[CS3704] Software Engineering

Dr. Chris Brown
Virginia Tech
9/20/2023

Announcements

- **Project Milestone 1 due Friday (9/22) at 11:59pm**
    - Lightning talk presentation slides
    - Proposal document

Requirements Engineering

Requirements Engineering
Requirements Elicitation
Prioritization
Requirements in Practice

Learning Outcomes

By the end of the course, students should be able to:

- **Understand software engineering processes, methods, and tools used in the software development life cycle (SDLC)**
- **Use techniques and processes to create and analyze requirements for an application**
- **Use techniques and processes to design a software system**
- **Identify processes, methods, and tools related to phases of the SDLC**
- **Explain the differences between software engineering processes**
- **Discuss research questions and current topics related to software engineering**
- **Create and communicate about the requirements and design of a software application**

Requirements

Goal: Understand customer requirements for the software system

- The *what* of the project

- Very difficult to “get right” the first time and evolve over the course of development
  - Remember the Top 3 reasons for project failure: **(2)** Incomplete and **(3)** Changing Requirements
- *Software Artifacts:* requirements documents, use cases, user stories,...

What are requirements?

Definition: Capabilities and conditions to which the system — and more broadly, the project — must conform. [Larman]

- Focusing on the **WHAT** *not* the **HOW**
- **Should always come *first* in the SDLC**

Warm-Up

Discuss: What is the difference between requirements analysis and requirements specification?

Requirements Engineering

- **Definition:** the a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real-world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies. **[Easterbrook]**

Requirements Engineering (RE) is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies

Not a phase or stage!

Communication is as important as the analysis

Quality means fitness-for-purpose. Cannot say anything about quality unless you understand the purpose

Designers need to know how and where the system will be used

Requirements are partly about what is needed...

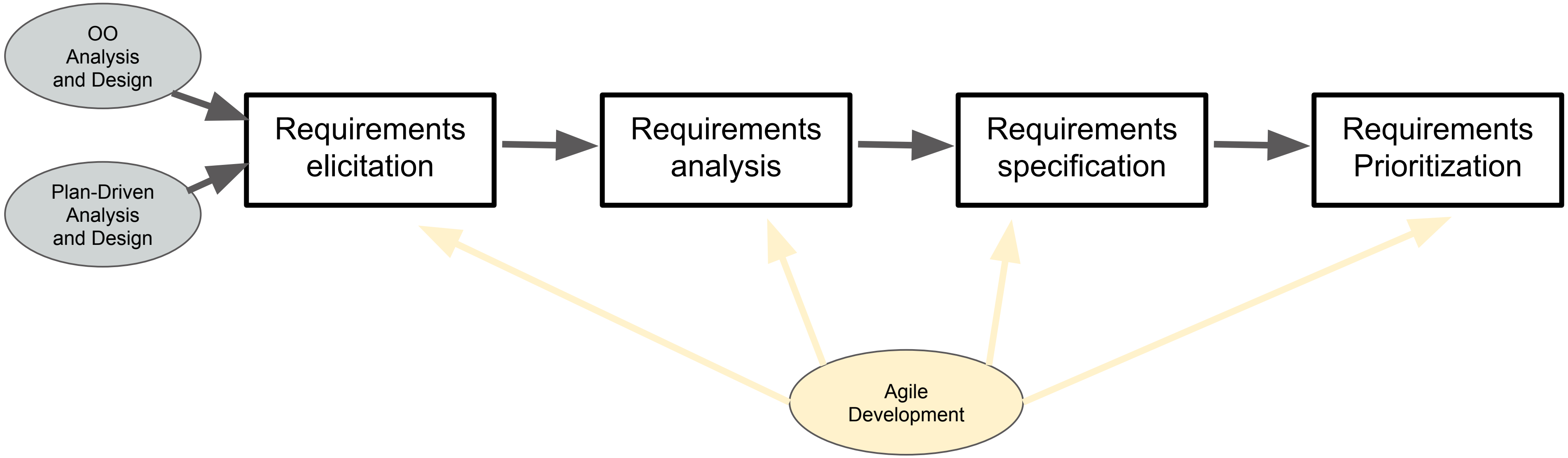
...and partly about what is possible

Need to identify all the stakeholders - not just the customer and user

Requirements Engineering (cont.)

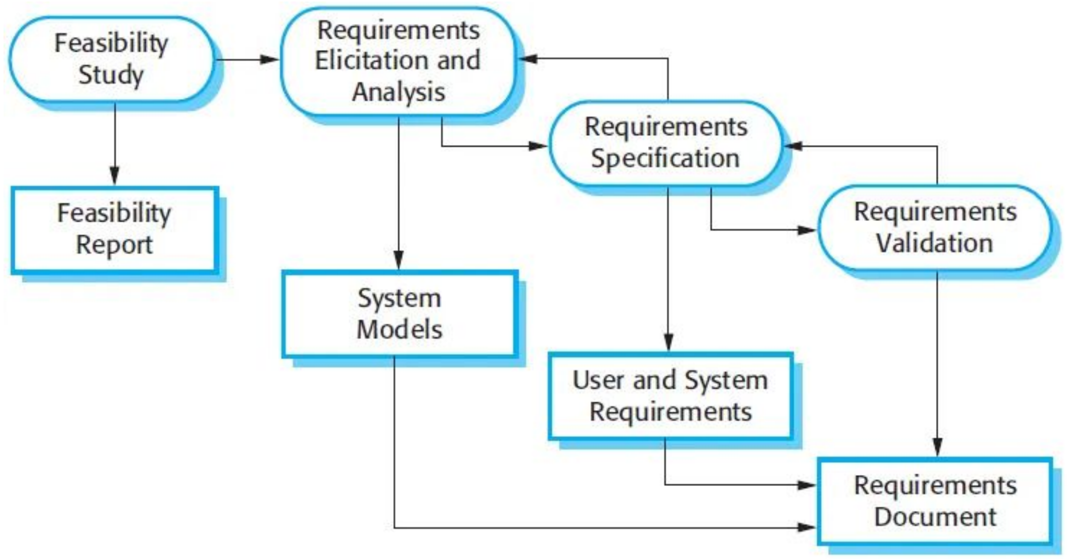
- *Requirements*: list of features demanded by the customer
    - Rarely one single customer in practice
    - Requirements may be hard to articulate, conflict,...
- *Engineering*: a discipline whose output needs to be carefully engineered.
    - understanding is gained from applying systematic analysis techniques, rather than documented specifications and models.

Requirements Engineering (cont.)


