ECE 321 Software Requirements Engineering

Lecture 5: Requirements elicitation – Learning and understanding user needs

The 3 steps of the requirements development process

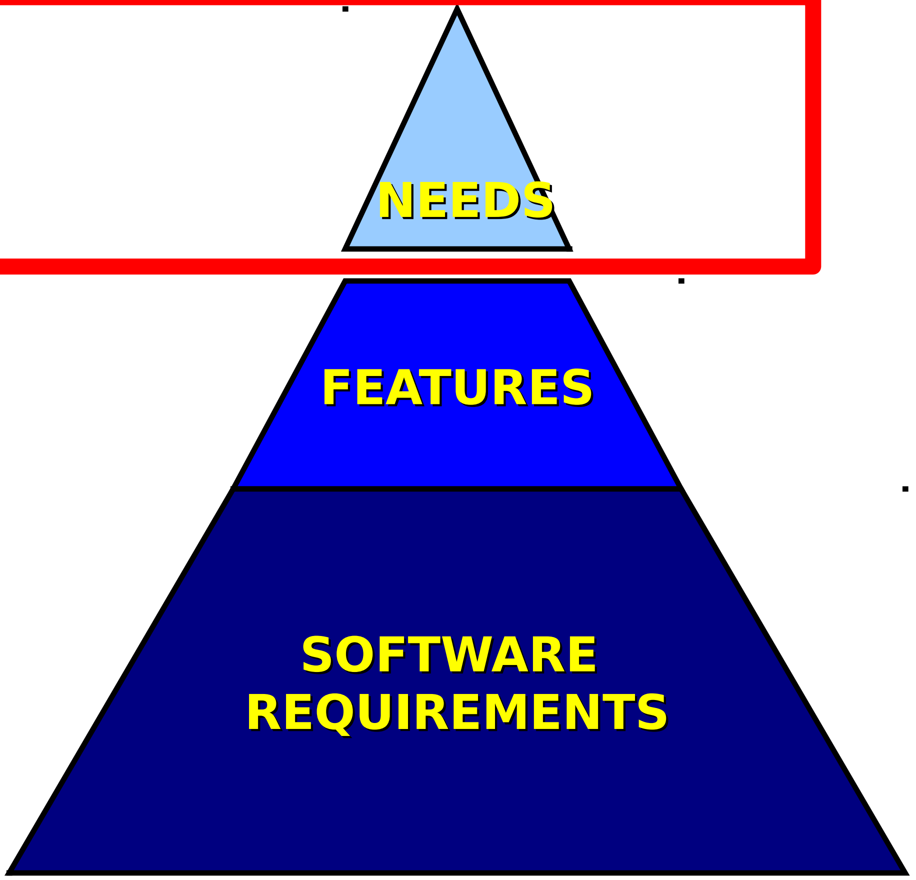
- **Requirement elicitation**
  - Understanding and analyzing the problem
  - Learning and understanding user needs
- **Requirement specification**
  - Developing a vision document
  - Developing requirement specification document
- **Requirement validation and verification**

The 3 steps of the requirements development process

- **Requirement elicitation**
  - Understanding and analyzing the problem
  - Learning and understanding user needs
- **Requirement specification**
  - Developing a vision document
  - Developing requirement specification document
- **Requirement validation and verification**

The software requirements pyramid

- problem domain
- solution domain



Needs vs. features

- Need
  - A reflection of the business, personal, or operational problem that must be addressed to justify a new system
  - The reason for building a system
- Feature
  - A service that the system provides to fulfill one or more stakeholder needs

Needs vs. features

Requirements are about learning !
user needs
and translating them into **features**
that are described in the
requirements specification document

Requirement elicitation in a nutshell

- Difficult
- Critical
- Error-prone
- Communication-intensive
- Collaborative partnership between stakeholders and developers

Information sources for learning user needs

- Interviews and discussions
  - I will discuss several techniques in this and next lecture
- Documents
  - Reviews of competing products
  - Descriptions of corporate and industry standards
  - Documents that describe business processes

More information sources for learning user needs

- Scenario analysis of user tasks
  - Use cases that describe particular tasks
- Events and responses
  - Recording of external events to which the system must react

Even more information sources for learning user needs

- Observing users at work
  - To validate collected information
  - To identify how the new system can improve the workflow
  - What is a challenge for this source?
    - People behave differently when they are being watched!

Yet more information sources for learning user needs

- Issue reports/support tickets for a current system
  - Contain problems but also feature requests!
  - Often available through help desk
  - Or look at issue tracking systems (e.g., Bugzilla or JIRA)

Still more information sources for learning user needs

- Marketing surveys and user questionnaires
  - May be used to select the most important root causes
  - May be used to test your understanding of requirements

Techniques for requirements elicitation

- Interviewing
- Workshops
- Use cases
- Prototyping

Interviewing is simple and direct

- Can be used in virtually every situation
- Need to make sure there is no bias in the questions
- Interview should listen and discuss
  - Interviewer can suggest things, but not force their opinion on the user
    - A very thin line!

Ask many why/what questions

- They help to understand the problem
- Use them to find out more about exceptions
  - “Why is this...?”
  - “What else could...?”
  - “What happens then...?”
  - “Why do you/don’t you...?”
  - “Does anyone ever...?”

Context-free questions to avoid bias

- Questions that do not bias user input
- After all context-free questions are asked, you can ask non-context-free questions
  - To give the interviewee a different view on the problem

Context-free vs. non-context-free questions

- Context-free
  - “Who is the user?”
  - “Who is the customer?”
  - “Where else can a solution to this problem be found?”
- Non-context-free
  - “You don’t think A is a good solution for B, do you?”

The key to a successful interview?

PREPARATION!

