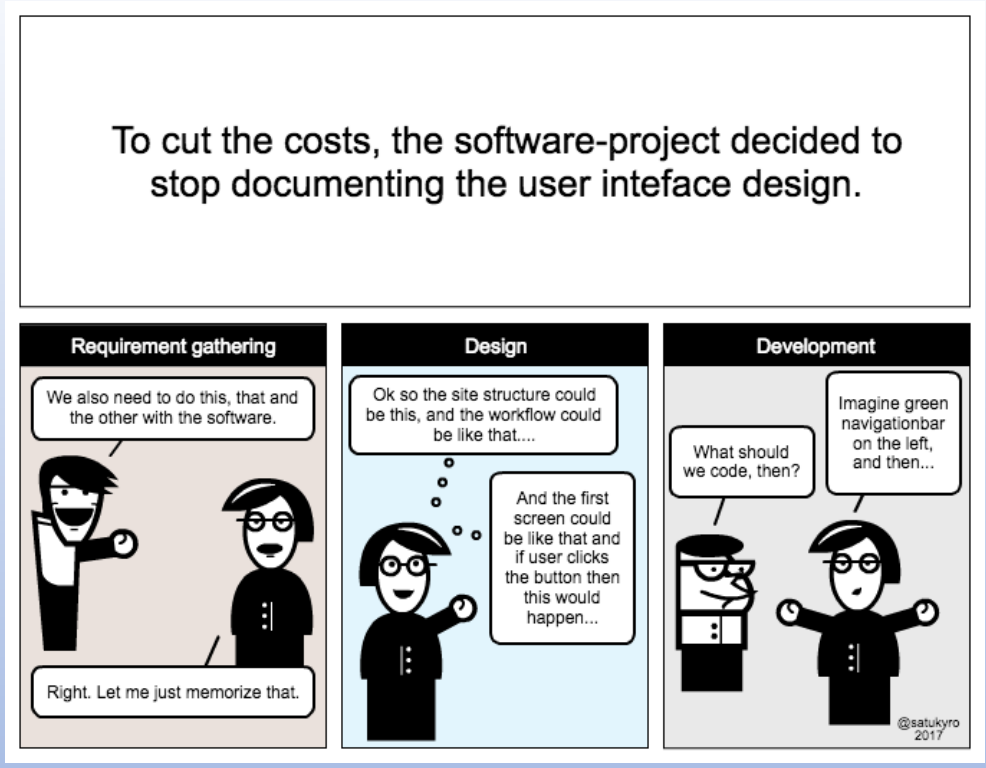
Specifications



CSCI2340: (Graduate) Software Engineering

Steven P. Reiss

Thanks for your flexibility

I will entertain project change requests

If you drop the course and have a project, please tell me.

If you weren't assigned a project, please tell me.

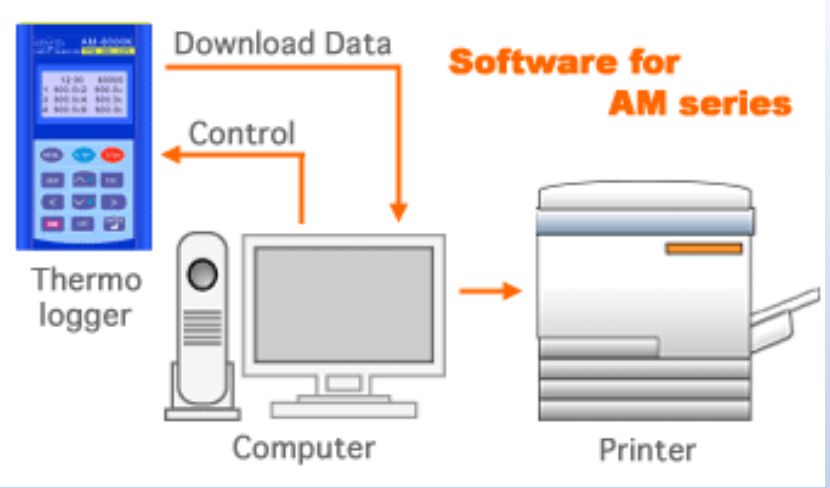
ANY QUESTIONS???