OO Analysis and Design
Plan-Driven Analysis and Design
Requirements elicitation
Requirements analysis
Requirements specification
Requirements Prioritization
Agile Development

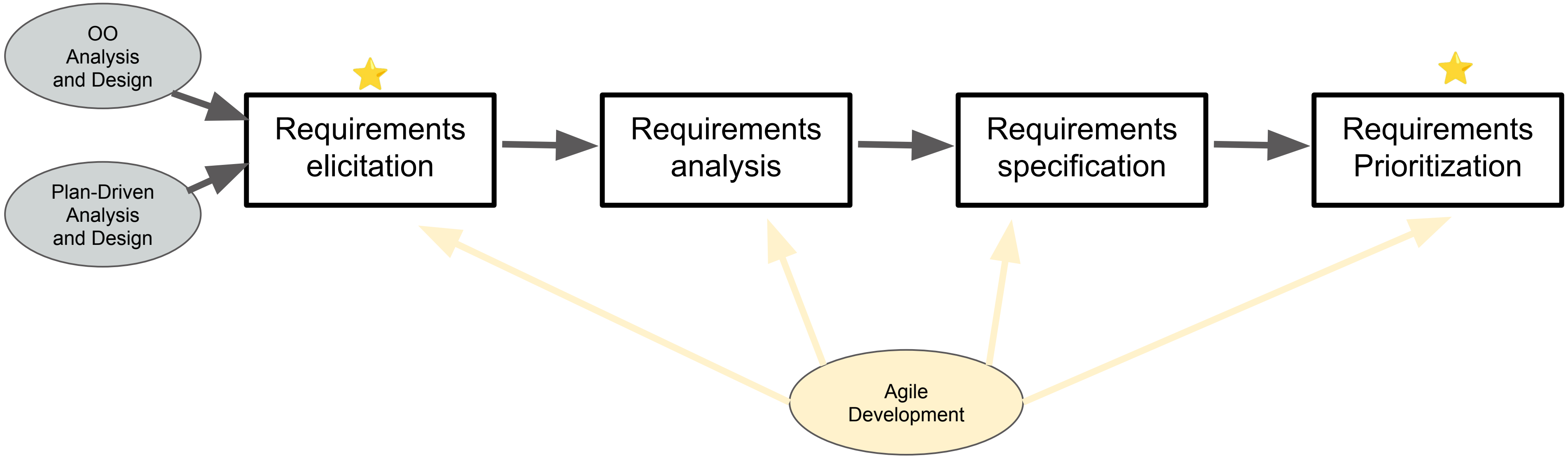
Requirements Engineering (cont.)

Another example



Feasibility study: can the identified be achieved using the current software and hardware technologies, under the current budget, etc. The feasibility study should be cheap and quick; it should inform the decision of whether or not to go ahead with the project.

Requirements Engineering (cont.)


OO Analysis and Design
Plan-Driven Analysis and Design
Requirements elicitation
Requirements analysis
Requirements specification
Requirements Prioritization
Agile Development

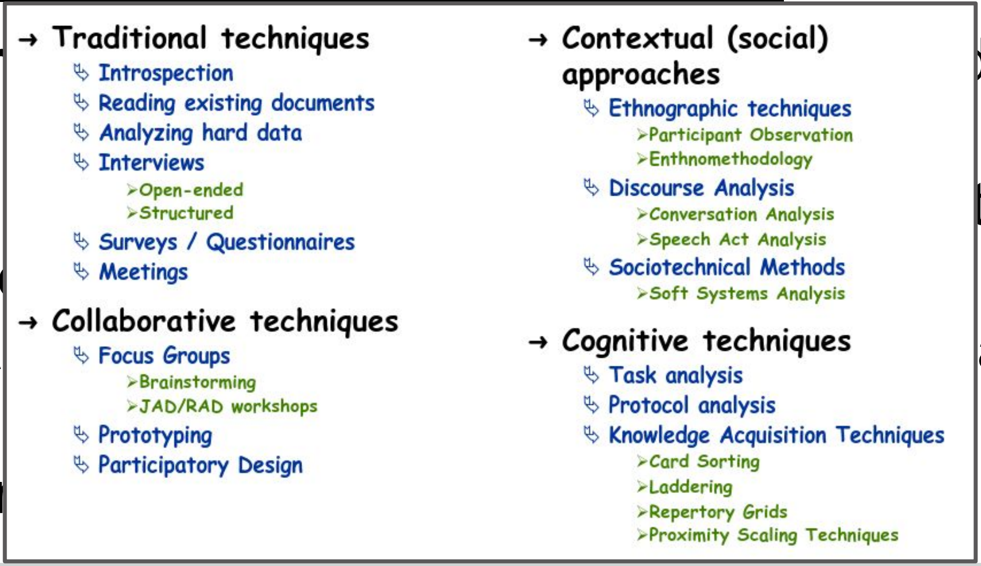
Requirements Elicitation

Discovering the requirements of a system.

aka Requirements gathering

Techniques for requirements elicitation:

- **Brainstor** deas and prune
- **Interview** th stakehold
- **Ethnogra** analyzes how peop
- **Strawma**



→ Traditional techniques
- Introspection
- Reading existing documents
- Analyzing hard data
- Interviews
  - Open-ended
  - Structured
- Surveys / Questionnaires
- Meetings

→ Collaborative techniques
- Focus Groups
  - Brainstorming
  - JAD/RAD workshops
- Prototyping
- Participatory Design

→ Contextual (social) approaches
- Ethnographic techniques
  - Participant Observation
  - Enthnomethodology
- Discourse Analysis
  - Conversation Analysis
  - Speech Act Analysis
- Sociotechnical Methods
  - Soft Systems Analysis

→ Cognitive techniques
- Task analysis
- Protocol analysis
- Knowledge Acquisition Techniques
  - Card Sorting
  - Laddering
  - Repertory Grids
  - Proximity Scaling Techniques

Elicitation Techniques (Group)

- Ex) Brainstorming, Focus Groups (interviews)
- Advantages
  - More natural interaction between people than formal interview
  - Can gauge reaction to stimulus materials (e.g. mock-ups, storyboards, etc)
- Disadvantages
  - May create unnatural groups (uncomfortable for participants)
  - Danger of Groupthink
  - May only provide superficial responses to technical questions
  - Requires a highly trained facilitator

Elicitation Techniques (Interviews)