Preparing for a successful interview

- Write down all questions you want to ask
  - Though try to have a natural conversation
- Check these questions throughout the interview to make sure that you asked everything
- Research the background of the stakeholder and the company to be interviewed
  - Don’t ask for answers you could have known

How to have an even more successful interview

- Write down answers (on paper, not electronically)
  - Or record them
    - Some people do not like this!
    - Recording may be inaudible, or get lost
    - Make sure to have a backup plan
- End each interview with the ‘three most important needs or problems’
  - After a few interviews, you will find common priorities

Things to keep in mind about asking questions

- Ask if it is OK to ask questions at the start!
- People do not like not knowing the answer
  - It is not an exam!
  - The solution is empathy: make sure to emphasize that not knowing something is OK
- You may not have time to ask all questions you want
  - Start with the important ones

A generic template for interviews 1/3

1) Establishing customer or user profile
2) Assessing the problem
3) Understanding the user environment
4) Recap for understanding

A generic template for interviews 2/3

5) Interviewer’s inputs on customer’s problem

6) Assessing your solution

7) Assessing the opportunity

8) Assessing reliability, performance, and support needs

A generic template for interviews 3/3

9) Other requirements

10) Wrap-up

11) Interviewer’s summary

Interviewing template: 1. Establishing a user profile 1/2

- “What are your key responsibilities?”
- “What outputs do you produce, for whom?”
  - Who is the person in the project
- “When is the solution successful for you?”
  - This can be different for each stakeholder

Interviewing template: 1. Establishing a user profile 2/2

- “Which problems interfere with your success?”
- “What makes your job easier/more difficult?”
  - Learn about existing practices that could/should be integrated into the solution

Interviewing template: 2. Assessing the problem

- “For which problems do you lack a solution?”
- For each problem, ask:
  - “Why does this problem exist?”
    - Search for root causes
  - “How do you solve it now?”
    - Learn about successful and failed solutions
    - Note: workarounds are not real solutions
  - “How would you like to solve it in the future?”

Interviewing template: 3. Understanding the user environment

- “Who are the users and what is their background?”
- “Which platforms are in use now and in the future?”
- “Are there additional relevant applications?”
  - Should we generalize the project?
- “What are your expectations for training time?”
- “What kind of user help do you need?”
  - Tutorial, hard copy manual, help system

Interviewing template: 4. Recap for understanding 1/2

- Communicate your understanding of the problem so far back to the user
  - List all problems in your own words and validate them
  - Ask if there are more problems
  - Ask how the problems are ranked

Interviewing template:
4. Recap for understanding 2/2

- Validating the understanding is important
  - You may have different backgrounds
  - You may use different terminology
  - The user knows more about the problem domain

Interviewing template:
5. Interviewer’s inputs on the problem 1/2

- This step is about your view of the problem
- List any needs or additional problems you suspect are of concern to the user

Interviewing template: 5. Interviewer’s inputs on the problem 2/2

- For each suggested problem ask:
  - “Is this a real problem?”
  - “What are the reasons for it?”
  - “How do you currently solve the problem?”
  - “How would you like to solve it?”
  - “How would you rank it in comparison to other problems?”

Interviewing template:
5. Interviewer’s inputs on the problem 2/2

!

You learn **similar** things
to the previous steps, but this time
they are from **your point of view**

Interviewing template:
6. Assessing your solution

- Summarize the key capabilities of your solution
- Present how you see the solution
  - “What if you could ...”
  - “How would you rank the importance of these ...”

Interviewing template: 7. Assessing the opportunity

- Evaluate the costs and benefits of the project
  - “Who needs this application?”
  - “How many types of users would use the application?”
  - “How would you value a successful solution?”

Interviewing template: 8. Assessing reliability, performance and support needs

- Learn specific needs about the characteristics of the solution
  - “What are your expectations for reliability?”
  - “What are your expectations for performance?”
  - “Will you support the application?”
  - “What are the security requirements?”
  - “What are installation and configuration requirements?”

Interviewing template: 9. Other requirements

- Use this step to clarify outside factors that may affect the development
  - “Are there legal, regulatory, or environmental requirements?”
  - “Can you think of any other requirements”

Interviewing template: 10. Wrap-up

- Make sure you and the user did not miss anything
  - “Are there any questions I should be asking you?”
- Plan further communication
  - “If I need to ask follow-up questions, can I call you?”
  - “Would you like to participate in the requirement review?”

Interviewing template: 11. Interviewer’s summary

- Similar to step 5, but now about the entire interview
- Concentrate on summarizing the ‘three’ highest-priority needs or problems for the user
- Seize the moment when everything is fresh in your mind!

One more thing about interviews

- Interviews are hard to substitute!
- Always start with an interview
- Use other techniques to learn more if necessary

Mathematics for requirement analysts

$$10 + 10 + 10 = \cancel{30}$$

Usually you will end up with
~10-15 **unique** requirements after getting the 10 most important ones for 3 people

Techniques for requirements elicitation

- Interviewing
- Workshops
- Use cases
- Prototyping

Workshops are a powerful technique

- Gather all key stakeholders
- A short, intensively focused period
- Involve both users and developers

The main goals of a workshop

- Building an effective team
  - Everyone is committed to one purpose
- All stakeholders get their say
- Agreement between the stakeholders and development team about what the project must do
  - A preliminary system definition
  - The goal is to have it available immediately

Preparing a workshop 1/3

- Sell the concept
  - Explain to the stakeholders how a workshop can help
    - Helps to avoid the ‘Not another meeting’ syndrome
  - Have a clear idea of the topic and contents of the workshop
  - Hold it, and they will come (hopefully)

Preparing a workshop 2/3

- Take care of the logistics
  - Make sure the right stakeholders are there
  - Provide ‘warm-up materials’ to **all** participants
  - Prepare the location of the workshop
    - e.g., travel, catering, lighting, room temperature
- Set the agenda

A typical agenda of a requirements workshop

- Introduction
- Context
- Brainstorming
- Lunch
- Brainstorming
- Feature definition
- Idea reduction and prioritization
- Wrap up

Preparing a workshop 3/3

- Have a facilitator/moderator
  - Preferably someone who is not a stakeholder
  - Keeps everything under control
  - Makes sure the agenda is followed
- Be professional
  - For stakeholders, the quality of the logistics is correlated with the quality of your project!

Running the workshop

- Tips and tricks
- Brainstorming and idea reduction
- Storyboarding

Tips and tricks for workshops

- Establish ground rules
- Stay in scope
- Capture items for later consideration
- Timebox discussions
- Keep everyone engaged

Tips: Establishing ground rules

- Participants should agree on some basic operating principles
  - Start and end on time
  - Return from breaks promptly
  - Hold one conversation at a time
  - Expect everyone to contribute
  - Focus comments and criticism on issues, not individuals
  - Etc.