Goals for Specifications

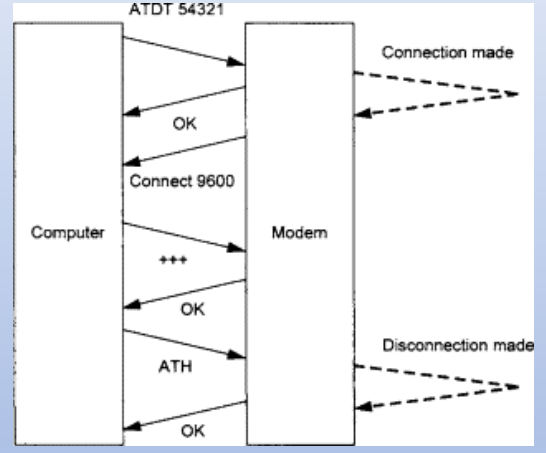
- Outline the solution
  - Requirements define the problem
  - Specifications define the solution
    - What needs to be done to solve the problem
- Provide a basis for better requirements
- Provide a basis for designing the system
  - You must know what you are building to build it
- Provide a basis for understanding what the costs
  - Team size, work & cost estimation, time estimation, …
- Provide a basis for understanding if the solution works
  - Will it meet the users' needs
  - Will it solve the problem for the users
- Provide a basis for prioritization

Specifications

- Detail **what** the application will do
  - From the programmer's point of view
  - Concentrate on commands, inputs, outputs
    - What commands are needed
  - Not **how** the commands are implemented
- Define the data the application will use
  - What information is needed
  - What information is used
  - Where does this information come from
  - Where does this information go
  - How the user interacts with the system
  - How the system interacts with other systems

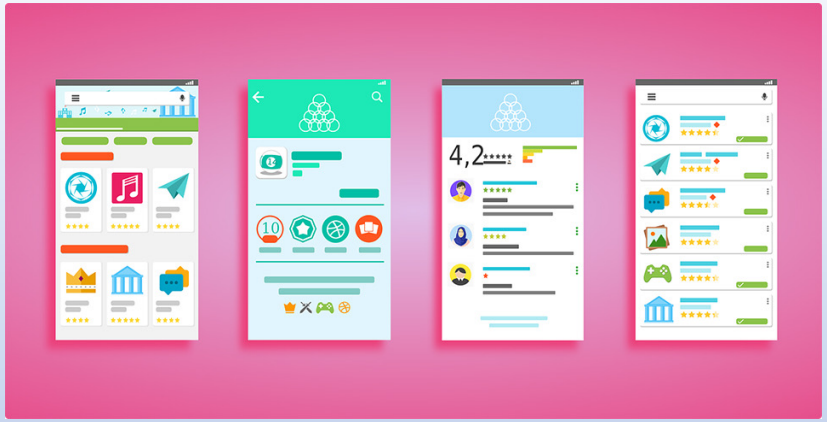


Define the Actions or Commands



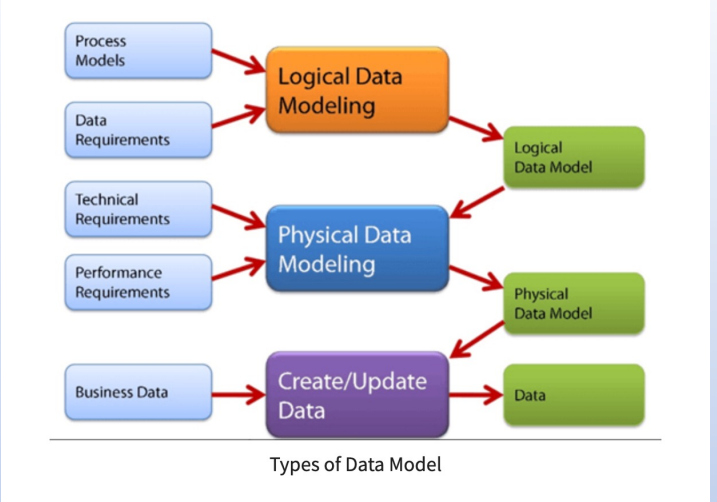
- Define each command
  - What are the inputs (what data is needed)
    - What data is provided by the user
    - What data is provided by the system
  - What are their outputs (what data is produced)
  - What is the mapping from input to output (what do they do)
- Commands should cover the requirements
  - Actions required by the system in the stories or use cases
  - Actions required by specific requirements
  - Have a complete (and extensible) set of commands