- Structured or unstructured (no pre-set agenda)
- Advantages
  - Rich collection of information
  - Good for uncovering opinions, feelings, goals, as well as hard facts
  - Can probe in depth and adapt follow-up questions to what the person tells you
- Disadvantages
  - Large amount of qualitative data can be hard to analyze
  - Hard to compare different respondents
  - Interviewing is a difficult skill to master

Elicitation Techniques (Ethnography)

- Social world is ordered; observe in natural setting
- Advantages
  - Contextualized
  - Reveals details that other methods cannot
- Disadvantages
  - Extremely time consuming!
  - Resulting “rich picture” is hard to analyze
  - Cannot say much about the results of proposed change

Elicitation Techniques (Prototype)

- Aka Strawman, Prototype, Joint/Rapid Application Development
- Advantages
  - Contextualized
  - Visual aids
  - WYSIWYG approach
- Disadvantages
  - WYSIWYG approach
  - Depends on prototype quality
  - Still not ecologically valid

Requirements Elicitation Difficulties

- Thin spread of domain knowledge
  - It is rarely available in an explicit form (i.e. not written down)
  - …distributed across many sources
  - …with conflicts between knowledge from different sources
- Tacit knowledge (The “say-do” problem)
  - People find it hard to describe knowledge they regularly use
- Limited Observability
  - The problem owners might be too busy coping with the current system
  - Presence of an observer may change the problem
    - E.g. Probe Effect: humans change behavior while being probed
    - E.g. Hawthorne Effect: humans change behavior while being observed

Requirements Elicitation Difficulties

- Bias
    - People may not be free to tell you what you need to know
    - People may not want to tell you what you need to know
        - The outcome will affect them

Example Scenario

You are observing a development team to elicit requirements for Stand-up Bot:

- **Implicit knowledge:** There is no document in which the unwritten rules for scheduling stand-up meetings are written down (i.e., no meetings before 10am)
- **Conflicting information:** Different developers have different ideas about what the rules for stand-up meetings should be
  - Less experienced/new developers find stand-up meetings useful for learning
  - More experienced developers find them useless
- **Say-do problem:** The scrum process described in class/educational settings may be quite different from what development teams actually do
- **Probe effect:** The development team scrum meeting while you are observing is different from the one they normally use
- **Bias:** The scrum master fears that you will computerize their job out of existence, so they deliberately emphasize the meeting scheduling to convince you it has to be done by a human!

TODO: Complete a stand-up meeting!

- **What I did.**
- **What I need to do next.**
- **What is blocking me.**

Analysis vs. Specification

- **Analysis:** process of *understanding* the problem and the requirements for a solution
- **Specification:** process of *describing* what a system will do

➔ *Analysis leads to Specification* – they are not the same!

Requirements Analysis is Hard

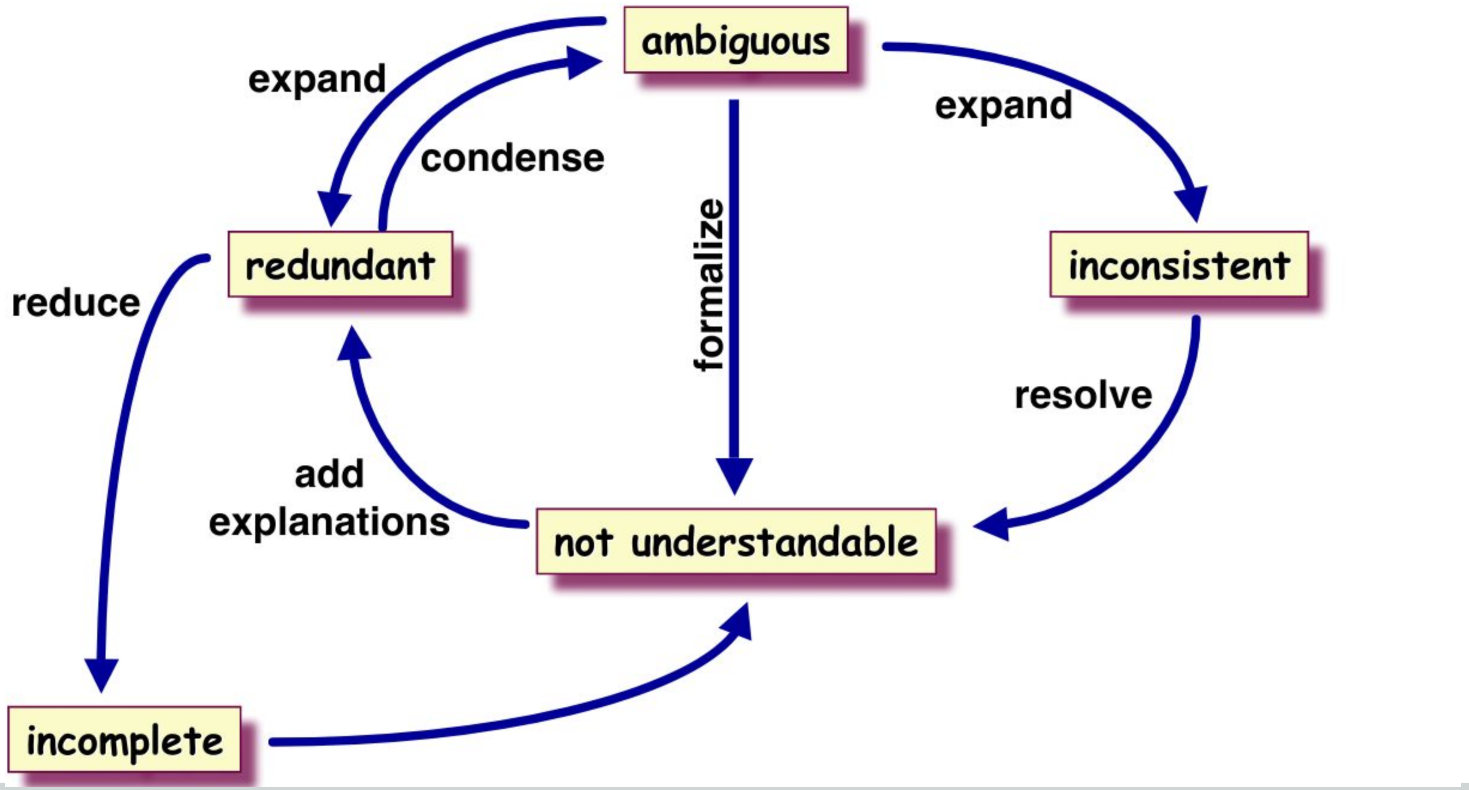
- Major causes of project failures
  - Lack of user input
  - Incomplete requirements
  - Changing requirements
- **Essential requirements analysis solutions**
  - Classification of requirements
  - Iterative requirements analysis
  - Use Cases

Requirements Specification

- Describing what the system will do

“Creating a business, product or a piece of software, is a complicated and long winded process…The software requirements specification, very often, is the developer’s bible, for guidance and direction.”

