Elec 4309 Senior Design

Wendell H Chun

Sept. 5, 2017

University of Colorado
Denver

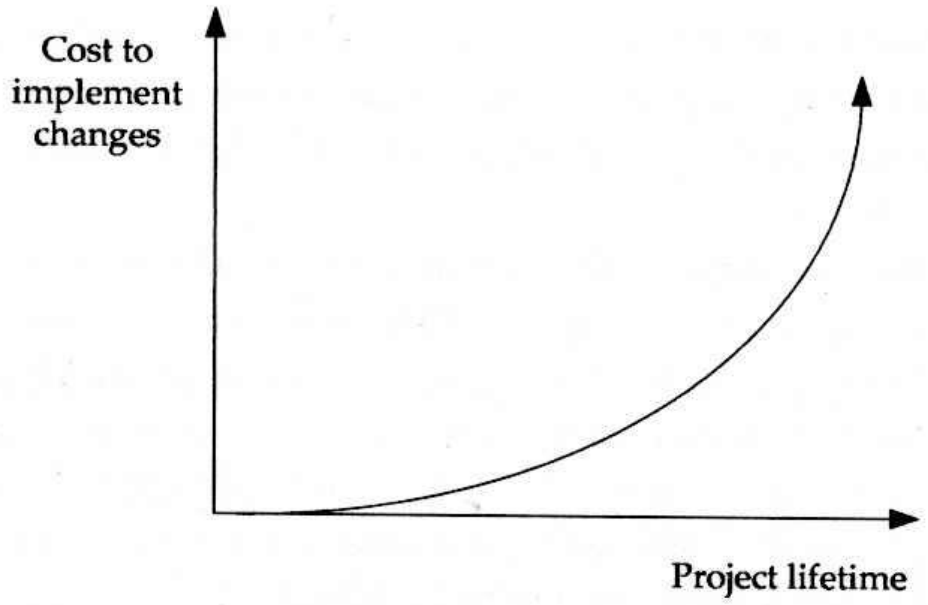
Engineering Design

- Engineering design is the process of devising a system, component, or process to meet desired needs. It is a decision making process in which the basic sciences and mathematics and engineering sciences are applied to convert resources optimally to meet a stated objective. Among the fundamental elements of the design process are the establishment of objectives and criteria, requirements, synthesis, analysis, construction, and testing….

Engineering Design Process

- Creative process
- Problem solving – the big picture
- No single "correct" solution
- Technical aspects only small part

Cost of Design Changes



- Costs increase exponentially as the project lifetime increases

Needs Identification

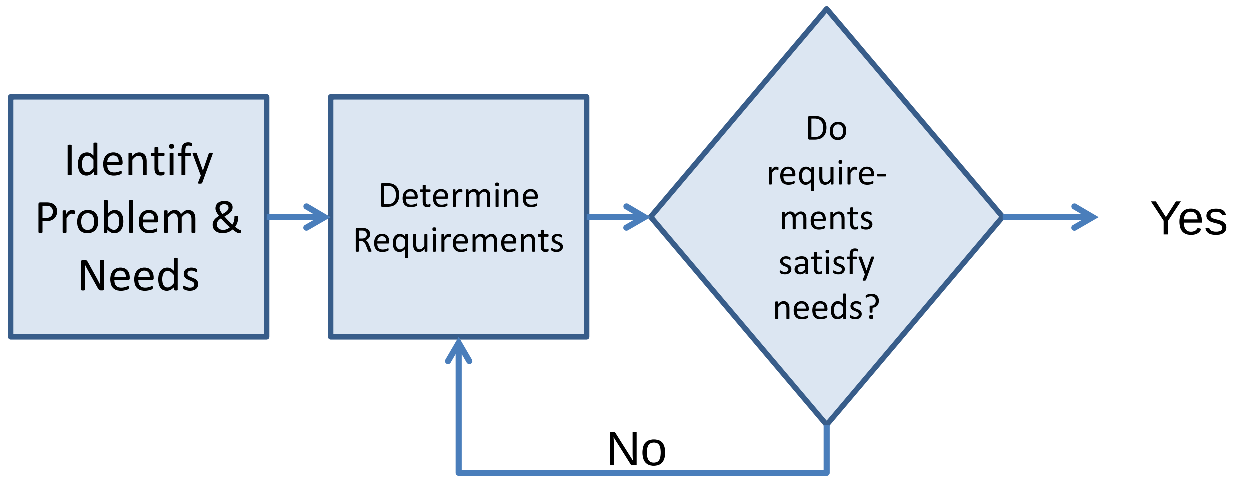
- What is the Problem?

1. Collect information
2. Interpret information
3. Organize needs hierarchy
4. Determine relative importance of needs
5. Review outcomes and process

Problem Statement

- Example:
  - Need: Drivers have difficulty seeing obstructions in all directions
  - Objective: design system to avoid accidents

Problem Identification and Requirements Specifications



A prescriptive design process for problem identification and requirements selection

Requirements Engineering

- The **process of** establishing the services that the **customer requires from a system** and the **constraints** under which it operates and is developed
- Requirements may be **functional** or **non-functional**:
  - Functional requirements describe system services or functions
  - Non-functional requirements is a constraint on the system or on the development process

University of Colorado
Denver

Requirements Engineering

- Very challenging activity
- Requires collaboration of people with different backgrounds
  - User with application domain knowledge
  - Developer with implementation domain knowledge
- Bridging the gap between user and designer
  - Concept of Operations: Description or Pictorial
  - Scenarios: Example of the use of the system in terms of a series of interactions between the user and the system
  - Use cases: Abstraction that describes a class of scenarios

