CS5127/6027: Requirements Engineering

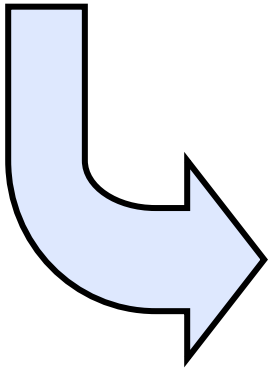
(Fall 2024)

Prof. Nan Niu (nan.niu@uc.edu)

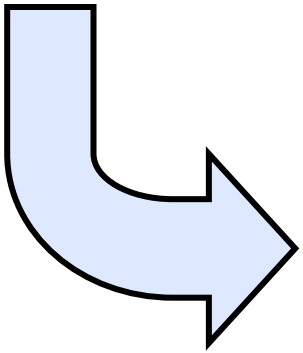
Office Hours: 10am-11am, Mondays, Rhodes 832

Today's Menu

Last Lecture (Friday 10/4):
Visual Modeling Notations

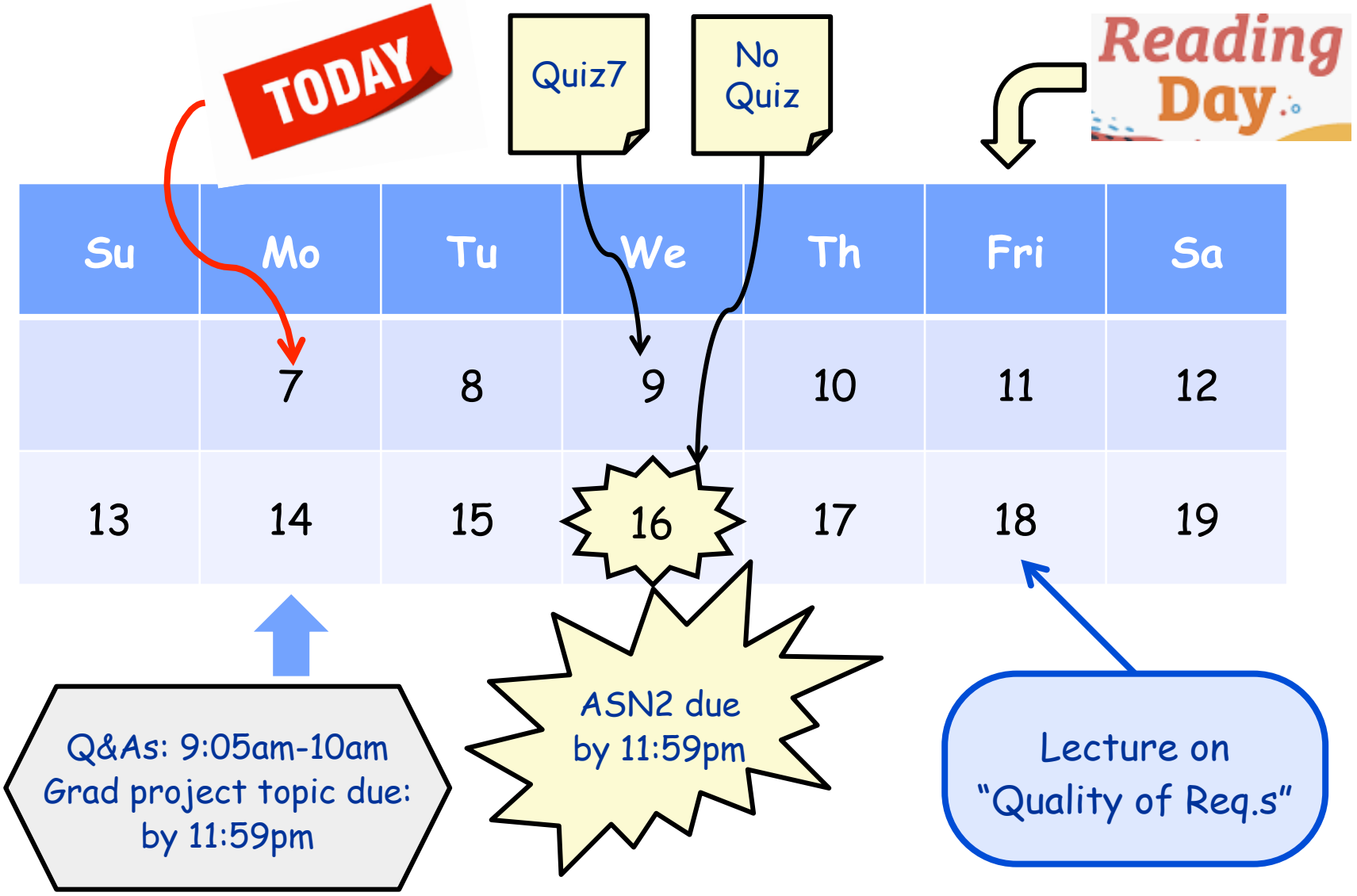


This Lecture (Monday 10/7):
Forms of Requirements



Next Lecture (Monday 10/14):
OPTIONAL: Q&As

Schedule Look-Ahead



"Form": practically speaking …

→ As a requirements engineer, how do I showcase (document) my work? How easy or difficult this task (RE activity) will be? How long will it take?

