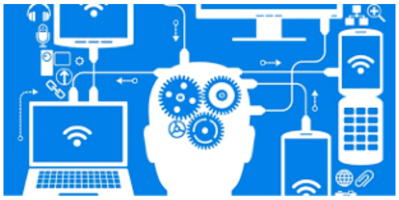
CECS-491-Software-Engineering-Project-I

Seminar Notes



August 30, 2018

Scenarios Vs. Use Cases:

Scenario:	Use Cases:
Hoag Hospital Sends in a RFP	
Due Date is Sept 14th	
Proof of insurance	
Proof of Ability to work with nuclear medicine	
Must have Cali GC license	
Must have Cali Business license	
Build an MRI Exam Room	

Scenario:	Use Cases:
Walmart sends in RFP	
Due Date Oct 7th	
Build a Warehouse	
Proof of Cali GC License	
Proof of EOE practice	
Proof of CA Business License	

Scenarios = Items to be done (Gathering Requirements)

- Also they are binding of Use Cases
    - ALL SCENARIOS become ONE USE CASE for “receiving RFP”
- Don’t need to know why (not within your DOMAIN)

- Where most system fails is not understanding system requirements
    - 1 to 2 years preferred time to study
    - “Scenario” is a good way to approach a system requirements

Use Cases:

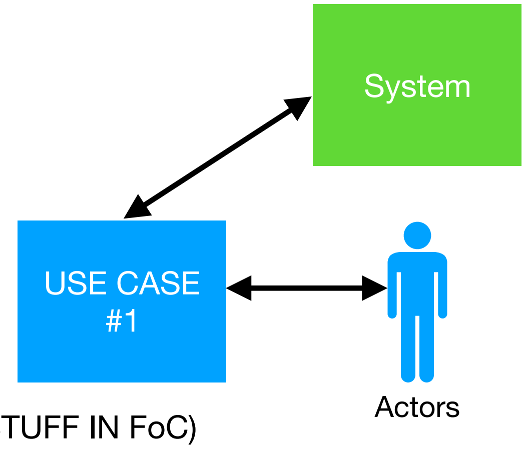
<u>Formatting</u>

Name:

Actors:

PreCondition: (What is true?)

- IE: Company Exists



FoC Flow of Control: (NO ERROR STUFF IN FoC)

Actor VS System

1. User presses log on button 2. Display UI for name & pwd

3. User enters name

4. User enters pwd

Post Conditions: (After FoC, what changes?)

- Assuming everything works, what is changed in our system

Error Conditions: (Problem occurring in FoC)

- We don’t ignore the error, we just put it in error block

IE: User enter wrong password, ask to try again

Quality/Non-Function Requirements: (Adverb of FoC)

- Additional Attributes that needs to be true (not PreCondition) for Functionalities

IE: A network guy making sure when user presses button, it goes to server and back in 3 seconds

- This typically dictates the Use-Case functionalities

September 4, 2018

Non- Functional Requirements

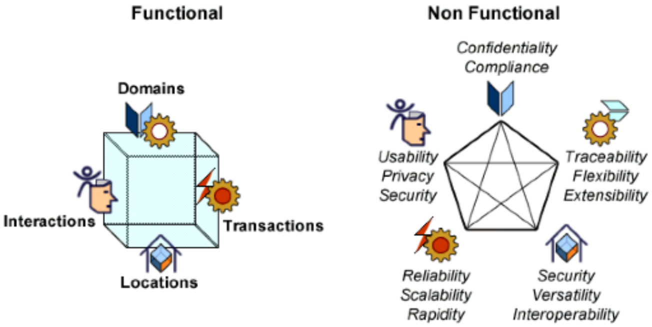
- Usability - the case with which a user can operate, prepare inputs for, and interpret outputs of a system or component
- Reliability - the ability of a system or component to perform its required functions under stated conditions for a specified period of time

  IE: Dependability, Robustness, Safety

  - Robustness IE: System has to be able to proceed even with bad inputs
- Performance - requirements are concerned with quantifiable attributes of the system

  IE: Response Time, Throughput, Availability and Accuracy
- Supportability - changes to the system after deployment

  IE: Adaptability, Maintainability

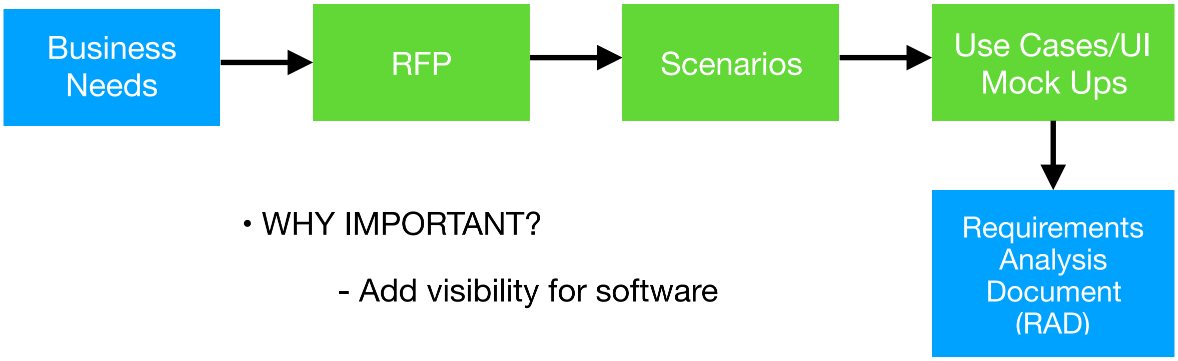


- Implementation Requirements - constraints on the implementation of the system, including the use of specific tools, programming languages, or hardware platforms
- Interface Requirements - are constraints imposed by external systems, including legacy systems and interchange formats
- Operations Requirements - Constraints on the administration and management of the system in the operational setting
- Packaging Requirements - constraint on the actual delivery of the system

IE: constraints on the installation media for setting up the software

- Legal Requirements - licensing, regulation, and certification issues.