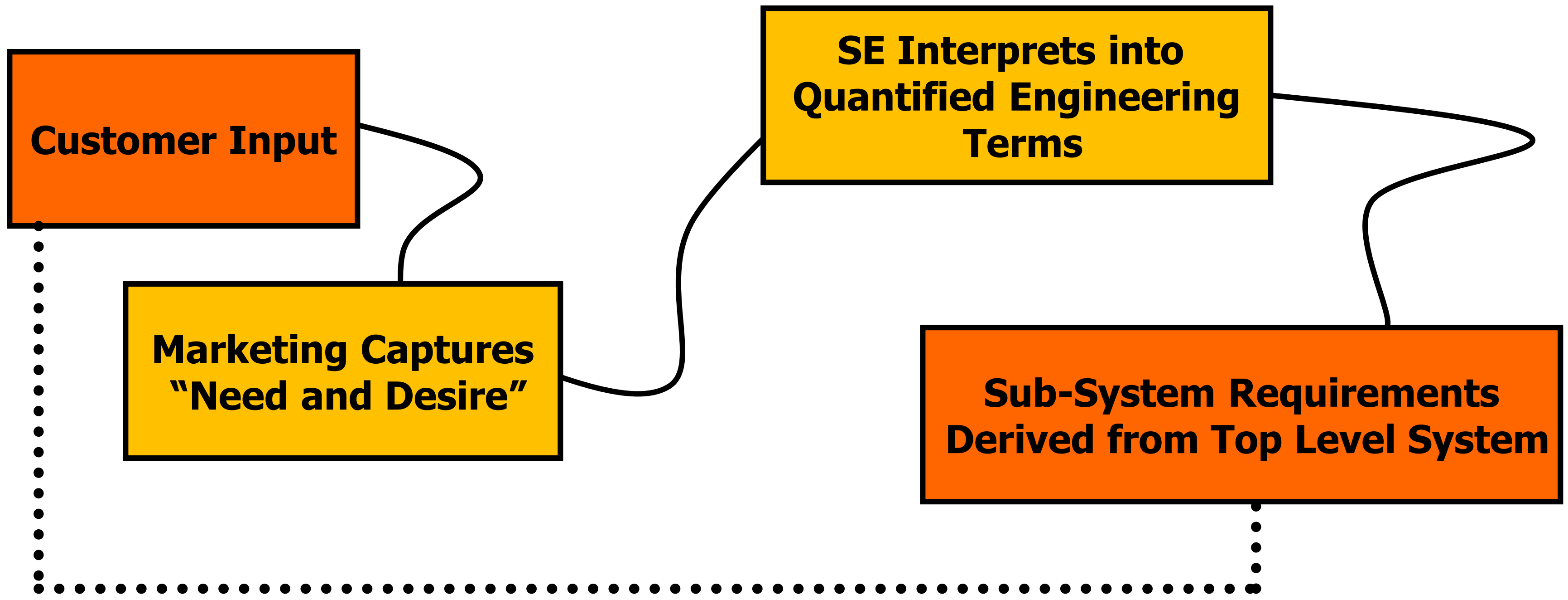
Types of Requirements Engineering

- Greenfield Engineering
  - Development starts from scratch, no prior system exists, the requirements are extracted from the end users and the client
  - Triggered by user needs
- Re-engineering
  - Re-design and/or re-implementation of an existing system using newer technology
  - Triggered by technology enabler
- Interface Engineering
  - Provide the services of an existing system in a new environment
  - Triggered by technology enabler or new market needs

What is a Requirement?

- It may range from a **high-level** abstract statement of a service or of a system constraint to a **detailed** mathematical functional specification
- This is inevitable as requirements may serve a dual function:
  - May be the **basis for a bid for a contract** - therefore must be **open to interpretation**
  - May be the **basis for the contract** itself - therefore must be **defined in detail**
  - Both these statements may be called requirements

"Writing Good Engineering Requirements…Communicating Input"


Customer Input
Marketing Captures "Need and Desire"
SE Interprets into Quantified Engineering Terms
Sub-System Requirements Derived from Top Level System

Goal: Build a Thought Process for What and Why?

Requirements Engineering Process

- Feasibility study
  - Find out if the current user needs be satisfied given the available technology and budget?
- Requirements analysis
  - Find out what system stakeholders require from the system
- Requirements definition
  - Define the requirements in a form understandable to the customer
- Requirements specification
  - Define the requirements in detail

Requirements Definition/Specification

- Requirements definition
  - A statement in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers
- Requirements specification
  - A structured document setting out detailed descriptions of the system services. Written **as a contract** between client and contractor
- Software specification - Example
  - A detailed software description which can serve as a basis for a design or implementation. Written for **developers**

Requirements Definition

- Should specify external behavior of the system so the requirements should not be defined using a computational model
- Includes functional and non-functional requirements:
  - Functional requirements are statements of the services that the system should provide
  - Non-functional requirements are constraints on the services and functions offered by the system

Writing Requirements Definitions

- Natural language, supplemented by diagrams and tables is the normal way of writing requirements definitions
- This is universally understandable but three types of problem can arise:
  - Lack of clarity. Precision is difficult without making the document difficult to read
  - Requirements confusion. Functional and non-functional requirements tend to be mixed-up
  - Requirements amalgamation. Several different requirements may be expressed together

Problems with Natural Language

- Natural language relies on the specification readers and writers using the same words for the same concept
- A natural language specification is over-flexible and subject to different interpretations
- Requirements are not partitioned by language structures