→ As a teammate or a software project manager, how do I review or assess my requirements engineer's work?

Today's Take-Aways

→ Most common form of requirements is: _____

→ A common req.s form in UML is: _____

→ Agile req.s are often expressed in: _____

The majority of req.s are written in

→Natural language (e.g., English)

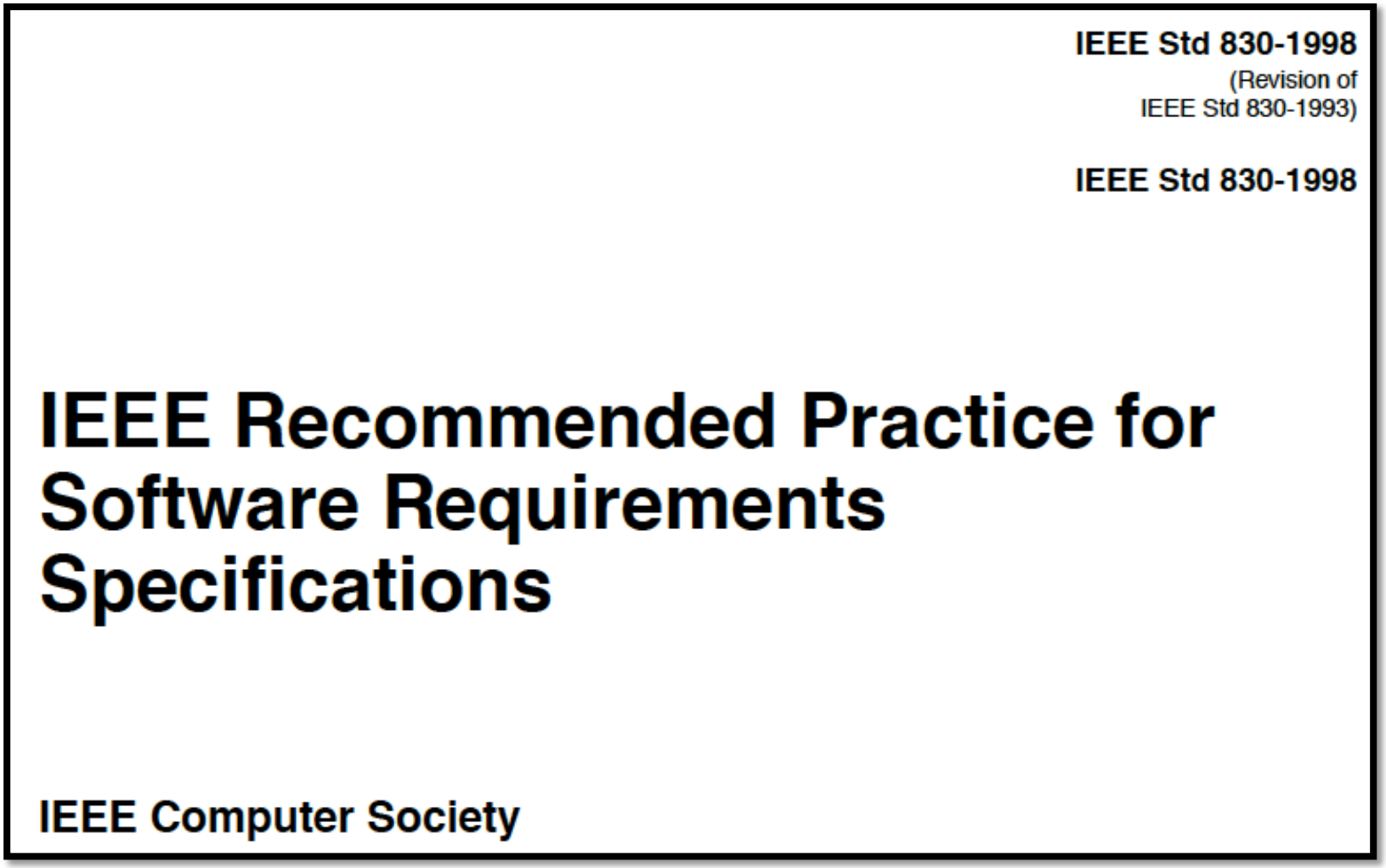
- because text is used universally to convey information and to communicate

- J. Aranda, *et al.* "Requirements in the wild: how small companies do it", *RE'07*, pages 39-48
- J. Dag, *et al.* "A linguistic-engineering approach to large-scale requirements management", *IEEE Software*, 22(1): 32-39, 2005.
- M. Friedewald, *et al.* "Status of the software industry in Germany", *Informatik Spektrum*, 24(2): 81-90, 2001.