Tips: Staying in scope

- Vision documents should be used to confirm whether newly proposed requirements fit in scope
- Stay at right level of abstraction
  - Not too vague, not too detailed :)
  - Often details can lead to heated discussions
  - At this point, these details are not necessary

Tips: Capturing items for later consideration

- Usually random questions and points will popup during the discussion
  - e.g., quality attributes, business rules, user interface ideas
  - Organize the information so it will not get lost
    - For example write them on an extra whiteboard
  - Do not discuss them directly – remember to stay focused
- Come back to these points later

Tips: Timeboxing discussions

- Allocate a fixed amount of time for each agenda item
  - Facilitator should keep track
  - Don’t be too strict, but try to stick to the schedule
- Spending too much time on one topic may lead to neglecting other topics entirely
  - Nobody wants to (or can) stay longer than the schedule

Tips: Keeping everyone engaged 1/2

- Stay away from workshops with too many people
  - They are slow and inefficient
- Consider running multiple workshops in parallel if needed
- Five or six active participants is about the right number

Tips: Keeping everyone engaged 2/2

- Facilitator must engage people who drifted out of the discussion
  - They could be too shy
  - They might be frustrated:
    - They see things going the wrong way
    - They feel not treated seriously

Brainstorming

- The most creative, innovative ideas often come from combining multiple, seemingly unrelated ideas
- Let people think and combine their inputs
  - Simple, easy to do and fun

Running a brainstorming session

- Requires supplies (sticky notes, markers, pins, “soft wall”, whiteboard)
- Needs a facilitator with authority to manage the session
- Has clear objectives
- Generate as many ideas as possible
- Mutate and combine ideas
- Try to stay away from criticism and debate

Idea reduction

- Pruning via acceptance and rejection of ideas
- Grouping of ideas
  - New features
  - Performance issues
  - Enhancements
  - User interface, etc.
- Prioritization of the gathered ideas

Storyboarding

- Provide a ‘concept view’ of the project
- Allow stakeholders to give an early reaction on this view
  - Often triggers a “Yes, great” or “No, this is not what we meant” reaction
- Inexpensive and easy to create
- Often used by movie industry

The types of storyboards

- Passive
  - Tell a story to the user
    - e.g., sketches, pictures, presentations
- Active
  - e.g., animations, automated presentations
- Interactive
  - Experience the system in a realistic manner
    - e.g., simulations, mock-ups, “almost prototypes”

Tips for storyboarding

- Do not invest too much in a storyboard
  - People will be scared to make changes
- Make it easy to modify
- Do not make the story too good
- Whenever possible, make the storyboard interactive

Techniques for requirements elicitation

- Interviewing
- Workshops
- Use cases
- Prototyping

Use cases define what a user intends to do with the system

- A use case is a sequence of interactions between a system and an actor
  - Actor is a person, software system, or hardware device that interacts with the system
- Use cases are very common in the object-oriented domain

Use cases discuss what a user needs to accomplish

- In contrast, traditional elicitation asks users what they want the **system** to do
- Use cases identify the who, what and how of system behaviour
-

A use case is a stand-alone activity that an actor can perform

- It might encompass a number of interactions between the actors and the system
- A high-level representation of the user requirements

An example of a use case-driven approach

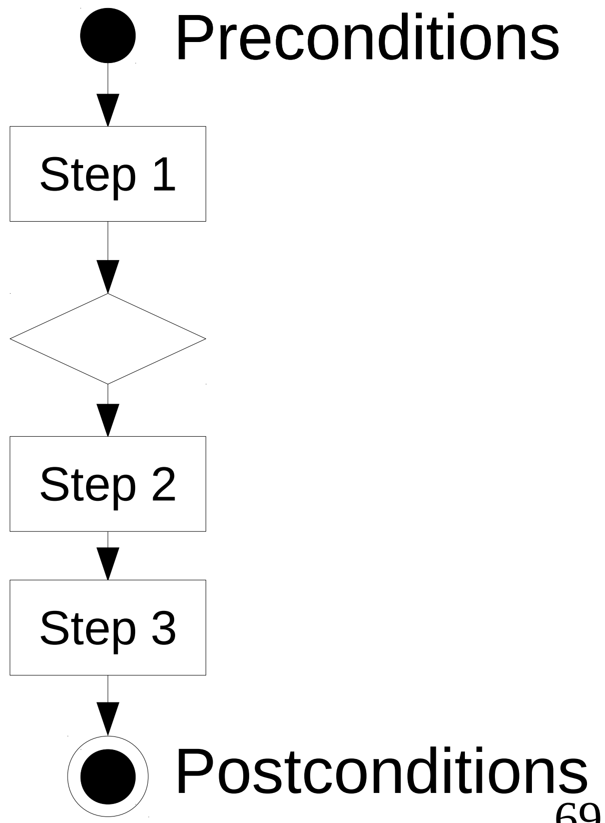
- (Please read the handout)

Use cases are a collection of related usage scenarios

- A scenario is a specific instance of a use case representing a single path through the use case
- **Normal course (primary scenario)**
  - The ‘usual’ sequence of events for this use case
- **Alternative courses (secondary scenarios)**
  - All other valid scenarios within the use case

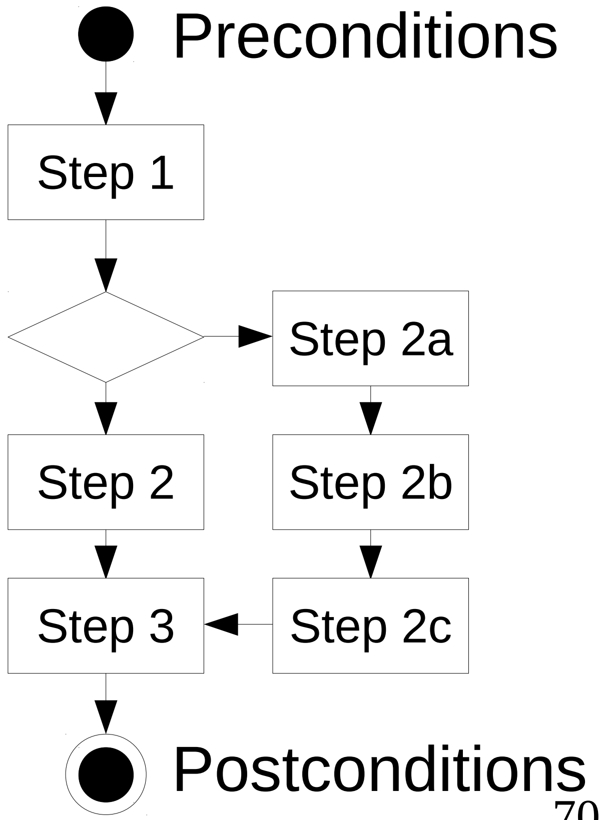
A collection of related usage scenarios

- A normal course for the “request a chemical” use case is to request a chemical which is available in the stockroom



A collection of related usage scenarios

- A normal course for the "request a chemical" use case is to request a chemical which is available in the stockroom
- Alternative can be to order chemical from a vendor



Documenting a use case 1/4

- A unique identifier
- A name “verb + object”
  - e.g. “Request a chemical”
- Info about who and when created and modified
- Description (natural language)

Documenting a use case 2/4

- A list of preconditions
  - All things that must be satisfied before the use case can begin
- A list of postconditions
  - The state of the system after the use case is successfully completed
- A numbered list of steps
  - They lead from pre to postconditions