Requirements Specification is Hard



Requirements Prioritization

Why is prioritization important?

- Need to select what to implement
  - Customers (usually) ask for way too much
  - Balance time-to-market with amount of functionality
  - Decide which features go into the next release
- For each requirement/feature, ask:
  - How important is this to the customer?
  - How much will it cost to implement?
  - How risky will it be to attempt to build it?

Triage

- Derived from medical usage: prioritizing patient care based on urgency and severity of wounds, illness, etc.
- Perform Triage
  - Some requirements *must* be included
  - Some requirements should definitely be excluded
  - That leaves a pool of “nice-to-haves”, which we must select from

Unified Modeling Language

- Definition: A visual language for specifying, constructing, and documenting the artifacts of systems
  – Standard diagramming notation for drawing pictures related to software
  – Includes 13 types of diagrams

UML Categories

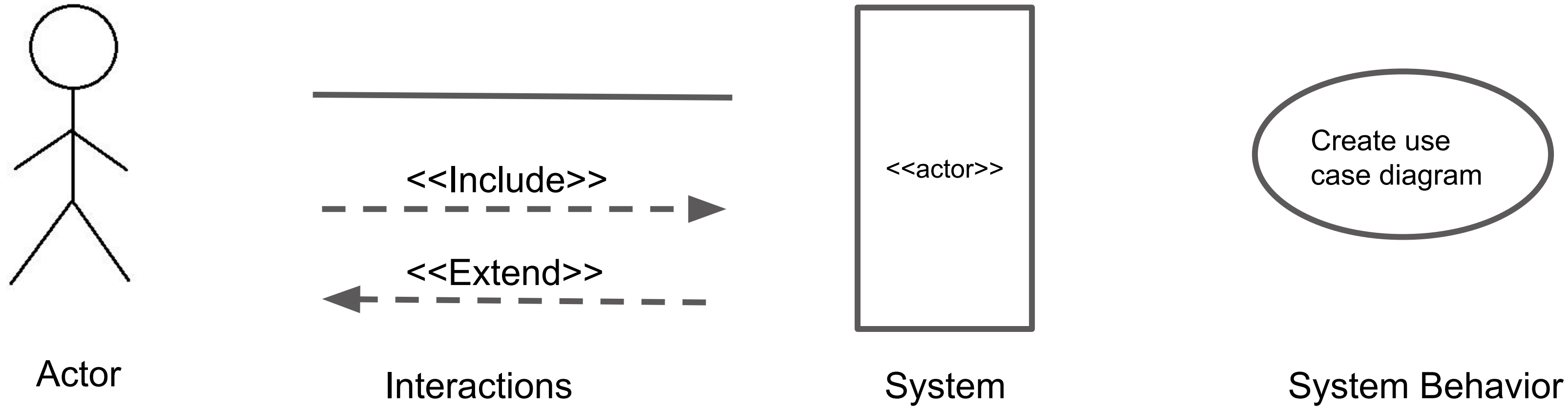
- **Structural UML diagrams**
  - Specifies the structure of the objects, classes or components
    - **<u>Class diagram</u>**
    - Object diagram
    - Package diagram

    ...
- **Dynamic UML diagrams**
  - Represent object interactions at runtime
    - **<u>Use case diagram</u>**
    - **<u>Sequence diagram</u>**

    ...

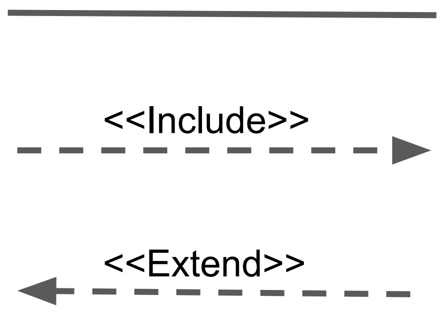
UML Use Case Diagram

- Graphical depiction of use cases
  - Defines *actors'* interactions with the *system*



Use Case Diagrams

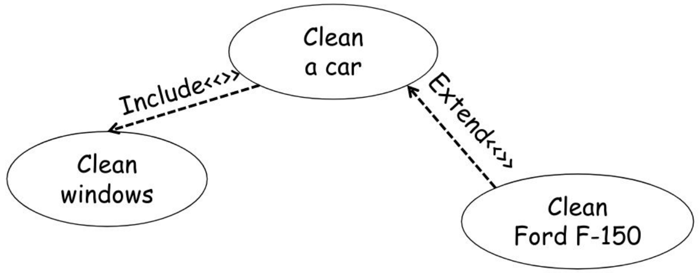
Interactions



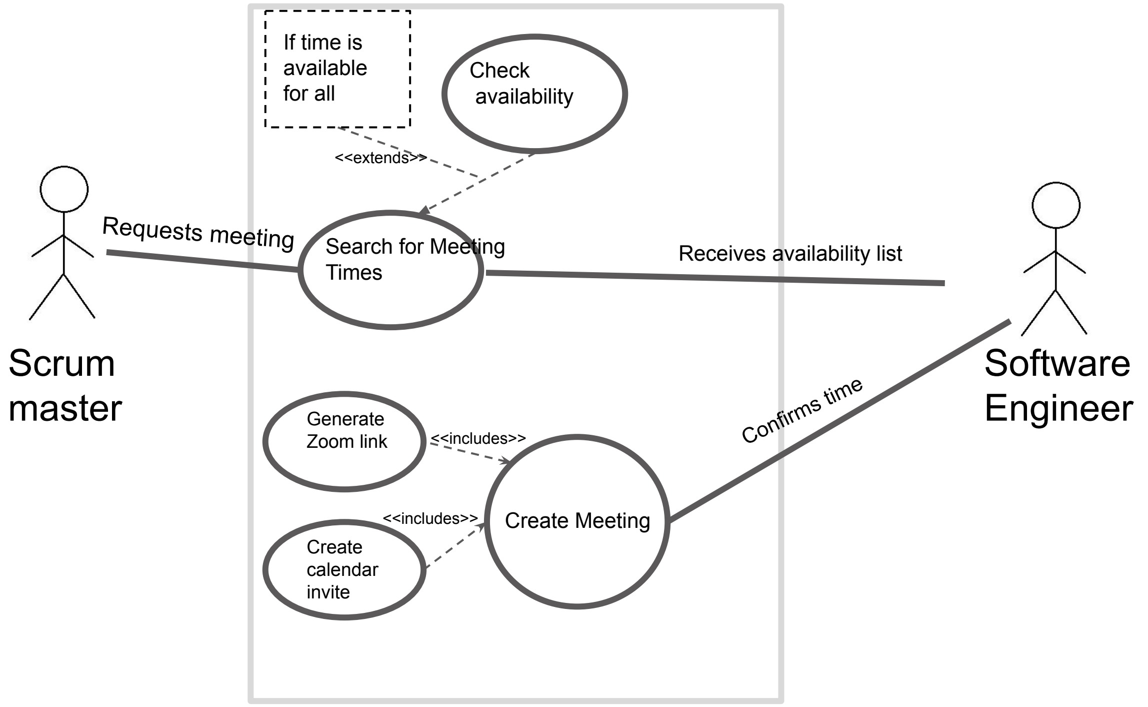
Association: relation between an actor and a use case