Some NL req.s examples

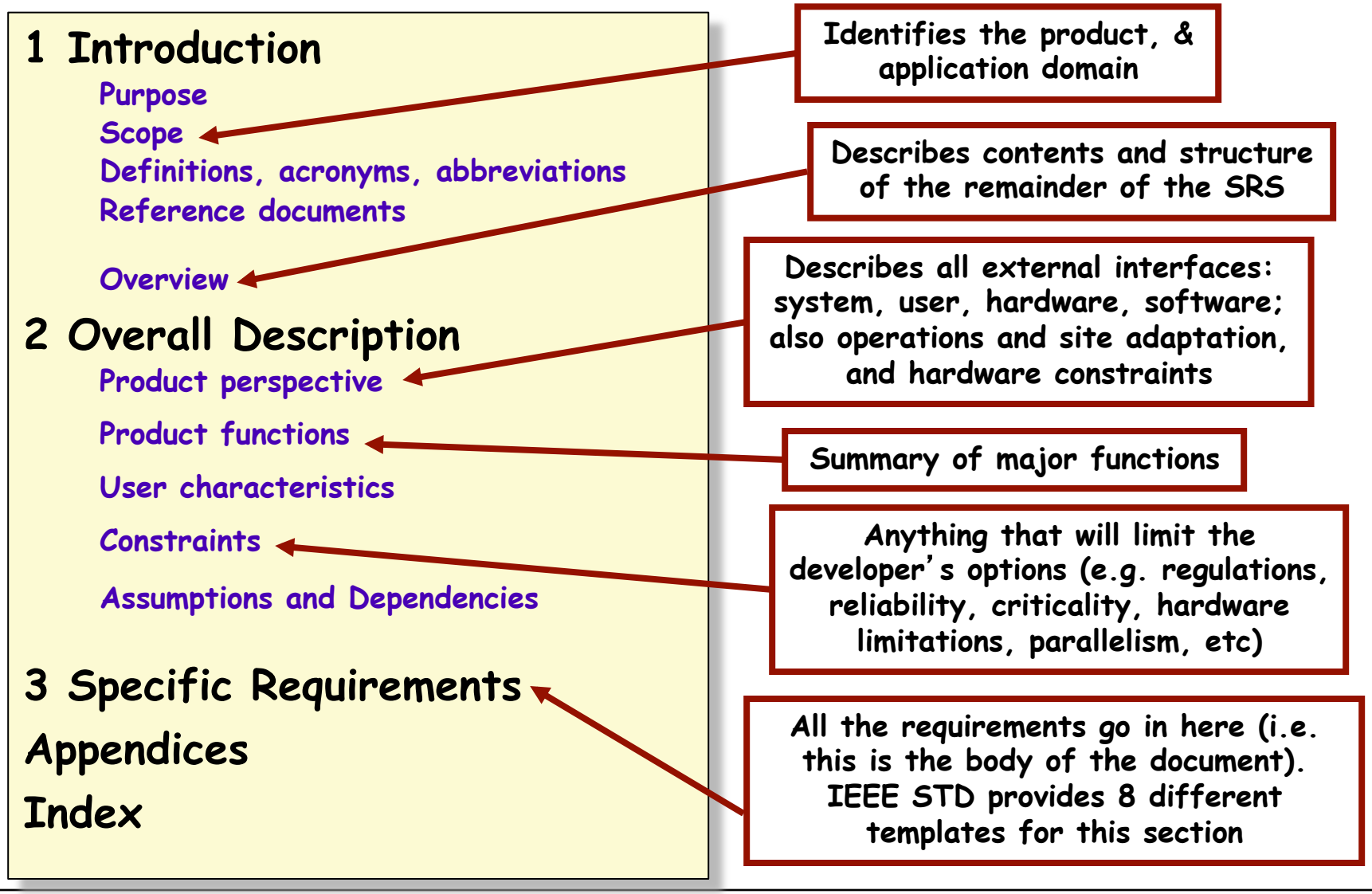
SRS (Software Requirements Specification)	Pacemaker	Platform for safety-critical systems' certification	Rosetta spacecraft mission
author (if known)	Boston Scientific	Simula Research Laboratory	Rosetta Project Team
# of pages	35	95	142
req.s sections (#)	• System req.s (8) • Diagnostics (9) • Therapy (9)	• High-level business process • Product-level & feature-level req.s (3 + 10) • Component-level req.s (5 functional areas + UI mock-ups) • One area, e.g., has 55 req.s	• General (7) • Env.al (5) • Structural (4) • Mechanism (5) • Thermal (5) • Attitude, orbit control, measurement (5) • … (7 more sect.s)

Reference in the course website



IEEE Standard for SRS

Source: *Adapted from IEEE-STD-830-1993* ***See also****, Blum 1992, p160*



IEEE STD Section 3 (example)

Source: *Adapted from IEEE-STD-830-1993.* ***See also****, Blum 1992, p160*

3.1 External Interface Requirements

- 3.1.1 User Interfaces
- 3.1.2 Hardware Interfaces
- 3.1.3 Software Interfaces
- 3.1.4 Communication Interfaces

3.2 Functional Requirements

this section organized by mode, user class, feature, etc. For example:

- 3.2.1 Mode 1
  - *3.2.1.1 Functional Requirement 1.1*
  - …
- 3.2.2 Mode 2
  - *3.2.1.1 Functional Requirement 1.1*
  - …
- ...
- 3.2.2 Mode n
  - ...