User Interface Specifications



- Commands can be invoked in many ways
  - Based on user requirements
  - Alternatives
    - Commands, drag and drop, menus, typing, external events, …
  - Exactly how is UI design; specifications can provide hints
- Commands can produce output in many ways
  - Based on user requirements
  - Alternatives
    - Text files, graphics, CSV, pdf, html pages, searchable lists, text output, …
  - Exactly how it is presented is UI design
    - Specifications can provide hints

Define the Underlying Data

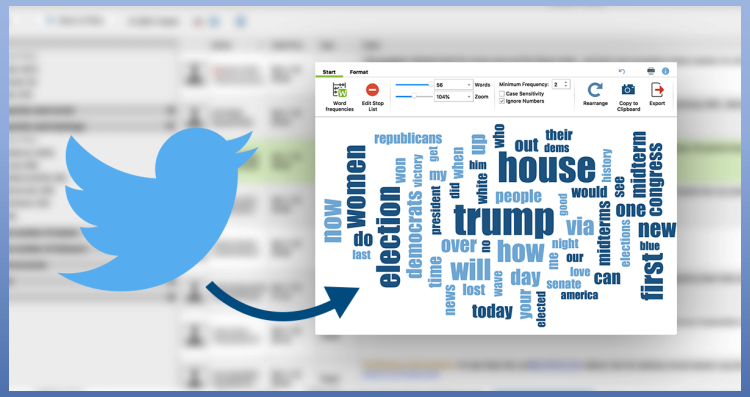


Types of Data Model

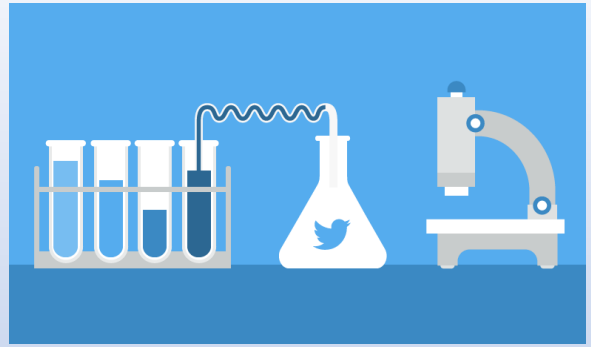
- What data does the system need to operate
  - Where does this data come from
  - How are you going to obtain this data
  - How are you going to store this data
  - Are there legal or other restrictions on this data
  - *How are you going to bootstrap this data*
  - *How are you going to keep the data up-to-date*
- What data does the system produce
  - Is the data to be used by other systems
  - Where does the data go (permanent, ephemeral, …)

Example: Twitter Data

- Problem:
  - I have about 3 billion geolocated US tweets over 10 years
  - Want to let user query these and see results
    - Motivation: track election, flu, covid, affluence, discrimination, …
    - Query by keywords (and, or), dates
    - Display results on a map, using color and other highlighting
    - Dynamic display of results over time
    - Output the results for further analysis
  - Should have a web interface
  - Should be easy to use (for cs0931)



Twitter Data



- Data was collected from twitter stream
  - Processed, cleaned and stored to be accessible
    - Missing a bit due to quirks in the feed (could be redone)
  - Twitter shut down the collection mechanism last summer
    - geo-located queries no longer work; no longer free
- Data Specifications:
  - What was gathered from the stream (CSV format with X fields)
  - How it is stored initially (CSV files with ~1M entries) (6000)
  - How it is updated (separate program)
  - Data is somewhat clean before it is used by the program
    - Duplicates eliminated; US only; Lat & long; zip; district; state; …

