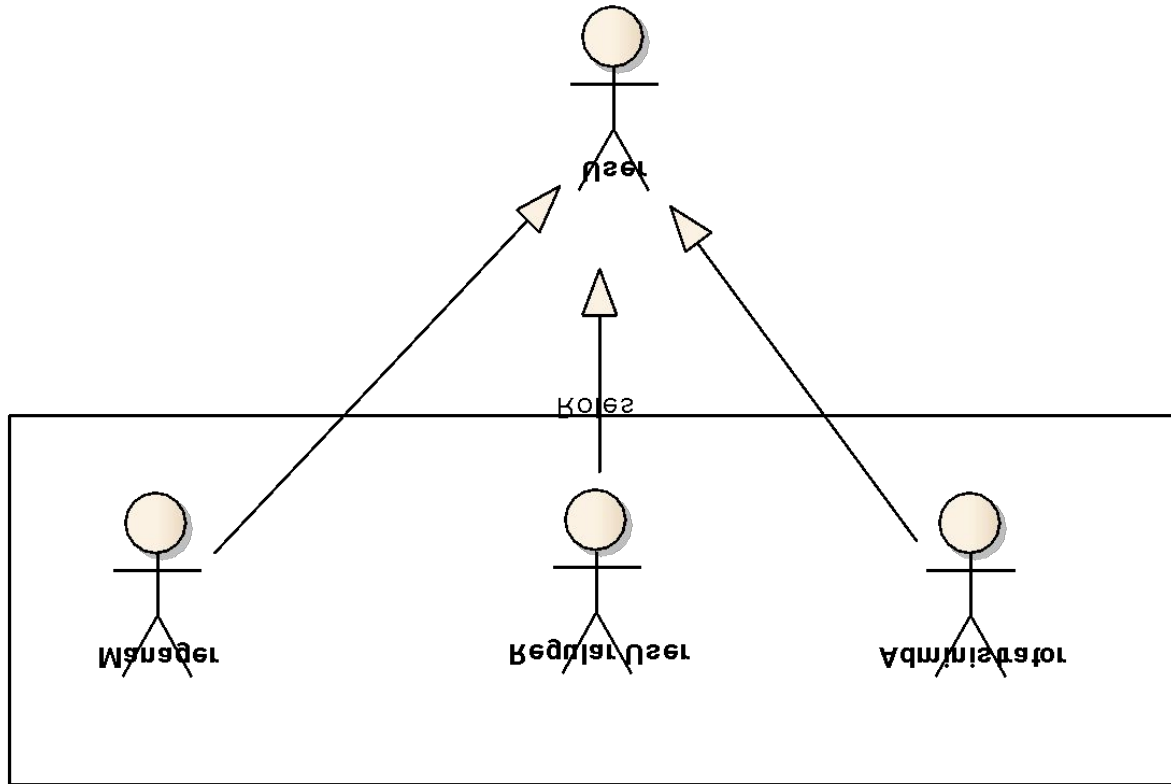
# Trying to rewrite requirements



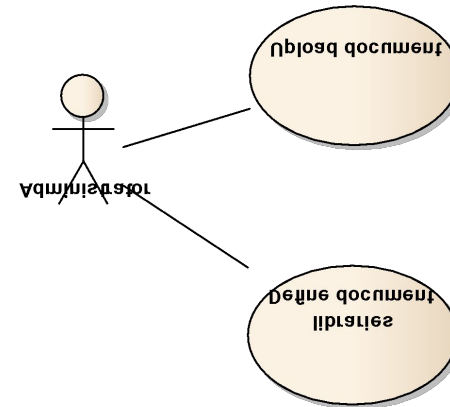
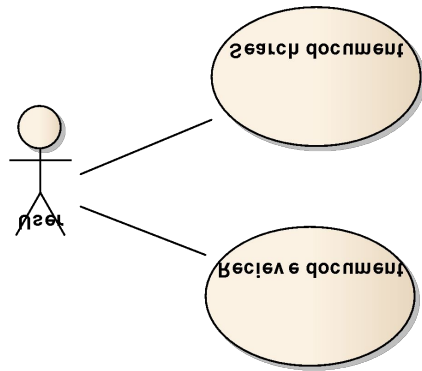
# Looking for Actors



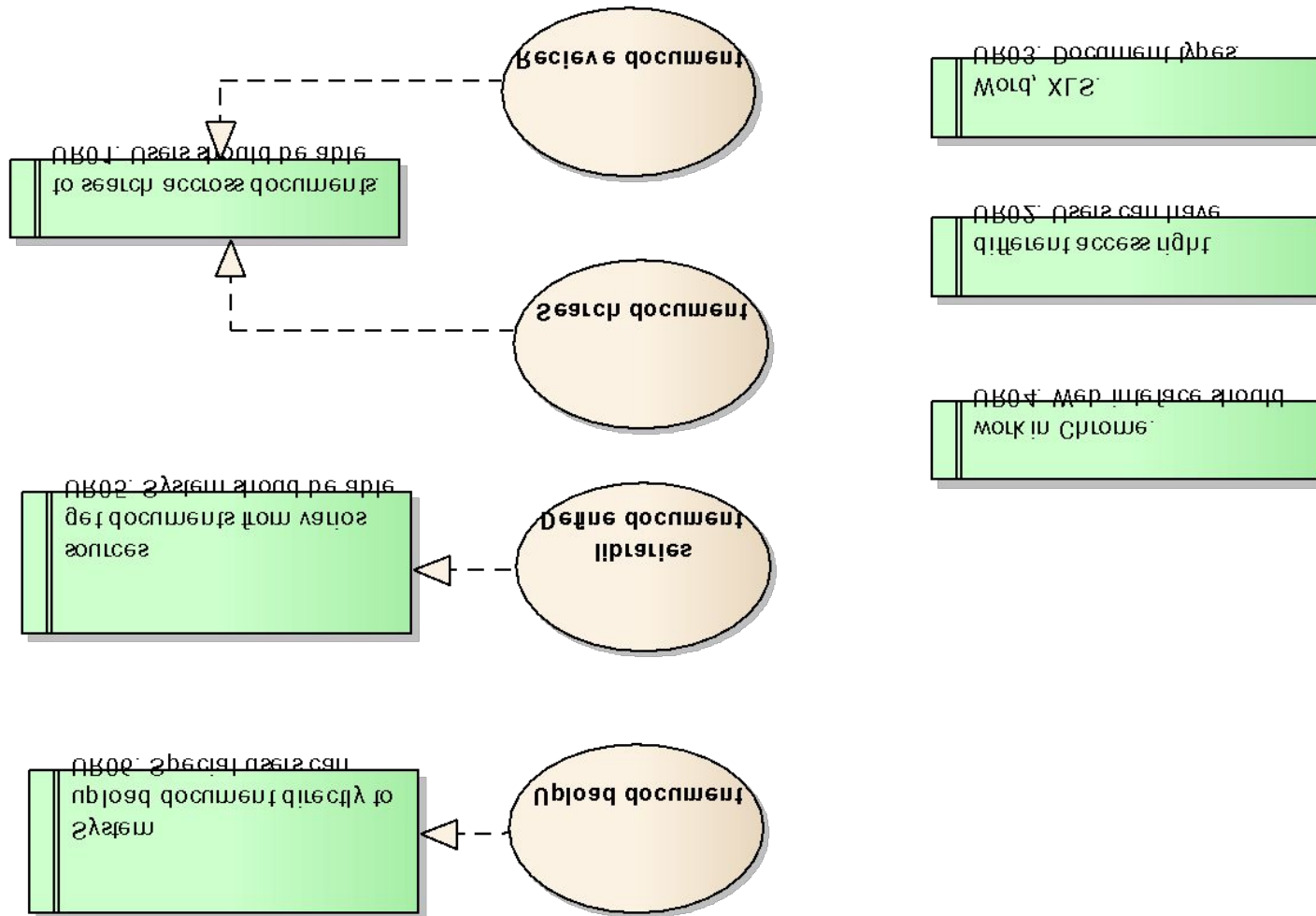
# Actors generalization



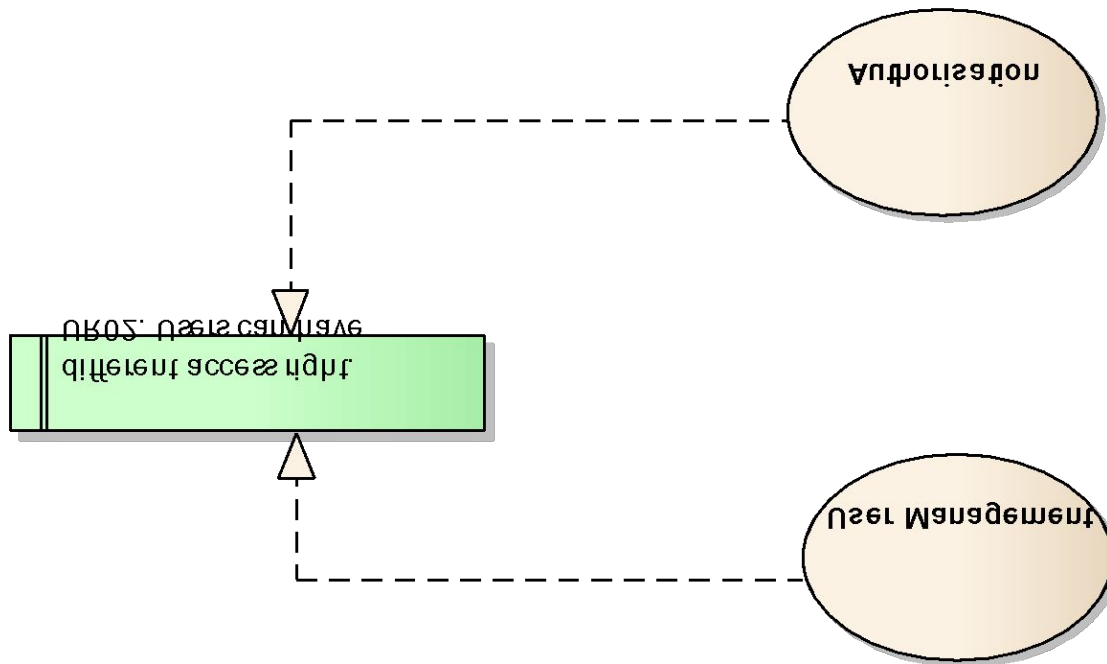
# UseCase analyzes



# UseCase analyzes. Traceability

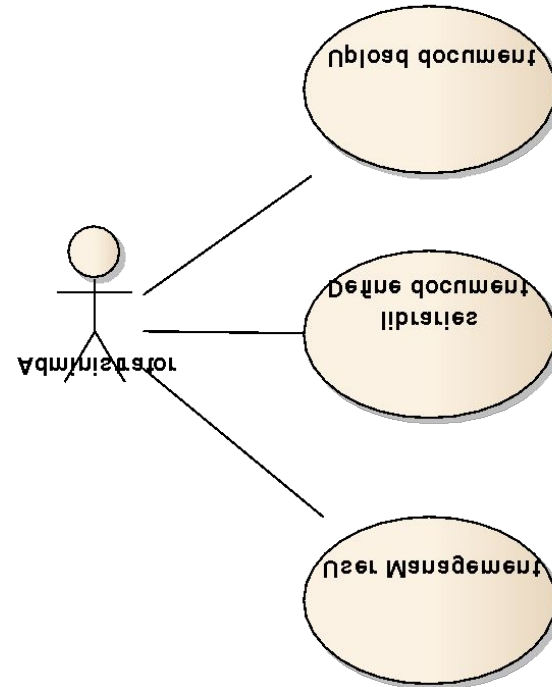
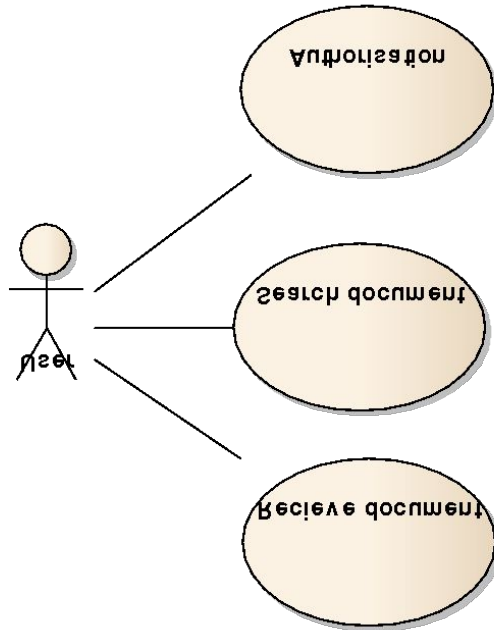