Definitions and Specifications

Requirements Definition

1. The design must provide a means of representing and accessing external files created by other tools.

Requirements Specification

1.1 The user should be provided with facilities to define the type of external files.

1.2 Each external file type may be have an associated tool which may be applied to the file.

1.3 Each external file type may be represented as a specification on the user's display

1.4 Facilities should be provided for the icon representing an external file type to be defined by the user.

1.5 When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Defining Requirements

- Editor requirement mixes up functional and non-functional requirements and is incomplete
- Easy to criticize but hard to write good requirements definitions
- Use of a standard format with pre-defined fields to be filled means that information is less likely to be missed or messed up

Requirements Rationale

- It is important to provide rationale with requirements
- This helps the developer understand the application domain and why the requirement is stated in its current form
- Particularly important when requirements have to be changed. The availability of rationale reduces the chances that change will have unexpected effects

Requirements Specification

- The specifications adds detail to the requirements definition. It should be consistent with it.
- Usually presented with system models which are developed during the requirements analysis. These models may define part of the system to be developed.
- Often written in natural language but this can be problematical.

Non-functional Requirements

- Define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Non-functional Classifications

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organizational requirements
  - Requirements which are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Non-functional Requirements Examples

- Product requirement
  - 4.C.8 It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Ada character set.
- Organizational requirement
  - 9.3.2 The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.
- External requirement
  - 7.6.5 The system shall provide facilities that allow any user to check if personal data is maintained on the system. A procedure must be defined and supported in the software that will allow users to inspect personal data and to correct any errors in that data.

Requirements Separation

- Functional and non-functional requirements should, in principle, be distinguished in a requirements specification.
- However, this is difficult as requirements may be expressed as whole system requirements rather than constraints on individual functions.
- It is sometimes difficult to decide if a requirement is functional or non-functional:
  - For example, requirements for safety are concerned with non-functional properties but may require functions to be added to the system

5 Principles to Good Requirements

1. Communicate Input to Design
   - What are we solving?
   - Why is this function important?
   - Clarity to Cross-Functional Team
2. Measurable & Testable
   - Verification and Validation are Possible
   - Subjective Requirements cannot be Verified
3. Requirements are Focused
   - Audience for Requirement is known
4. Provide Value to Development
   - Based on Need: Answer WHY?
5. Free of Specific Design Content

Specific Goals for Different Requirements

- Customer (Marketing)
  - Needs: “Must Have”
  - Desires: “Nice to Have”
  - What are we trying to Solve?
- System (Top Level Engineering)
  - Translate Customer Input to Engineering Requirements
  - Convert Subjective to Objective
  - Conduit for Communication:
    - Customer to Development Team
- Sub-System (Engineering)
  - Higher Resolution
  - Specific to a Functional Domain
  - Often Communication Tool for Sub-Contractor
  - Direct Communication to Test Team

Goal:
Communication of Customer’s Problem Statement

"Writing Good Engineering Requirements…Communicating Inputs"

Avoid the Following:

- ✗ Good requirements with Subjective Content
  - Use of "should, might, higher, faster…." all are vague and subjective.
    - » Action: Use Positive and Objective Terms.
    - » Example: "**Shall** move at 100 meters per second".
- ✗ Conflicting Requirements
- ✗ Multiple Objectives for a Single Key
    - » Action: Create Separate Keys for each Objective

  Example: Velocity **OR** Acceleration

  Example: Size **OR** Weight
- ✗ Comparative Requirements

  Example: "Shall be at least 20% Better than best available"

Goal:
Avoid Common Pitfalls

University of Colorado Denver

Wicked Problems

- Most **large engineering systems and software systems** address wicked problems
- Problems which are so **complex** that they can **never be fully understood** and where understanding **evolves** during the system development
- Therefore, requirements are normally both **incomplete** and **inconsistent**

System Models

- **Different models** may be produced during the requirements analysis activity
- Requirements analysis may involve three structuring activities which result in these different models:
  - **Partitioning**. Identifies the structural (part-of) relationships between entities
  - **Abstraction**. Identifies generalities among entities
  - **Projection**. Identifies different ways of looking at a problem
- Using modeling techniques, e.g. UML software example

System-level Requirements

- Some requirements place constraints on the system as a whole rather than specific system functions
- Example:
  - The time required for training a system operator to be proficient in the use of the system must not exceed 2 working days.
- These may be emergent requirements which cannot be derived from any single sub-set of the system requirements

Reasons for Inconsistency

- Large systems must improve the current situation. It is **hard to anticipate the effects** that the new system will have on the organization
- **Different users** have different requirements and priorities. There is a constantly shifting **compromise** in the requirements
- System **end-users** and organizations who **pay** for the system have different requirements
- Prototyping is often required to clarify requirements

The Requirements Document

- The requirements document is the official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

Requirements Document Structure

- Non-functional requirements definition
  - Define constraints on the system and the development process
- System evolution
  - Define fundamental assumptions on which the system is based and anticipated changes
- Requirements specification
  - Detailed specification of functional requirements
- Appendices
  - System hardware platform description
  - Database requirements (as an ER model perhaps)
- Index

Requirements Document Requirements

- Specify external system behavior
- Specify implementation constraints
- Easy to change
- Serve as reference tool for maintenance
- Record forethought about the life cycle of the system, i.e. predict changes
- Characterize responses to unexpected events

Requirements Checking

- **Validity**. Does the system provide the functions which best support the customer’s needs?
- **Consistency**. Are there any requirements conflicts?
- **Completeness**. Are all functions required by the customer included?
- **Realism**. Can the requirements be implemented given available budget and technology

Review Checks

- Verifiability. Is the requirement realistically testable?
- Comprehensibility. Is the requirement properly understood?
- Traceability. Is the origin of the requirement clearly stated?
- Adaptability. Can the requirement be changed without a large impact on other requirements?