Twitter Commands

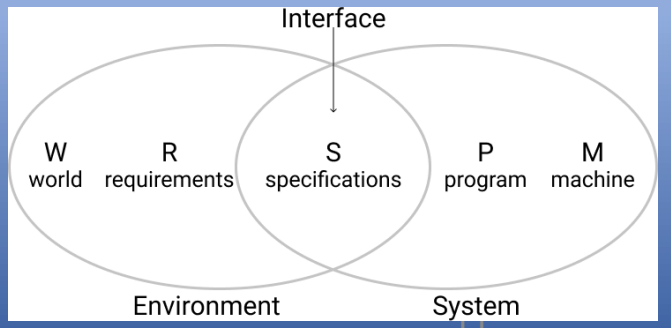
- Search
  - Inputs: time range, keywords, logic
  - Outputs: list of tweets
- Refine Search
  - Inputs: list of tweets, time range, keywords, negative words
  - Outputs: refined list of tweets
- Display
  - Inputs: list of tweets, type of region, map, range transformations
  - Outputs: interactive web page
- Restrict Display
  - Inputs: display, time range, change transformations
  - Outputs: updated web page
- Load Data from raw twitter CSV files
- Export Data to CSV or other formats

Specifications Are Not Designs

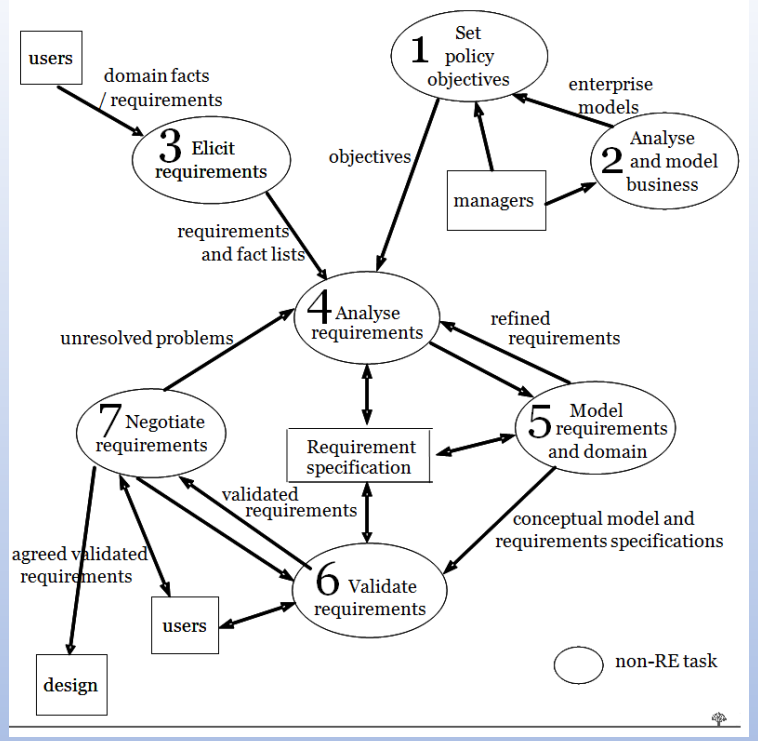
- WHAT not HOW
- Do **not** identify specific technologies to use
  - Unless mandated by outside requirements (e.g., CSV files)
- Do **not** determine how or where tasks are done
  - Client, server, cloud, browser …
  - Languages or technologies to be used (unless required)
  - Algorithms or processing (unless required)
- Do **not** provide detailed user interface designs
- You do not want to overcommit at this point
  - Understand the full specs before design
  - Specifications will change as requirements change

Specifications versus Requirements

- Fine line between the two
  - Requirements are from the user point of view (problem)
  - Specifications are from the programmer's point of view (solution)
  - But both describe the problem and how it may be solved
- Additional information in the specifications
  - User and external interfaces
  - Commands with their inputs and outputs
  - Data used and generated by the system



Requirements Traceability



- Motivation
  - Want to understand WHY for each command
  - Importance or priority of each command
  - Resolve conflicts between specifications
  - Understand what needs to change
    - As requirements change
- Relate specifications to requirements
  - Each specification should be based on 1 or more requirements
  - Should be able to go both ways

