

**Types of requirements**

* Functional
  + Describe what the system should do
* Non-functional
  + Constraints on the services or functions offered by the system
  + System performance. Security, availability

**Types of Non-functional Requirements**

* Product: execution speed, memory requirements, acceptable failure rate, probability, usability
* Organizational: policies and procedures
* External: interoperability with other systems, legal requirements, ethical requirements

**User Requirement Challenges**

* Ambiguity: Human language is different than user language is different than system language
* Confusion: Functional vs. non-functional vs. system goals vs. design information
* Amalgamation: Single stated requirement may actually contain several requirements

**Ethnography: Observational technique used to understand social and organizational requirements**

* Effective at discovering
  + Requirements derived from the way people actually work rather than how they are supposed to work
  + Derived from cooperation and awareness of co-**stakeholder’s activities**

**Requirements Validation**

* Checkpoint or ensuring that the requirements as specified truly define the system the customer wants
* Look for validity, consistency, completeness, realism, and verifiability

S**tructural Models**

* Capture the system components and their inter-relationships
  + Class diagrams, Generalization, Aggregation

**Behavioral Models**

* Data-flow models
  + Work is often driven by information flowing through an organization and the manner in which the organization processes, consumes, and disseminates it
* Event-driven (state machine) models

Data Models: More static view of information than the data flow diagram

Test Cases

* Effectiveness: Credibly show proper operation
* Efficiency: Repeatable, Self-documenting, Easy to develop and maintain
* Strategies: Test normal and abnormal, Use realistic data, Test boundaries

Object-Orientation

* Developing software solutions involves tow significant translations:
  + The user’s environment and The system design
* Identifies the entities, referred to as objects, that make up the user’s environment
* Develops a model of these objects including their attributes and behaviors
* Implements this design as an application

**Unified Modeling Language (UML)**

* Systems design is complex and abstract
* Expressing it is difficult using “human” language which presents a significant risk of misinterpretation

Reusability

* Reuse has tremendous potential benefits
  + Gives more return on investment
* Impact of new development
  + Time required to build a new application will be shortened if we can reuse existing components

**Incremental vs. Prototyping**

* Incremental development consists of a series of planned efforts designed result in a complete system to user specification
* Prototyping can be used to facilitate incremental development by incrementally improving the prototype into the finished system
* Throw away prototyping on other hand used to produce communication vehicles

***DISCUSSION QUESTIONS***

**1.How has software development improved as a result of the “software crisis” discussed in lecture?, how is it still the same?**

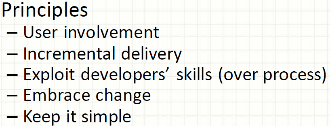
* Different techniques are used when creating a software (especially a sophisticated one)
  + Ex. Reusing parts of a software
  + Creating smaller pieces of the software
  + Making the software as flexible as possible in case of changes.
* It is still the same today because there are always new improvements in technology.
  + New updates
  + New versions
  + New (Better) methods
  + It isn’t too different with the software crisis back then.

**2.Discuss benefits of the Waterfall approach as well as shortcomings of the Agile approach.**

* Waterfall approach:
  + Tedious
  + Time-consuming
  + Many iterations

Less-prone to mistakes

Success is higher (given that the software engineer wasn’t in a hurry)

* Agile Approach:
  + Fast and Easy
  + Easier to build a prototype
  + Can get the big picture done

Higher failure-rate:

Functions were missed

Non-Functions were missed

The software is not working

**4.Why are software systems particularly well-suited for automation?, under what circumstances is it difficult create a software system which automates?**

* A lot of the methods are repeated numerous times that it would be best to automate things instead (if applicable)
* Some of those methods also require multiple iterations which would be time-consuming for a human’s perspective.
* A software system with automation would have to deal with errors in the real world. The “vagueness” and “randomness” of everything.
  + This is the problem because there are too many factors to consider.
  + But it would be possible to narrow down the problems with an “Error Exception” which is very commonly used in a lot of programming languages today.

**5.Why does a compiler consistently translate source code accurately while humans frequently misunderstand each other?**

* Because whatever the source code was made from, the source code is ***absolute***.
* For example, if the source code is based from the English alphabet, then obviously we shouldn’t see any Chinese characters within the source code.
  + The compiler is made to translate the source code made from the English alphabet…
* But for humans, the problem in the real world is that if you speak to another person in English, that doesn’t mean they understand it.
  + Maybe they only speak Spanish?
  + Maybe they have hearing issues?
  + There are too many factors to consider.

**6.Why is it difficult if not impossible to prove perfection?**

* Because it is just a matter of being content
  + ***Is it enough?***
  + What is enough?
    - What were the requirements?
    - Maybe I missed something?
* If things were narrowed down and the requirements were clear, then it is easier to see what to do to meet success.
  + But is it really enough to just stick with the requirements, or should there be more to be done?
  + Then the cycle repeats if the person’s perfectionism is really extreme.

**7.Discuss the concept of defensive design and give an example.**

* Handling errors, being ready for anything that could go wrong.
* Must handle errors safely. Ongoing operation must continue to work correctly.

**8.Discuss the benefits of reuse and steps that should be taken to make components reusable.**

* The component should conform to a standardizedmodel which enforces interfaces, documentation, deployment, etc.
* A reusable component should be able to exist independent of other components
* Public access (methods, knowledge of attributes, etc.)
* must be available but strictly controlled

**9.Discuss differences and similarities between H-C-I and C-C-I designs.**

* Human Computer Interaction
* Computer Computer Interaction

**10.What about “mission critical” makes development more difficult?**

* Expensive and dangerous scenarios where error is ABSOLUTELY NOT ALLOWED.
  + Ex. Space Rocket launches/missile attacks

**Design and architecture**

* Levels of abstraction (large/small)
* Designing the \_\_ (foundation, structure, supports)
* **Control Styles (centralized / event-based)**
* **Think of client server model and botnet for centralized or distributed systems example archit.**
* CASE- comp aided SW engineering

