

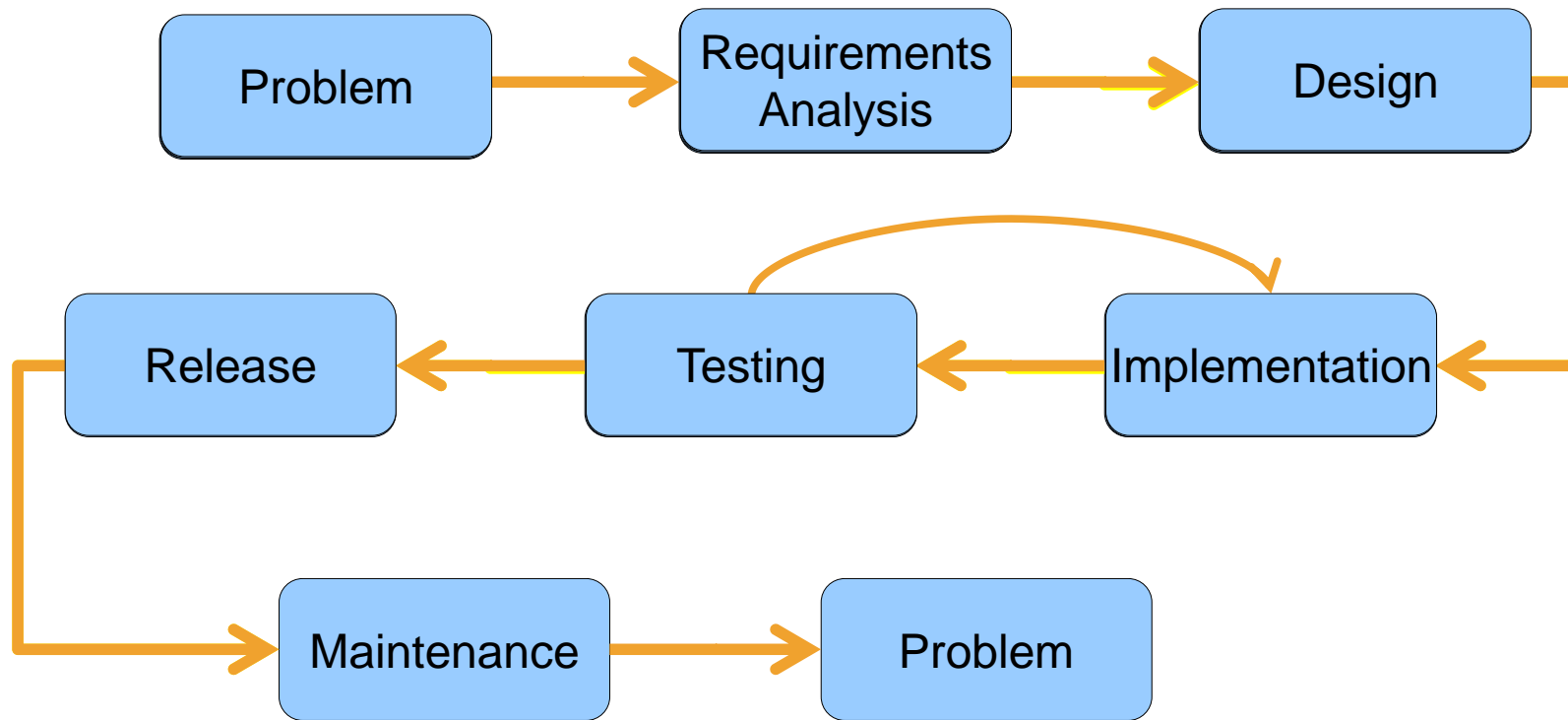
CS 4320 / 7320

Software Engineering

Semester Project
Requirements

What is the SDLC?

Where does Requirements Analysis fit?



What is a software requirement?

A **requirement** is a property that **must be exhibited** by the software system in question in order to **solve some problem** in the real world.

Each requirement must be **verifiable** within **available resource constraints**.

