CS1530, LECTURE 7

REQUIREMENTS ENGINEERING

WHAT ARE REQUIREMENTS?

- "A set of statements that describe the user's needs and desires"
- A set of statements which indicate **what** the program is supposed to
- Avoid "how" to do it (this belongs in design constraints, which can be part of a requirements specification, but are not requirements *per se*)
- Note that software may be considered complete without meeting all of the requirements. This is not an ideal goal but something that happens on a regular basis.
- Also note that design constraints are often elicited as part of requirements engineering

EXAMPLE REQUIREMENTS

- The system shall turn off automatically if the measured internal temperature is greater than 50 degrees Celsius.
- The system shall enable the HIGHTEMP status message if the measured internal temperature is greater than 40 degrees Celsius.
- If the database subsystem does not respond to a query within five seconds of the initial request, the message “WAITING…” shall appear on the status bar.
- Usernames on the system shall consist of between three and ten alphanumeric characters, case-insensitive.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

- This is a listing of all requirements, quality attributes, design constraints, and other details necessary to describe a given software system to be developed
- Can be formal or informal
  - Formal: IEEE 830 format
  - Informal: Web page drawn on back of a napkin
- Variations include the product backlog in Agile Scrum

WHY HAVE A SRS?

- Software team needs to know what to make
- Customers need to specify what they want
- The SRS allows those two sides to come together
- Considering the requirements early on in a document allows us to reduce development time, understand project sooner, make estimates in terms of cost/resources/schedule, prioritize features, scope, quality, etc.

REQUIREMENTS ENGINEERING PROCESS

- Elicitation
- Documentation / definition
- Specification
- Prototyping
- Analysis
- Review and validation
- Agreement and acceptance
- *Note: not all of the above are necessary on every software project!*

REQUIREMENTS ELICITATION

- Determining what the requirements should be based on interactions with the customer, user, and other stakeholders
- Sometimes called “requirements gathering”, but I think this minimizes the amount of work necessary!
- Various methods: interviews, paper prototyping, descriptions, interacting withs similar software
- Methods of elicitation will vary based on user/customer type, skill level, etc.

REQUIREMENTS ELICITATION

- This is hard work and involves specialized skills, often very domain-focused (e.g. background in finance, or medicine, etc.)
- Have to be able to take the vapor of a customer's idea and distill it into a format suitable for development team
- During this phase, also try to determine prioritization, root out internal/external inconsistency, avoid stating "how" problems are to be solved
- "Requirement engineers" or "system engineers"

LEVELS OF ELICITATION

- **High level:** What is the basic business rationale for the software project? What are the issues it is seeking to ameliorate? What are the problems it is trying to solve?

- **Low level:** Specifically, what do you want the program to do? What kinds of trade-offs should we make, in terms of quality attributes? How should the system interact with other actors (users, APIs, other systems, etc.)?

HIGH-LEVEL REQUIREMENTS ELICITATION

- **Opportunity / needs -** What problem are we trying to solve?
- **Justification-** What will be the payback of spending time/resources on this project?
- **Scope -** What are the limits of what we are trying to solve?
- **Major constraints -** Are there any constraints (in terms of budget, resources, schedule) on the development of the project?
- **Major functionality -** What are the key features that you want the software to do?
- **Success factor -** How will we know that our project is successful? By what metrics?
- **User characteristics -** What are the salient characteristics of the users?

DETAILED REQUIREMENTS ELICITATION

- **Individual functionality** - What should the program do, specifically?
- **Business flow** - How will this software and its functionality interact with other aspects of the business?
- **Data, formats, and information needs** - How should data be transferred to/from external systems? In what format? As part of what flow?
- **User interfaces** - How should users interact with the system, specifically?
- **System interfaces** - With what other systems does the system interact with? What should the system do if it cannot interact properly with them?
- **Other attributes / constraints** - Reliability? Performance? Security? How should we prioritize and decide to make trade-offs?

REQUIREMENTS ANALYSIS

- Once we have requirements, we need to analyze them:
  - Categorization
  - Prioritization

CATEGORIZATION METHODOLOGIES

- Cluster by dimension (e.g. Individual Functionality, Business Flow, Data Format, User Interfaces, System Interfaces, Other Attributes / Constraints)
- Via Use Cases (e.g. by actor or individual use case)
- Viewpoint-oriented requirements definition (VORD) - A complex process by which multiple stakeholders offer their viewpoints on requirements, which are then refined and mapped to the system

PRIORITIZATION

- Based on..
  - Current customer demands
  - Competition and current market conditions
  - Future customer needs
  - Immediate sales advantage
  - Critical problems in the existing product
  - Technical reasons (e.g., one requirement depends on another)