# UseCase analyzes. Traceability

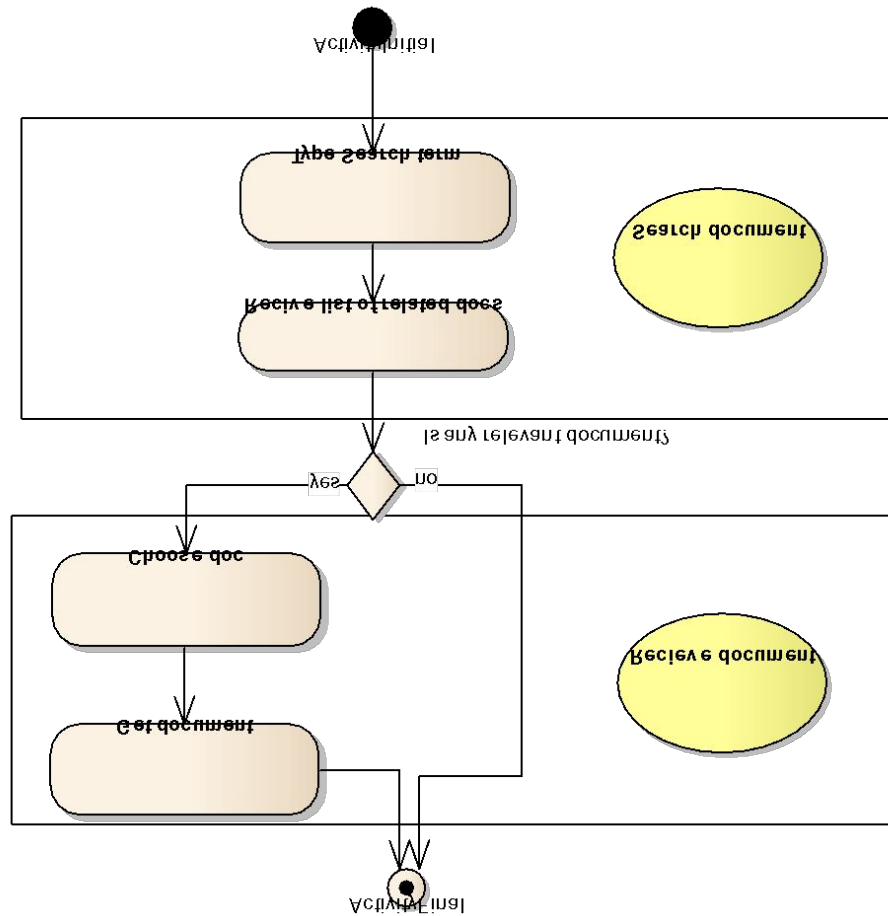




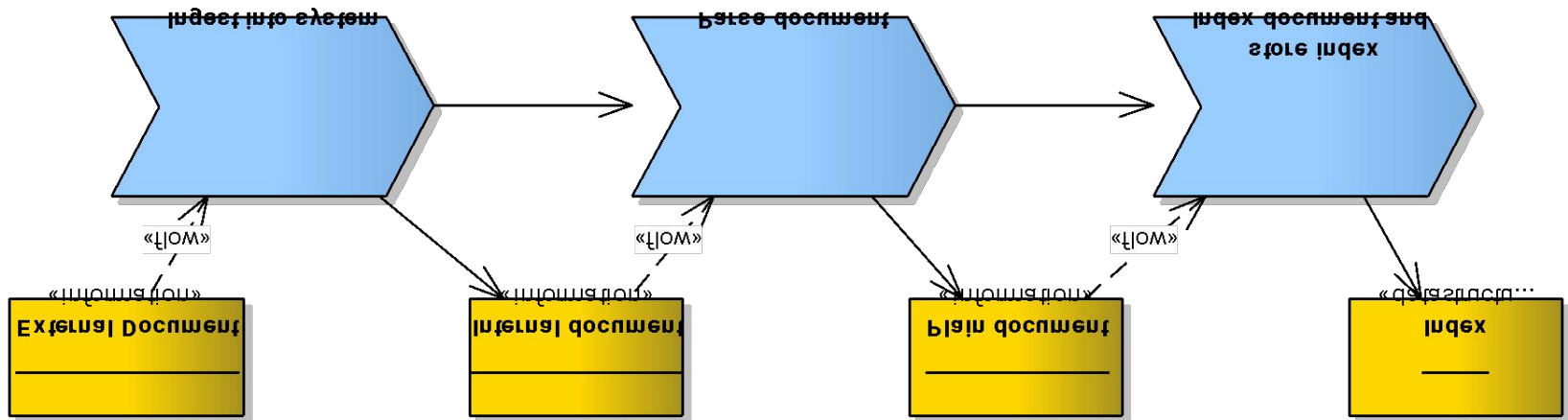
# UseCase analyzes. Finally



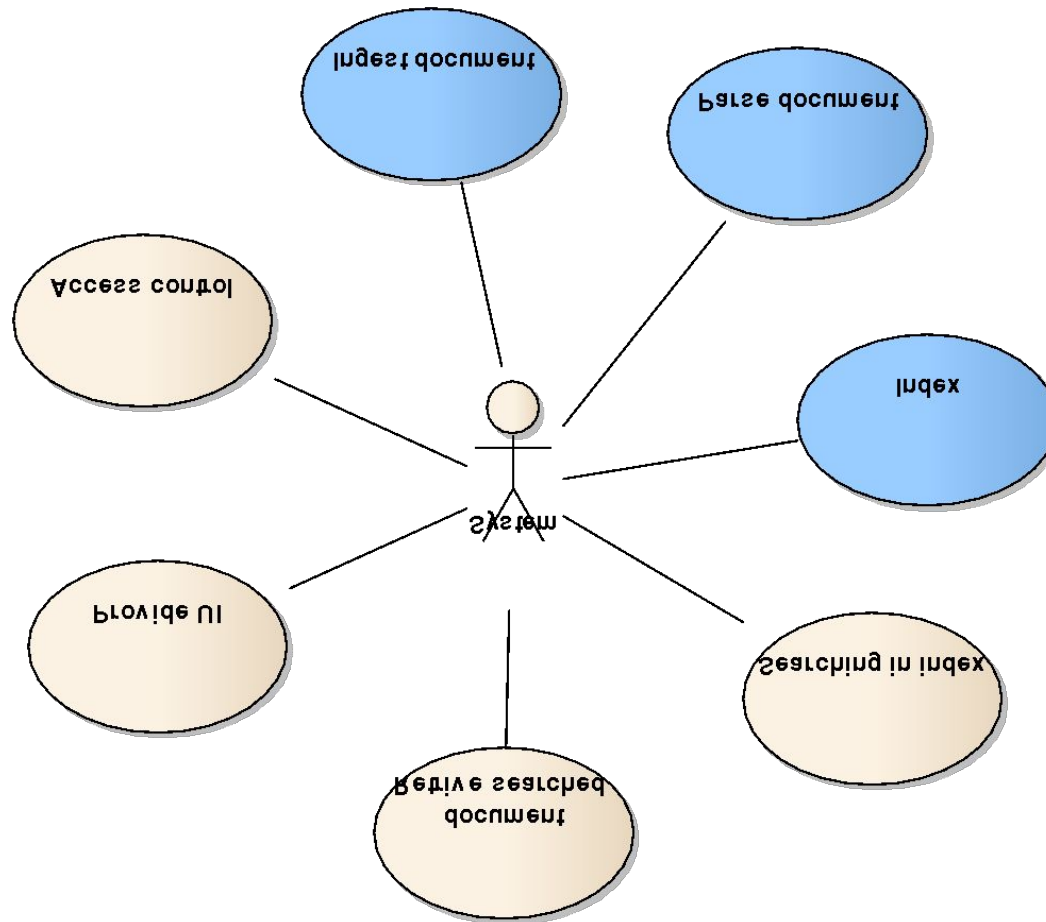
# Activity



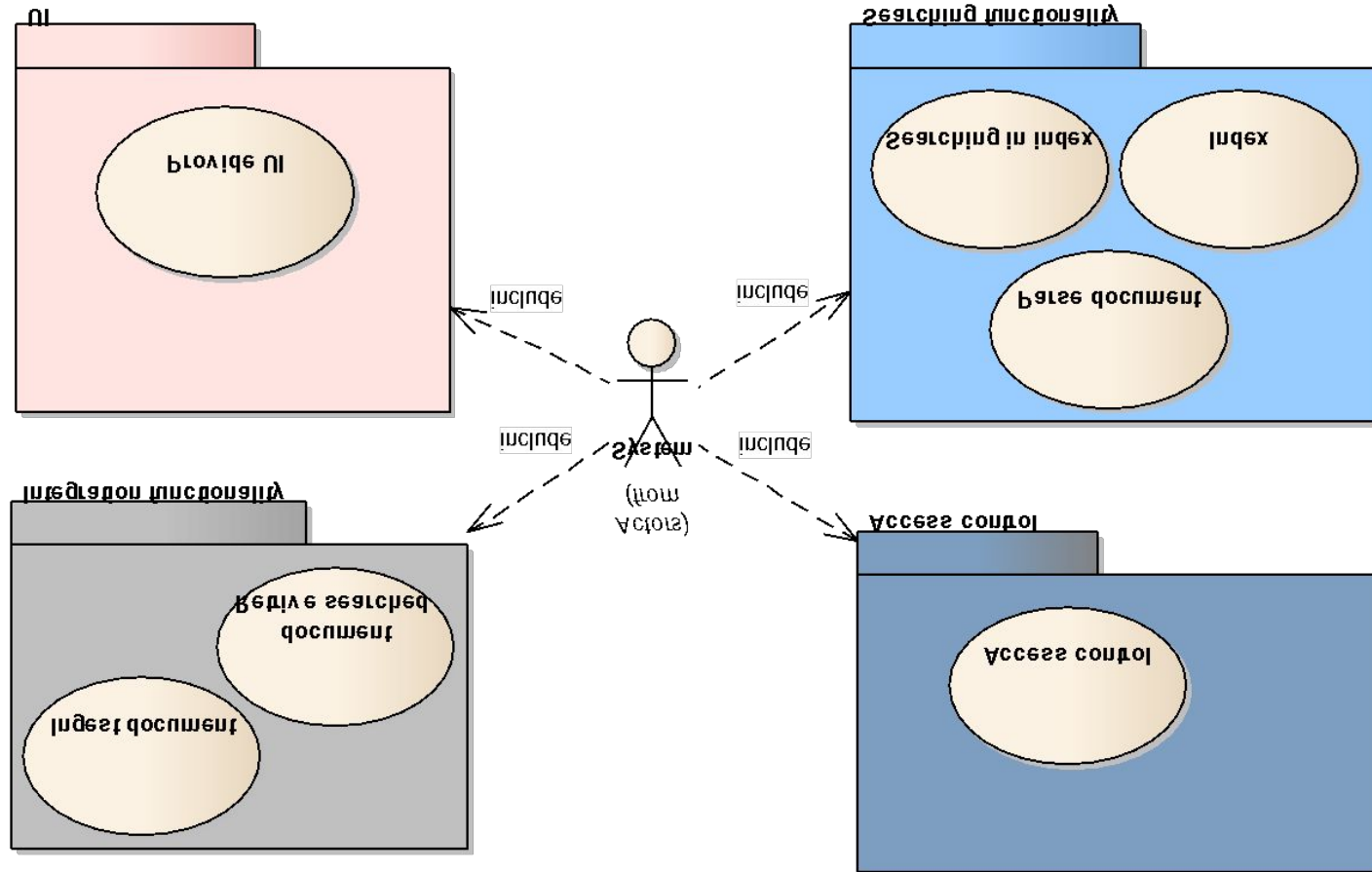
# System analyses



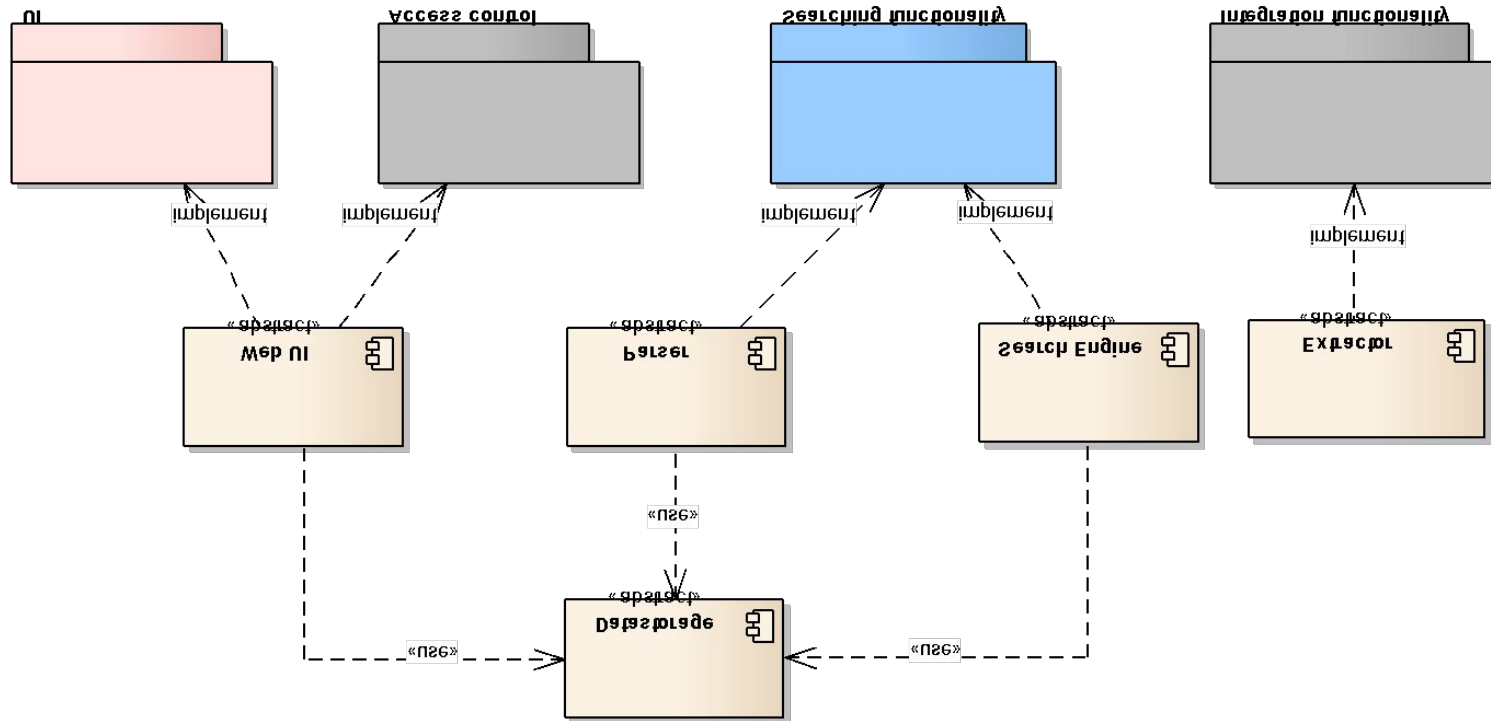
# System UseCases



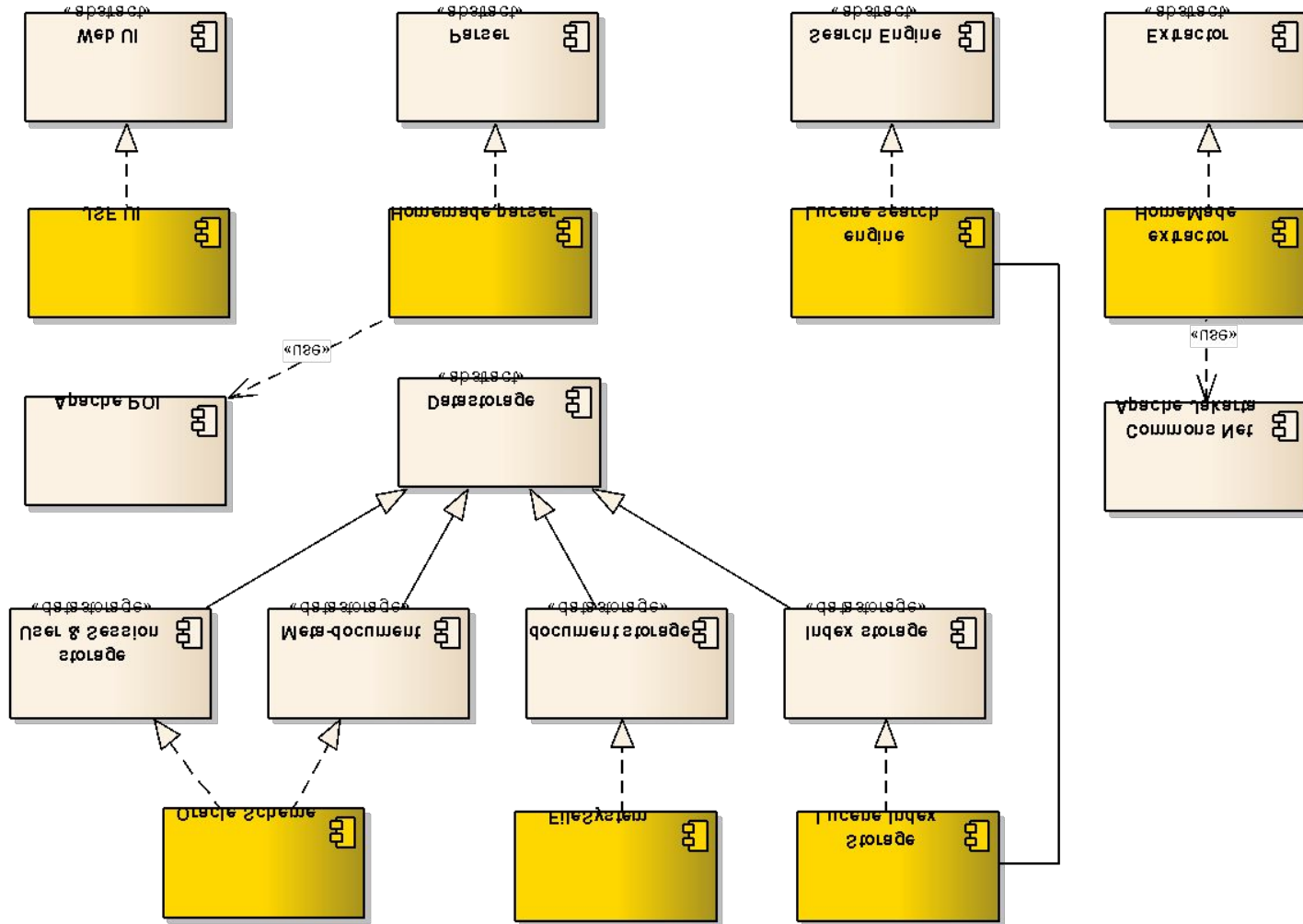