Requirements Reviews

- Regular reviews should be held while the requirements definition is being formulated
- Both client and contractor staff should be involved in reviews
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage

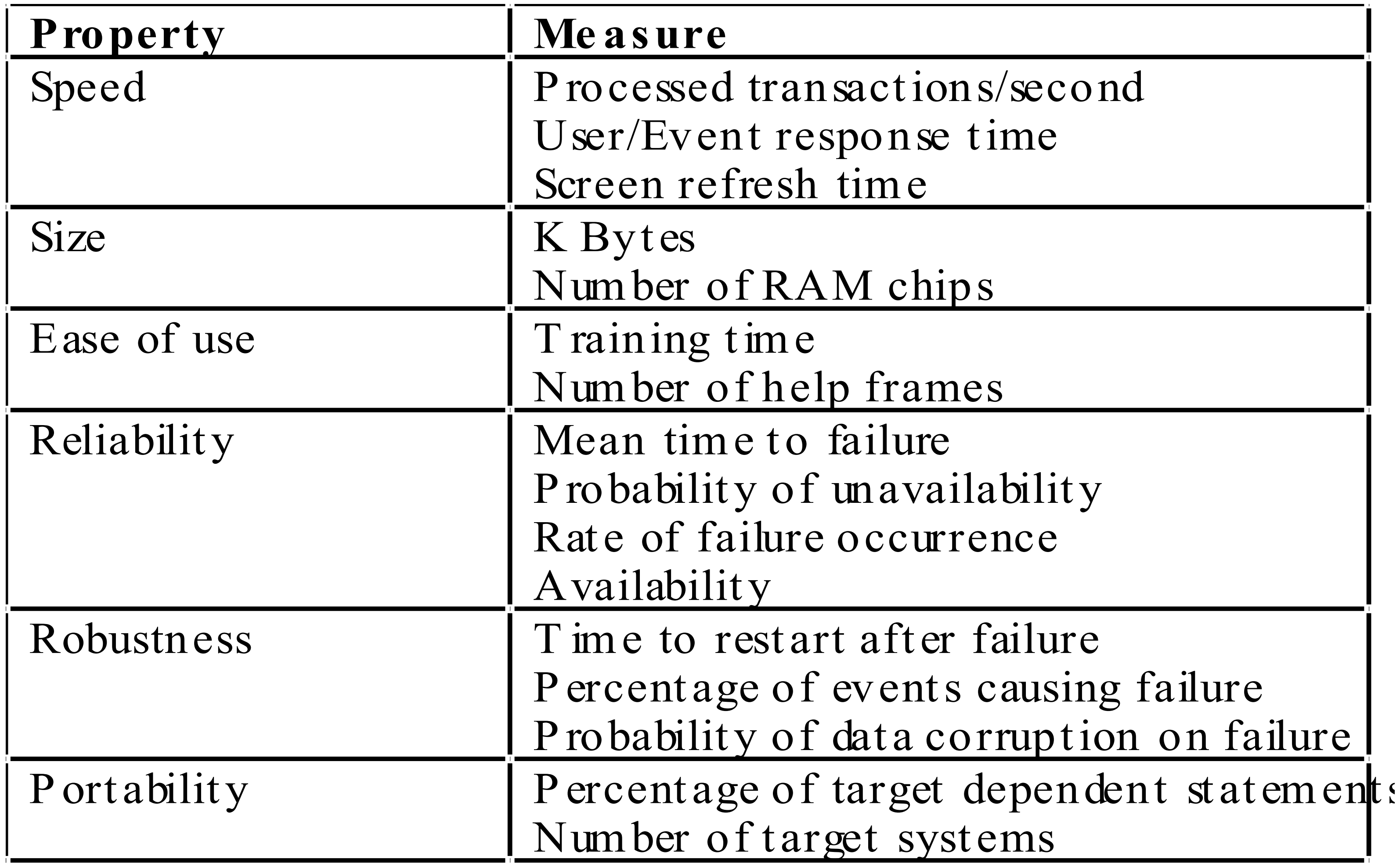
Requirements Validation

- Concerned with demonstrating that the requirements define the system that the customer really wants
- Requirements error costs are high so validation is very important:
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error
- Prototyping is an important technique of requirements validation

Requirements Verifiability

- Requirements should be written so that they can be objectively verified
- The problem with this requirement is its use of vague terms such as ‘errors shall be minimized’
  - The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.
- The error rate should be quantified:
  - Experienced controllers should be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users should not exceed two per day.

Requirements Measures



Property	**Measure**
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statement Number of target systems

University of Colorado Denver

Requirements Validation

- **Critical step in the development process:**
  - Usually after requirements engineering or requirements analysis. Also at delivery.
- **Requirements validation criteria**
  - Correctness:
    - The requirements represent the client’s view.
  - Completeness:
    - All possible scenarios through the system are described, including exceptional behavior by the user or the system
  - Consistency:
    - There are functional or nonfunctional requirements that contradict each other
  - Realism:
    - Requirements can be implemented and delivered

Additional Requirements Validation Criteria

- Traceability:
  - Each system function can be traced to a corresponding set of functional requirements
  - With AOSD we can improve traceability of requirements.
  - Goal: can we map requirements to aspects?