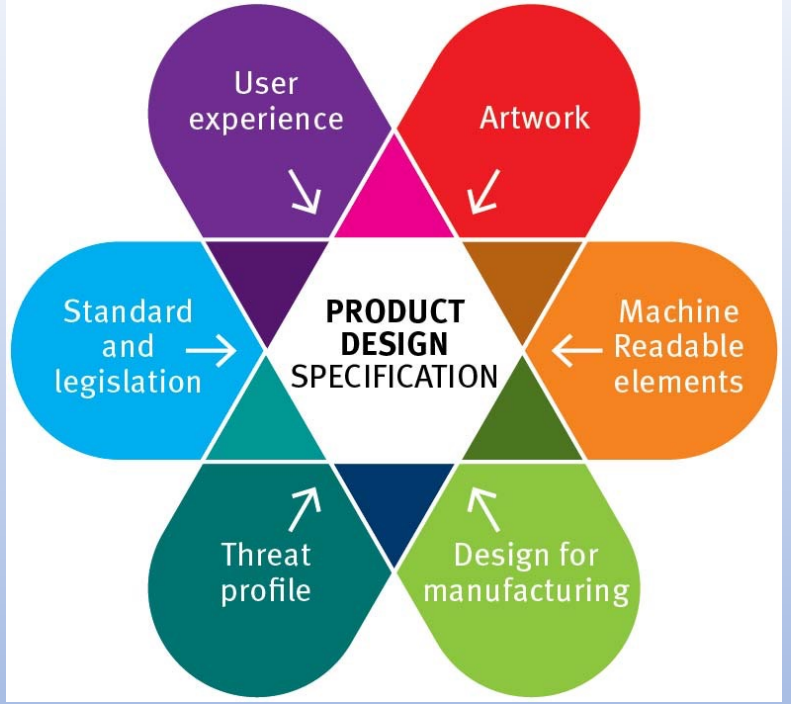
Requirements Specifications

- These are often viewed as a single phase
  - Talking with both users and the developers
  - Agreeing on a problem and a potential solution
  - Automatic traceability
- Sometimes the results are merged
  - **Requirements Specifications** Document
  - Tends to lose the user influence and problem understanding
    - Specifications emphasized; requirements deemphasized
  - But it is easier to maintain one set of documents rather than two
- Do requirements first
  - Then merge the specifications into the requirements document
  - Have an on-going dialog with the stakeholders

Representing Specifications

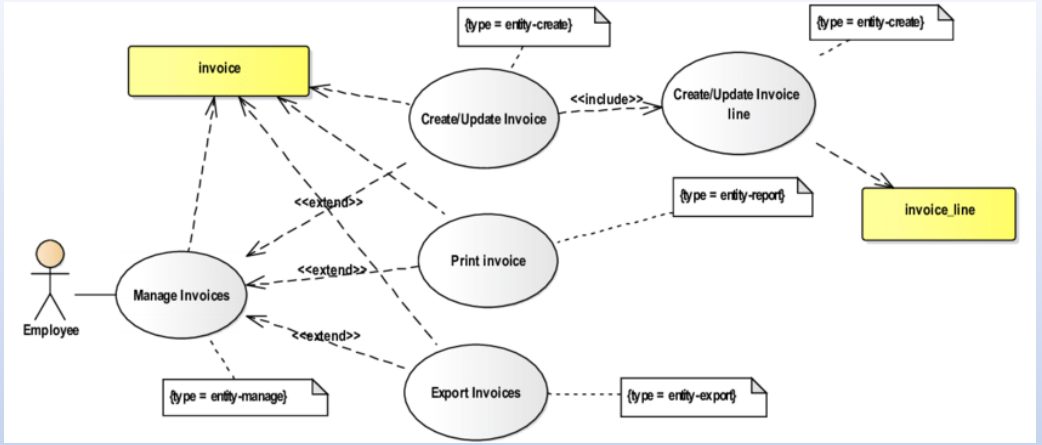
- Natural language
  - Akin to requirements document
- Organized outline
  - Organized by topic, component, purpose
  - Numbered points (for reference)
- Links to corresponding requirements
- Description of the what it does
  - Commands or Actions
  - Effects on outside entities
  - How the user might invoke them

Representing Specifications



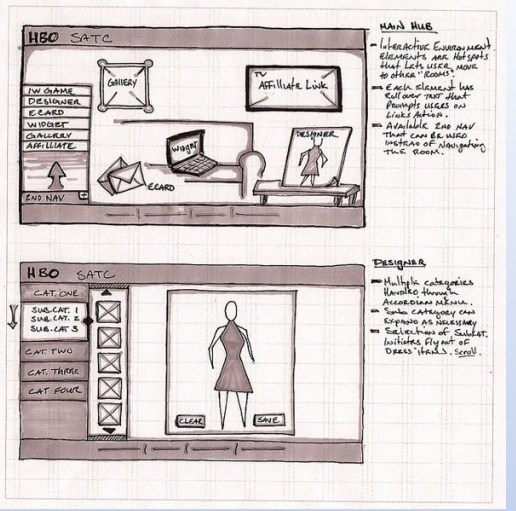
- Augment the requirements document
  - With specifications (what) information
  - Commands, inputs, outputs
- This is the simplest approach
  - Provides links to requirements
  - Provides a concrete understanding of the requirements
  - Ensures that all requirements are considered
  - Offers a basis for getting user feedback
- Augment with ideas for the user interface

