Intro to UML

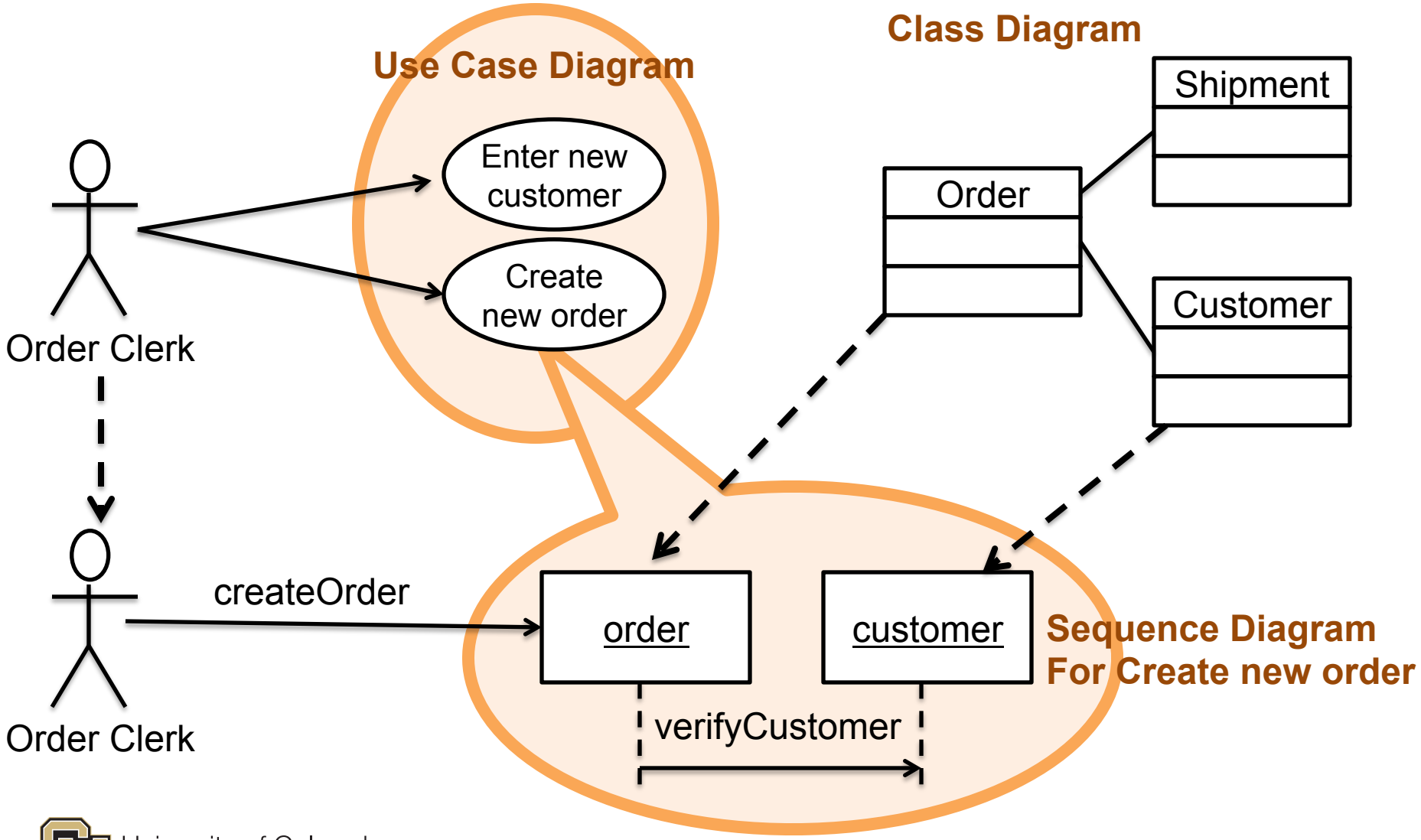
CSCI-4448 - Boese

University of Colorado **Boulder**

Objectives

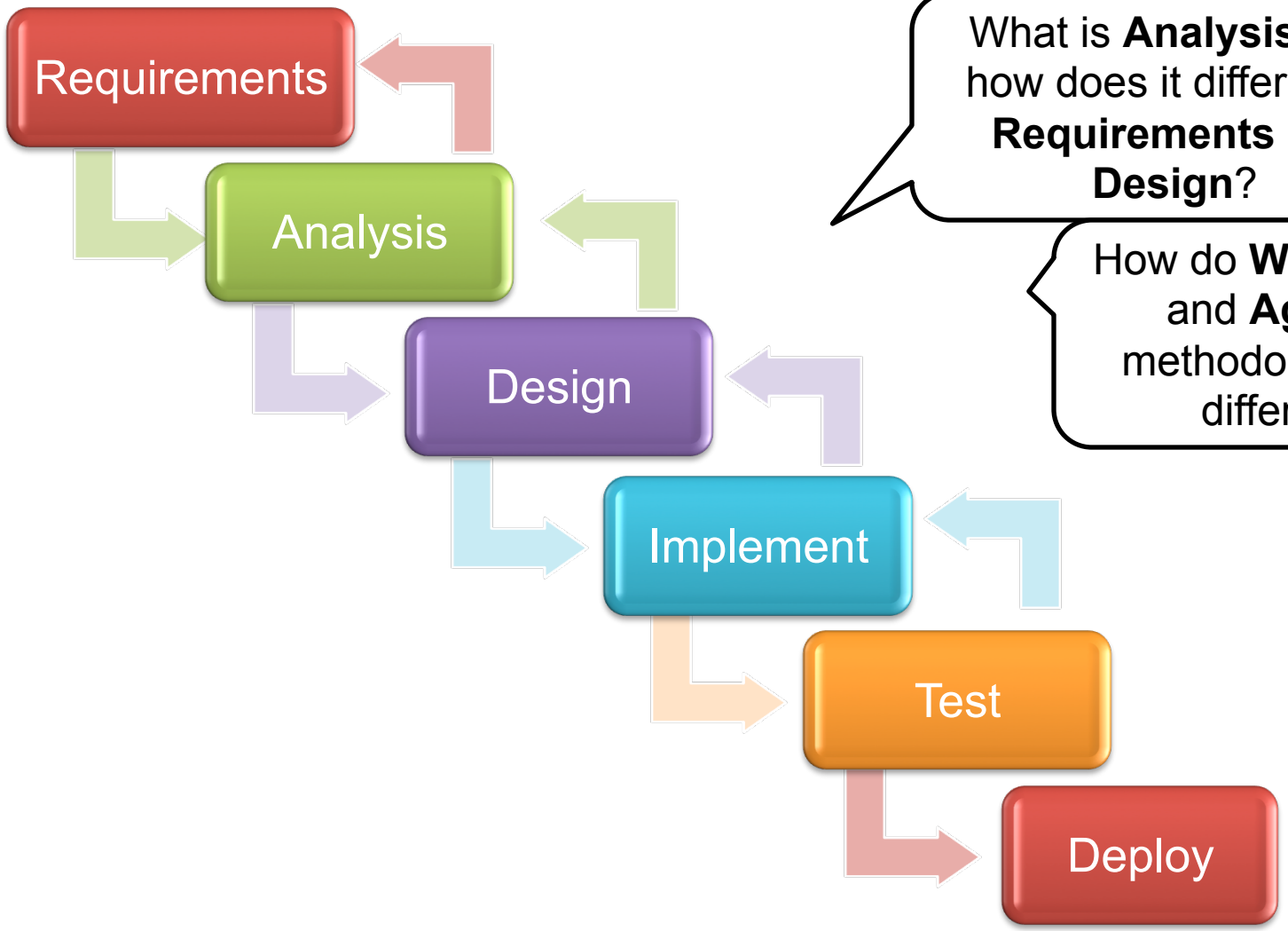
- Understand an overview of SDLC
- Understand how to determine and analyze requirements
- Know how/when to apply Unified Modeling Language (UML)
- Know how to represent requirements/design in diagrams
  - Use Cases: Diagrams and Written
  - Activity Diagram
  - System Sequence Diagram
  - Class Diagrams
  - State Machine Diagram
- Integrating Object-Oriented Models