Documenting a use case 3/4

- The primary scenario (steps)
- Secondary scenarios (steps)
- Inclusion of other use cases
- Priority
- Frequency of use
- Business rules
- Special requirements
- Assumptions

Documenting a use case 4/4

- Exceptions
  - Conditions that prevent a task from succeeding
    - e.g. chemical is not commercially available for “Request a Chemical” use case
  - How to handle an exception
    - Define what is the best way to deal with unsuccessful use case
    - Show an appropriate message to the user
    - Helps to build robust software

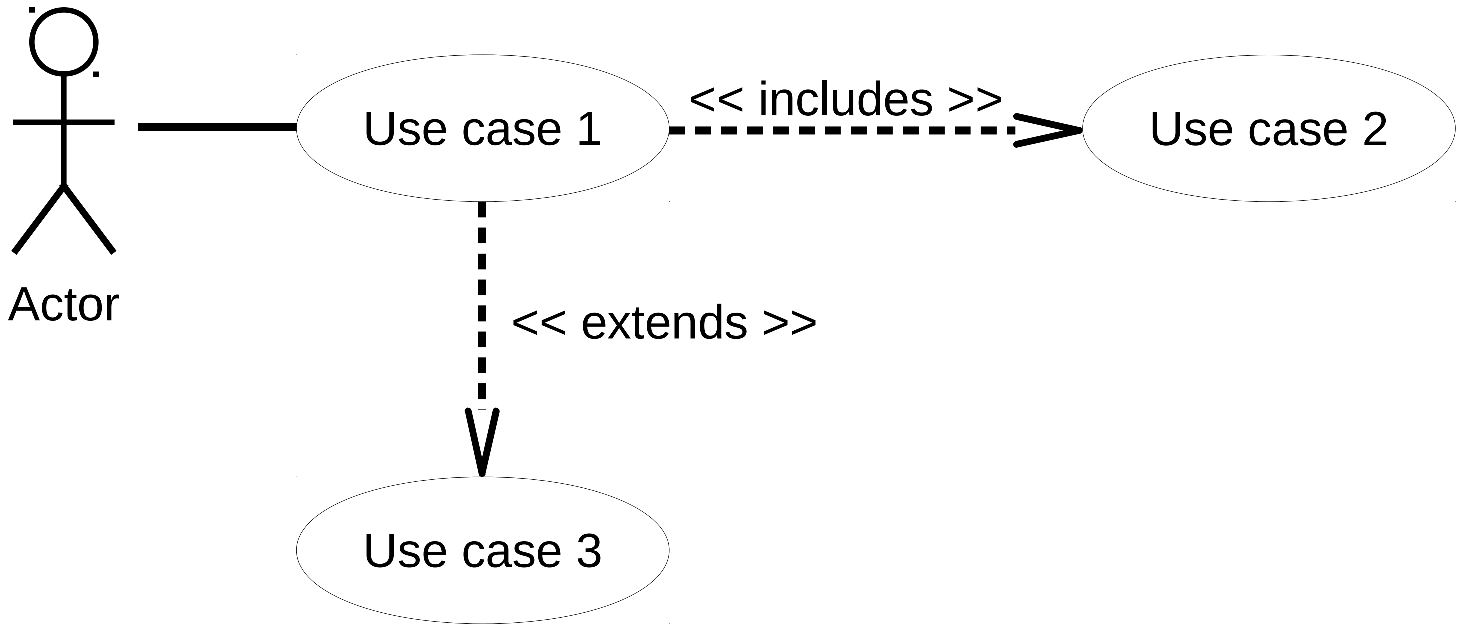
How to identify use cases?

- Identify the actors first, then the business processes in which each participates
- Identify the external events to which the system must respond and relate them to actors and use cases
- Express business processes in terms of scenarios
  - And generalize these scenarios into use cases

Use case models

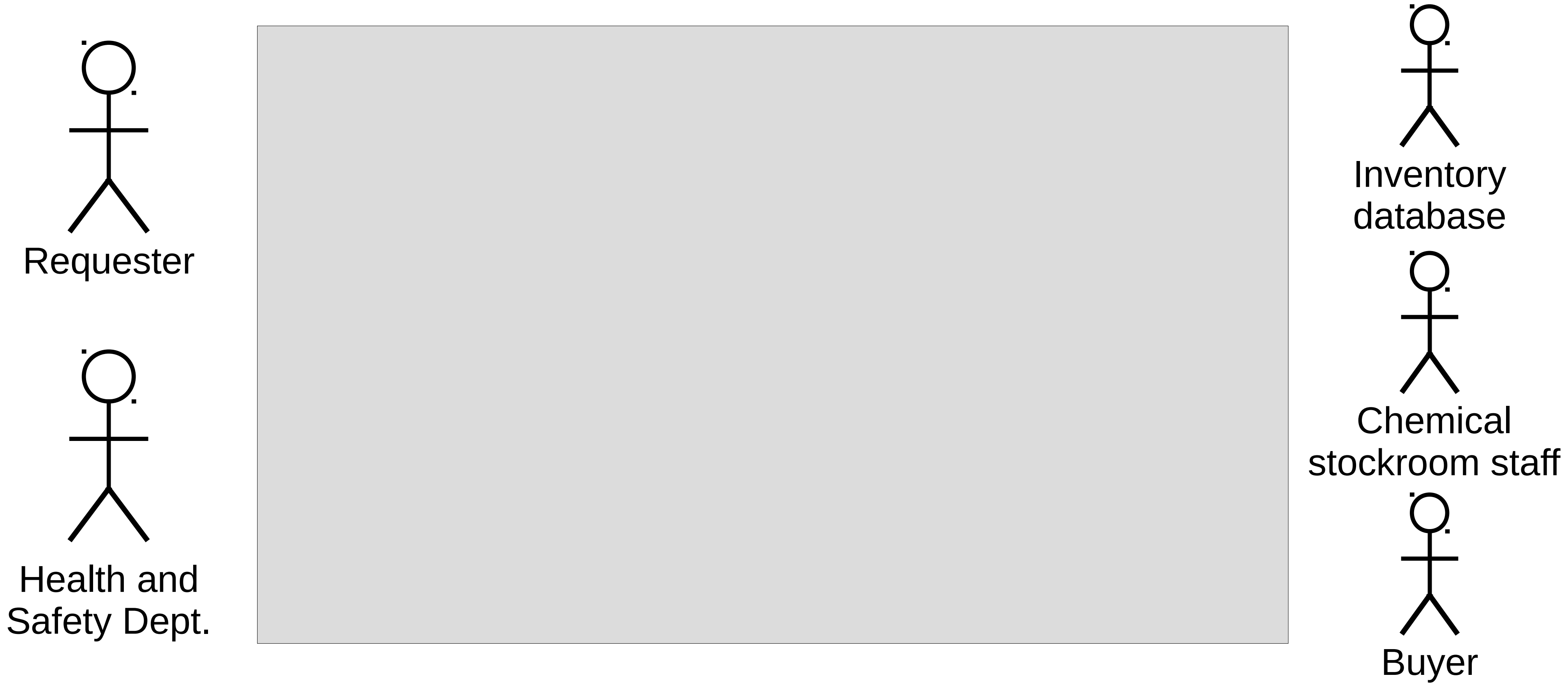
- Consist of all the actors of the system and all use cases through which they interact with the system
- Also contains relationships between use cases
- Can be visualized using a use case diagram in UML notation