# System Functional decomposition



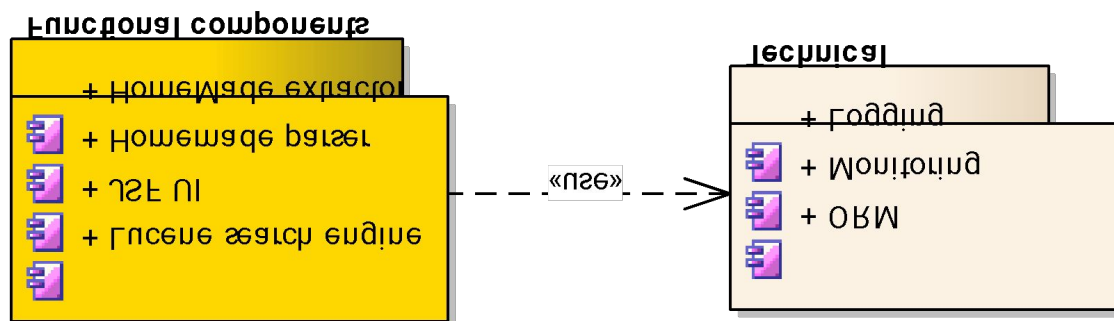
# System Components: A-architecture



# System Components

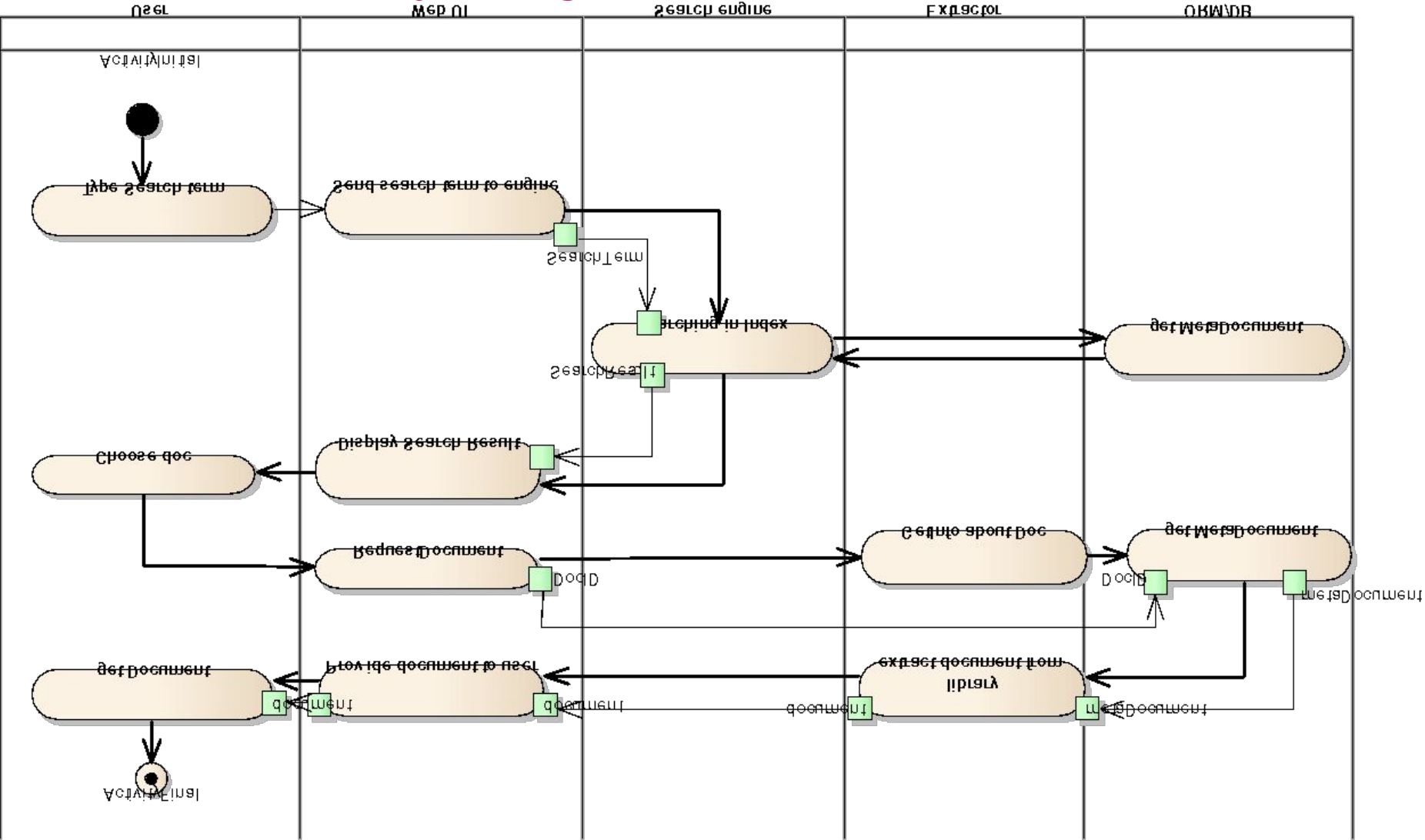


# Application Components + Technical components

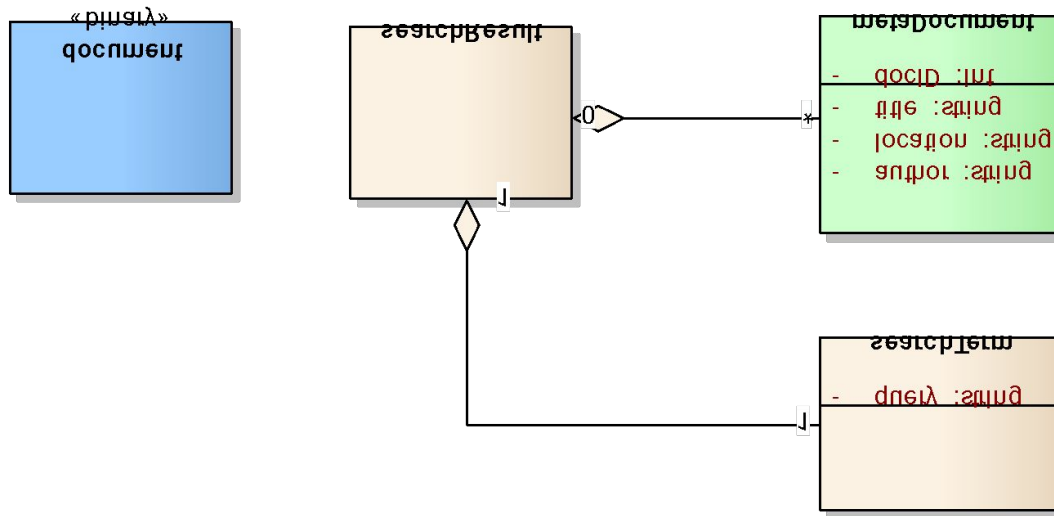




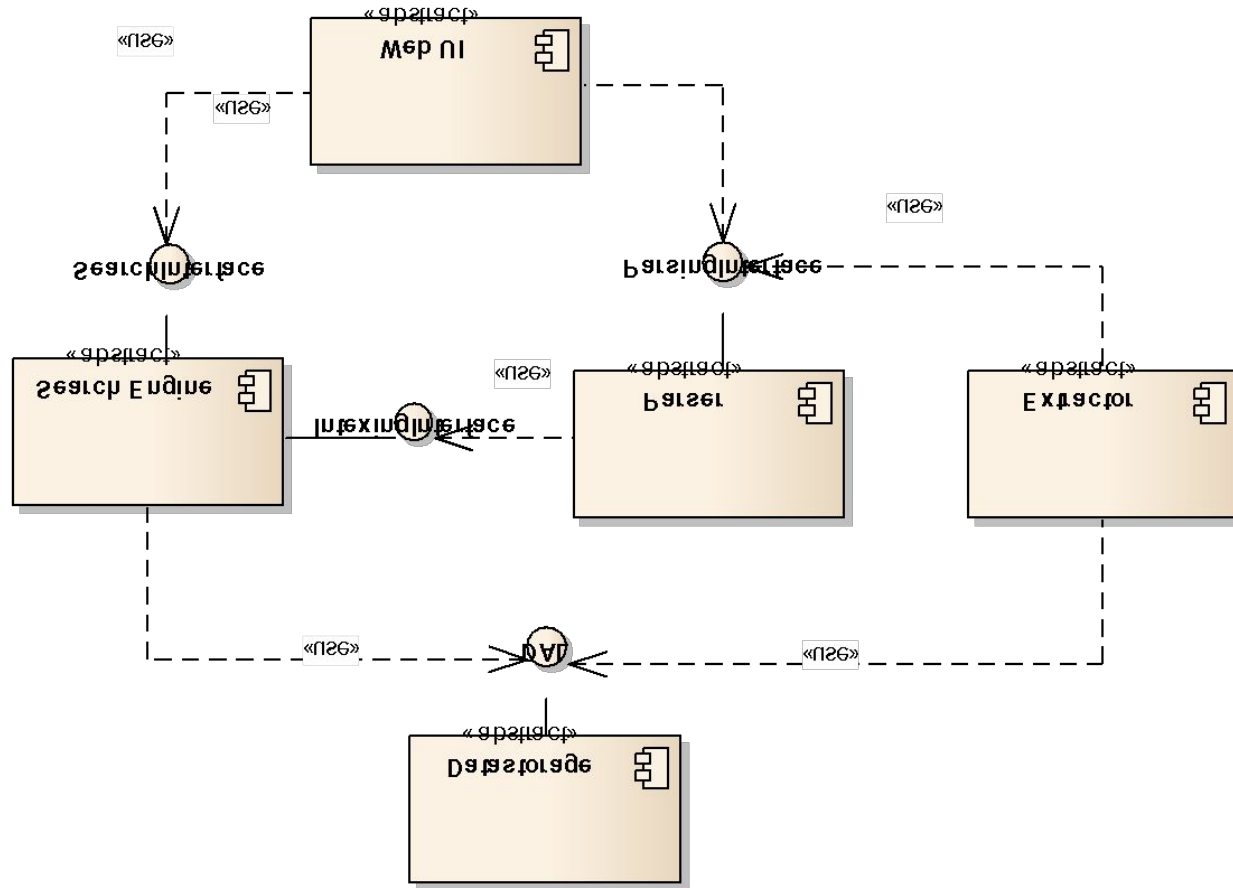
# Detailed Activity Diagram



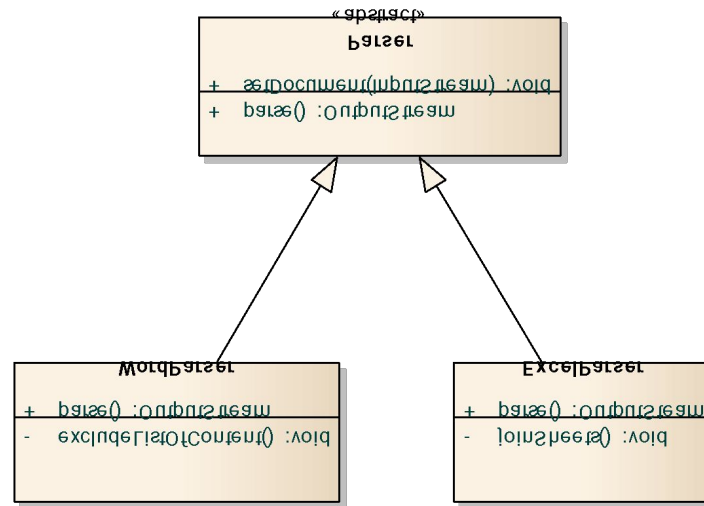
# Entity Classes



# Interfaces

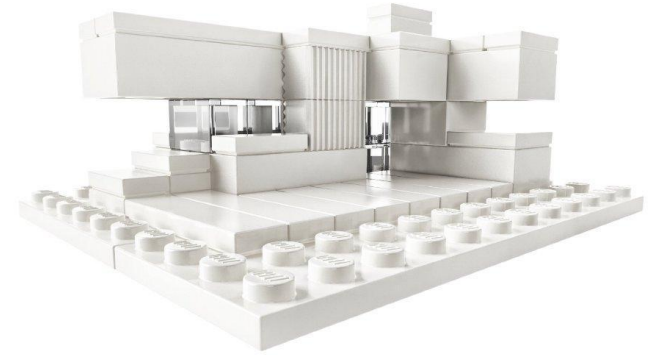


# Behavior classes



# Questions?

Design example





# Coffe

10 minutes  
18:50



LIFE IS FOR SHARING.

# Multilayer application pattern



# Multilayered architecture: Why?

## Benefits from the box

- Abstraction
- Isolation
- Manageability
- Performance
- Reusability
- Testability