Design Methodologies

"EXPLORER" METHOD:

- Typically used for new design ideas or research. It is useful in initial design and specification stages, and is often used when in "unfamiliar territory":

  1) Move in some direction; e.g. toward the library, telephone, domain expert's office, etc.
  2) Look at what you find there.
  3) Record what you find in your notebook.
  4) Analyze findings in terms of where you want to be.
  5) Use results of analysis to choose next direction.
  6) Back to 1) and continue exploring

Design Methodologies: Tops-Down

- Also called “functional decomposition”
- Implementation details considered only at the lowest level
- Top-down design, is not so clean and linear in practice
- Often implementation level commitments are made at high levels in the design process

Design Methodologies

BOTTOM-UP DESIGN:

- Opposite of top-down
- Start at the bottom with detail design
- To do this, you must have some idea of where you are going. So, often this becomes...

HYBRID DESIGN:

- Combines aspects of both top-down and bottom-up
- More practical design approach then pure top-down
- Start with a top-down approach, but have feedback from the bottom

Design Methodologies

CASE-BASED:

- Research a specific, similar design case study
- Model your process on that

INCREMENTAL REDESIGN:

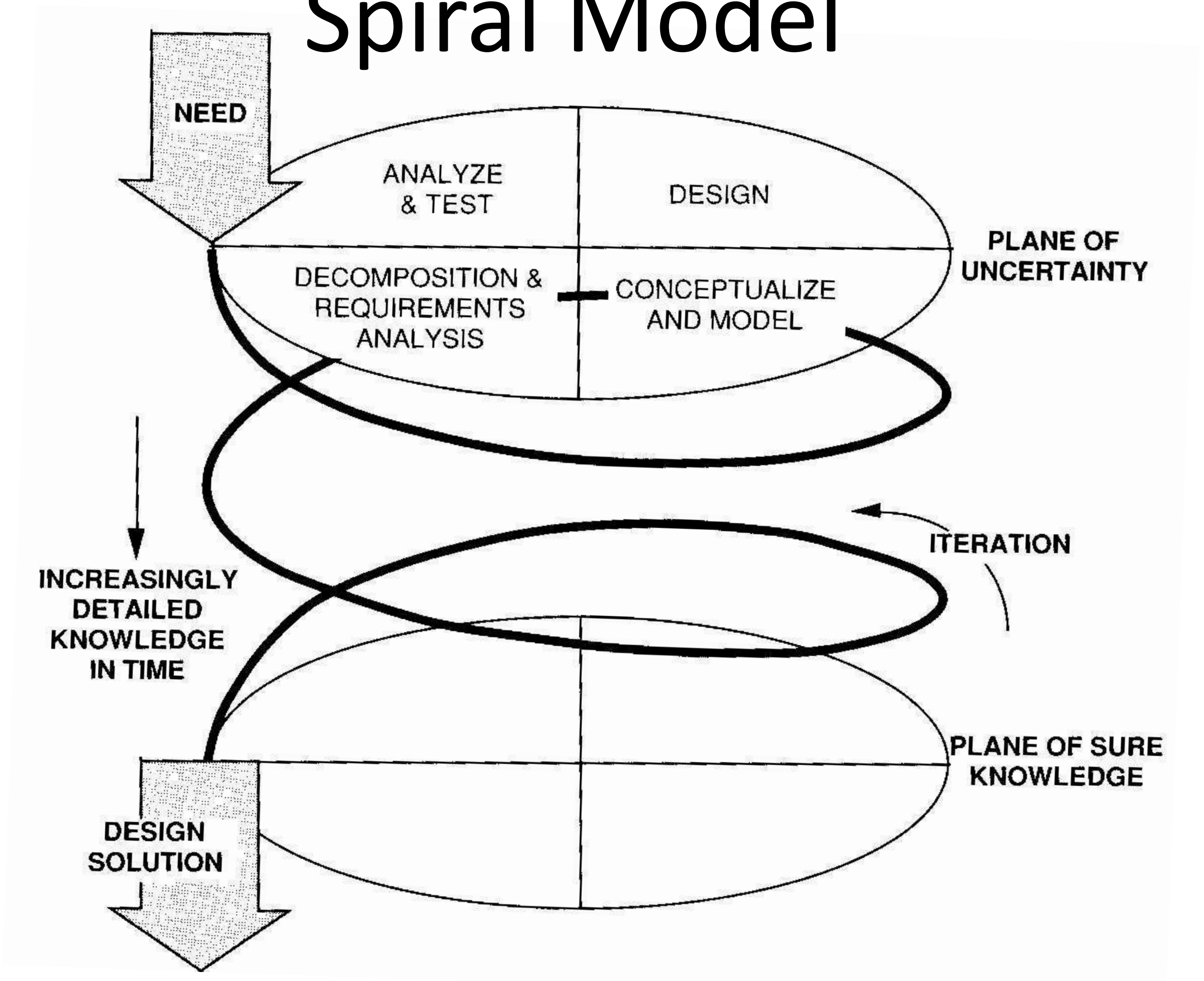
- Find an existing design and "unravel" the design from the bottom up
- Modify as required
- Detailed and least global aspects of the design are explored and redesigned, if necessary, first

Design Methodologies

ITERATIVE REFINEMENT:

- An iterative top-down approach
- First a rough, approximate and general design is completed
- Then we do it finer, more exact and more specific
- This process continues iteratively until the complete detail design is done

Spiral Model



Design Considerations

The design of a component or system may be influenced by a number of requirements. If a requirement affects design, it is called a design consideration. For example, if the ability to carry large loads without failure is important, we say that strength is a design consideration. Most product development projects involve a number of design considerations:

- Strength/stress
- Distortion/stiffness
- Wear
- Corrosion
- Safety
- Reliability
- Friction
- Usability/utility
- Cost
- Processing requirements
- Weight
- Life
- Noise
- Aesthetic considerations
- Shape
- Size
- Scrapping/recyclability
- Thermal properties
- Surface finish
- Lubrication
- Marketability
- Maintenance
- Volume
- Liability