Use cases in UML

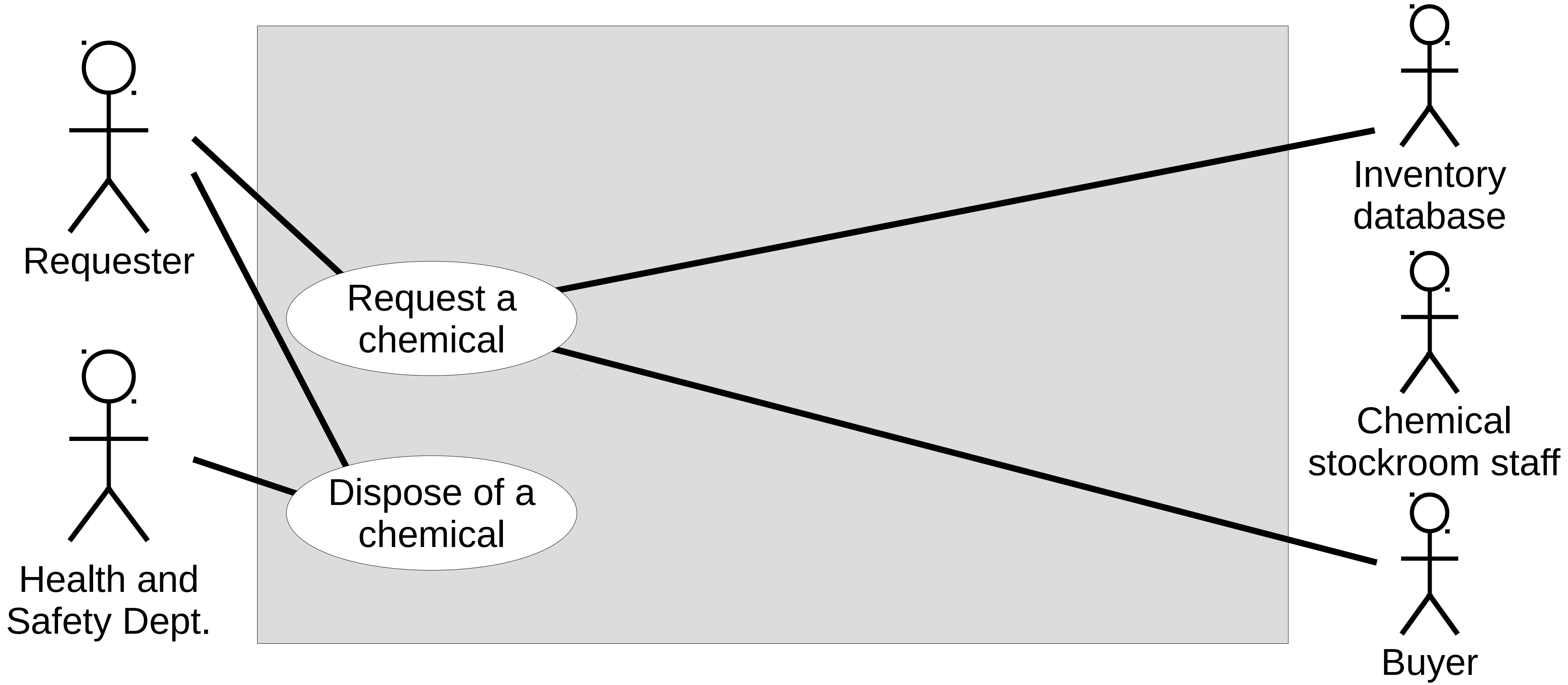


Use case diagram for Chemical Tracking System

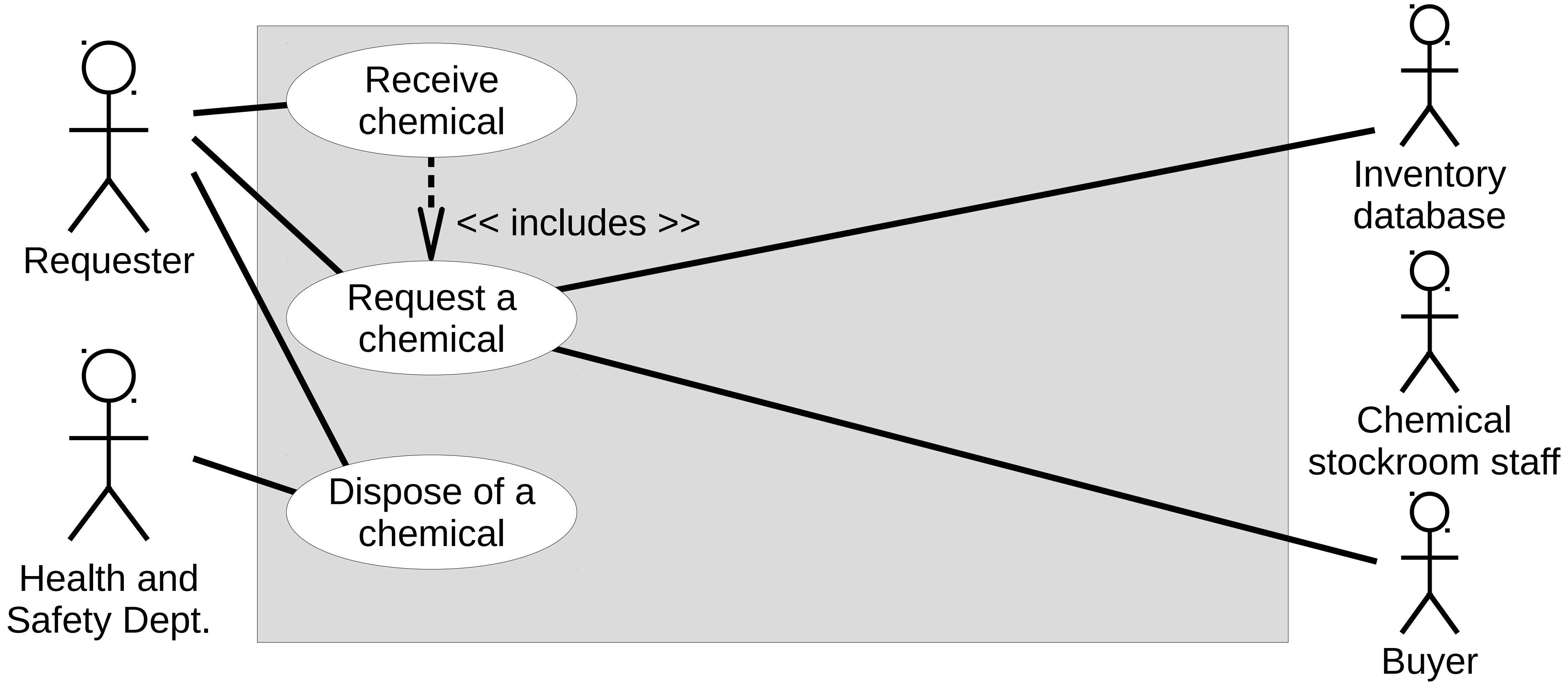
Use case diagram for Chemical Tracking System