Diagrams Overview



Overview SDLC

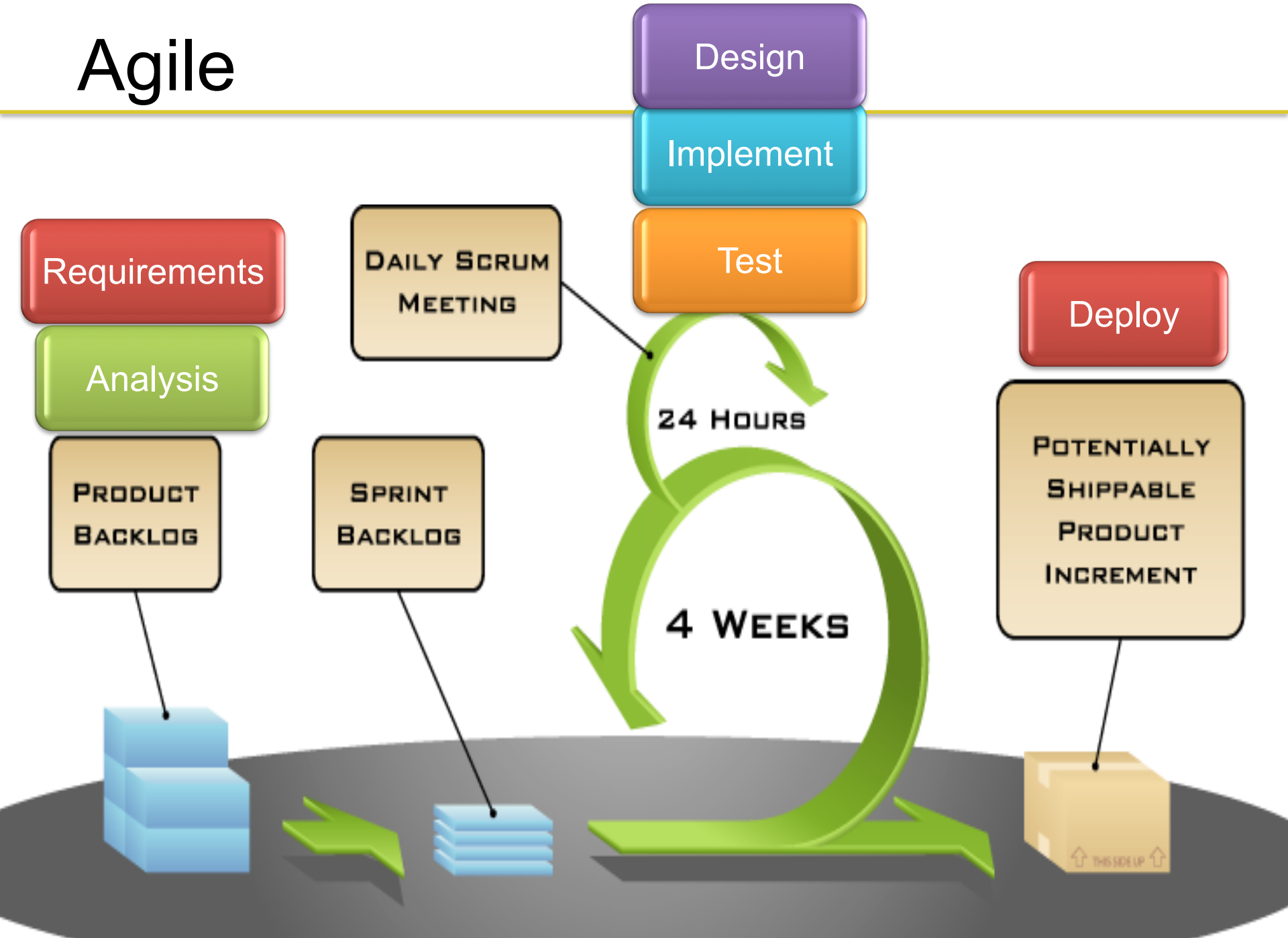
SDLC



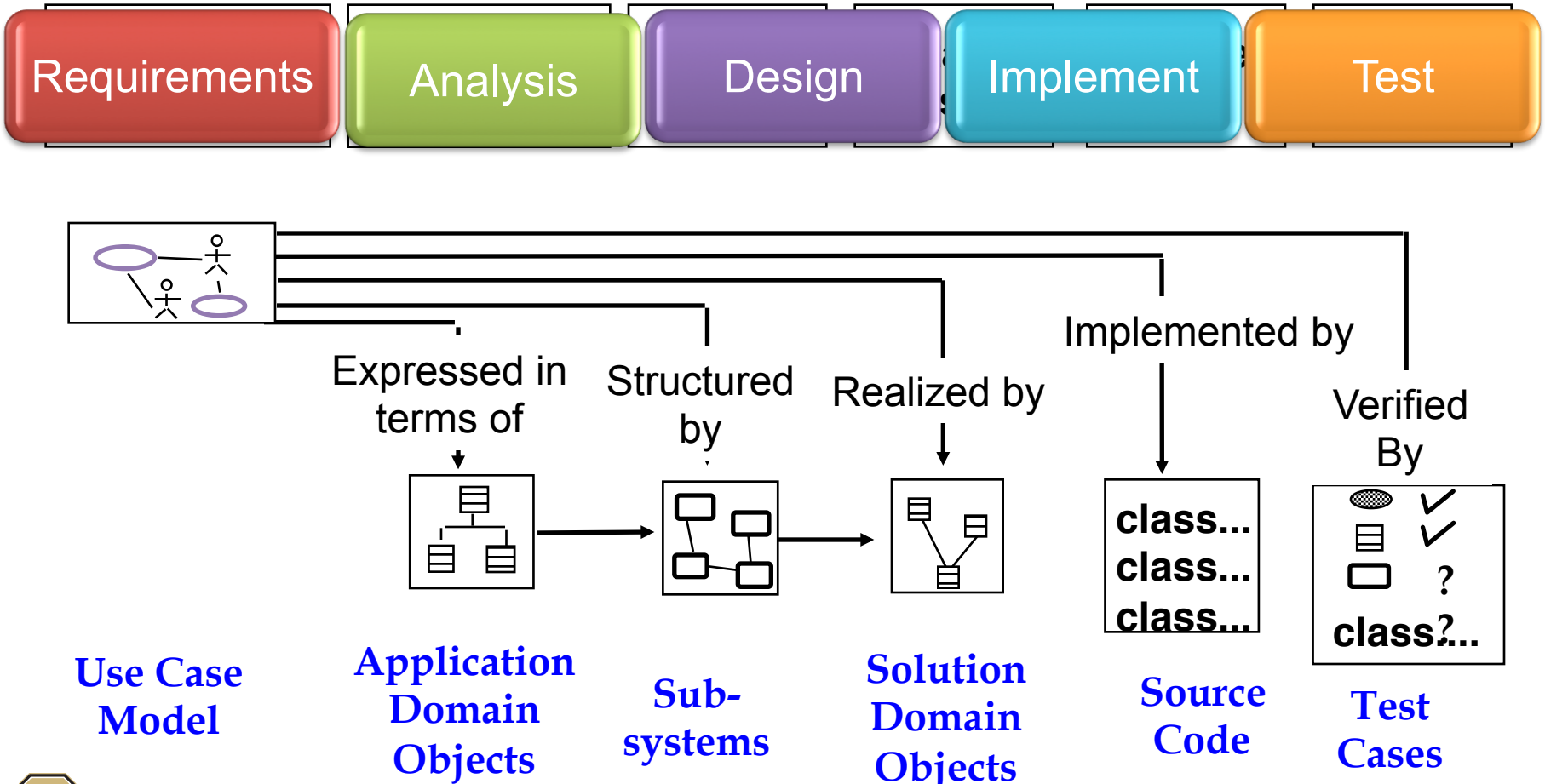
What is **Analysis** and how does it differ from **Requirements** and **Design**?

How do **Waterfall** and **Agile** methodologies differ?

Agile


Design
Implement
Test
Requirements
Analysis
Daily Scrum Meeting
24 Hours
Deploy
Potentially Shippable Product Increment
Product Backlog
Sprint Backlog
4 Weeks