3.3 Performance Requirements

Remember to state this in measurable terms!

3.4 Design Constraints

3.4.1 Standards compliance

3.4.2 Hardware limitations

etc.

3.5 Software System Attributes

- 3.5.1 Reliability
- 3.5.2 Availability
- 3.5.3 Security
- 3.5.4 Maintainability
- 3.5.5 Portability

3.6 Other Requirements

Other Standards

NASA-DID-P200

Requirements
- 1.0 Introduction
- 2.0 Related documentation
- 3.0 Requirements approach and tradeoffs
- 4.0 External interface requirements
- 5.0 Requirements specification
  - 5.1 Process and data requirements
  - 5.2 Performance and quality engineering requirements
  - 5.3 Safety requirements
  - 5.4 Security and privacy requirements
  - 5.5 Implementation constraints
  - 5.6 Site adaptation
  - 5.7 Design goals
- 6.0 Traceability to parent's design
- 7.0 Partitioning for phased delivery
- 8.0 Abbreviations and acronyms
- 9.0 Glossary
- 10.0 Notes
- 11.0 Appendices

NASA-STD-2100-91 29 July 1991
Military Standard, Software Development and Documentation

MIL-STD-498 5 Dec 1994
Military Standard, Software Development and Documentation

MIL-STD-1679A (Navy) 22 Oct 1983
Software Development

Req.s Eng.

- Problem theory
  - E, S |- R
- Solution specification
  - SRS (IEEE STD 830)

UML (Unified Modeling Language)

- Nonproprietary standard for modeling SW systems
  - Managed by OMG (Object Mgmt Group)
  - Current version: UML 2.2
    - http://www.uml.org/
  - Commercial tools
    - Rational, Together, Visual Architect…
  - Open-source tools
    - ArgoUML, StarUMML, Umbrello …
- Designed by committee
  - Use case diagrams
  - Class diagrams
  - Message sequence charts (sequence diagrams)
  - Activity diagrams
  - State Diagrams (uses Harel's statecharts)
  - Module Diagrams
  - …

omg.org

Why "unifying"?

- Convergence of different notations used in OO methods (late 1980s-early 1990s), mainly
  - OMT (James Rumbaugh & colleagues), OOSE (Ivar Jacobson), & Booch (Grady Booch)
- They also developed the Rational Unified Process '99

25 year at GE Research, where he developed OMT, joined (IBM) Rational in 1994, CASE tool OMTool

At Ericsson until 1994, developed use cases and the CASE tool Objectory, joined IBM Rational since 1995

Developed the Booch method ("clouds"), ACM Fellow 1995, and IBM Fellow 2003

Use Cases

→ What is a use case?

- Each different way that an actor interacts with a system is a use case
  - "a description of a *sequence of actions* that a system performs that yields an *observable result of value to a particular actor*" [Booch]
  - All the use cases need to be enumerated (or the requirements will not be complete)
- A description of a set of possible scenarios, with a common purpose
- Typically written in natural language
- No internal description of the system implementation; just the interaction

→ Combining use cases

- UC1 "uses" UC2
  - the process of doing UC1 always involves doing UC2 at least once
  - a brief, mnemonic way to think about the "uses" arrow: "UC1 has a UC2"
- UC1 "extends" UC2
  - the process of UC1 is a special case behavior of the same type as the more general process of UC2