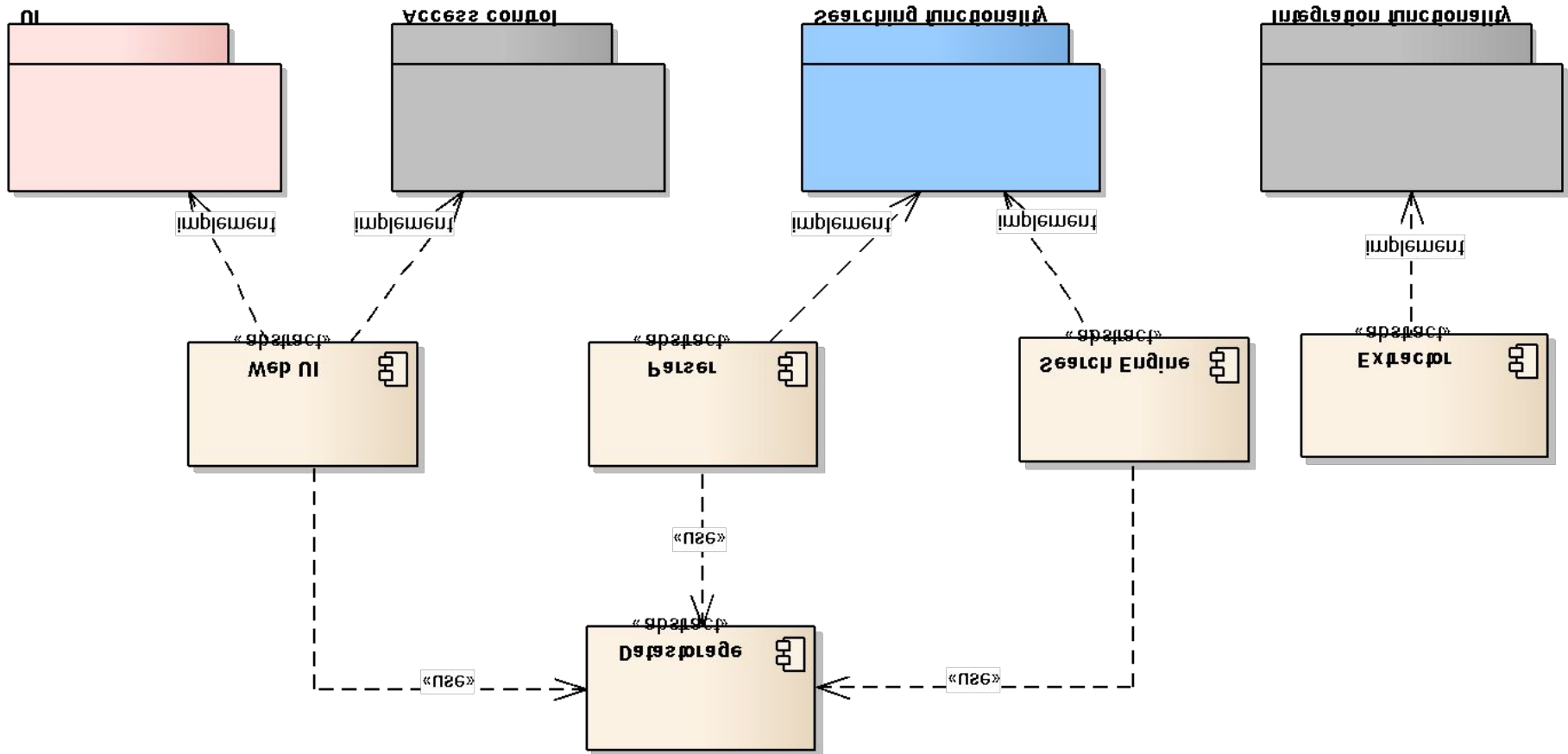
# 3-tier Architecture

Presentation layer

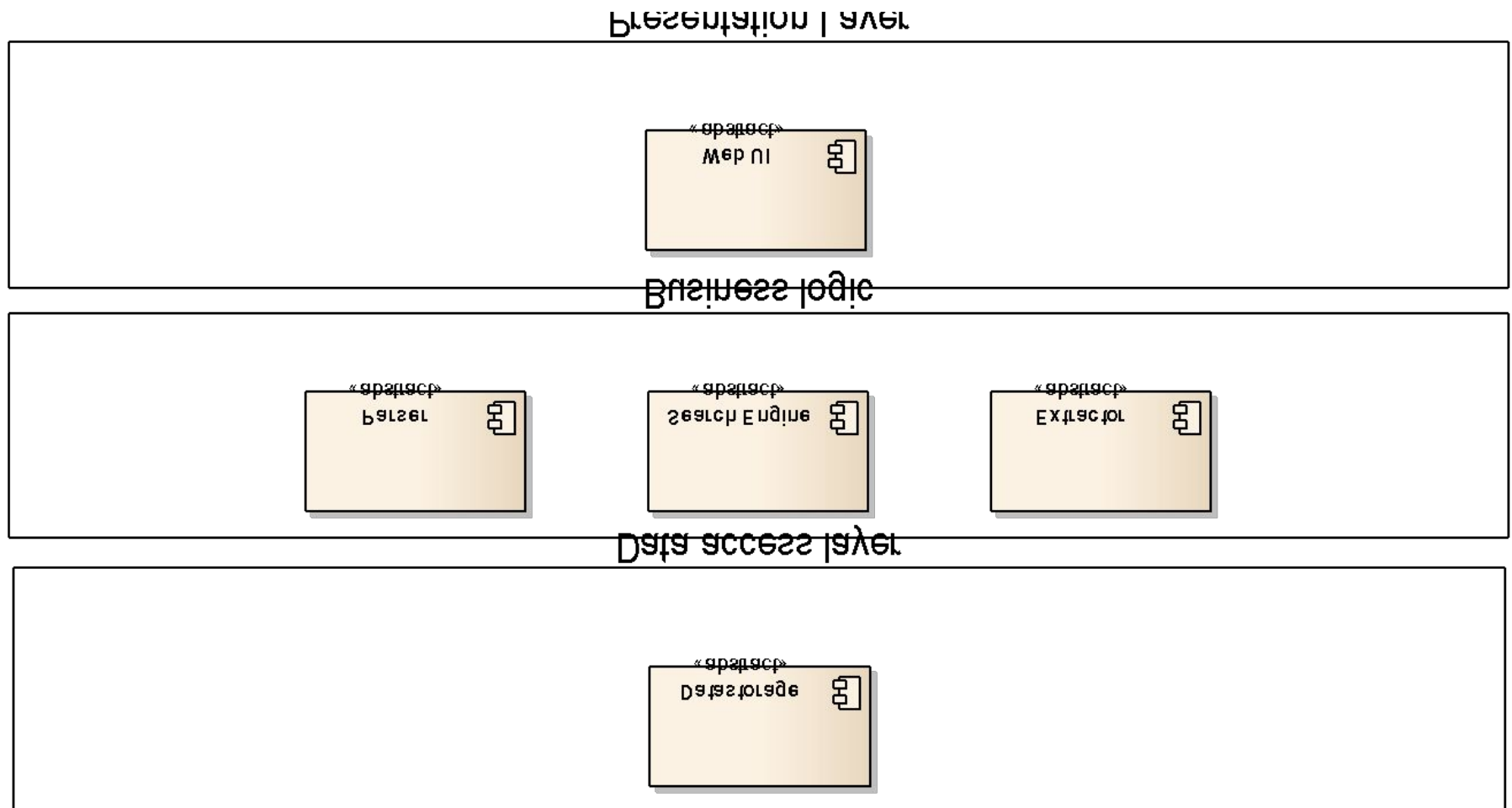
Business Logic layer

Data Access Layer

# 3-tier Architecture: our application



# 3-tier Architecture: our application



# n-tier Architecture

*Presentation layer*

*Service Layer*

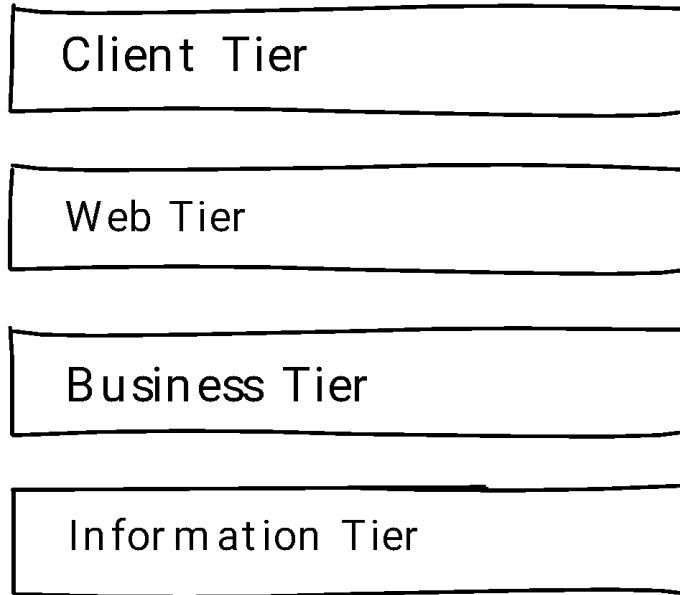
*Business Logic layer*

*Infrastructure layer*

*Data Access Layer*

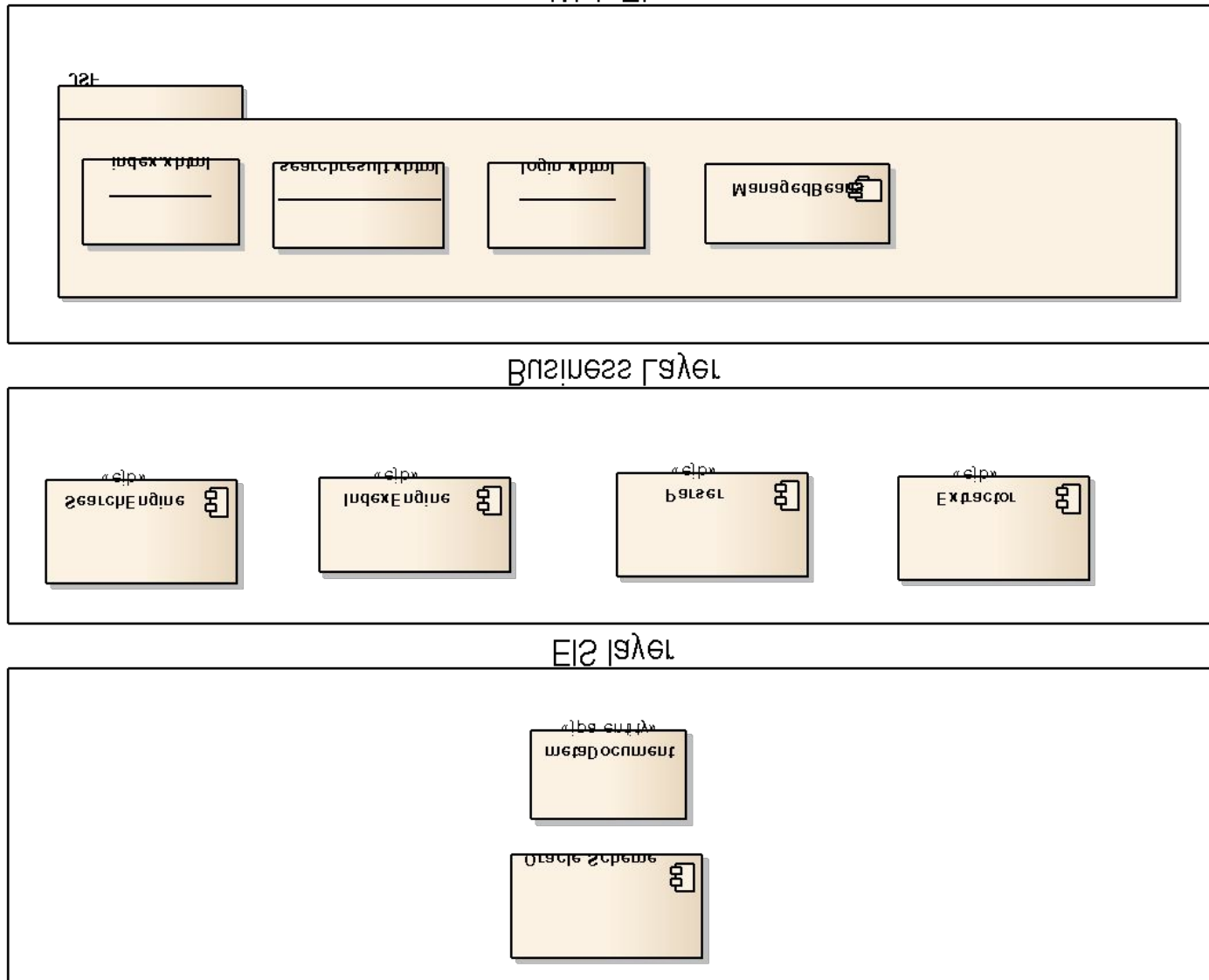
*Persistence layer*

# n-tier Architecture: JEE



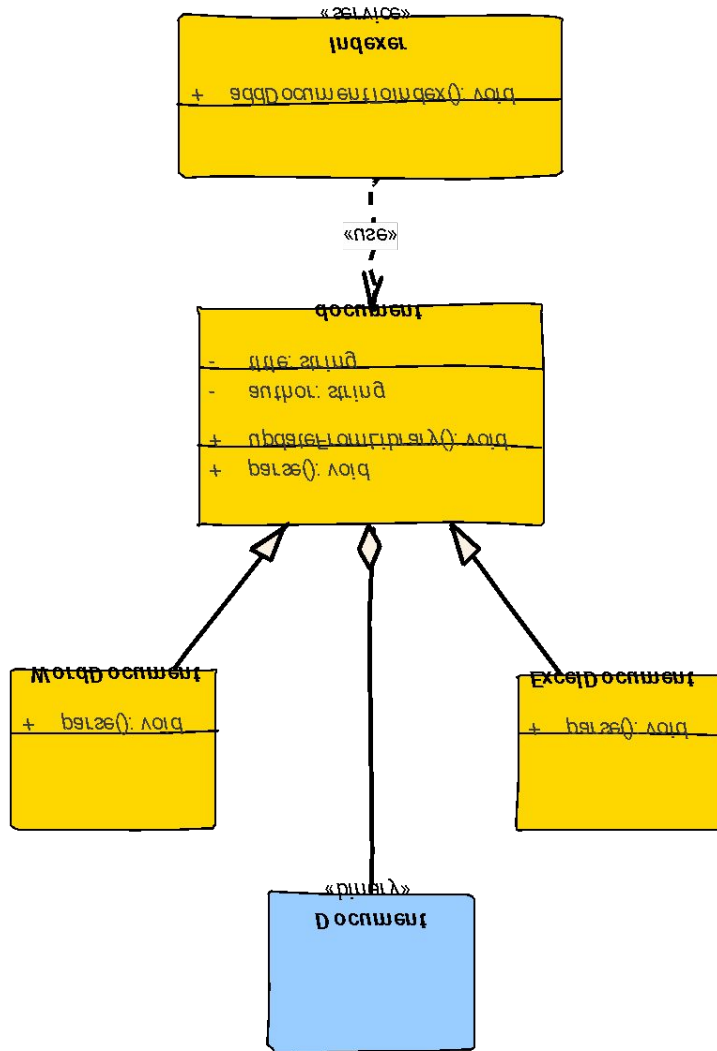
- Rich application or JS/Html application
- JSP / JSF / Struts / Spring MVC ...
- EJ  
B
- JPA + Database