Includes dependency: a base use case includes a sub use case as component

Extends dependency: a use case extends the behavior of a base use case.

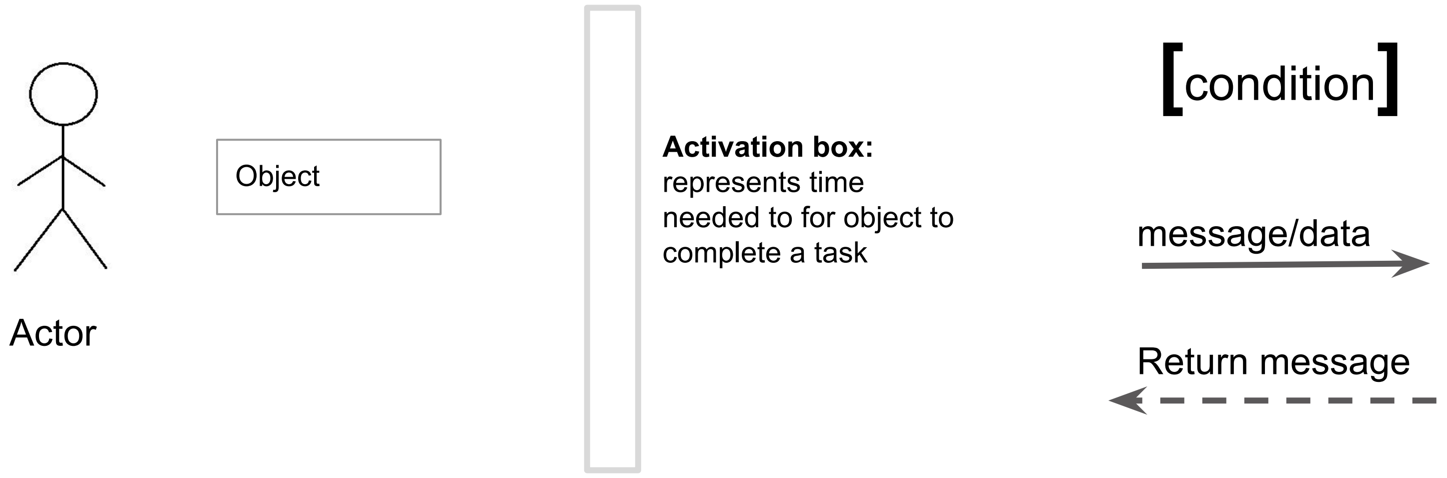


Example: Stand-Up Bot

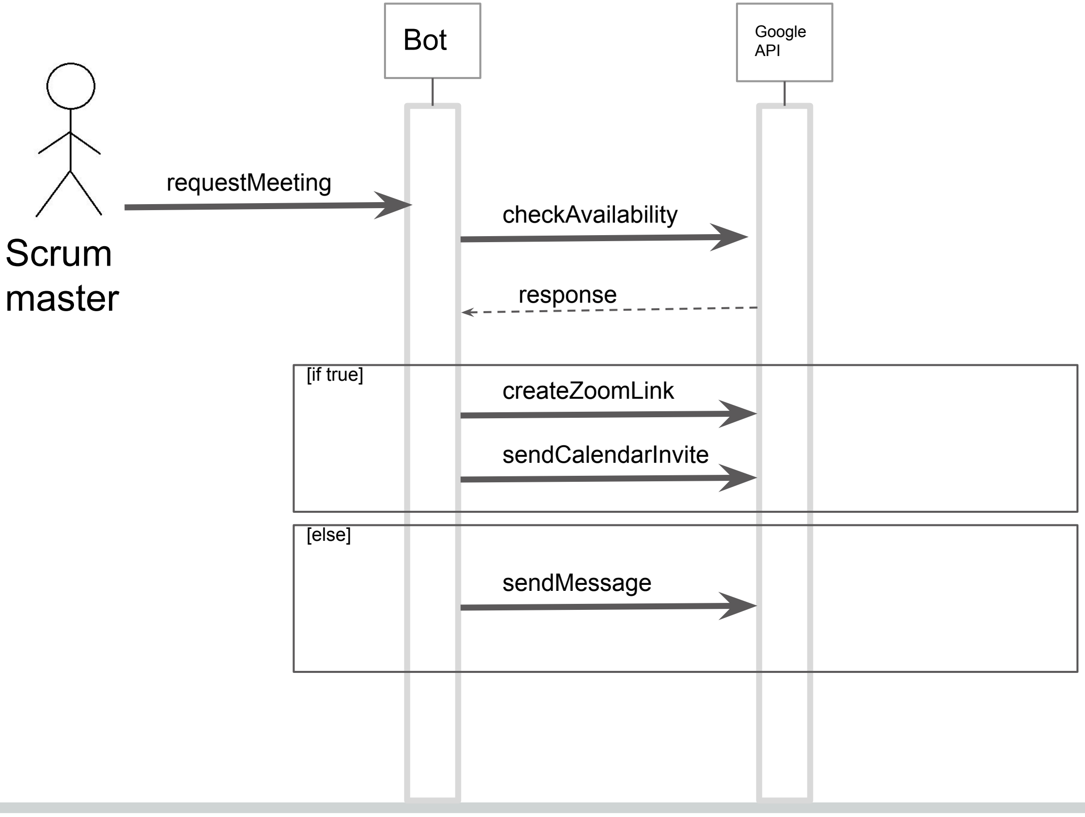


UML Sequence Diagram

- Graphical Depiction


Actor
Object
Activation box: represents time needed to for object to complete a task
[condition]
message/data
Return message

