Software Requirements Engineering
Introduction
Topics

- Set the context for requirements engineering
  - Requirements engineering as knowledge acquisition and transformation
- Introduce terms, concepts, and activities of requirements engineering
- Why engineer requirements?
Problem and Solution

Problem

Solution

Liberal Arts

Software Engineering
(Requirements Engineering)

Computer Science

Natural Sciences

Mathematics

What?

How?
Requirements Engineering as a Human Activity System

“Requirements engineering draws on cognitive and social sciences to provide both theoretical grounding and practical techniques for eliciting and modeling requirements”

- **Psychology** - an understanding of the **difficulties** people may have in **communicating** their needs
- **Anthropology** - a methodological approach to **observing human activities** to help understand how computer systems may help or hinder those activities
- **Sociology** - an understanding of the **political and cultural changes** caused by a new system.
- **Linguistics** – communication and the use of **language**
- **Philosophy** – understanding stakeholder world view, **beliefs**, **goals**, motivations, logic, agreements, **what is knowable**, etc.
Requirements Engineering Requires Creativity

- Creativity - production of something original and useful
- The creative process:
  - **Fact finding** – gain domain knowledge
  - **Problem finding** – anticipate all possible pitfalls and issues
  - **Idea finding** – generate as many ideas as possible
  - **Solution finding** – which ideas are the most effective and value adding
  - Develop plan of action
"The Five Orders of Ignorance"

- **Software** is a medium for the **storage of knowledge**
  - The product is the stored knowledge; analogous to a book
- The **key challenge** is **knowing what to build - acquiring** the necessary **knowledge**
- So software development produces products as a **knowledge-acquiring** activity
- Hacking is a process of acquiring knowledge as you write the code
  - The final product is contaminated with the legacy of the code that is changed or discarded along the way, the "unknowledge".
  - Prototyping is a name for following this process intentionally.

[Armour, CACM, 10/2000, Vol 43, No 10, P. 17]
### “The Five Orders of Ignorance”

If software development is the acquisition of knowledge, it can also be viewed as the reduction or elimination of ignorance.

<table>
<thead>
<tr>
<th>Order</th>
<th>Description</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0&lt;sup&gt;th&lt;/sup&gt; Order</td>
<td>I know enough to build the software</td>
<td>I have the answer</td>
</tr>
<tr>
<td>1&lt;sup&gt;st&lt;/sup&gt; Order</td>
<td>Lack of knowledge - I know I don’t know something and can do something about it</td>
<td>I have the question and it can be answered</td>
</tr>
<tr>
<td>2&lt;sup&gt;nd&lt;/sup&gt; Order</td>
<td>Lack of awareness - I don’t know that I don’t know</td>
<td>The real problem – I don’t have the answer or the question; where most projects begin</td>
</tr>
<tr>
<td>3&lt;sup&gt;rd&lt;/sup&gt; Order</td>
<td>Lack of process – I don’t know an effective way to find out I don’t know that I don’t know</td>
<td>Find and use methods to frame and eventually answer the question; requirements engineering!</td>
</tr>
<tr>
<td>4&lt;sup&gt;th&lt;/sup&gt; Order</td>
<td>Meta ignorance – I don’t know about the five orders of ignorance</td>
<td>Now you do!</td>
</tr>
</tbody>
</table>
Requirements Engineering as Modeling

- **Systems and users do work**, sometimes in collaboration, sometimes independently
- Requirements engineering is about how to discover and document the work
- **Abstraction hierarchy** – system -> features -> functions -> tasks -> subtasks -> actions
Requirements Engineering as Knowledge Acquisition, Transformation, and Communication

Problem Domain, Stakeholder and User Goals and Needs

Analysis Models

Software Requirements Specification

Design Models and Solution Understanding

What
- Functional capabilities
- Non-functional quality attributes
- Constraints or conditions

How
- Software architecture

Stakeholders, Customers, Users

Requirements Analyst

Software Engineer
Communication

How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it

How the project was documented
What operations were installed
How the customer was billed
How it was supported
What the customer really needed
What is a Requirement?