# n-tier Architecture: JEE



# Alternatives

## Domain Driven Design



### Specific

- No strict layers
- Entity and their behavior combined

### Pro

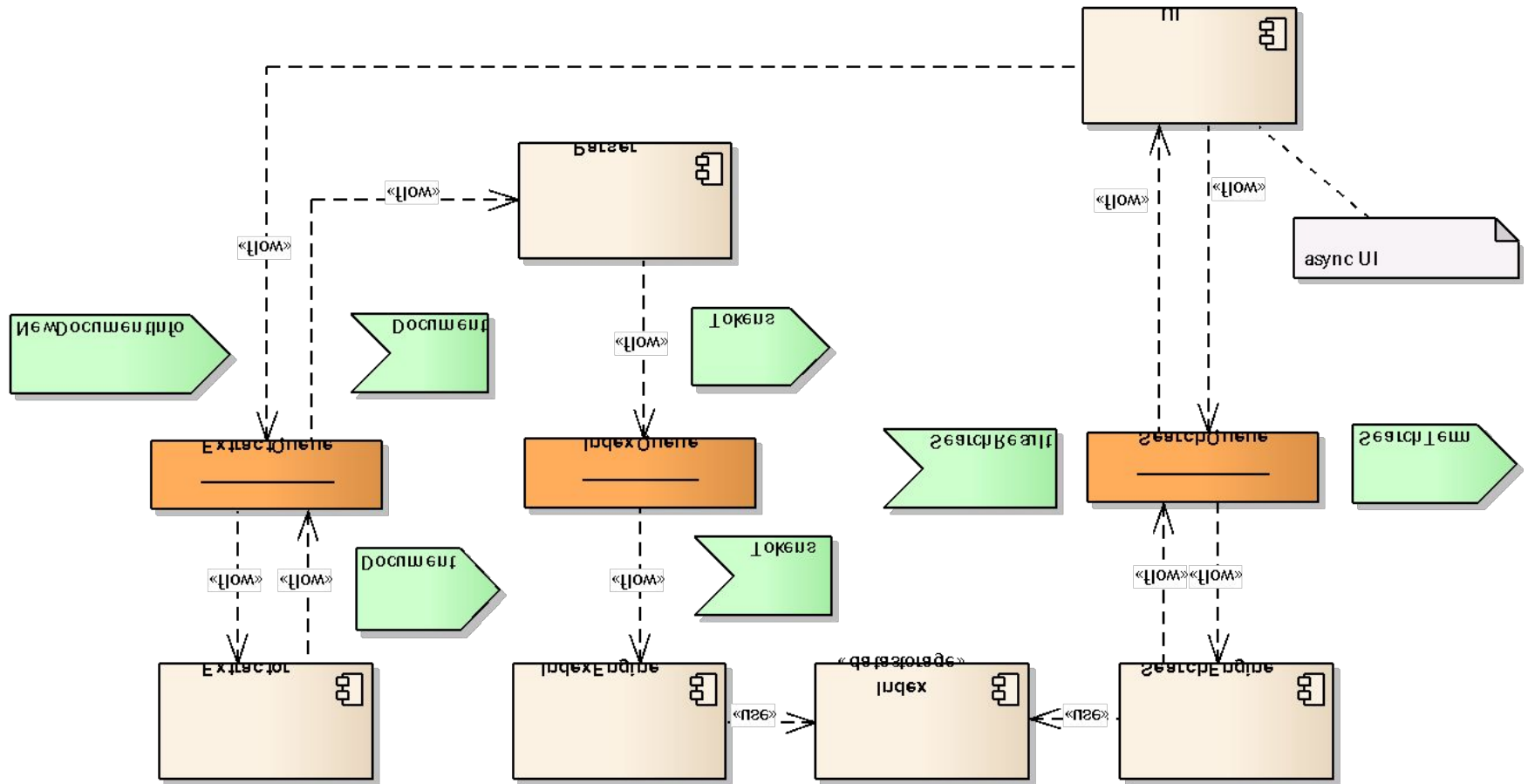
- Easy to extend domain model
- Easy to test
- Easy to map

### Cons

- Hi-efforts/costs

# Alternatives

## Event Based

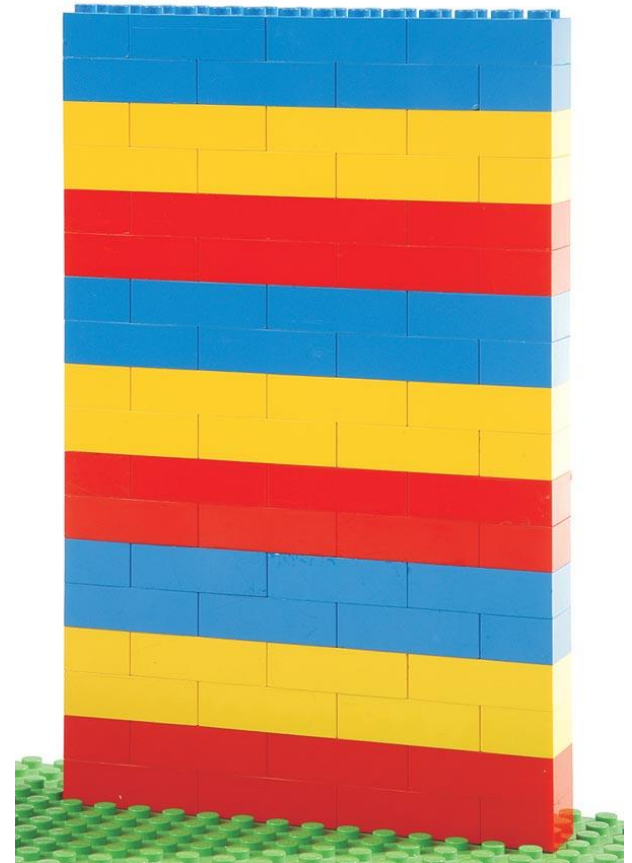




# Important

Please start your enterprise application from n-tiers architecture

- It is clear and easy to understand
- It is proven by time
- Most current EA using layered architecture
- JEE designed for layered architecture
- You don't have any strong NFR



# Questions?

Multilayered architecture





# Coffe

15 minutes  
XX:XX

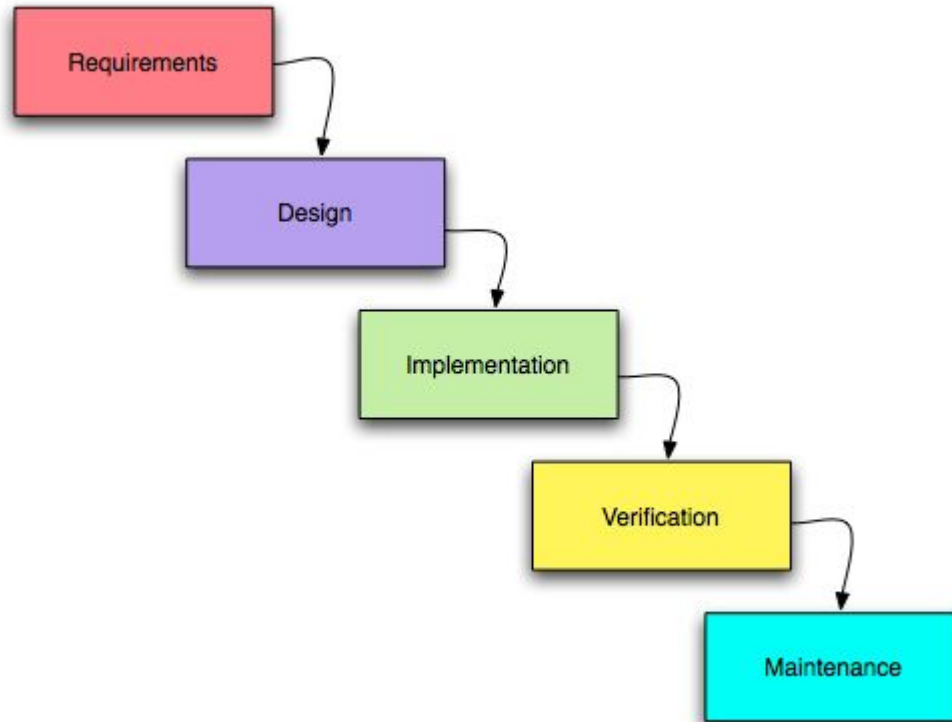


LIFE IS FOR SHARING.