Use Case Components

Each use case should contain the following elements:

1. **Title** (active verb phrase, states main goal)
2. **Description** (This may be several paragraphs. Context is important. You are describing the use case in some detail, and since many of the use cases will involve users changing parameters on data visualizations, you should be exceedingly clear about this type of thing.)
3. **Triggers** (What prompts the use case to start?)
4. **Actors** (Who is involved?)
5. **Preconditions** (This includes things like “data loaded”. Or, project is flagged as “of interest”; etc.)
6. **Main Success Scenario** (Goals) (What does it look like when the user’s work is successful in the system?)
7. **Alternate Success Scenarios** (For a data analysis and “data playing” focused project like this one, there could be several different success scenarios for each use case. “Sees visualization” is **not** a success scenario. “Compares four different projects on “indicator X” and saves “project trackers” for each one could be a success scenario.)
8. **Failed End Condition** (“crashes” is not a failed end condition. “User is unable to discern the difference between two projects because they are similar on the available indicators” might be).
9. **Extensions**
10. **Steps of Execution** (Requirements)
11. A **use case diagram**, following the UML Standard for expressing use cases.

Use Case Example

Each use case should contain the following elements:

1. **Title** (active verb phrase, states main goal)
2. **Description** (This may be several paragraphs. Context is important. You are describing the use case in some detail, and since many of the use cases will involve users changing parameters on data visualizations, you should be exceedingly clear about this type of thing.)
3. **Triggers** (What prompts the use case to start?)
4. **Actors** (Who is involved?)
5. **Preconditions** (This includes things like “data loaded”. Or, project is flagged as “of interest”; etc.)
6. **Main Success Scenario** (Goals) (What does it look like when the user’s work is successful in the system?)
7. **Alternate Success Scenarios** (For a data analysis and “data playing” focused project like this one, there could be several different success scenarios for each use case. “Sees visualization” is **not** a success scenario. “Compares four different projects on “indicator X” and saves “project trackers” for each one could be a success scenario.)
8. **Failed End Condition** (“crashes” is not a failed end condition. “User is unable to discern the difference between two projects because they are similar on the available indicators” might be).
9. **Extensions**
10. **Steps of Execution** (Requirements) (Functional and Technical)
11. **A use case diagram**, following the UML Standard for expressing use cases.

Contribute to Existing Dataset

Description

A user wishes to add SNC files for a dataset already uploaded to the server. The user will provide a link to the dataset that will be found on the server using its SHA1 code.

Functional Requirements

- 1. Place to upload link to data.
- 2. Place to upload scripts (if provided).
- 3. Place to upload Jupyter Notebooks (if provided).
- 4. Docker config files (if provided) or link to "Generate Config Settings".

Technical Requirements

- 1. Back end version control for scripts
- 2. Jupyter Hub Installation
- 3. Working, public web server.

Primary Actors

- 1. Researchers
- 2. Students
- 3. Data scientists
- 4. System admins

Pre Conditions

- 1. User wished to upload SNC files.

Main Success Scenario

Dataset has been found on server and SNC files have been successfully uploaded to site and connected to the dataset's manifest.

Failed End Condition

User is unable to upload SNC files or dataset not found on server.

Trigger

User clicks "Contribute to Existing Database"

Dependent Use Case

NONE

What You Will be Evaluated On

1. (20%) Requirements Elicitation: Is there evidence that you evaluated the repository and asked clarifying questions.
2. (40%) Completeness of Use Case
 1. Are All components included; or their absence explained?
 2. Is there grounding in the project to explain where the use case originated? (Creativity and interpretation are allowed)
3. (20%) Is the Diagram Syntactically Correct UML
4. (20%) Are the use cases consistent: Is it evident that the team collaborated? Does it appear there are 5 use cases that are inconsistently complete and clear (i.e., divide and conquer without collaboration) [This would be “bad”]

Product requirements may be...

Functional

- Capabilities, features, or functions the software will execute
- Finite test steps can be written for it
- A description of a **behavior** that a system will exhibit under specific conditions. [Wiegers]

Non-functional

- Constraints on the solution
- Quality attributes
 - Performance
 - Maintainability
 - Reliability
 - Security
 - Interoperability
 - Etc...
- A description of a **property or characteristic** that a system must exhibit or a **constraint** that it must respect. [Wiegers]

System Requirements

A system is an interacting combination of elements to accomplish a defined objective. These include **hardware**, software, firmware, people, information, techniques, facilities, services, and other **support elements**.

System requirements are requirements for the **system as a whole**.

Software requirements are derived from system requirements. [SWEBOK 1.6]

A system requirements is a **top-level requirement** for a product that contains multiple subsystems, which could be all software or software **and hardware**. [Wiegers]

System Requirements

- Pre-requisites that often define the operating environment
- Architecture
- Hardware
 - *Storage*
 - *Memory*
 - *CPU*
 - *Connectivity etc.*

User Requirements

- Identify the different classes of user in your system
- What are the goals of that user? What do they want to be able to do?