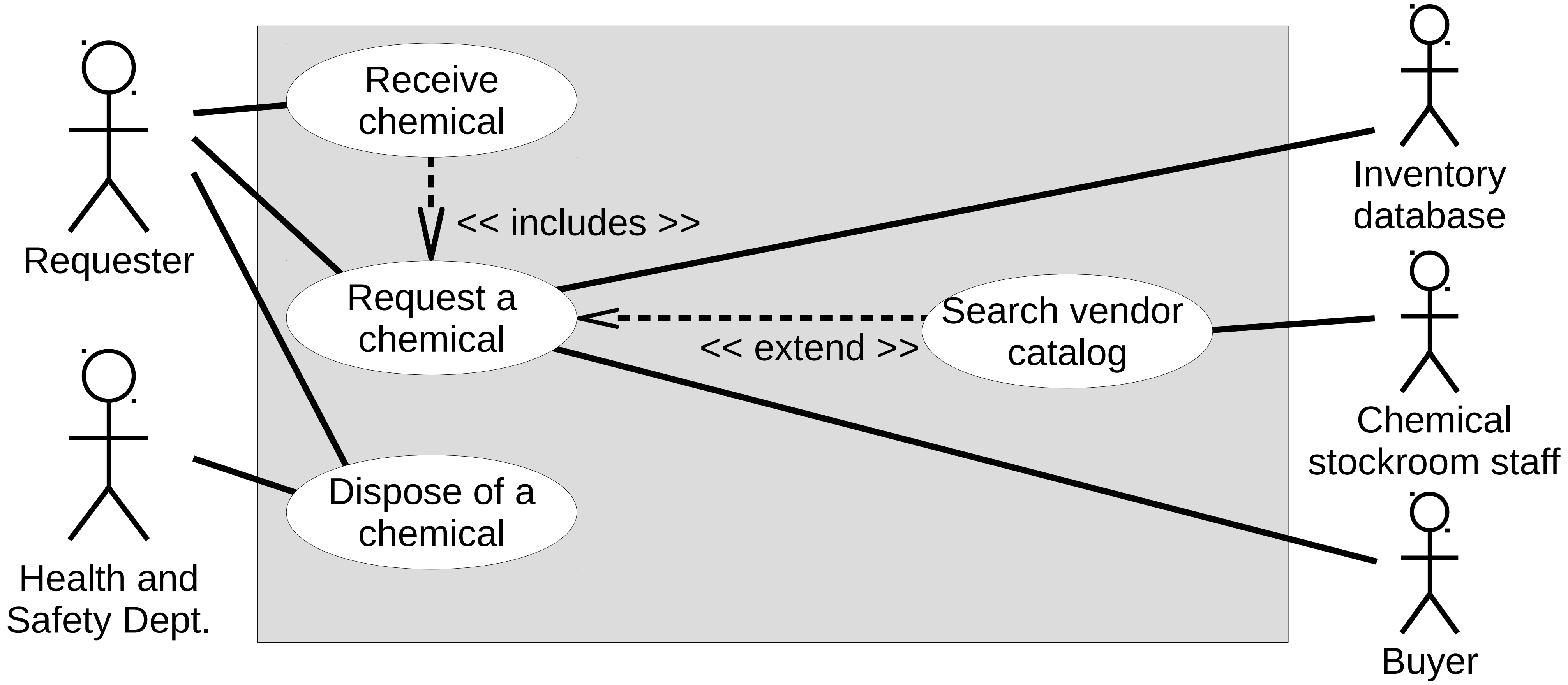
Use case diagram for Chemical Tracking System



Use case diagram for Chemical Tracking System



Use case diagram for Chemical Tracking System



An example of a use case

Use Case ID	**UC-1**	Use Case Name	**Request a Chemical**
Created by	Tim	Last Updated by	Janice
Date Created	12 / 04 / 2018	Date Last Updated	12 / 27 / 2018

An example of a use case

Use Case ID	**UC-1**	Use Case Name	**Request a Chemical**
Created by	Tim	Last Updated by	Janice
Date Created	12 / 04 / 2018	Date Last Updated	12 / 27 / 2018
Actors	Requester		

An example of a use case

Use Case ID	**UC-1**	Use Case Name	**Request a Chemical**
Created by	Tim	Last Updated by	Janice
Date Created	12 / 04 / 2018	Date Last Updated	12 / 27 / 2018
Actors	Requester		
Description	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system satisfies the request either by offering the Requester a new or used container of the chemical from the chemical stockroom or by letting the Requester create a request to order from an outside vendor.		

An example of a use case

Use Case ID	**UC-1**	Use Case Name	**Request a Chemical**
Created by	Tim	Last Updated by	Janice
Date Created	12 / 04 / 2018	Date Last Updated	12 / 27 / 2018
Actors	Requester		
Description	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system satisfies the request either by offering the Requester a new or used container of the chemical from the chemical stockroom or by letting the Requester create a request to order from an outside vendor.		
Preconditions	1. User's identity has been authorized. 2. User is authorized to request chemicals. 3. Chemicals inventory database is online.		