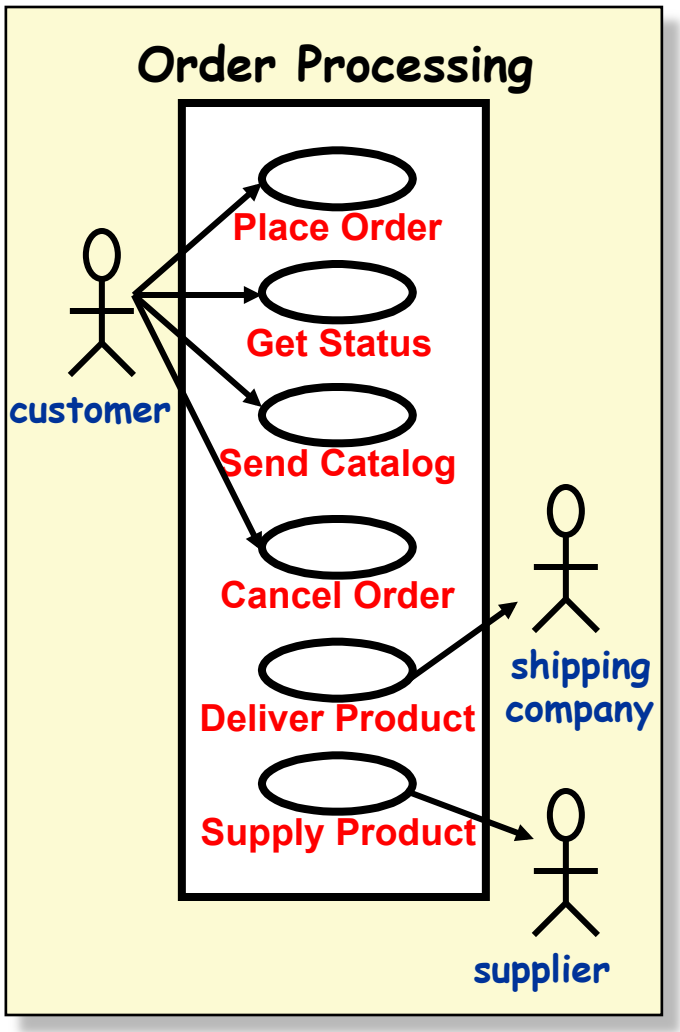
→ Advantages & Disadvantages

- detailed characterization of all possible interaction with the system
- helps in drawing system boundary, and scoping the requirements
- don't confuse use cases with a precise specification

Source: *Adapted from Rumbaugh 1997, p123-124*

Use Cases - Example

Use Case Diagram



Use case diagrams represent the functionality of the system from user's point of view

An **actor** is a model for an external entity which interacts (communicates) with the system (software-to-be)

- Role, e.g., "customer", "admin"
- External system, e.g., "CC auth.", "GPS"
- Physical environment, e.g., "weather"

System boundary of the software-to-be

- Scoping the problem & solution

Classifier of the software-to-be, e.g., "Order Processing" in the example

Use Case (UC), e.g., 

→ A UC represents a class of functionality provided by (& visible from the outside of) the system

→ A UC can be described textually, with a focus on the event flow between actor and system

→ The textual UC description consists of 6 parts

1. Unique name
2. Participating actors
3. Entry conditions (i.e., preconditions)
4. Exit conditions (i.e., postconditions)
5. Flow of events
6. Special requirements (e.g., exceptions)

Use Case Description - Example

Use Case Description

Name: Place Order

Precondition: A valid user has logged into the system.

Description:

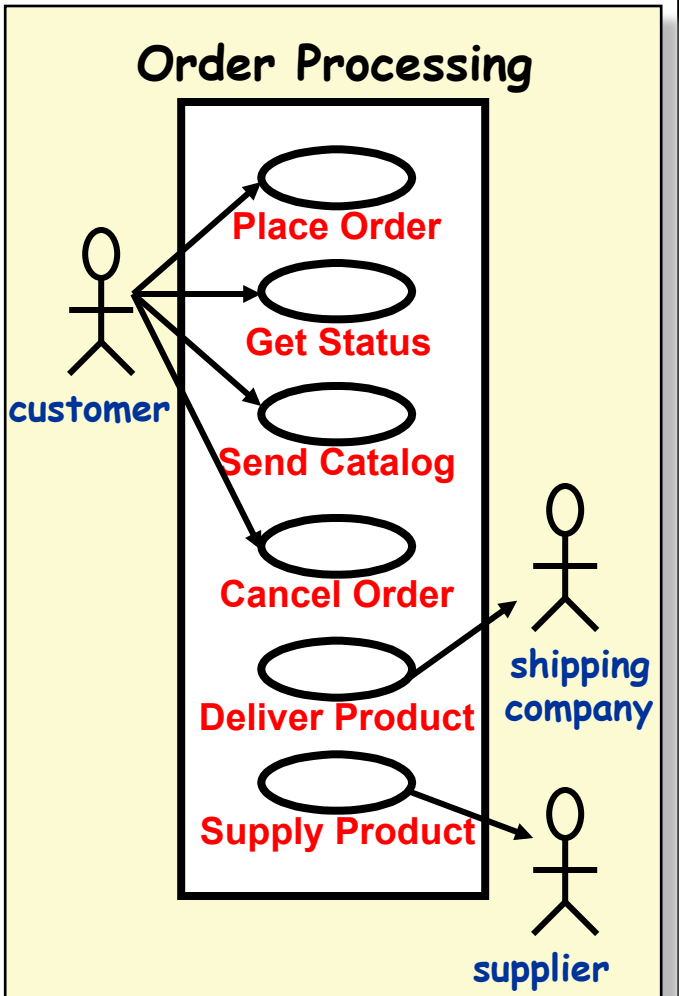
1. The use case starts when the customer selects Place Order.
2. The customer enters his or her name and address.
3. If the customer enters only the zip code, the system will supply the city & state.
4. The customer will enter product codes for the desired products.
5. The system will supply a product description and price for each item.
6. The system will keep a running total of items ordered as they are entered.
7. The customer will enter credit card payment information.
8. The customer will select Submit.
9. The system will verify the information, save the order as pending, and forward payment information to the accounting system.
10. When payment is confirmed, the order is marked Confirmed, an order ID is returned to the customer, and the use case ends.

Exceptions:

In step 9, if any information is incorrect, the system will prompt the customer to correct the information.

Postcondition: The order has been saved in the system and marked confirmed.