*A **prospective customer** shall be able to **add items to a shopping cart**.*

*A **customer** shall be able to **check out**.*

Example: User Requirements to Functional Requirements



User Requirement for a Web Browser:

- The user shall be able to edit bookmarks.

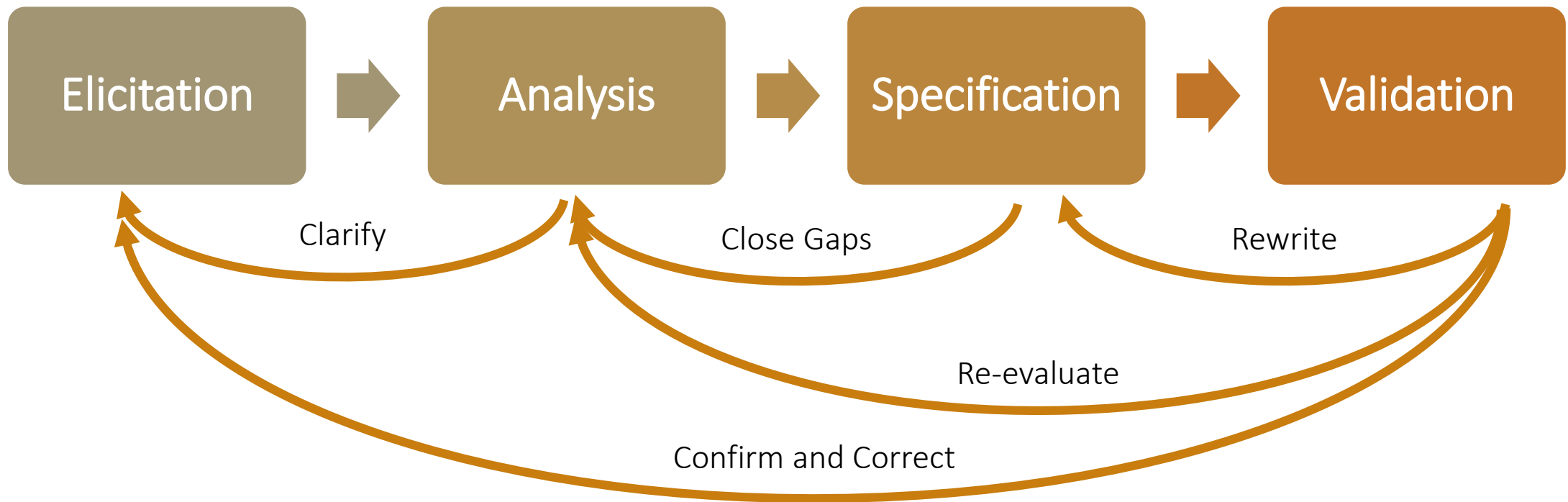
Functional Requirements for a Web Browser:

- The system shall display bookmarks as a collapsible and expandable hierarchical tree.
- The user shall be able to resequence bookmarks.
- The system shall display bookmark properties.
- The user shall be able to modify a bookmark's name, URL, and description. [Wiegers]

A few practical properties...

- Priority rating 
to enable trade-offs when faced with finite resources
- Status value 
to enable project progress tracking
- Unique identifier **1.2a**
to enable tracking and configuration management

Requirements Process



Requirements Elicitation Sources

- Goals (Business Requirements)
- Domain Knowledge
- Stakeholders
- Business Rules
- Operational Environment
- Organizational Environment

Requirements Elicitation Techniques

- Interviews
- Scenarios (use cases)
- Prototypes
- Facilitated meetings
- Observation
- User Stories
- Analyzing competitor products
- Analyzing existing system being replaced

Requirements Analysis – Why?

- Detect and resolve conflicts
- Discover the bounds of the software and how it must interact with its organizational and operational environment
- Elaborate system requirements to derive software requirements

Requirements Analysis – How?

Classify requirements

to help think about them in an organized way

- Functional vs non-functional
- Derived from stakeholder or emergent
- Product or process
- Priority
- Scope
- Volatility/stability

Requirements Analysis – How?