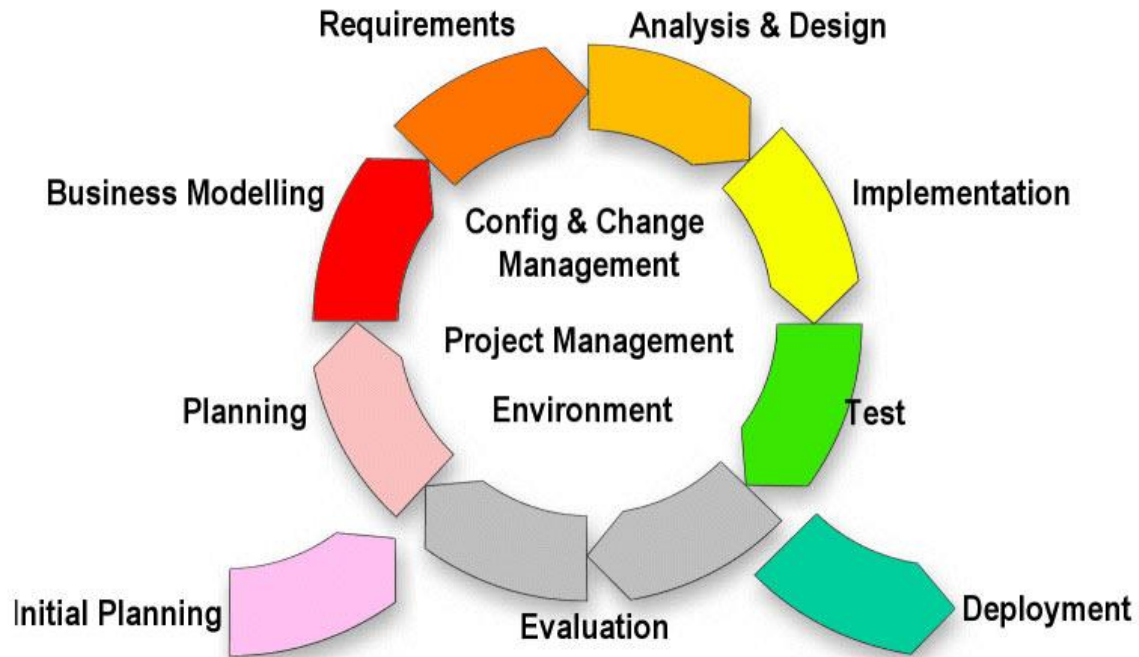
# Architect Role in PLC



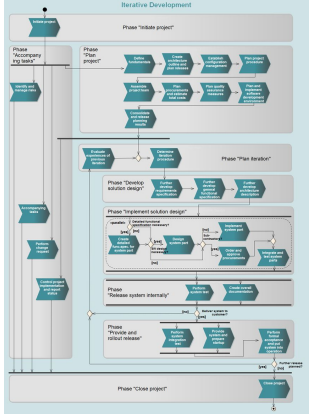
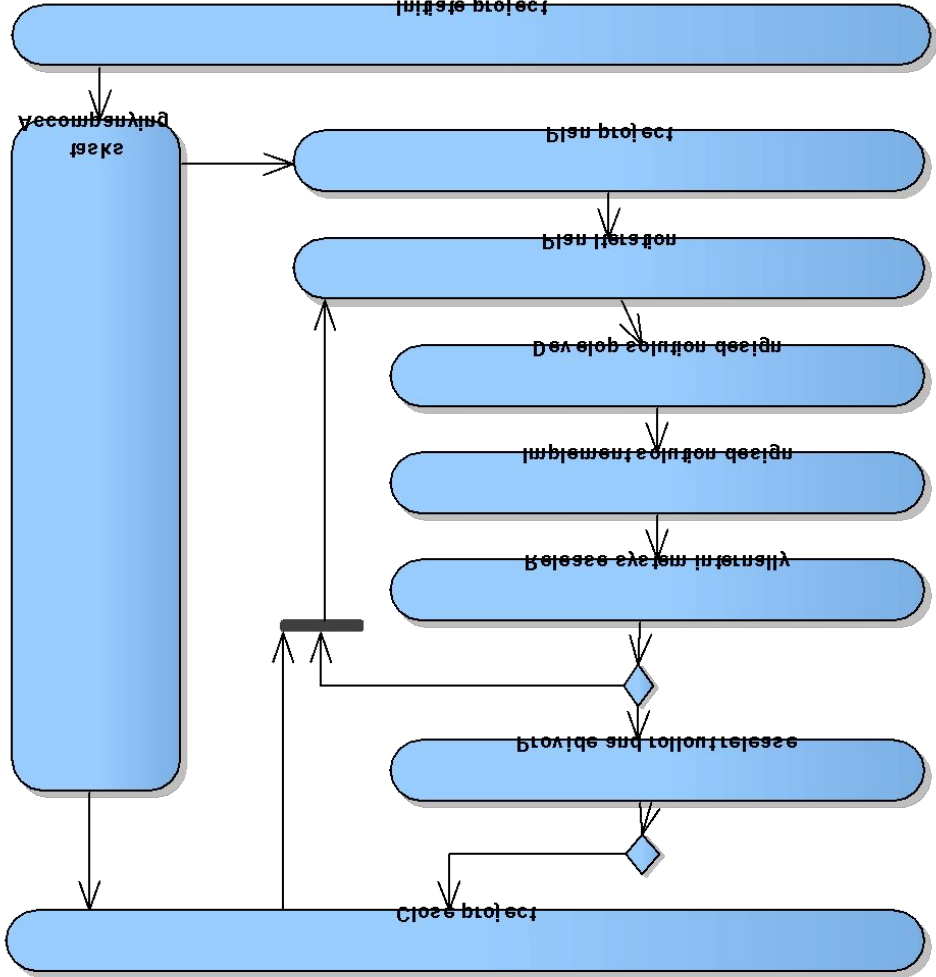
# PLC Waterfall



# PLC RUP



# PLC SE-Book Iterative development



**Details:**

<http://sebook.t-systems.com/en/1116135a14c0b91.html>



# PLC SE-Book Architect in PLC

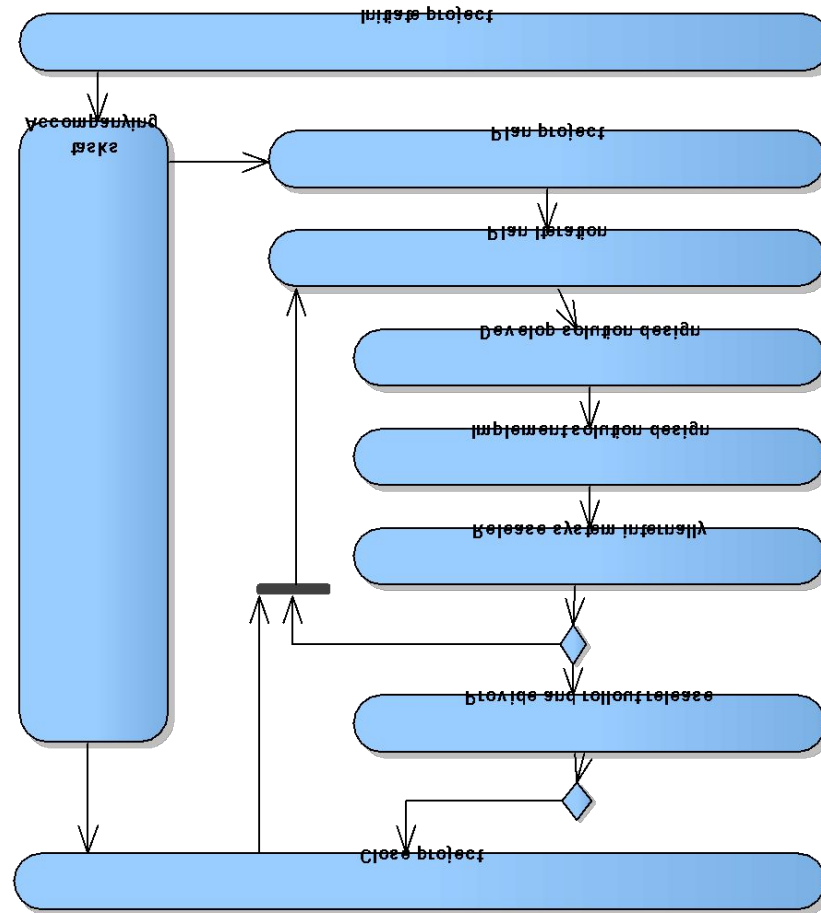


**Discussion: In which project steps Architect should be involved?**



# PLC SE-Book Architect in PLC

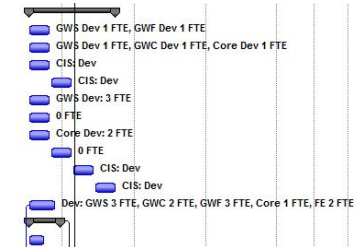
Short answer – Architect or Chief Architect should be involved in almost all project steps.



# PLC SE-Book Architect tasks: Plan project

- Provide effort estimation
- Provide technical risks list
- Validate WBS and Dependencies in Project plan
- Support in Configuration management

432		Software development	80 days
433		DEM1-10597 - CSV Import - Mapping Tabellen Gateway - Interimslosu	20 da
434		Catch All topic - plan need to be finalized	20 da
435		DRV Issues - list to be prepared	20 da
436		DEM1-9720 - I-SP-FE: Enhancements SP Data - GWS Part	20 da
437		DEM1-10041 - I-SP-FE: Enhancements SP Actions - GWS Part	20 da
438		DEM1-8273 - Assignment TO-, CC-, BCC-Header - Gateway	20 da
439		DEM1-10373 - G10 Events for mailboxes without user - Gateway	20 da
440		DEM1-9542 - Migration to new GW: New GW mustn't deliver Mails alre	20 da
441		DEM1-6777 - Multiple Tenants using one Security Token	20 da
442		DEM1-6710 - Definition of multiple De-Mail accounts	20 da
443		Bugfix	25 days
444		Module Test	30 da
445		Creation	15 da

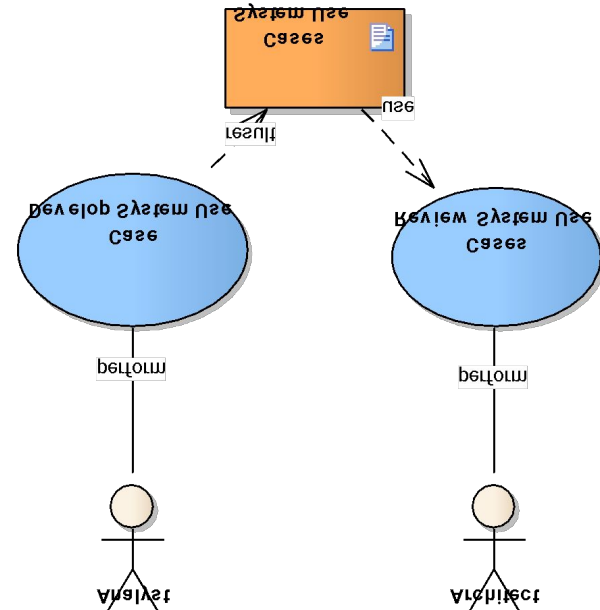


## WBS - Work Breakdown Structure

# PLC SE-Book Architect tasks

## Requirements analyses

- Support requirements development
- Validate and review
  - System Use Cases
  - Requirements
  - GUI Prototype
  - Interface agreement
  - Traceability matrix (UC vs Req)
  - Logical Data Model



**explained below**

# Example: Traceability Matrix Bonus

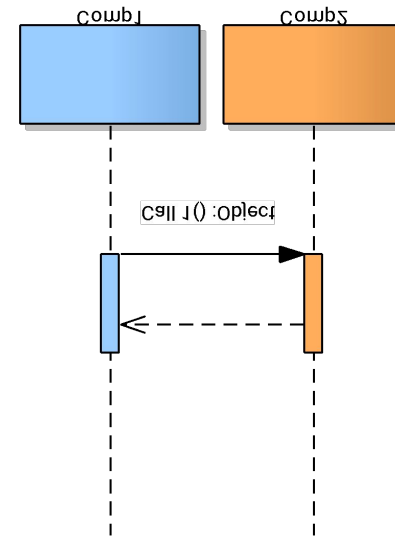
Relationships: - direct only	<b>UC2: Check Order...</b> Check Order Status	<b>UC2.2: Basic Flow</b> BEGIN The use case starts when the Shopper chooses to check on the status of a previous order. IDENTIFY SHOPPER The system requests information to...	<b>UC2.3: TRACK...</b> TRACK PACKAGES At BF VIEW ORDER DETAILS, the order has already shipped; the Shopper chooses to view tracking information for the order. The...	<b>UC3.1</b> A Shop selecte consid purcha case c purcha allowin
<b>FEAT1: Secure payment...</b> Secure payment method				
<b>FEAT2: Easy browsing</b> Easy browsing for available titles				
<b>FEAT4: Ability to check...</b> Ability to check the status of an order				
<b>FEAT8: Shopper...</b> Shoppers should be able to register once for all...				
<b>FEAT9: Shipping Status</b> Shoppers should be able track any package that has...				
<p>FEAT1: Secure payment method</p> <p>UC2: Check Order Status</p>				

**Rational RequisitePro**

# PLC SE-Book Architect tasks

## Develop solution design

- Design:
  - Database
  - Components
  - Interfaces
- Review
  - Test Plan / Test Specification
- Provide/support:
  - Prototyping
  - Traceability (UC vs Comp)



**Know your onions**

# PLC SE-Book Architect tasks: Implementation

- Create program
- Test program
- Defect fixing
- Ensure Code Quality
- Refactoring
- Align code and architecture