An example of a use case

Use Case ID	**UC-1**	Use Case Name	**Request a Chemical**
Created by	Tim	Last Updated by	Janice
Date Created	12 / 04 / 2018	Date Last Updated	12 / 27 / 2018
Actors	Requester		
Description	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system satisfies the request either by offering the Requester a new or used container of the chemical from the chemical stockroom or by letting the Requester create a request to order from an outside vendor.		
Preconditions	1. User's identity has been authorized. 2. User is authorized to request chemicals. 3. Chemicals inventory database is online.		
Postconditions	1. Request is stored in the Chemical Tracking System. 2. Request was emailed to the chemical stockroom or to a Buyer.		

Scenarios for the use case

Normal Course	**1.0 Request a Chemical from the Chemical Stockroom** • Requester specifies the desired chemical. • System validates that the chemical is valid. • System lists the containers of the desired chemical that are in the chemical stockroom. • Requester has an option to View the Container History for any container. • Requester selects a specific container or asks to place a vendor order (alternative course 1.1). • Requester enters the information to complete the request. • System stores the request and emails it to the chemical stockroom.

Scenarios for the use case

Normal Course	**1.0 Request a Chemical from the Chemical Stockroom** • Requester specifies the desired chemical. • System validates that the chemical is valid. • System lists the containers of the desired chemical that are in the chemical stockroom. • Requester has an option to View the Container History for any container. • Requester selects a specific container or asks to place a vendor order (alternative course 1.1). • Requester enters the information to complete the request. • System stores the request and emails it to the chemical stockroom.
Alternative Course	**1.1 Request a Chemical from a Vendor** (branch after step 5) • Requester searches vendor catalogs for a chemical. • System displays a list of vendors with available container sizes, grades, and prices. • Requester selects a vendor, container size, grade, and number of containers. • Requester enters the information to complete the request. • System stores the request and emails it to the Buyer.

Exceptions for the use case

Exceptions	**1.0.E.1 Chemical is not valid** (at step 2) • System display a message “The chemical does not exist”. • System asks the Requester whenever he or she wishes to request another chemical or to exit. 3a. Requester asks to request another chemical. 4a. System starts normal course over again. 3b. Requester asks to exit. 4b. System terminates the use case.

Exceptions for the use case

Exceptions	**1.0.E.1 Chemical is not valid** (at step 2) • System display a message "The chemical does not exist". • System asks the Requester whenever he or she wishes to request another chemical or to exit. 3a. Requester asks to request another chemical. 4a. System starts normal course over again. 3b. Requester asks to exit. 4b. System terminates the use case. **1.0.E.2 Chemical is commercially not available** (at step 5) • System display a message "No vendors for that chemical". • System asks the Requester whenever he or she wishes to request another chemical or to exit. 3a. Requester asks to request another chemical. 4a. System starts normal course over again. 3b. Requester asks to exit. 4b. System terminates the use case.

Extra info for the use case

Includes	**UC-22** View Container History

Extra info for the use case

Includes	**UC-22** View Container History
Priority	High

Extra info for the use case

Includes	**UC-22** View Container History
Priority	High
Frequency of Use	Approximately 5 times per week by each chemist; 100 times per week by each member of chemical stockroom staff

Extra info for the use case

Includes	**UC-22** View Container History
Priority	High
Frequency of Use	Approximately 5 times per week by each chemist; 100 times per week by each member of chemical stockroom staff
Business Rules	**BR-28** Only staff who are authorized by their laboratory managers may request chemicals

Extra info for the use case

Includes	**UC-22** View Container History
Priority	High
Frequency of Use	Approximately 5 times per week by each chemist; 100 times per week by each member of chemical stockroom staff
Business Rules	**BR-28** Only staff who are authorized by their laboratory managers may request chemicals
Special Requirements	1. The system must be able to import a chemical structure in the standard encoded form from any of the supported chemical drawing packages

Extra info for the use case

Includes	**UC-22** View Container History
Priority	High
Frequency of Use	Approximately 5 times per week by each chemist; 100 times per week by each member of chemical stockroom staff
Business Rules	**BR-28** Only staff who are authorized by their laboratory managers may request chemicals
Special Requirements	1. The system must be able to import a chemical structure in the standard encoded form from any of the supported chemical drawing packages
Assumptions	1. Imported chemical structures are assumed to be valid

Extra info for the use case

Includes	**UC-22** View Container History
Priority	High
Frequency of Use	Approximately 5 times per week by each chemist; 100 times per week by each member of chemical stockroom staff
Business Rules	**BR-28** Only staff who are authorized by their laboratory managers may request chemicals
Special Requirements	1. The system must be able to import a chemical structure in the standard encoded form from any of the supported chemical drawing packages
Assumptions	1. Imported chemical structures are assumed to be valid
Notes and Issues	1. Tim will find out whenever management approval is needed to request a chemical on the Level 1 hazard list. Due date 1 / 4 / 2018

Fully-dressed vs. casual use cases

- Fully-dressed (what you just saw) can be useful
  - When user representatives are not closely engaged with the developers
  - Application is complex and high-risk
  - A comprehensive set of test cases will be developed based on the use cases
  - Development is done in collaborative/remote manner

A more casual use case example

User Actions	System Responses
1. Specify the desired chemical 4. If desired, ask to view the history of any container 5. Select a specific container (done) or ask to place a vendor order (alternative course 1.1)	2. Verify that the chemical requested is valid 3. Display a list of containers of the desired chemical that are in the chemical stockroom’s current inventory

A more casual use case example

User Actions	System Responses
1. Specify the desired chemical 4. If desired, ask to view the history of any container 5. Select a specific container (done) or ask to place a vendor order (alternative course 1.1)	2. Verify that the chemical requested is valid 3. Display a list of containers of the desired chemical that are in the chemical stockroom’s current inventory

Tips for use cases

- Do not define too many
  - Try to be at the appropriate level of abstraction
  - Use secondary scenarios instead of new use cases, if possible
- Keep use cases simple
  - With a reasonable number of branching conditions
- Do not include user interface design in the use cases
  - “System presents choices” instead of “system displays drop-down list”

More tips for use cases

- Do not include data definitions in the use cases
  - Will lead to duplicate and inconsistent definitions across use cases
  - Collect them in a project-wide dictionary
- If a user cannot understand a use case, there is a problem!
- If you define a brand new system, users may have a hard time defining use cases
  - Try to work with prototypes or storyboards

Things to keep in mind about use cases

- Not everything can be defined as a use case
- They cannot be used for example to define
  - Qualities
  - Other non-functional requirements

Techniques for requirements elicitation

- Interviewing
- Workshops
- Use cases
- Prototyping

Prototyping

- A prototype is a partial implementation of a software system
- Built to help understand requirements
- Effective in addressing ‘Yes, but” and “I don’t know” syndromes
  - “I don’t know what I need, but I’ll know when I see it”