Use Case Diagram



Use cases in practice

Dataset	LOC (K)
iTrust	18.3

Use Case: Schedule Appointments

Sub-flows:

[S1] The system shall enable the administrator to add a new entry for an appointment type, including its type name with up to 30 alpha characters and duration in the unit of minutes [E1].

[S2] The LHCP (licensed health care professional) schedules an appointment with a patient, and enters comment (optional) up to 1000 characters [E2].

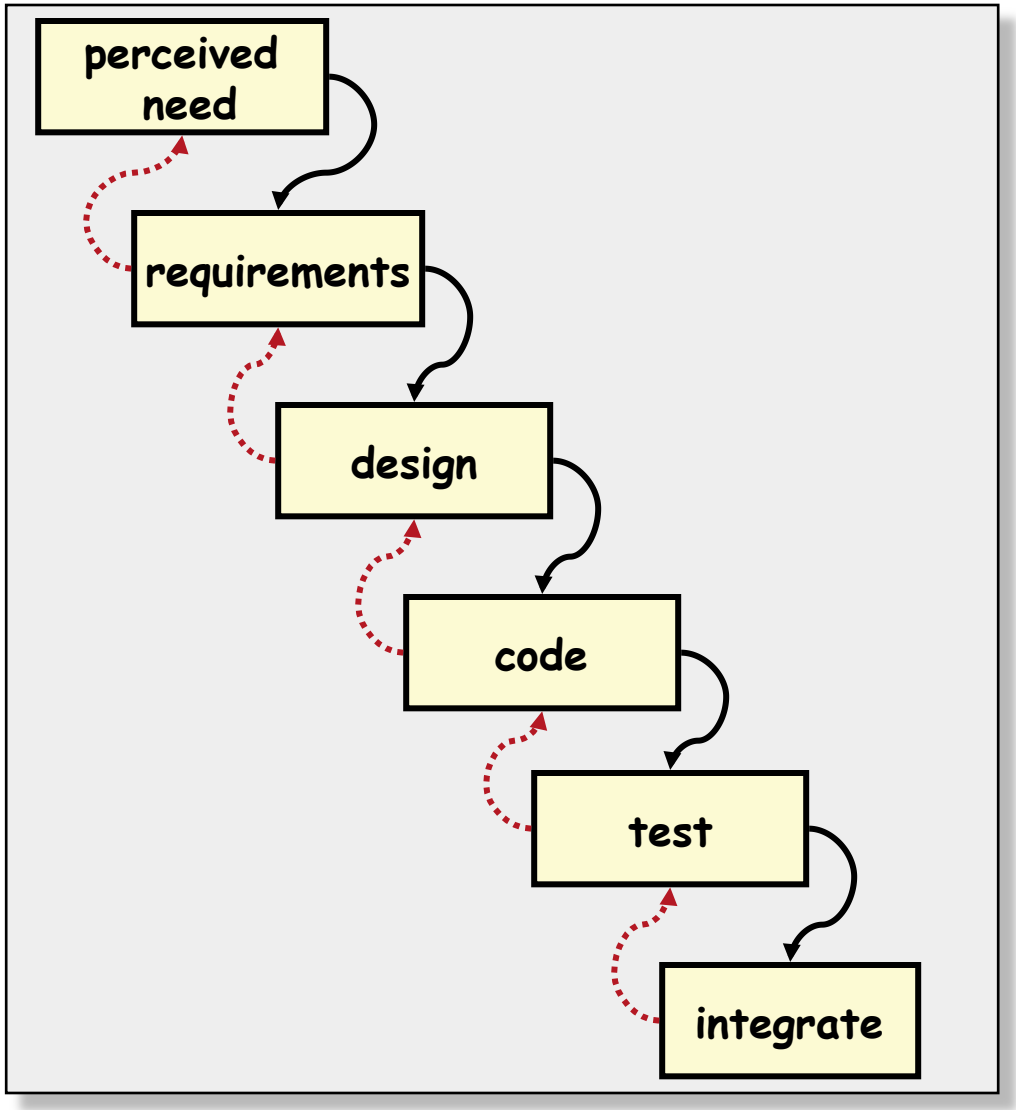
[S3] The patient selects an LHCP from his or her provider list. The patient selects the type of appointment, enters the appointment date and start time. If the requested appointment time does not conflict with any existing appointment for the LHCP, the request is saved. If the requested appointment time does conflict with an existing appointment, the patient is presented with a list of the three next non -overlapping available appointment times within 7 days of the requested date. The patient selects one of these appointments and the request is saved.

Alternative Flows:

[E1] The user inputs invalid information and is prompted to try again.

[E2] The comment is empty and the text "No Comment" (without link) is displayed instead of the "Read Comment" link.