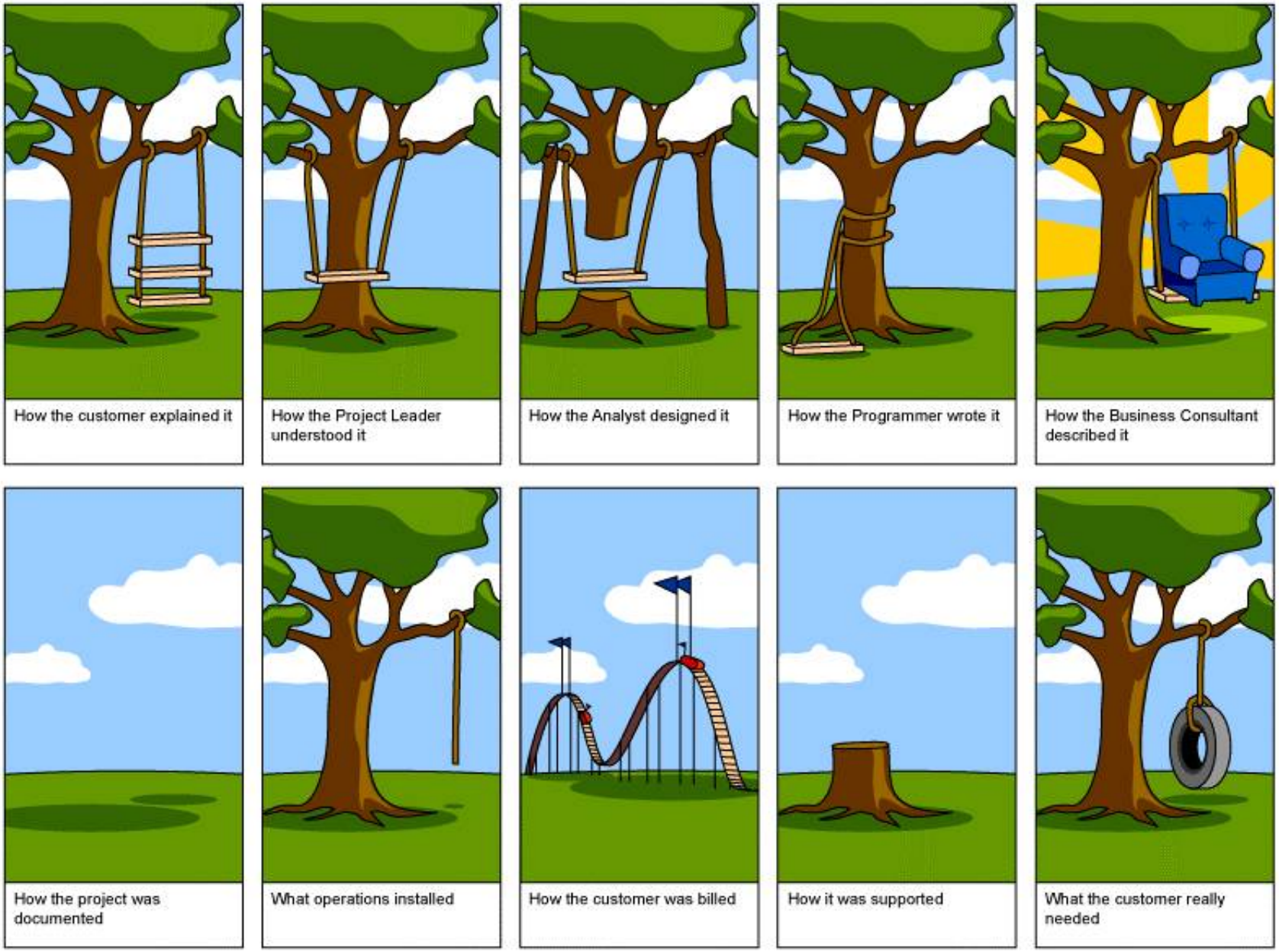
Software Lifecycle Activities ...and their models

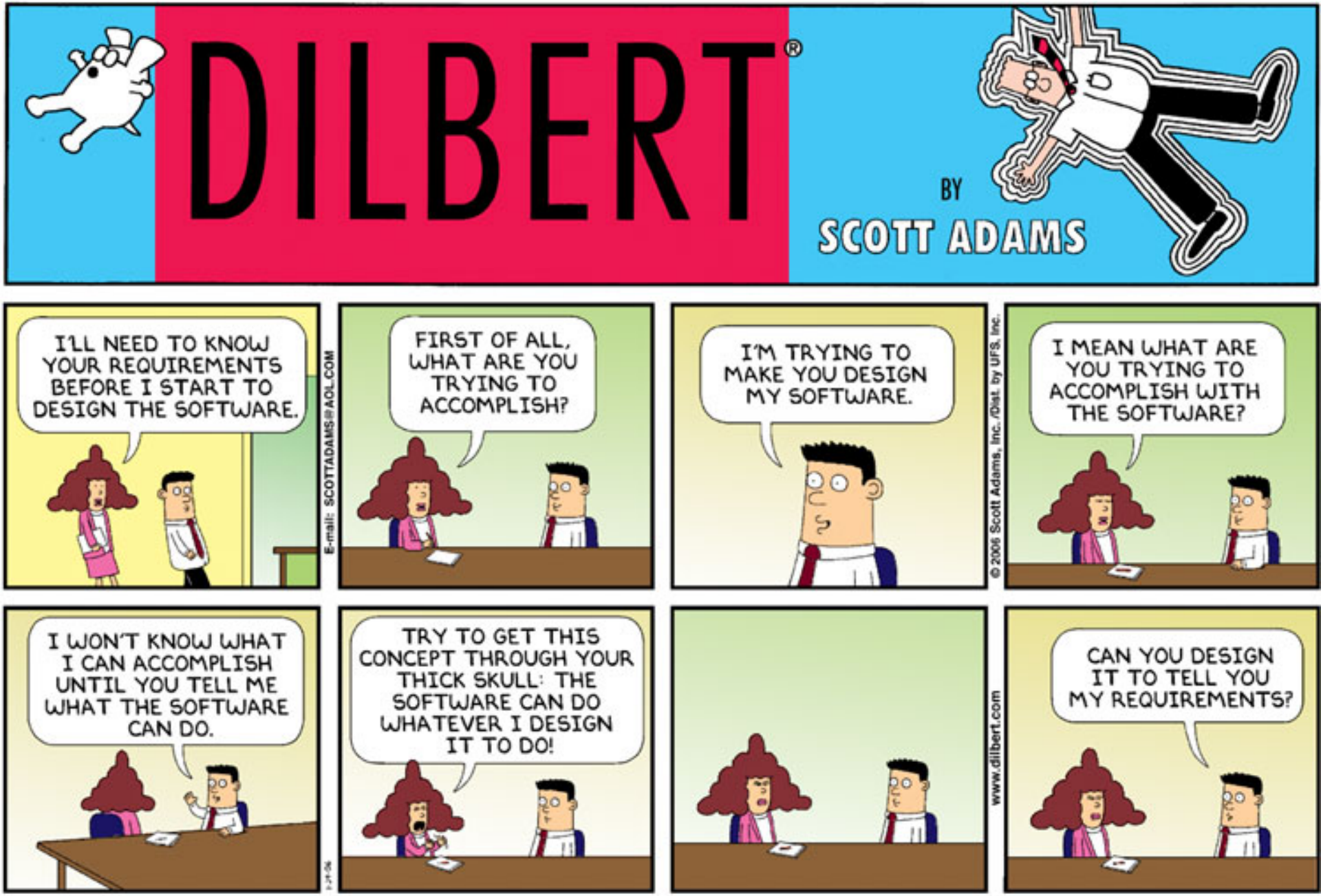


Determining Requirements

Determining Requirements

- With client
- List
- Client will tell you
  - What they ***want***
  - What they ***think*** **they need**
- Your job
  - Determine what they ***actually*** **need**