Design Considerations

- WORST CASE DESIGN
  - Component variation
  - Environmental conditions
  - Use computer simulations

Role of Management in Stimulating Creativity

- Recognize individuality
- Be tolerant of mistakes
- Be supportive under stress
- Techniques include:
  - Competitive teams
  - Idea bank of unused ideas for possible reuse
  - Encourage interaction – even in how offices are laid out

Project Management is Part of Design

- Work breakdown structure
  - Hierarchical breakdown of tasks and deliverables need to complete project

- Activity
  1. Task – action to accomplish job
  2. Deliverable – e.g. circuit or report

Project Management is Part of Design

- Define for each activity:
  1. Work to be done
  2. Timeframe
  3. Resources needed
  4. Responsible person(s)
  5. Previous dependent activities
  6. Checkpoints/deliverables for monitoring progress

WBS Example (Part of Design)

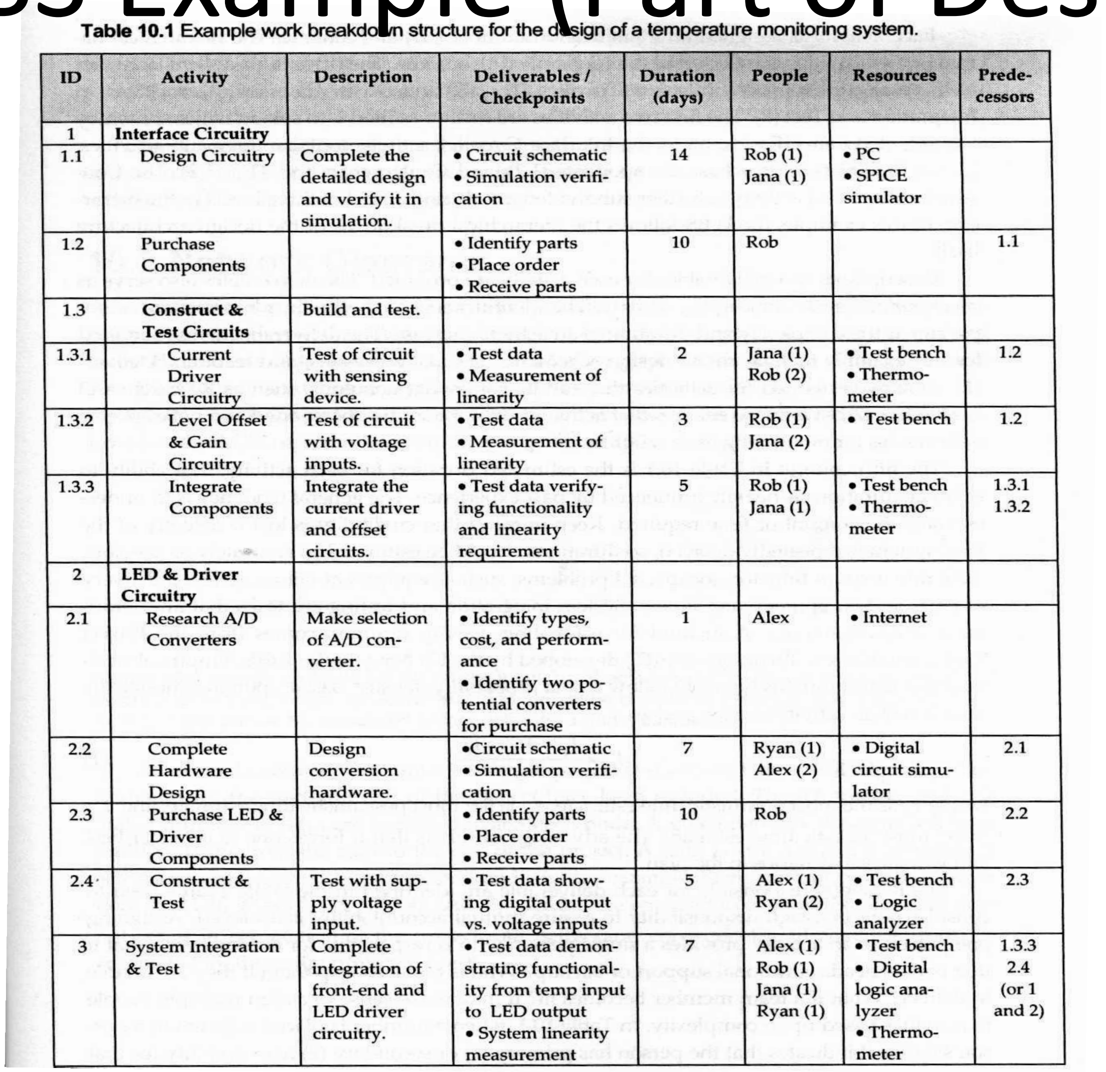
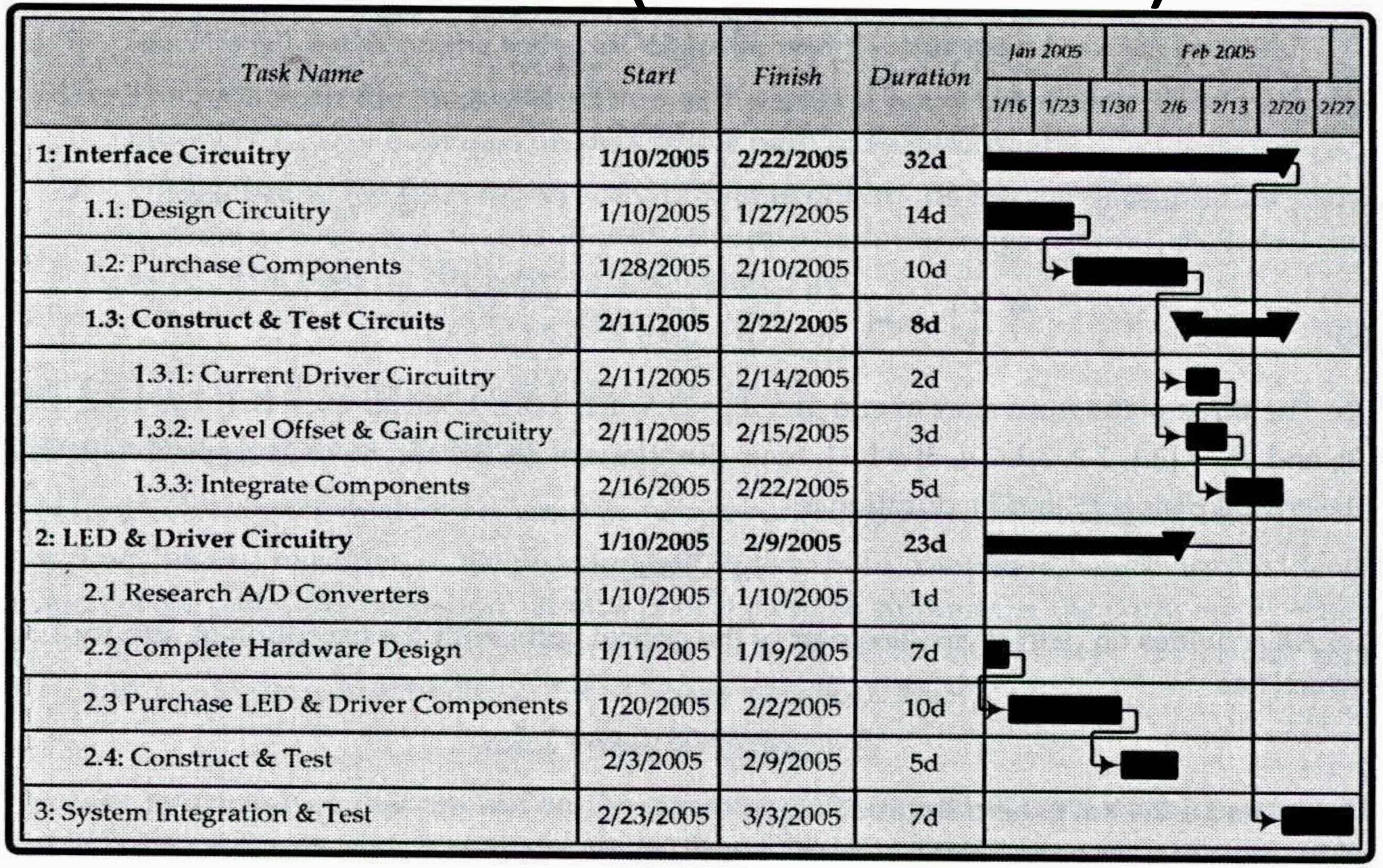