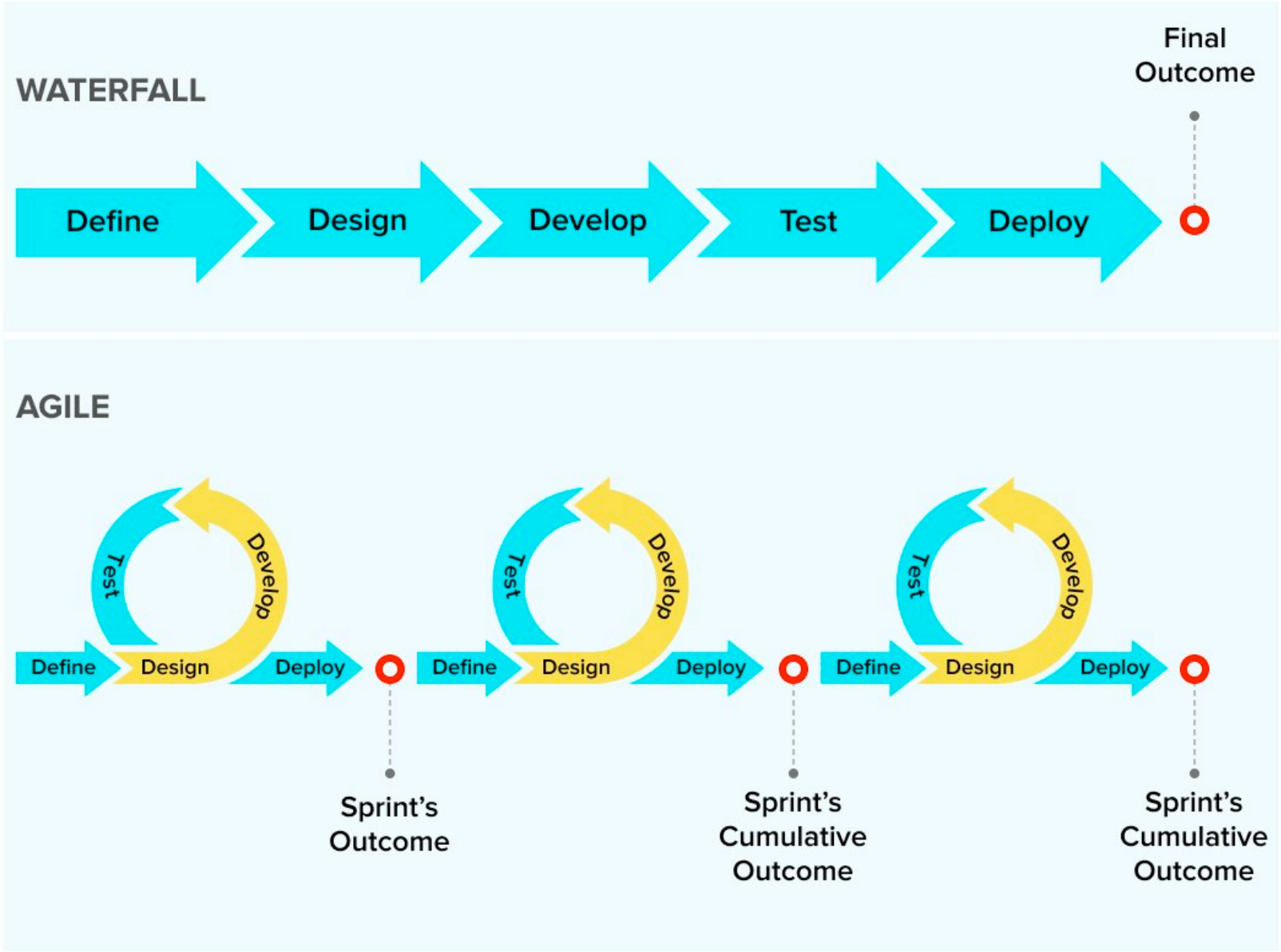
SRS ⟷ Waterfall Model



→ View of development:
- a process of stepwise refinement
- largely a high level management view

→ Problems:
- Static view of requirements - ignores volatility
- Lack of user involvement once specification is written
- Unrealistic separation of specification from design
- Doesn't accommodate prototyping, reuse, etc.

Source: *Adapted from Dorfman, 1997, p7 & Loucopoulos & Karakostas, 1995, p29*


WATERFALL
Final Outcome
Define
Design
Develop
Test
Deploy
AGILE
Test
Develop
Define
Design
Deploy
Sprint's Outcome
Sprint's Cumulative Outcome
Sprint's Cumulative Outcome