Specification Use Cases



- Like requirements use cases
  - But detailing system behavior
    - Explicit commands, inputs, outputs
  - What the user does to interact with the system
  - What the system does in response to the user
  - What the user interface might look like
- Create a user interface story board for each use case/story
  - Sequence of sketches showing the application
  - Showing the sequence of interactions from the use case
  - Annotations on what actions are taken from one to the next
- This can be done as extensions of the original use cases

User Interface Sketches



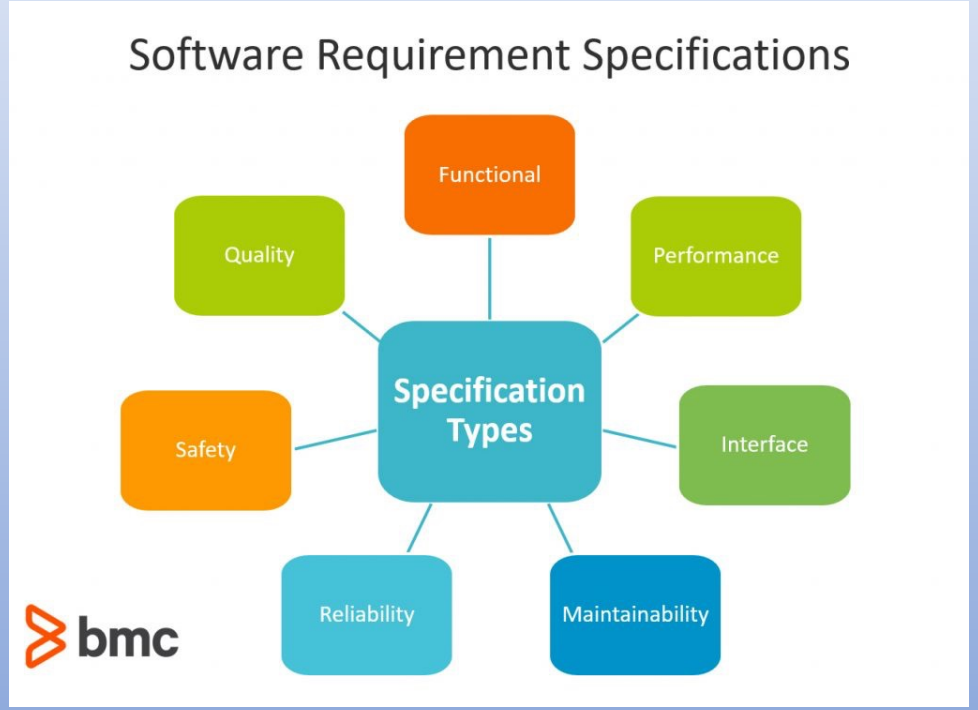
- Very quick sketch of what the interface might look like
  - Not a user interface design
  - More defining a set of components (menus, dialogs, drawings, …)
- Multiple sketches or pop-outs from a sketch
  - To show interactions
  - Think of creating a story board or story book
- These are not a design
  - Not an image of the actual interface
  - Show feasibility
  - Just show the commands and how they might be implemented
  - Give a basis for doing a full UI design

Specifications and Users

- Specifications provide feedback to users
  - Can ask would this solve your problem
  - Can ask if the interactions, commands are sufficient
  - Can ask if the interactions, commands are appropriate
- Users will give you more concrete requirements
  - Once they understand what you are thinking about
  - Once they understand what can and can't be done
  - Once they understand what is easy and what is difficult
- Don't feel committed to your initial specifications
  - Take user feedback into account
  - Specifications are going to change