**Yes, we can code, at least after worktime**

# PLC SE-Book Architect tasks: Test

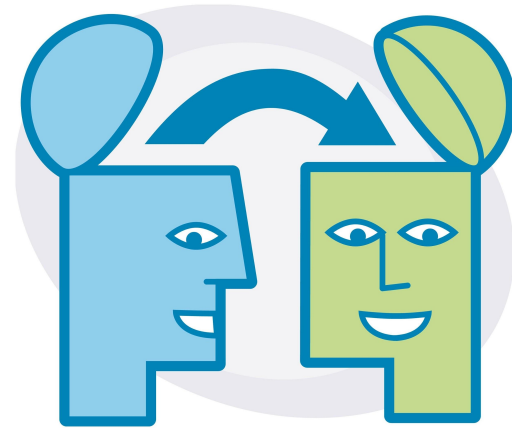
- Review test strategy
- Should be involved in Critical defect analyses



**We hate QA activities, but we do it.**

# PLC SE-Book Architect tasks: Rollout

- Prepare Transfer to Operation
- Support Productive Operation



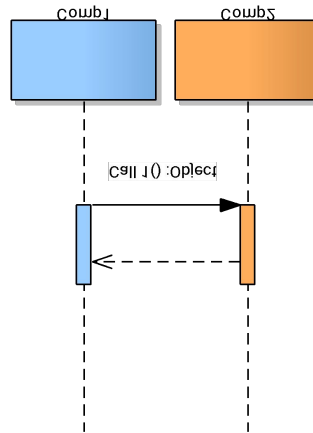
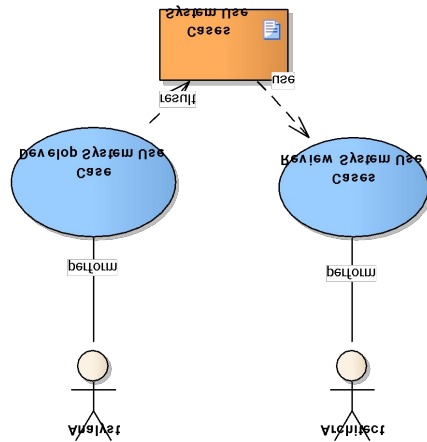


# PLC SE-Book Architect tasks: Close project

- Determine and Analyze KPIs and Derive Measures

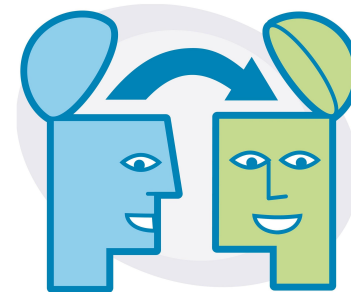


# PLC SE-Book All activities should be documented



432		Software development	80 days
433		DEM1-10597 - CSV Import - Mapping Tabellen Gateway - Internalisu	20 day
434		Catch All topic - plan need to be finazed	20 day
435		DRV Issues - list to be prepared	20 day
436		DEM1-9720 - I-SP-FE: Enhancements SP Data - GWS Part	20 day
437		DEM1-10041 - I-SP-FE: Enhancements SP Actions - GWS Part	20 day
438		DEM1-8273 - Assignment TO - CC - BCC-Header - Gateway	20 day
439		DEM1-10373 - G10 Events for mailboxes without user - Gateway	20 day
440		DEM1-9542 - Migration to new GW: New GW mustn't deliver Mails air	20 day
441		DEM1-6777 - Multiple Tenants using one Security Token	20 day
442		DEM1-6710 - Definition of multiple De-Mail accounts	20 day
443		Bugfix	25 days
444		Module Test	30 day
445		Creation	15 day

	GWS Dev 1 FTE, GWF Dev 1 FTE
	GWS Dev 1 FTE, GWC Dev 1 FTE, Core Dev 1 FTE
	CIS Dev
	CIS: Dev
	GWS Dev: 3 FTE
	0 FTE
	Core Dev: 2 FTE
	0 FTE
	CIS: Dev
	CIS: Dev
	Dev: GWS 3 FTE, GWC 2 FTE, GWF 3 FTE, Core 1 FTE, FE 2 FTE



Many many documents...

# Questions?

## Architect in PLC

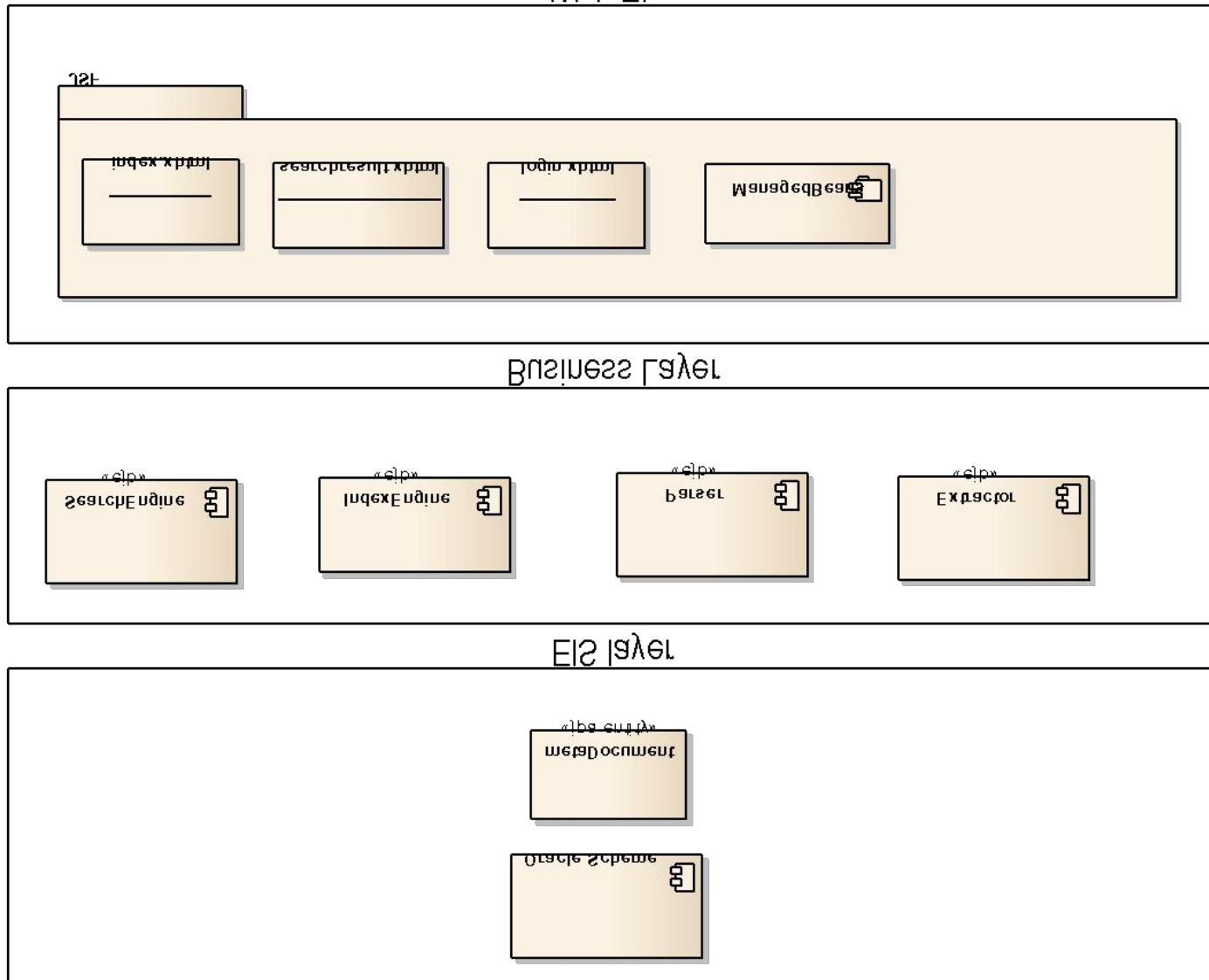


# Resume



LIFE IS FOR SHARING.

# n-tier Architecture: JEE



# Key architecture principles and practices

## Common design practices

- Prefer composition to inheritance
- **Separate the areas of concern between layers**
- Be explicit about how layers communicate with each other.
- Keep design patterns consistent within each layer
- Do not mix different types of components in the same logical layer.
- Keep the data format consistent within a layer or component
- **A component or an object should not rely on internal details of other components or objects.**
- **Do not overload the functionality of a component.**
- Keep crosscutting code abstracted from the application business logic as far as possible
- **Define a clear contract for components.**
- Establish a coding style and naming convention for development.
- Maintain system quality using automated QA techniques during development.
- Consider the operation of your application.

**Thank you!**



LIFE IS FOR SHARING.

# References





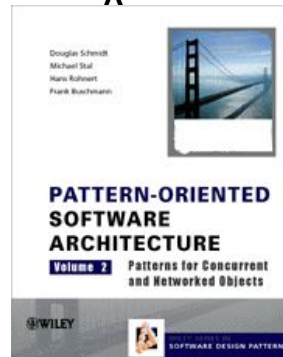
# Sources

## Architecture itself

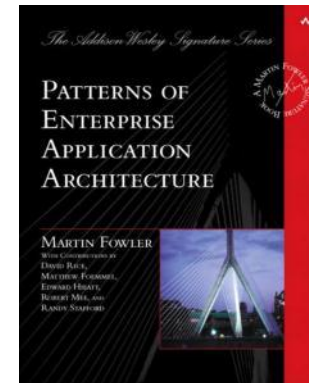
Rozanski&Woo



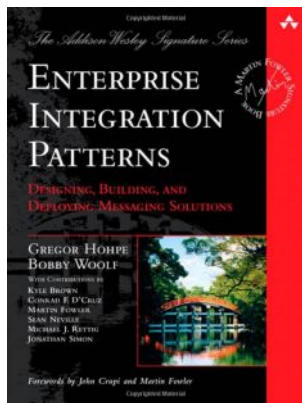
POS



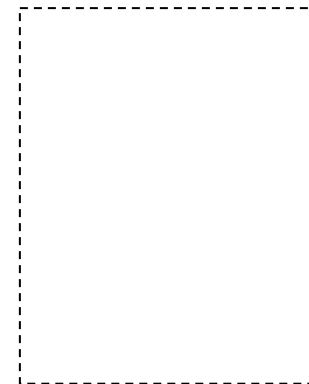
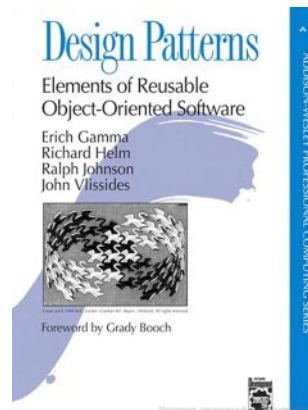
Fowl



EI



GO



# Sources

## Architecture practice / PLC aspects

- CMMI® for Development, Version 1.3
- Carr, Marvin et al, *Taxonomy-Based Risk Identification*, CMU/SEI-93-TR-006. Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, June 1993.
- Microsoft Application Architecture Guide, 2nd Edition

# Sources

## SOA / EDA

- The Growing Role of Events in Enterprise Applications. Five forces. July 2003, Roy W. Schulte, Gartner
- “Event-Driven Architecture Complements SOA”, by Roy W. Schulte, Yefim V. Natis, July 2003, by Gartner
- “2.0 The Mission and Future of Integration” 2004, Gartner “Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions” by Hohpe G., Woolf B., 2004
- “Applied SOA: Conquering IT Complexity through Software Architecture”, by Yefim V. Natis, May 2005, by Gartner, Inc. “Event-driven architecture” by Hohpe G., 2006

# Sources

## Requirements

- IEEE Recommended Practice for Software Requirements Specifications IEEE Std 830-1998
- Requirements management using IBM Rational RequisitePro / Peter Zielczynski
- WRITING EFFECTIVE USE CASES. Alistair Cockburn

# Alternative view on JEE and Architecture

Pure Simple Java (Антон Кекс - Как нам спасти Java?)



[http://www.youtube.com/watch?v=TSAlj04\\_tkA](http://www.youtube.com/watch?v=TSAlj04_tkA)