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed



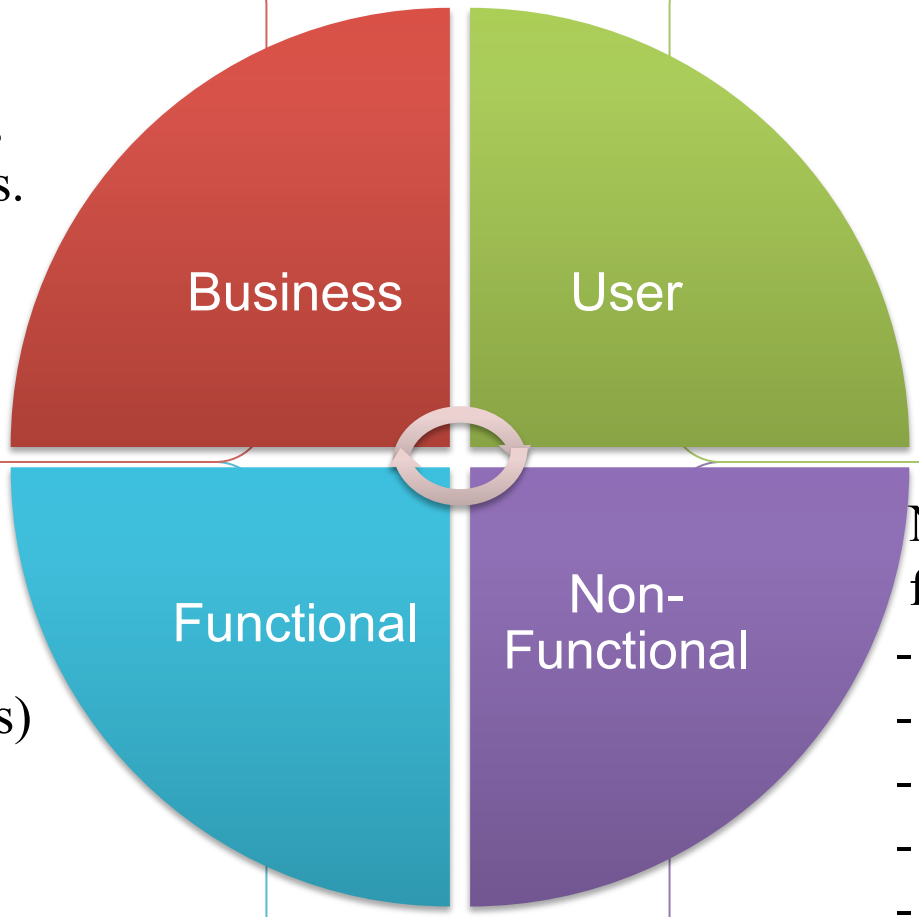
Determining Requirements

- Focus on listing out the client's needs
  - Spreadsheet
  - Document
  - Drawings/sketches
  - Diagrams
- As long as it is clearly specified
  - *These requirements are used at the end of the project to designate whether you fulfilled the contract! Also used for User Acceptance Tests*
  - *Danger: ambiguous requirements that can be interpreted multiple ways!*
- Add ID number to each requirement for reference
  - Useful for ensuring test cases cover the requirements
  - Referencing during meetings
- Additional readings
  - http://www.slideshare.net/guest24d72f/8-characteristics-of-good-user-requirements-presentation
  - Requirements: An Introduction http://www.ibm.com/developerworks/rational/library/4166.html

Types of Requirements

Specific rules based on business needs and procedures.

Processes a user will do. (use-cases)

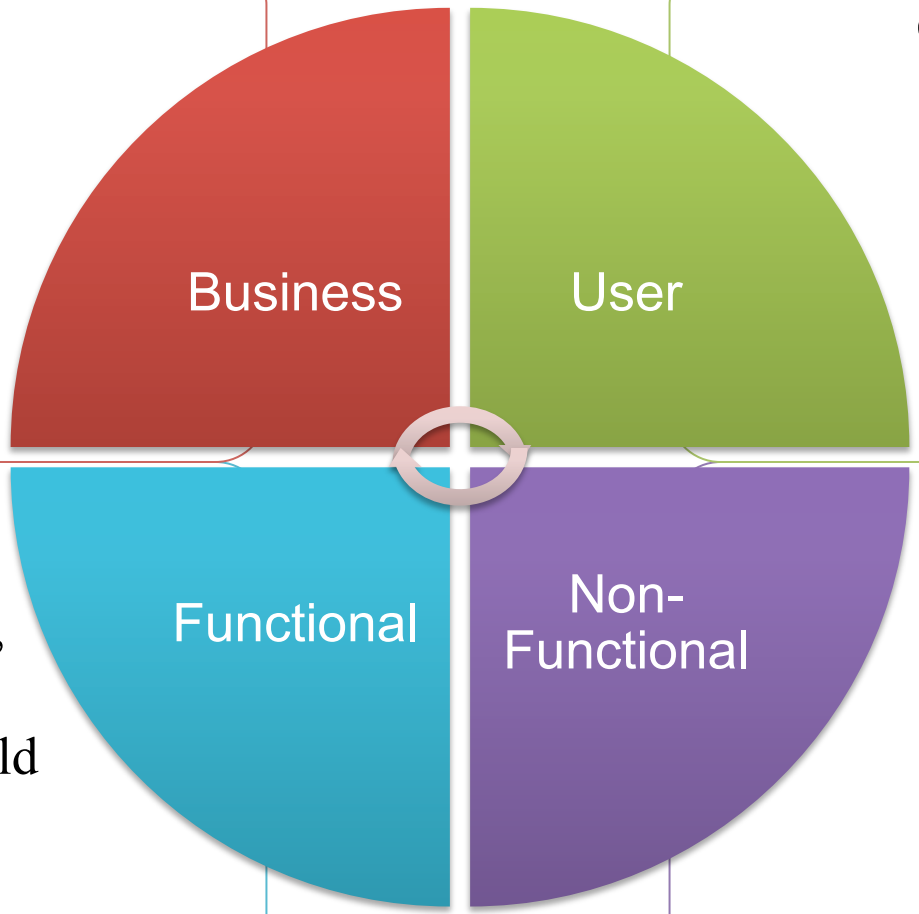


Specific system behaviors (usage scenarios)

Non-functional features such as
- Reliability,
- Performance,
- Security,
- Maintainability
- Usability

Types of Requirements - Examples

Login names must be their company's email address.