Why prototyping works

- Often users do not know what they really want (unless they see or touch it)
- We do not want to guess what to build
- Critiquing is much easier than creating

Benefits of prototyping

- Enables early feedback
- Closes gaps in understanding the requirements
- Stimulates a user’s thinking
- More fun than reading a requirement document

How to use prototypes

- To clarify and complete requirements
  - Usually they are a preliminary implementation of a part of the system that is not well understood
- To explore design alternatives
  - e.g., different GUI designs, different code optimization techniques
- To build incrementally
  - Add to the prototype incrementally
    - Watch out - A prototype is not always the correct foundation for a product!
    - Often we throw away the prototype

Examples of prototypes

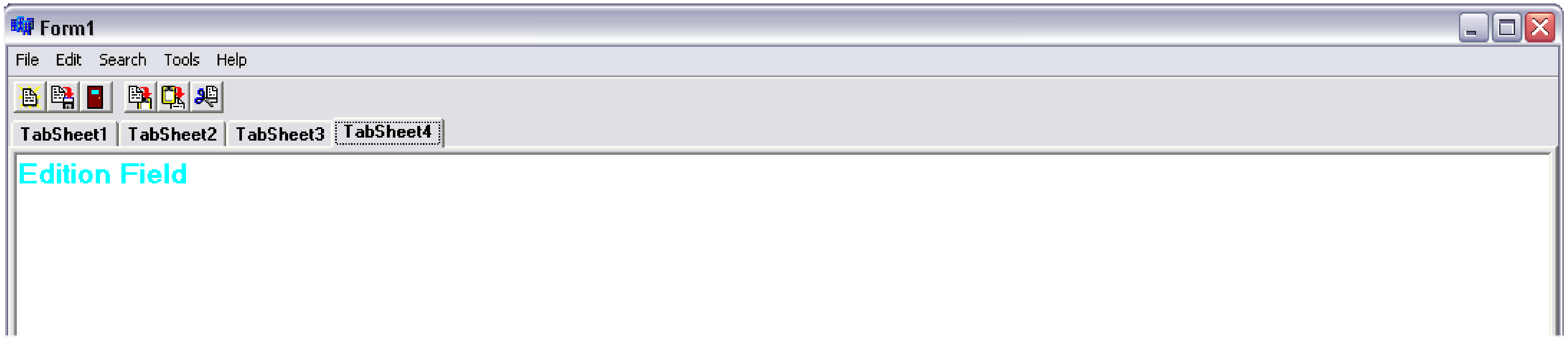
- An airplane prototype will actually fly (hopefully)
- Usually not the case for software prototypes
  - They might not do anything useful
  - Working models
  - Visual displays
  - Simulations
  - GUI designs

Horizontal prototyping

- Cover a wide range of system functionality
- The ‘usual’ software prototype
- Focuses on showing (portions of) the user interface
  - Look (colours, graphics, controls, etc.)
  - Navigation
  - System responses are usually hardcoded
  - Usually no ‘real’ functionality
    - Users should focus on requirements and workflow rather than details of screen elements

Example of horizontal prototype

- Multi-editor prototype
  - Can we start the development?



Vertical prototyping

- Construct a few requirements but in a quality manner
- Works like a real system, but concerns only a limited slice of the entire application

When to use vertical prototyping?

- When you want to know more about a specific part of the requirements
  - When needing to evaluate solutions (e.g., database schema)
  - When testing critical timing requirements
  - Etc.

Throwaway vs. evolutionary prototype

- Throwaway prototype
  - Build as quickly and cheaply as possible
  - Forget about software qualities
  - Uses shortcuts and alternative technologies
- Evolutionary prototype
  - Uses the same architecture as the final product
  - System will be built by evolving the prototype

Paper prototypes

- Not every prototype involves programming
- Paper prototypes show what the screen will look like
  - Use paper, coloured sticky notes, markers etc.
- Easy to develop and adapt
- Consider them as the first option of prototyping

Electronic prototypes

- Many available tools
  - Many programming IDEs (integrated development environments) support prototyping
    - e.g., MS Visual Studio, Eclipse, IntelliJ
  - Scripting languages
    - e.g., Perl, Python, Matlab, JavaScript
  - Drawing tools
    - e.g., MS Visio, MS PowerPoint, Google Apps

Customer input must be categorized

- Users will not give you their feedback in a nicely organized list
- Often they only realize what they need while talking to you

User input categories

- Business requirements
- Use cases
- Business rules
- Functional requirements
- Quality attributes
- Constraints
- Data definitions
- Solution ideas

Customer input: business requirements

- Anything that describes the financial, marketplace or other business benefit that customer wishes to gain
  - “Increase revenue by $$ per month”
  - “Save $$ per year on electricity”

Customer input: use cases

- General statements of user goals or business tasks that user needs to perform
  - “I need to <do something>”
  - “I need to print a mailing label for a package”
  - “I need to enter my authorization code”
  - Etc.

Customer input: business rules

- Only certain user classes can perform an activity under special conditions
  - “Must comply with <some corporate policy>”
  - “If <some condition is true> then <something happens>”

Customer input: functional requirements

- Describe the observable behaviours the system will exhibit under certain conditions and actions
  - Usually they make up the bulk of the requirement specifications
    - “The analyst must be able to sort the list in forward and reverse alphabetical order”
    - “The user must be able to place an order with an external vendor”

Customer input: quality attributes

- Define how well the system performs some action
  - Look for keywords such as “fast”, “user-friendly”, “robust”, “easy”, “secure”, “efficient”

Customer input: external user requirements

- Describe the connections between the system and the rest of the universe
  - Separately for user interface, interface with other software and interface with hardware
    - “Must use information stored in <some other system>”
    - “Must be able to read (or write) files in <some format>”

Customer input: constraints

- Restrict the options that are available to the developer
  - “Files submitted may not exceed 10MB in size”
  - “The browser must use 128-bit encryption”
  - “Application must be written in <a specific programming language>”

Customer input: data definitions

- Describe complex business data
  - The format
  - The data type
  - The allowed values
  - The default value
  - Composition
    - “Our phone numbers are 7 digits plus 3 digits for the extension”

Customer input: solution ideas

- Much of what users present as requirements fits into this category
  - e.g., a specific way of interaction with the system
    - Analyst must learn the real requirement, not an instance described by the user
  - User says: “... then select the province where I want to send the package from a drop-down list...”
  - Real requirement is: the system should permit the user to specify the province where the package should be sent

Discussed so far

- Different elicitation techniques
  - Not a single perfect one
- Interviews
- Workshops
- Use cases
- Prototyping
- The types of customer input / feedback