Table 10.1 Example work breakdown structure for the design of a temperature monitoring system.

ID	Activity	Description	Deliverables / Checkpoints	Duration (days)	People	Resources	Predecessors
1	**Interface Circuitry**						
1.1	Design Circuitry	Complete the detailed design and verify it in simulation.	• Circuit schematic • Simulation verification	14	Rob (1) Jana (1)	• PC • SPICE simulator	
1.2	Purchase Components		• Identify parts • Place order • Receive parts	10	Rob		1.1
1.3	**Construct & Test Circuits**	Build and test.					
1.3.1	Current Driver Circuitry	Test of circuit with sensing device.	• Test data • Measurement of linearity	2	Jana (1) Rob (2)	• Test bench • Thermometer	1.2
1.3.2	Level Offset & Gain Circuitry	Test of circuit with voltage inputs.	• Test data • Measurement of linearity	3	Rob (1) Jana (2)	• Test bench	1.2
1.3.3	Integrate Components	Integrate the current driver and offset circuits.	• Test data verifying functionality and linearity requirement	5	Rob (1) Jana (1)	• Test bench • Thermometer	1.3.1 1.3.2
2	**LED & Driver Circuitry**						
2.1	Research A/D Converters	Make selection of A/D converter.	• Identify types, cost, and performance • Identify two potential converters for purchase	1	Alex	• Internet	
2.2	Complete Hardware Design	Design conversion hardware.	•Circuit schematic • Simulation verification	7	Ryan (1) Alex (2)	• Digital circuit simulator	2.1
2.3	Purchase LED & Driver Components		• Identify parts • Place order • Receive parts	10	Rob		2.2
2.4	Construct & Test	Test with supply voltage input.	• Test data showing digital output vs. voltage inputs	5	Alex (1) Ryan (2)	• Test bench • Logic analyzer	2.3
3	System Integration & Test	Complete integration of front-end and LED driver circuitry.	• Test data demonstrating functionality from temp input to LED output • System linearity measurement	7	Alex (1) Rob (1) Jana (1) Ryan (1)	• Test bench • Digital logic analyzer • Thermometer	1.3.3 2.4 (or 1 and 2)

University of Colorado Denver

Schedule (Gantt Chart)