Customer Support users can click on "Tickets" and see a list of active tickets.

When creating a support ticket, populate the "created by" field with the user's logon id.

Security
Single Sign-on, Database easily replaceable by other relational DB

Requirements Examples

User Requirements			
ID	**Description**	**Priority**	
US-01	As a system admin, I want to see a dashboard of weekly usage by user type so that I can monitor the usefulness of the system.	Medium	
US-03	As a customer service rep, I want to be notified of failures so that I can respond to all users within an hour.	High	

Non-Functional Requirements		
ID	**Description**	**Priority**
NFR-01	Performance: Upon providing correct credentials, user shall be directed to the landing page within 7 seconds max.	Medium
NFR-03	Platform Constraints: Login functionality shall behave the same on different platforms (Linux/Windows/iOS/mobile).	High

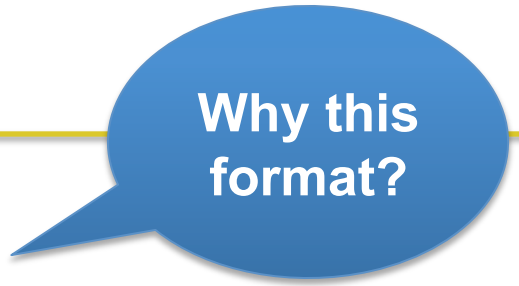
Non-Functional Requirements

Category	Example questions
Usability	• What is the level of expertise of the user? • What user interface standards are familiar to the user? • What documentation should be provided to the user?
Reliability *(including robustness, safety, and security)*	• How reliable, available, and robust should the system be? • Is restarting the system acceptable in the event of a failure? • How much data can the system loose? • How should the system handle exceptions? • Are there safety requirements of the system? • Are there security requirements of the system?
Performance	• How responsive should the system be? • Are any user tasks time critical? • How many concurrent users should it support? • How large is a typical data store for comparable systems? • What is the worse latency that is acceptable to users?
Supportability *(including maintainability and portability)*	• What are the foreseen extensions to the system? • Who maintains the system? • Are there plans to port the system to different software or hardware environments?

Non-Functional Requirements

Implementation	• Are there constraints on the hardware platform? • Are constraints imposed by the maintenance team? • Are constraints imposed by the testing team?
Interface	• Should the system interact with any existing systems? • How are data exported/imported into the system? • What standards in use by the client should be supported by the system?
Operation	• Who manages the running system?
Packaging	• Who installs the system? • How many installations are foreseen? • Are there time constraints on the installation?
Legal	• How should the system be licensed? • Are any liability issues associated with system failures? • Are any royalties or licensing fees incurred by using specific algorithms or components?

Agile Requirements

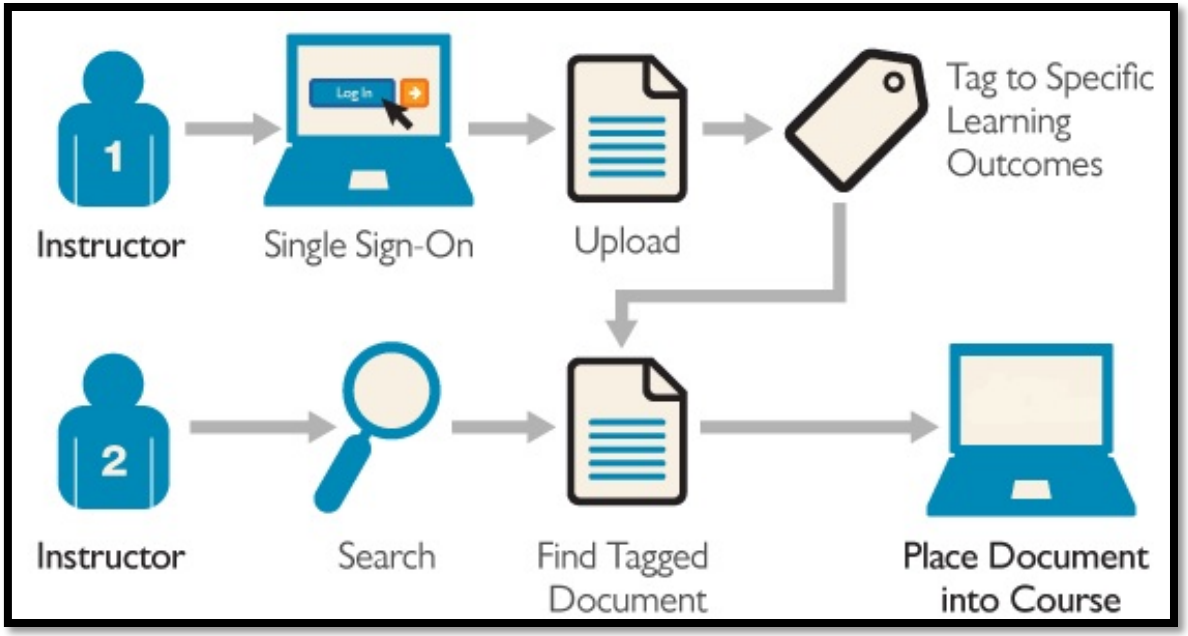


- **As a** <user>,
  I need <a task>
  so that I <can accomplish a goal>.

- **As a** user, **I need to** indicate folders not to backup **so that** my backup drive isn't filled up with things I don't need saved.
- **As an** administrator, **I want** the login usernames to be the same as their company email address **so that** we can prevent spam accounts.