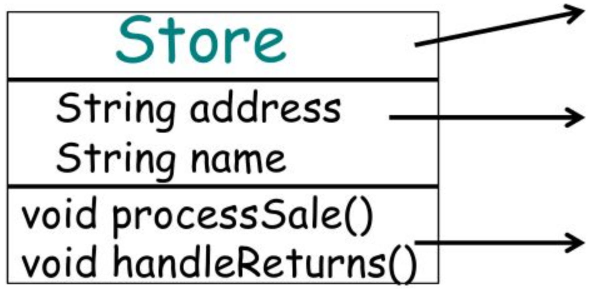
Example: Stand-Up Bot



UML Class Diagram

- A visual representation of main objects and their relations for a system.
- Elements
    - Classes containing: Attributes, Operations
    - Various relationships: Association, Aggregation, Composition, Generalization

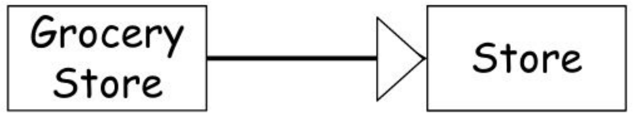
Class Diagram Example



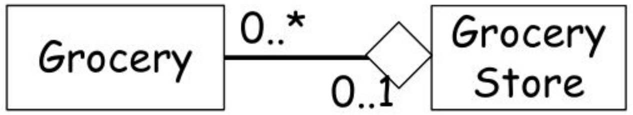
Class name: abstract concept

Attributes: properties relevant to the use case

Operation (Method signatures): behaviors of the class



Generalization: "is-a" relationship. A sub-class inherits all attributes and operations of its super class.



Aggregation: "has-a" relationship. The container and elements can exist independently from each other

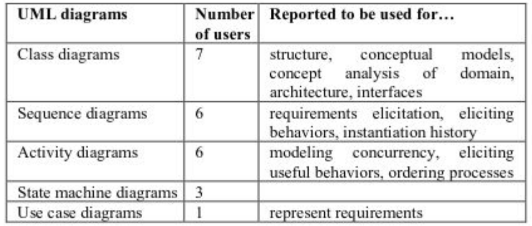
How is UML Really Used?

“UML has been described by some as ‘the lingua franca of software engineering’. Evidence from industry does not necessarily support such endorsements. How exactly is UML being used in industry – if it is? This paper presents a corpus of interviews with 50 professional software engineers in 50 companies and identifies 5 patterns of UML use.” [Petre]

NONE!	**70%**
SELECTIVE	22%
AUTOMATIC CODE GEN	6%
RETROFIT	2%
WHOLE	0%

Of those that reported using it…

TABLE II. ELEMENTS OF UML USED BY THE 11 ‘SELECTIVE’ USERS.



UML diagrams	Number of users	Reported to be used for…
Class diagrams	7	structure, conceptual models, concept analysis of domain, architecture, interfaces
Sequence diagrams	6	requirements elicitation, eliciting behaviors, instantiation history
Activity diagrams	6	modeling concurrency, eliciting useful behaviors, ordering processes
State machine diagrams	3	
Use case diagrams	1	represent requirements

Discuss: What is your experience with UML diagrams? Have you had to use them outside of classes? (and in HW2)

How is UML Really Used? (cont.)

1. No UML (35)
2. Retrofit (1)
   - Don’t really use UML, but retrofit UML in order to satisfy management or comply with customer requirements
3. Automated code generation (3)
   - Not used for design/requirements, but to generate code automatically
4. Selective (11)
   - UML is used in design in a personal, selective, and informal way, for as long as it is considered useful
5. Organizational approach to UML
   - ***0***

How is UML Really Used? (cont.)

Why not?

- *Lack of context:* “UML deals primarily with software architecture rather than the ‘whole’ system, and hence that it lacks context.”
- *Overhead of understanding notation:* “UML is considered *unnecessarily complex*. Issues of comprehension included both software developers and stakeholders...variations in understanding and interpretation among developers that led to problems…”
- *Issues of synchronization/consistency:* “issues of synchronization or consistency…between different views/diagrams, and in terms of maintaining the coordination between the model and the implementation.”

Why Learn UML?

“UML is to the modeling we do every
day as Latin is to the language we use every day.”

“The context and meaning in which I made this statement was: Latin is not a practical language for communication today. Nevertheless, we learn Latin in school still today because it has a very large set of grammar features (con jugations, declinations, genders, tenses, . . .), such that the grammars of almost all (European) languages in use today are subsets of Latin grammar and can be more easily understood by the students using Latin as “grammar reference architecture.” In that same sense, UML as “modeling reference architecture” contains all possible diagrams you could ever need, but it does not necessarily mean that you would or should use all of them in a given practical situation.”

Remember: Agile Modeling [8/30]

- Models are inaccurate
    - ***Only tested code demonstrates true design!***
- Diagrams/models are typically thrown away, if used at all.
    - Used to facilitate creativity
- Modeling should focus on the smaller percentage of unusual, difficult, and tricky parts of the design
    - Developers should do design modeling in pairs (or triads) and in parallel
    - Use simple tools and notation

Requirements in Practice

How do modern companies analyze, specify, model, and engineer requirements now?