After Use Cases: 🟩 = Requirements Elicitation Analysis



- WHY IMPORTANT?

  - Add visibility for software

  - Instead of seeing the construction of building, with software, client can criticizes the Scenarios or Use case before hand with DUE Dates

Design the Solution:

1. ID Objects of system
2. ID Classes of system
3. Decide Methods and Attributes of Classes of system
4. Class Diagrams
5. Sequence Diagrams
   - Use to show the processes of Use Cases

Build the Solution:

1. Build the Code

September 6, 2018

RAD (Requirement Analysis Document)

Introduction:

Purpose: ..

Goal:

- what software should do

Scope of the system:

- What could

Objective and Fulfillment Criteria:

IE: Add/Update/Delete User

Synopsis:

- describe the RAD in summary

Current System:

- Show what is currently implemented
- Manual input system (Often Microsoft Office)
    - IE: Excel spreadsheet

Proposed System:

Overview:

- Typically the outline of uses cases

Functional Requirement:

- English narrative of the flow of control

Non-Functional Requirement: (minimum of 3 relating to your use cases)

- Objective non-functional requirement

IE: System should be in English Language only

- You can see this!!!

IE: System has a “HELP” icon to guide user

- Performance is easy to be objective
- Error handling and Extreme Condition are objective

Documentations are non functional

System Maintenance

Security

Skip Scenario

User Cases:

- The first thing for uses cases flow of control must be actor

Mock UP:

What user interface should look like?

September 11, 2018

Identifying Candidate System Component

Part of Speech	Compound	Example
Proper Noun	Object	Richard
Common Noun	Class	Toy
Doing Verb	Method	Buy
Being Verb	Classification	is an
Heving Verb	Composition	has an
Stative Verb	Invariance	are owned
Model Verb	Data Semantics	must be
	Precondition	
	Postcondition	
Adjective	Attribute	unsuitable
Transitive Verb	Method	enter
Intransitive Verb	Exception Event	depend

API = public method of a class

3 Type of Classes

Entity - represent the objects in our application that we are modeling

- Normally Persistent (usually have a record of it, aka stored somewhere)

Boundary - represent the objects to connect to outside world

- Allow our system to interact with actors (both people and other software)
- We write these objects

  IE: UI class

Control - contain the code for executing the use cases.

- normally one-to-one relationship
  - between use case and control

September 13, 2018

List of Objects

Boundary Object - Interfaces to connect to outside world

Control Object - One to one

List of Methods (Public Methods)

Mandatory format

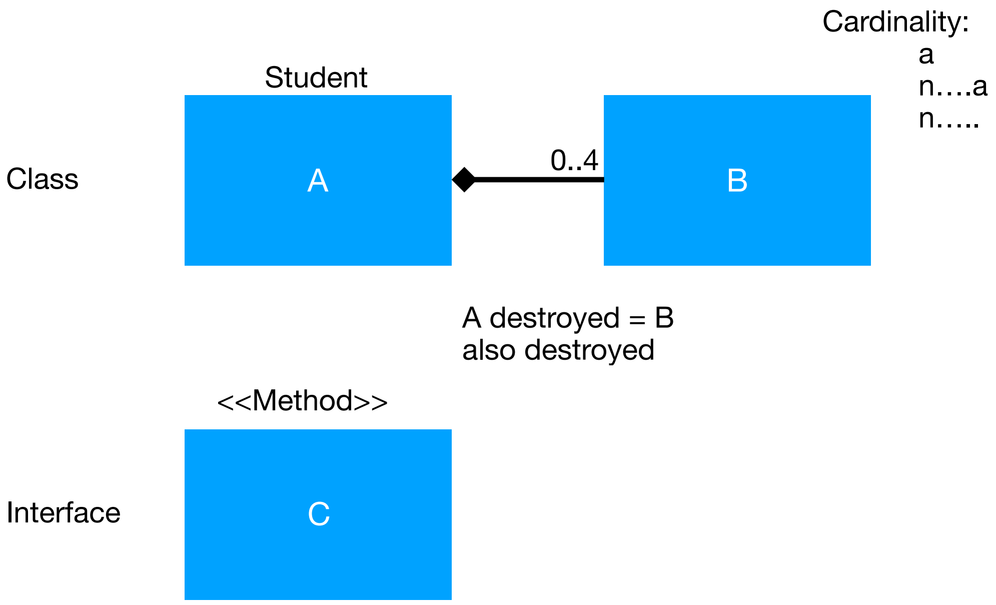
Follow the format of the document

Precondition: What is to be true prior for method to work

- Specific to this method

Postcondition

UML Being Used:



Sequence Diagram:

Follow Examples

We don’t parenthesis on action that is not done by method

- FIRST CALL MUST BY “CREATE”

September 18, 2018

Sequence Diagram