Vision

- Once you have the client's vision,
  a "best-practice" is to write the vision (statement or diagram) at the top of all documents/presentations…
- If the vision changes, that is a BIG DEAL and can void the contract
  (or increment the cost for the contract for new features!)



Requirements Analysis

- Are the requirements feasible?
  - Time, budget, resources
- Specify with diagrams
  - *(diagram the requirements, not the design. Diagramming the design comes later)*
- Why do analysis?
  - Fully **understand** the requirements
  - Identify **gaps** in the requirements
  - Identify **errors** and **conflicts** in requirements
  - Use to **verify completion** of project/contract
    *** Very important to avoid issues over contract/money!

Models

Requirements Analysis vs Design Analysis

Modeling in analysis

- **Functional models:**
  What is the current state? Activity/Use Case Diagram
  What are the functions of the system?
- **Object models:**
  What is the structure of the system? Class Diagrams
- **Dynamic models:**
  How does the system react to external events? Sequence Diagrams
  State Diagrams

- **System Model:** Object model + functional model + dynamic model

Requirements Analysis Questions

1. What are the transformations?  Functional Modeling

Create *scenarios and use case diagrams*

- Talk to client, observe, get historical records

2. What is the structure of the system?  Object Modeling

Create *class diagrams*

- Identify objects.
- What are the associations between them?
- What is their multiplicity?
- What are the attributes of the objects?
- What operations are defined on the objects?

3. What is its behavior?

Create *sequence diagrams*  Dynamic Modeling

- Identify senders and receivers
- Show sequence of events exchanged between objects.
- Identify event dependencies and event concurrency.

Create *state diagrams*

- Only for the dynamically interesting objects.

UML

UML – Unified Modeling Language

- **Model** = abstraction describing a system/subsystem
- **Notation** = graphical or textual rules for depicting models
- **CASE tools** = translate model notation → source code
- Standard for modeling software
- OMG – Object Management Group
- Commercial tools
  - Rational (IBM)
  - Together (Borland)
  - Umbrello (open source)

Model-Driven Software Development

Reality: A stock exchange lists many companies. Each company is identified by a ticker symbol

Design Analysis results in an object model (UML Class Diagram):

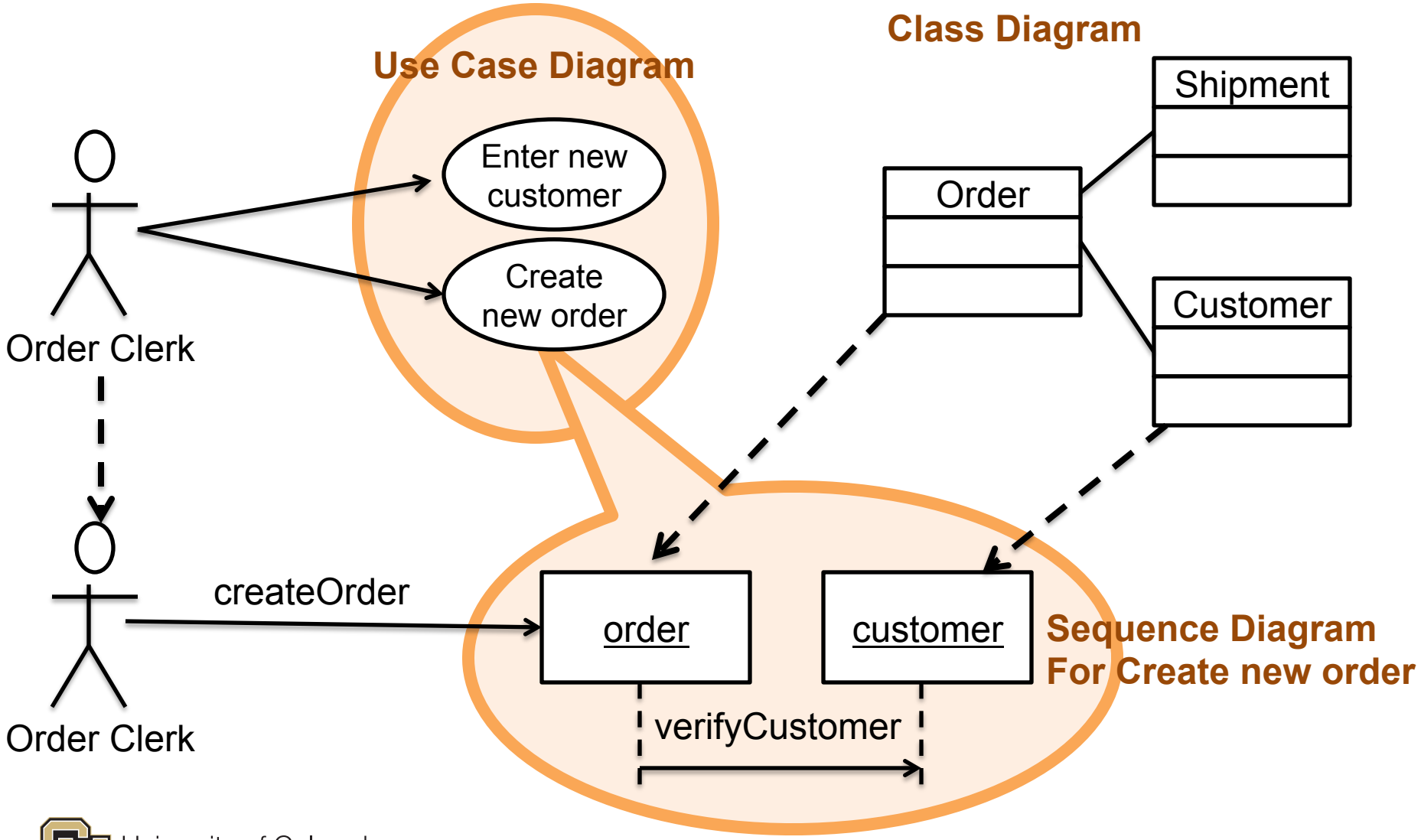
Implementation results in source code:

```
public class StockExchange
{
   ArrayList<Company> companies = new ArrayList<Company>();
}
public class Company
{
   private String tickerSymbol;
}
```