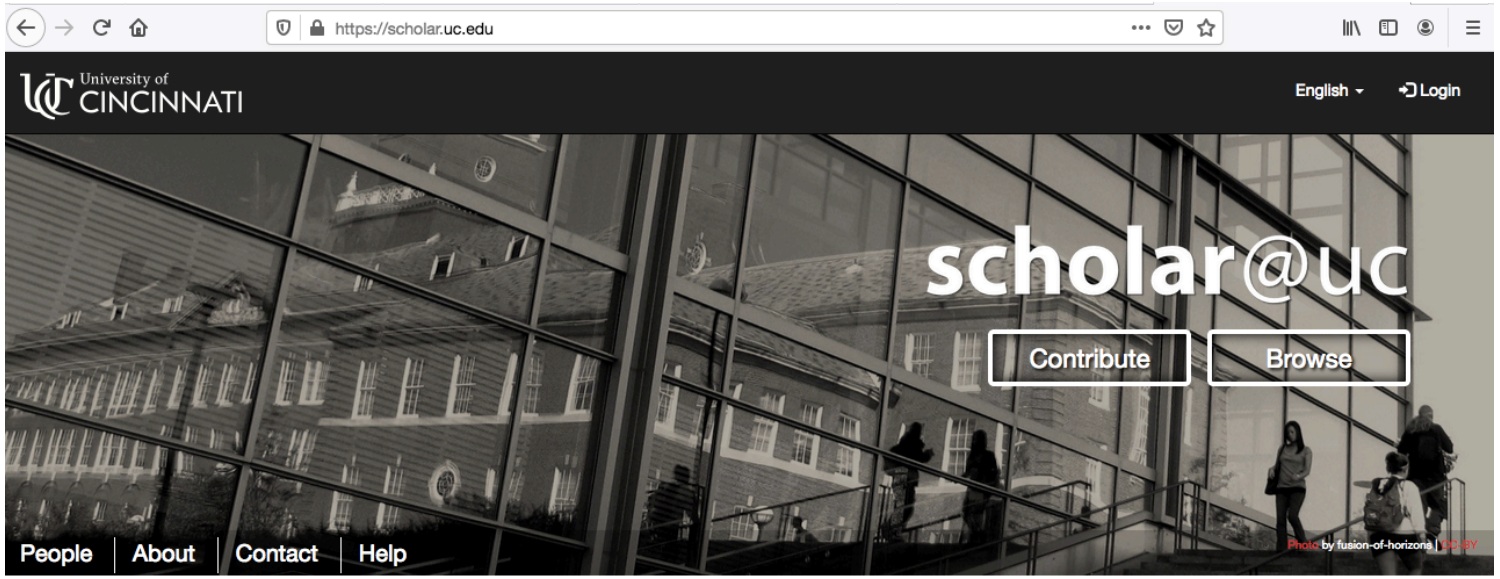
User Stories

- Only capture the essential elements of a requirement
  - *who* it is for
  - *what* it expects from the system
  - (optionally) *why* it is important

"As a <type of user>, I want <goal>, [so that <some reason>]"

e.g., "As an administrator, I want to receive an email when a contact form is submitted, so that I can respond to it."

User stories in practice

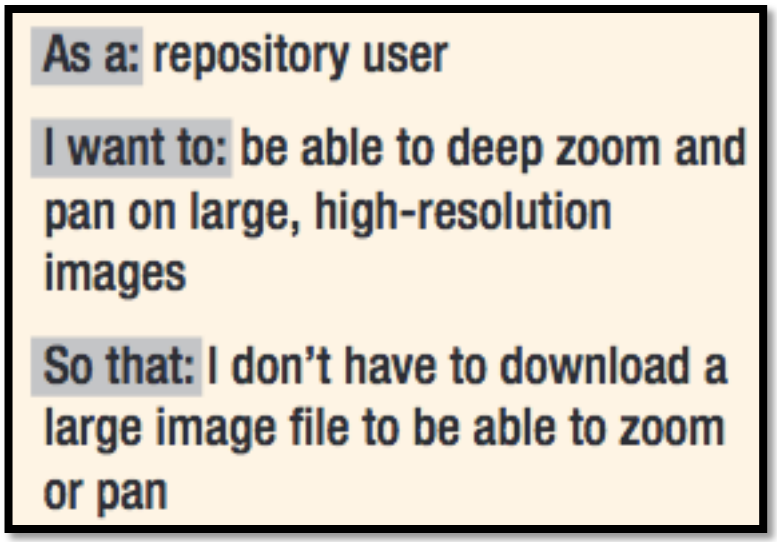


→ Scholar@UC https://github.com/uclibs/scholar_use_cases

In-class exercise

1. A student is filling out her college application and retrieves her cumulative GPA from the school's LMS (learning management system).

2. After a guidance counselor reschedules an appointment with a student, the system sends the student a notification email.



3. After entering the grades from a recent test, a teacher tells her students the average, high, and low scores

4. After making changes to his assignment, a student submits a revised version of the document that replaces the original.

Difference between SRS and US

1 Introduction
- Purpose
- Scope
- Definitions, acronyms, abbreviations
- Reference documents
- Overview

2 Overall Description
- Product perspective
- Product functions
- User characteristics
- Constraints
- Assumptions and Dependencies

3 Specific Requirements

Appendices

Index

As a: repository user

I want to: be able to deep zoom and pan on large, high-resolution images

So that: I don't have to download a large image file to be able to zoom or pan

Which one is better? What does "better" mean? *(next Friday's class: "Quality of req.s")*

Today's Take-Aways

→Forms: NL (SRS), UC, US

→To-do

- Review today's slides
- Complete Quiz7 *before* 11:59pm, Wednesday (Oct 9)
- Next Monday's class (Oct 14) is *optional* [Q&As]
- Grad project topic due: Monday (Oct 14)
- ASN2 due: Wednesday (Oct 16)
- Attend next Friday's class (Oct 18) on the "Quality of req.s"