- **Requirement** [Merriam-Webster On-Line Dictionary]
  - a: something wanted or needed: **necessity**
  - b: something essential to the existence or occurrence of something else: **condition**

- Software requirement is a **condition** or **capability** [IEEE-STD-610.12-1990 Software Engineering Glossary]
  - Needed by a user to solve a problem or achieve an objective
  - Met or possessed by a system to satisfy formally imposed documents such as specifications or standards
Types of Requirements

- **Business** – business objectives, the “business case”
- **Business rules** – policy, guideline, regulation, algorithm
- **User** – goals and tasks for a class of user; HCI
- **Quality Attributes** – non-functional level of service
- **System** – high level requirements for the system at large
- **External Interface** – interfaces to users and/or other systems
- **Constraints** – development choice restrictions
- **Functional** – behavior the system must exhibit to satisfy user and business requirements
Goals vs. Requirements?

- System functions and features *must* address goals
- **Tasks** are the **means to the end** (goals)
- **Goals are the motivation** for observed behaviors
- Goals drive usage behavioral patterns
Why Engineer Requirements?

- The most **significant contributors** to **project failure** relate to **requirements** (Standish Group’s CHAOS Reports)
  - Most failed projects fail due to changing customer requirements
- Meeting your project’s requirements defines success
- We can engineer a higher probability of success
Why Software Fails

- Unrealistic or unarticulated project goals
  - Inaccurate estimates of needed resources

- Badly defined system requirements
  - Poor reporting of the project's status
  - Unmanaged risks

- Poor communication among customers, developers, and users
  - Use of immature technology
  - Inability to handle the project's complexity
  - Sloppy development practices
  - Poor project management
  - Stakeholder politics
  - Commercial pressures

A Notable Example of Requirements Failure

(CNN) -- NASA lost a $125 million Mars orbiter because a Lockheed Martin engineering team used English units of measurement while the agency's team used the more conventional metric system for a key spacecraft operation, according to a review finding released Thursday.

The units mismatch prevented navigation information from transferring between the Mars Climate Orbiter spacecraft team in at Lockheed Martin in Denver and the flight team at NASA's Jet Propulsion Laboratory in Pasadena, California.
“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, …

……

No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.”

[Brooks, The Mythical Man-Month 1995]
Ethical Considerations

- Software engineering ethics starts with requirements
- Consider questions such as …
  - Public and market value
  - Unintended consequences, especially of scale
  - Quality
  - User diversity, cultural values
  - Safety
  - Privacy
  - Practicality within constraints
  - Legal compliance
  - ….
Requirements Engineering  Process Frameworks

- We will survey techniques from a variety of sources
- We will utilize elements of the Unified Modeling Language (UML) from the Rational Unified Process (RUP)
- It is essential to learn some “systematic, disciplined, quantifiable approaches” to engineering of software requirements
  - the “classics” like basic arithmetic
What About Agile Methodologies?

“Everyone is using agile style incremental, iterative development now; UML is just too heavy”
“Just outline requirements and code”

- Agile techniques won’t fit every project, especially for large and complex systems
- Requirements engineering concerns and techniques still apply
- As we discuss the classics, we will periodically consider agile style considerations

Note: Some knowledge of agile methods is assumed
Vocabulary

- **Domain (application)**
  - The target area of interest, usually the business context
  - E.g., health care, scientific research, a product market, an enterprise sales department

- **Business**
  - The sponsoring institution – profit, non-profit, commercial (external products and systems), enterprise (internal products and systems)

- **Problem**
  - A business issue to be resolved or objective to be accomplished by the solution

- **Solution**
  - The system or product that addresses the problem requirements

- **System**
  - The inputs, processing, and outputs comprising the solution
  - “System” will be used generally as the abstraction for what is to be built
Vocabulary (cont)

- **Product**
  - The specific incarnation of the solution – a system, subsystem, application

- **Project**
  - All of the activities managed by some process necessary to create a finished system or product solution to satisfy the domain problem

- **User**
  - The individual (or external system) that interacts with the system features

- **Stakeholder**
  - Individuals (or external systems) that have some vested interest in the success of the system
    - E.g., management, users, customers, developers, service providers, etc.

- **Customer**
  - The individual who pays for and receives the completed system or product
Sources

- Unified Process for Education (UPEDU – Yoopeedoo) (http://www.yoopeedoo.org/ then navigate the UPEDU link)
- Data Analysis Center for Software (DACS) (http://www.dacs.dtic.mil/)
  Requirements Management Gold Practice (http://www.goldpractices.com/practices/mr/index.php)
- IEEE-STD-830-1998 Recommended Practice for Software Requirements Specifications
- Software Engineering Body of Knowledge (http://www.swebok.org/)
  – See also http://www.standishgroup.com/