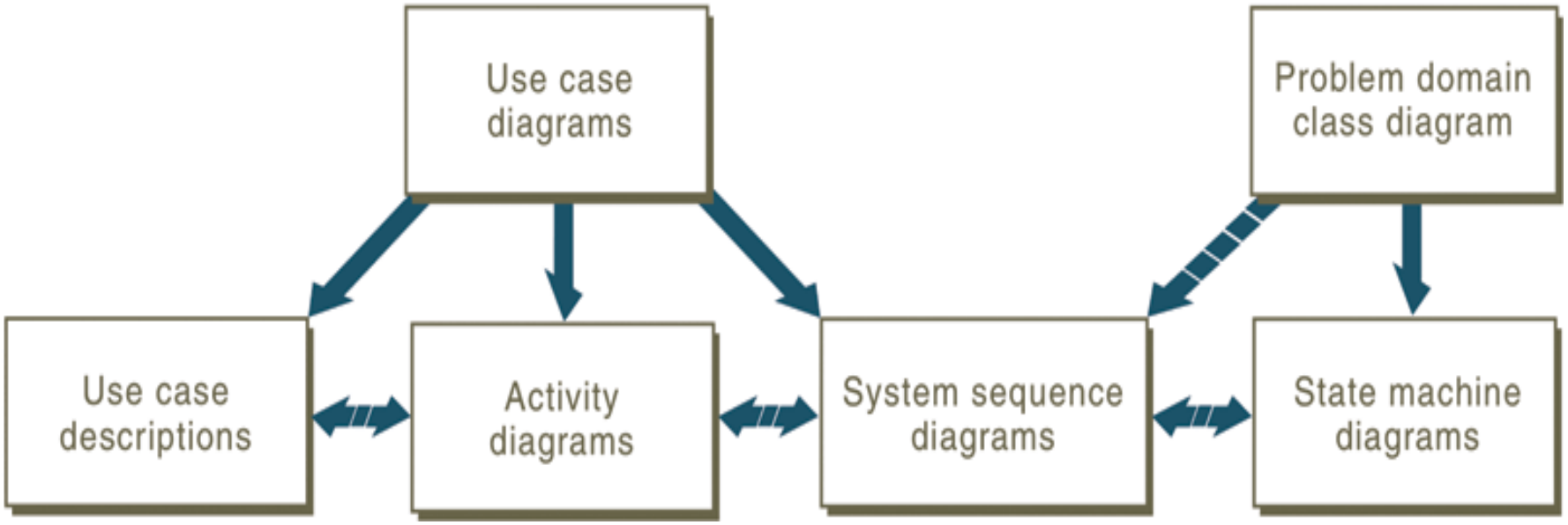
UML First Pass

- **Use case diagrams**
  - Describe the functional behavior of the system as seen by the user
- **Activity diagrams**
  - Describe the dynamic behavior of a system, in particular the workflow.
- **Class diagrams**
  - Describe the static structure of the system: Objects, attributes, associations
- **Sequence diagrams**
  - Describe the dynamic behavior between objects of the system
- **Statechart diagrams**
  - Describe the dynamic behavior of an individual object

Diagrams Overview



Relationships Between OO Requirements Models



Diagrams Overview

- No data flow diagrams in UML → replaced with Activity Diagrams

Use case diagram:

– Relationships between the system and outside environment

Activity diagram:

– Workflow of business processes.

– Shows concurrent and alt. flows.

– Use 2+ ADs to model different use case scenarios.

– Use Swim Lanes (UML 2)

Class diagram:

– Classes and their attributes and behaviors

Sequence diagram:

– Timing or sequence relationships between actors and classes.

– Shows inputs and outputs and sequence of messages between an external actor and the system objects during a use case or scenario.

State machine diagram:

– Explore behavior of an object.

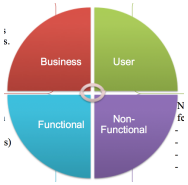
– An object can exist in only one configuration at a time.

– Shows the life of an object in states and transitions.

Relationship of Requirements Gathering and Analysis to Diagrams

Problem statement

Requirements Elicitation Process



User, Business Functional, Non-functional Requirements

Use Cases, Activity Diagrams Sequence Diagrams (user perspective)

Object-Oriented Analysis Process

Static Analysis
Dynamic Analysis

- Class Diagrams,
- State Diagrams,

Refined Sequence Diagrams (The object level)

Full Requirements Document

Requirements Analysis Document

1. Introduction
   - 1.1 Purpose of the system
   - 1.2 Scope of the system
   - 1.3 Objectives and success criteria of the project
   - 1.4 Definitions, acronyms, and abbreviations
   - 1.5 References
   - 1.6 Overview
2. Current system
3. Proposed system
   - 3.1 Overview
   - 3.2 Functional requirements
   - 3.3 Nonfunctional requirements
     - 3.3.1 Usability
     - 3.3.2 Reliability
     - 3.3.3 Performance
     - 3.3.4 Supportability
     - 3.3.5 Implementation
     - 3.3.6 Interface
     - 3.3.7 Packaging
     - 3.3.8 Legal
   - 3.4 System models
     - 3.4.1 Scenarios
     - 3.4.2 Use case model
     - ~~*3.4.3 Object model*~~
     - ~~*3.4.4 Dynamic model*~~
     - 3.4.5 User interface—navigational paths and screen mock-ups
4. Glossary

UML Core Conventions

- All UML Diagrams denote graphs of nodes and edges
  - Nodes are entities and drawn as rectangles or ovals
  - Rectangles denote classes or instances
  - Ovals denote functions

- Names of **Classes** are not underlined
  - `SimpleWatch`
  - `Firefighter`
- Names of **Instances** are underlined
  - `myWatch:SimpleWatch`
  - `Joe:Firefighter`
- An edge between two nodes denotes a relationship between the corresponding entities