PRIORITIZATION: ANALYTICAL HIERARCHY PROCESS

- Requirement priorities are determined in relationship to other requirements (scale of 1 - 9). This is the **intensity value**.
- For each requirement, sum up the intensity, and divide each element by the sum of the column
- Now sum up each of the rows
- This normalizes the requirements as percentages and should always (barring rounding errors) give us a total of ~100%

EXAMPLE REQUIREMENTS

- 1 - The Rent-A-Cat system shall allow users to rent and return cats.
- 2 - The Rent-A-Cat system shall only allow usernames of between three and ten alphanumeric characters.
- 3 - The Rent-A-Cat system shall allow users to search for cats by name.
- 4 - The Rent-A-Cat system shall allow users to search for cats by length of tail.

PAIRWISE COMPARISON MATRIX

	1	2	3	4
1	1	6	3	9
2	1 / 6	1	1/2	2 / 3
3	1 / 3	2	1	3
4	1 / 9	3 / 2	1 / 3	1

Note an inconsistency here!

NORMALIZED PAIRWISE COMPARISON MATRIX

	1	2	3	4
1	1	6	3	9
2	1 / 6	1	1/2	2 / 3
3	1 / 3	2	1	3
4	1 / 9	3 / 2	1 / 3	1
TOTAL	1+1/6+1/3 +1/9= 1.611	6+1+2+3/2 = 10.5	3+1/2+1+ 1/3 = 4.833	9+2/3+3+1 = 13.667

NORMALIZED PAIRWISE COMPARISON MATRIX

	1	2	3	4
1	1 / 1.611	6 / 10.5	3 / 4.833	9 / 13.667
2	(1 / 6) / 1.611	1 / 10.5	(1/2) / 4.833	(2 / 3) / 13.667
3	(1 / 3) / 1.611	2 / 10.5	1 / 4.833	3 / 13.667
4	(1 / 9) / 1.611	(3 / 2) / 10.5	(1 / 3) / 4.833	1 / 13.667
TOTAL	1.611	10.5	4.833	13.667

NORMALIZED PAIRWISE COMPARISON MATRIX

	1	2	3	4
1	0.621	0.571	0.621	0.658
2	0.103	0.010	0.103	0.049
3	0.207	0.190	0.207	0.220
4	0.069	0.143	0.069	0.073
TOTAL	1.611	10.5	4.833	13.667

NORMALIZED PAIRWISE COMPARISON MATRIX

	1	2	3	4	TOTAL
1	0.621	0.571	0.621	0.658	2.471 / 4
2	0.103	0.010	0.103	0.049	0.265 / 4
3	0.207	0.190	0.207	0.220	0.824 / 4
4	0.069	0.143	0.069	0.073	0.354 / 4

NORMALIZED PAIRWISE COMPARISON MATRIX

	1	2	3	4	TOTAL
1	0.621	0.571	0.621	0.658	0.618 = 61.8%
2	0.103	0.010	0.103	0.049	0.066 = 6.6%
3	0.207	0.190	0.207	0.220	0.206 = 20.6%
4	0.069	0.143	0.069	0.073	0.089 = 8.9%

REQUIREMENTS TRACEABILITY

- Why?
  - Verify that all features have been developed and requirements met
  - Test based on requirements (assess quality)
  - Ensure that we are not doing anything unnecessary which cannot be traced back to requirements (scope creep)

TYPES OF TRACEABILITY

- Backward from traceability - Links requirement to where it originated (person, document, etc.)
- Forward from traceability - Links requirement to design / implementation in the system
- Backward to tracebility - Link design / implement to requirement whence it came
- Forward to traceability - Links origination of requirement to requirement itself
- There may also be a need for requirements to link to each other (e.g. co- or pre-requisites)

REQUIREMENTS DEFINITION

- Requirements may be listed using text (“The system shall…”) or diagrams (abstract drawings of how the system should work)
  - UML (Unified Modeling Language)
  - DFD (data flow diagram)
  - ER (Entity relationship diagram)

PROTOTYPING

- We will rarely be able to implement a user-facing system entirely from requirements in a way that meets all of the customer's needs
- Can modify requirements from feedback on prototype
- Low-fidelity prototyping ("paper prototyping")
- Various tools :
  - Balsamiq Mockups
  - Atomic
  - Origami Studio

SIGNING OFF

- Usually a group of stakeholders will agree to a final SRS
- This may be very formal (involving numerous legal personnel) or informal (email)
- Changes should be monitored -
  - In Agile environment, these are expected, but should still be closely monitored!
  - In traditional methodologies, often involves submitting a requirement change request. Usually a more involved process.