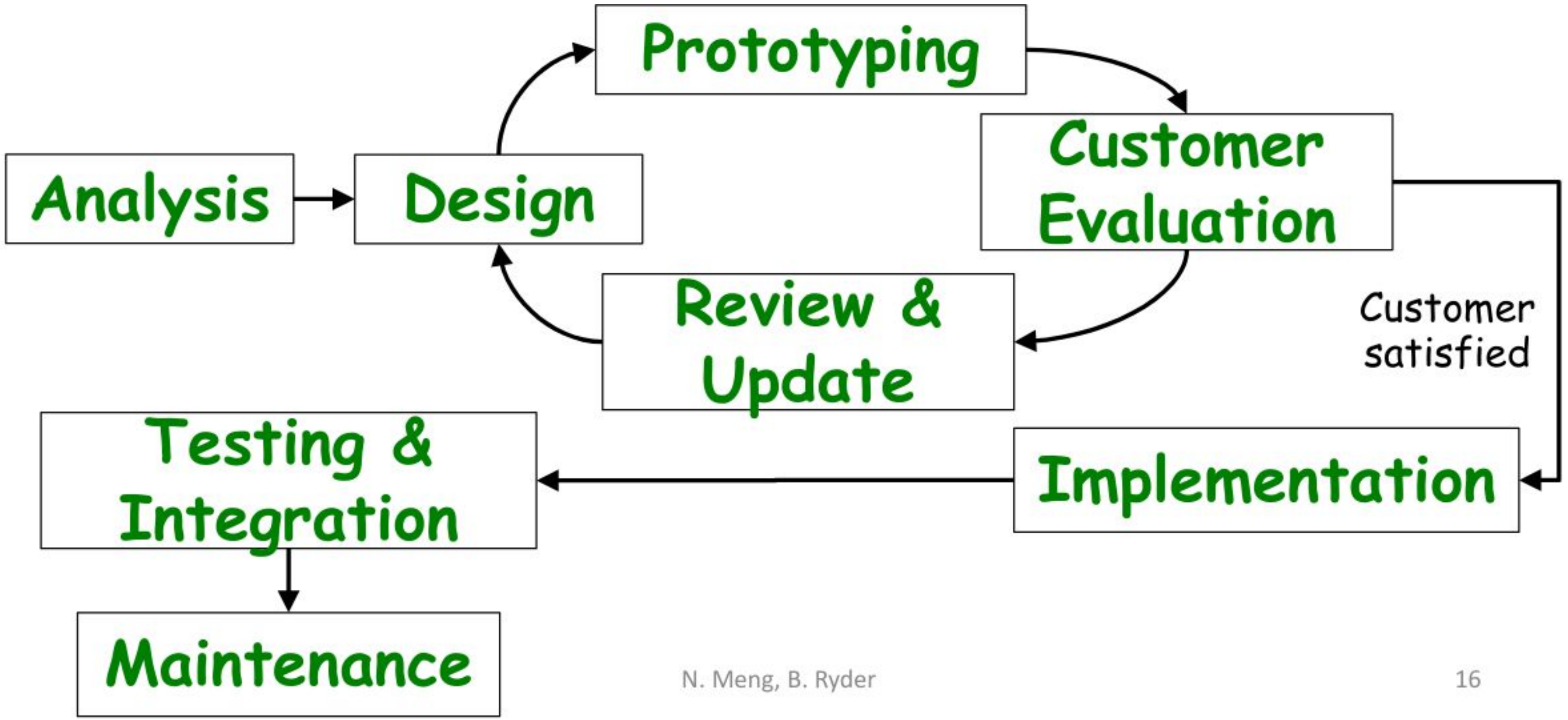
1. They don’t
2. Requirements/Business analysts
3. Prototypes
4. Less formal modeling (flowcharts, BPMN)
5. Gantt charts
6. Gap analysis
7. Automated techniques*

Requirements Analysts

What do requirements analysts do?

- Provide a starting point
  - Some notion that there is a “problem” that needs solving
    - e.g. dissatisfaction with the current state of affairs
    - e.g. a new business opportunity
    - e.g. a potential saving of cost, time, resource usage, etc.
  - A Requirements Analyst is an agent of change
- The requirements analyst must:
  - identify the “problem”/”opportunity”
    - Which problem needs to be solved? (identify problem Boundaries)
    - Where is the problem? (understand the Context/Problem Domain)
    - Whose problem is it? (identify Stakeholders)
    - Why does it need solving? (identify the stakeholders’ Goals)
    - How might a software system help? (collect some Scenarios)
    - When does it need solving? (identify Development Constraints)
    - What might prevent us solving it? (identify Feasibility and Risk)
  - and become an “expert” in the problem domain

Prototyping

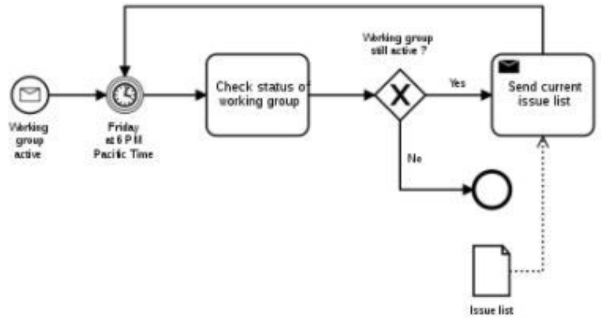


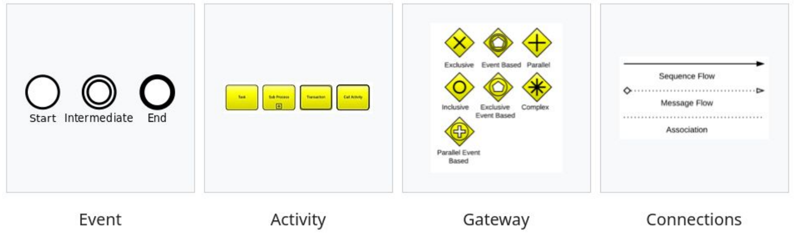
Modeling: Flowcharts

- Flowcharts depict sequential flow and control logic of a related set of activities. They are useful for both technical and non-technical members.
    - Visual representation
    - Without the syntax, complexity of UML!

Modeling: Business Process Model Notation

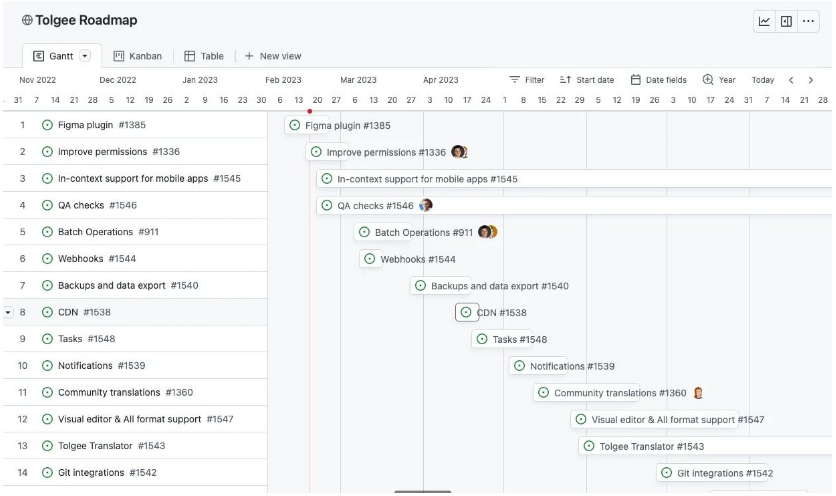
- Used to create graphs that simplify understanding and coordinate sequences between different participants in events in a set of related activities.