Task Name	Start	Finish	Duration
1: Interface Circuitry	**1/10/2005**	**2/22/2005**	**32d**
1.1: Design Circuitry	1/10/2005	1/27/2005	14d
1.2: Purchase Components	1/28/2005	2/10/2005	10d
1.3: Construct & Test Circuits	**2/11/2005**	**2/22/2005**	**8d**
1.3.1: Current Driver Circuitry	2/11/2005	2/14/2005	2d
1.3.2: Level Offset & Gain Circuitry	2/11/2005	2/15/2005	3d
1.3.3: Integrate Components	2/16/2005	2/22/2005	5d
2: LED & Driver Circuitry	**1/10/2005**	**2/9/2005**	**23d**
2.1 Research A/D Converters	1/10/2005	1/10/2005	1d
2.2 Complete Hardware Design	1/11/2005	1/19/2005	7d
2.3 Purchase LED & Driver Components	1/20/2005	2/2/2005	10d
2.4: Construct & Test	2/3/2005	2/9/2005	5d
3: System Integration & Test	2/23/2005	3/3/2005	7d

Figure 10.3 Gantt chart for the temperature display project created using Microsoft Visio^{TM}.

Project Management

- Guidelines:
  - Project plan after design plan complete
  - Double time estimates and add 10%
  - Assign a lot of integration and test time
  - Remember lead times for parts ordering
  - Assign tasks based on skills and interests
  - Track progress versus plan
  - Plans change

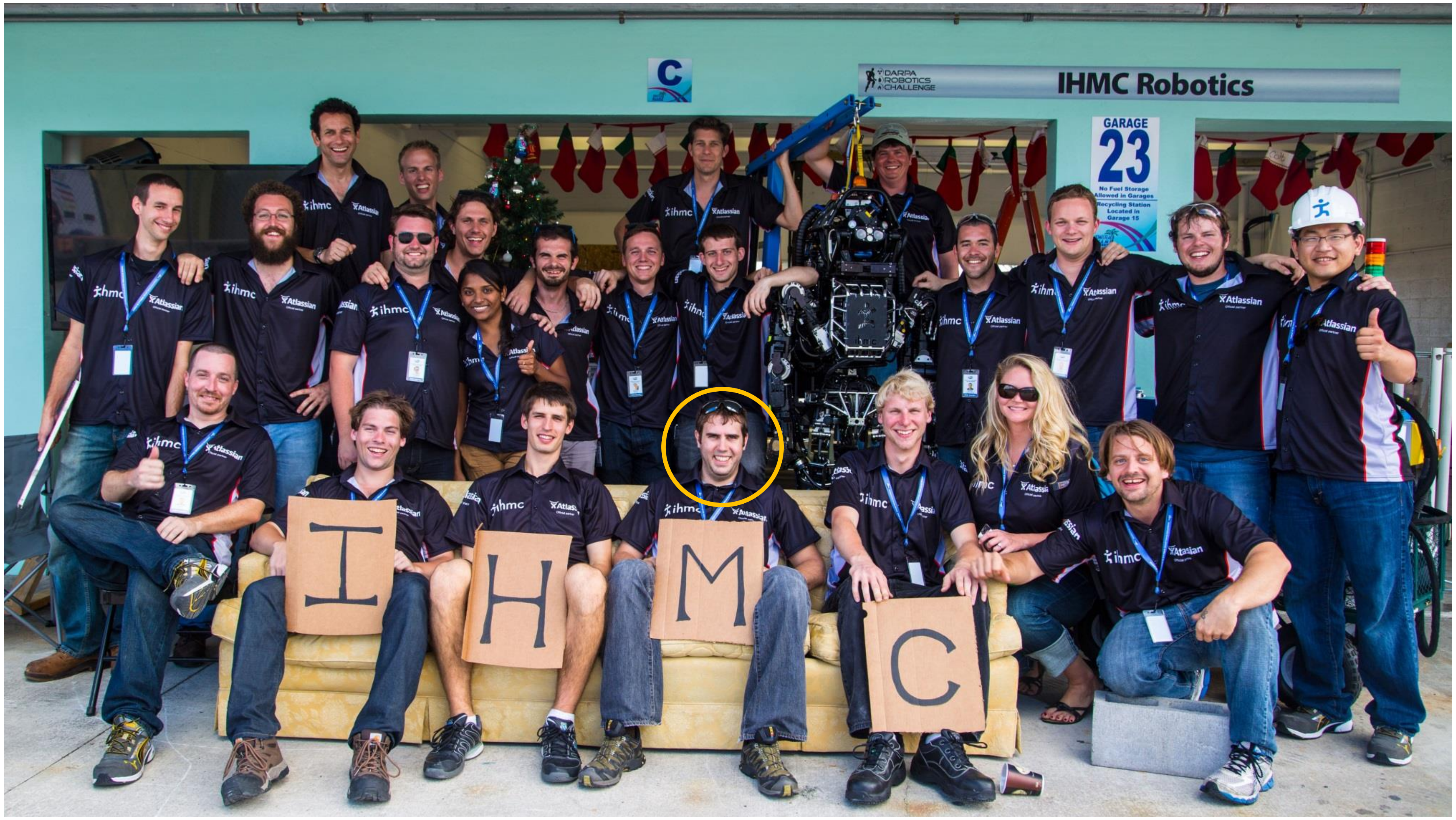
Project Communications

Focus on needs of specific audience

- Who?
  - level of knowledge
  - their motivation – needs
- Why?
  - to persuade
  - to inform

This is Where we are on Teams Today

My Colleague Peter on His Team


DARPA ROBOTICS CHALLENGE
IHMC Robotics
C
GARAGE 23
No Fuel Storage Allowed in Garages
Recycling Station Located in Garage 15
I
H
M
C