Organize requirements

to help track them and look for interactions

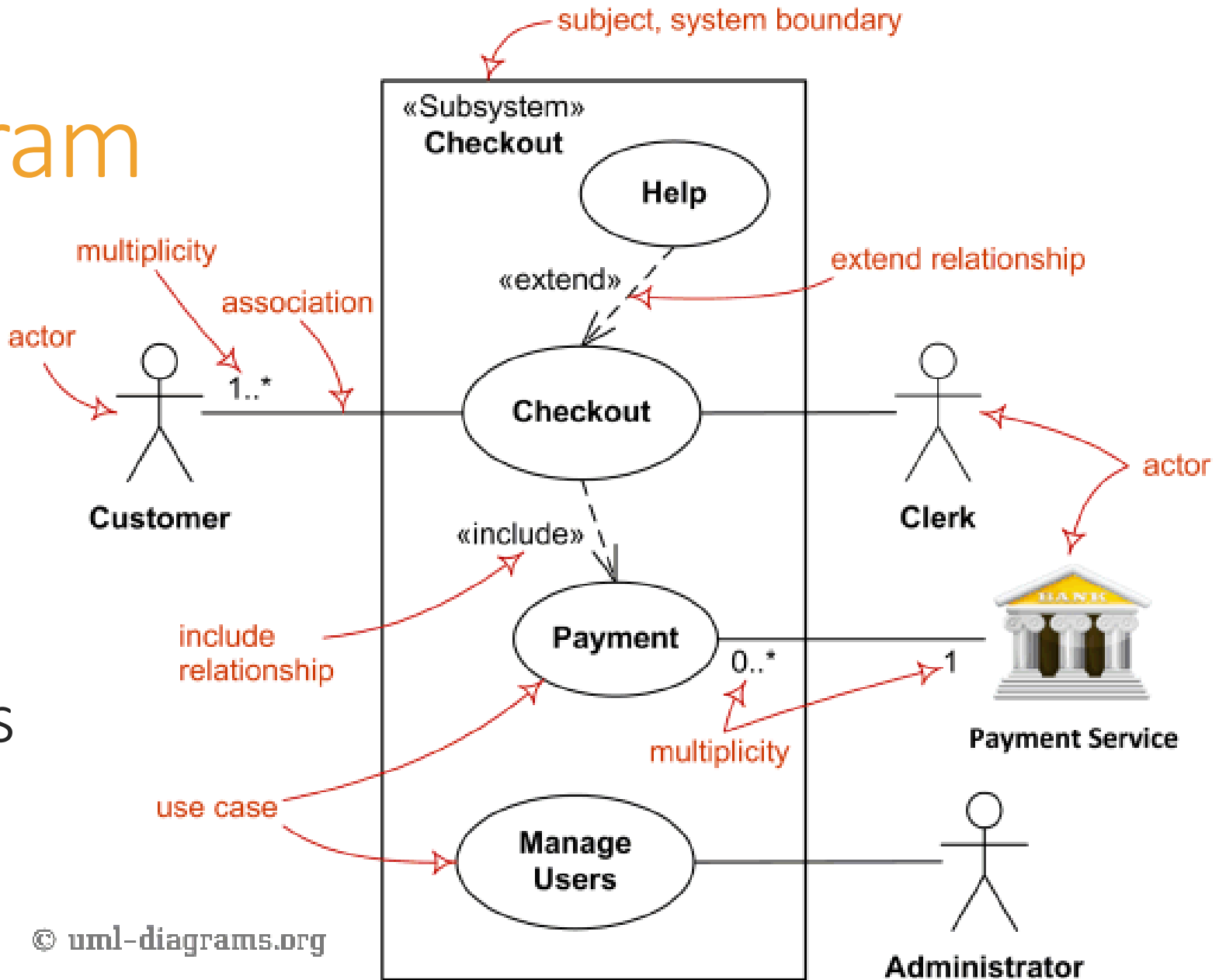
- By Business Process? Features? Subsystems?
- Leave room in your identifiers for additions and changes

Conceptual modeling to understand the problem

- Use Case diagrams, data models, others as deemed useful

Use Case Diagram

- Actors
- Use cases (functions)
- Included functions (required)
- Extended functions (optional)



Requirements Validation

- **Clear?** –unambiguous; one possible meaning to all readers
- **Correct?** – accurately states a user or external need
- **Consistent?** – no conflicts or contradictions
- **Complete?** – nothing missing
- **Feasible?** – can be implemented within resource constraints
- **Verifiable?** – we can tell if the requirement has been achieved

Requirements Validation

- Requirements reviews
 - Include a customer representative
- Prototyping
 - Especially for dynamic behaviors, user interfaces, critical features
 - Use low-quality prototypes to keep focus on topic
- Model Validation
- Acceptance Tests
 - Identifying and designing acceptance tests

Requirements Management

What to manage:

- Documentation
- Tracing - connecting up requirements, design, code, testing
- Change Management

How to manage:

- Specialized tools
- Simpler, cheaper, but less satisfactory: spreadsheets