Working group active
Friday at 6 PM Pacific Time
Check status of working group
Working group still active ?
Yes
Send current issue list
No
Issue list



Gantt Chart (aka Roadmap)

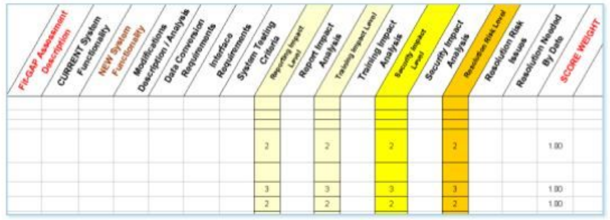
- Provide a visual representation of requirements along with scheduled timelines to help visualize the start and end dates of tasks.
    - GitHub projects



1	Figma plugin #1385
2	Improve permissions #1336
3	In-context support for mobile apps #1545
4	QA checks #1546
5	Batch Operations #911
6	Webhooks #1544
7	Backups and data export #1540
8	CDN #1538
9	Tasks #1548
10	Notifications #1539
11	Community translations #1360
12	Visual editor & All format support #1547
13	Tolgee Translator #1543
14	Git integrations #1542

Gap Analysis

- Evaluate the gaps in a product performance to determine whether a requirement is met or not.
    - Usually after a working product is available
    - Used by business analysts to present current state and target state of product



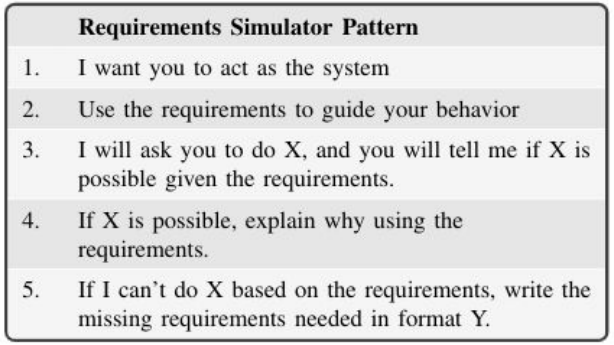
FIT-GAP Assessment Description	CURRENT System Functionality	NEW System Functionality	Modifications Description / Analysis	Data Conversion Requirements	Interface Requirements	System Testing Criteria	Reporting Impact Level	Report Impact Analysis	Training Impact Level	Training Impact Analysis	Security Impact Level	Security Impact Analysis	Resolution Risk Level	Resolution Risk Issues	Resolution Needed By Date	SCORE WEIGHT	
							2		2		2		2			1.00	
							3		3		3		3			1.00	
							2		2		2		2			1.00	

Automation

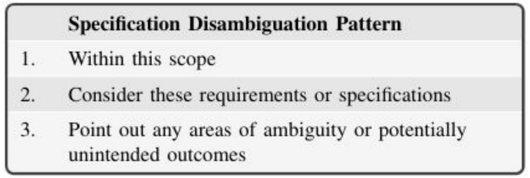
- Requirements elicitation, analysis, specification, etc. are time-consuming and error-prone.
- Researchers have explored various techniques (i.e. static analysis, natural language processing, etc.) to streamline this process, but in general automated requirements activities and documentation is largely unexplored (needs human input).
  - Ex) Static requirements analysis tools (i.e. ScopeMaster) can help identify potential problems so they can be resolved quickly and efficiently.
- But still a way to go before these techniques can be adopted mainstream by software development teams for requirements activities.

Automation (cont.)

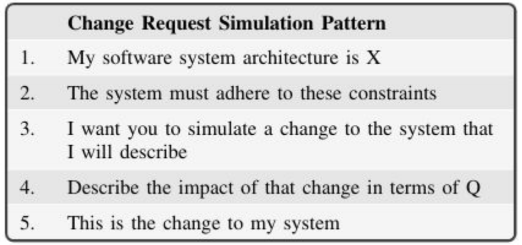
Example: **ChatGPT**



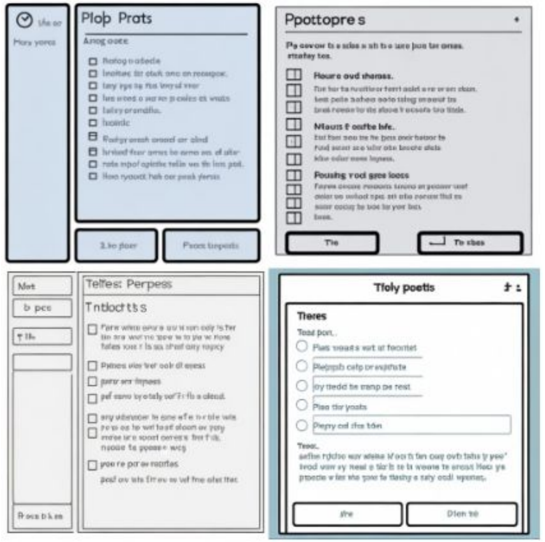
	Requirements Simulator Pattern
1.	I want you to act as the system
2.	Use the requirements to guide your behavior
3.	I will ask you to do X, and you will tell me if X is possible given the requirements.
4.	If X is possible, explain why using the requirements.
5.	If I can't do X based on the requirements, write the missing requirements needed in format Y.



	Specification Disambiguation Pattern
1.	Within this scope
2.	Consider these requirements or specifications
3.	Point out any areas of ambiguity or potentially unintended outcomes



	Change Request Simulation Pattern
1.	My software system architecture is X
2.	The system must adhere to these constraints
3.	I want you to simulate a change to the system that I will describe
4.	Describe the impact of that change in terms of Q
5.	This is the change to my system



"Significant human involvement and expertise is currently necessary to leverage LLMs effectively for automating common software engineering tasks…output requires close scrutiny from human users at this point…to ensure output of LLMs is accurate and helpful in practice." [White, 2023]

Attendance Quiz

https://forms.gle/oSeSec8zeJnkbEKf8

Next Class

- **Project Workday (9/22) in class**
- **PM1 due Friday (9/22) at 11:59pm**
  - Lightning Talk slides
  - Project proposal
- **Lightning talks next week!**

References

- Na Meng, Barbara Ryder. Virginia Tech
- Sarah Heckman, Chris Parnin. NC State
- Steve Easterbrook. University of Toronto
- https://www.simplilearn.com/what-is-requirement-analysis-article
- “ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design”. Jules White, et al., 2023
- “UML in Practice”, Marian Petre. 2013
- “‘No shit” or “Oh, shit!’: responses to observations on the use of UML in professional practice”, Marian Petre. 2014