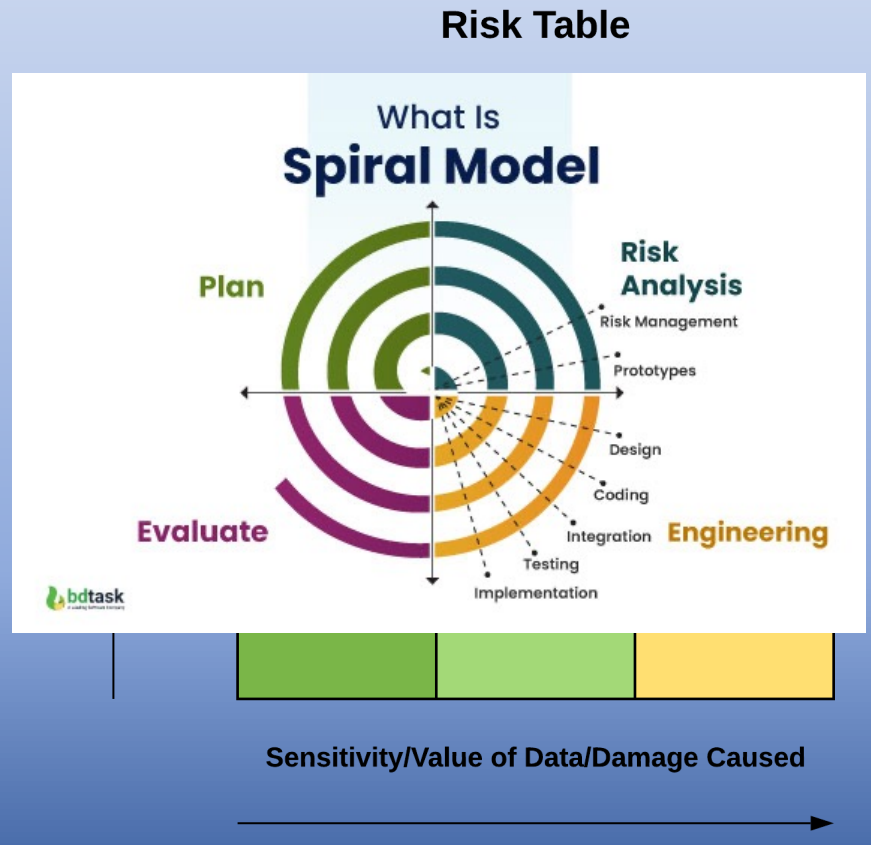
Non-Functional Specifications

- Non-functional Requirements
- Non-functional Specifications
  - Performance
  - Platforms
  - Security
  - Privacy
  - Accessibility
- External systems to interact with
- Existing systems to build on



Risk Analysis

- Understanding the risks in the project is part of specifications
- Do a risk analysis
  - What can go wrong
  - What is going to be difficult
  - What do you not understand
  - How well understood is the user interface
  - How well understood are the users
  - How well understood are the algorithms …
- This is information you will need for design
  - Recall the spiral model of development
- Include the risks in the specifications
  - Enumerate and categorize them



Maintaining Specifications

- Specifications are going to change
  - Requirements change
  - Needs change
  - Users change
  - Competition changes
  - Design & implementation provide opportunities and obstacles
- You will need to know the current specifications
  - To understand where you are, to direct design and coding
  - To prioritize development
  - To trace decisions back to requirements
- Specifications are not throwaway
  - You want to maintain the requirements-specifications thru development

Research in Software Specifications

- Using specifications to detect API misuse
- Formal specifications (e.g., temporal logic)
- Using specifications to plan security mechanisms
- Generating specifications from UI tests
- Automatic documentation generation

Homework / Further Reading

- Develop specifications for the programming assignment
  - Feel free to discuss and exchange requirements
  - Canvas assignment (due 9/17)
- Further Reading
  - Text book, chapter 4
  - IBM Rational Description of Use-Case Diagrams
  - IEEE Software Requirements Specifications

PROJECT

- Initial team meeting
  - We will break into project teams
- Agenda
  - Introduce yourselves
  - Set up a weekly time to meet
  - Start discussing requirements & specifications
    - Start getting ideas for what specifically your system will do
    - Create a shared requirements document if you haven't done so
    - Using the shared requirement specifications document
      - Can separate or merge the two
    - Determine what stakeholders to talk to (and assign people to do so)
  - I'll road around, or you can just ask for me stop by