UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Agile Method

IT 3105

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

Topics

1. Agile method
2. History of agile method
3. Agile Manifesto
4. Principles of agile methods
5. Extreme programming
6. Scrum
7. Advantages of scrum
8. Problems in agile development

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Agile Software Development

- Rapid software development processes are designed to produce useful software **quickly**.
- The software is not developed as a single unit but as a **series of increments**, with each increment including new system functionality.

History of agile method

1980 – 1990s	It was believed that the best way to achieve better software was through thorough planning
Feb. 2001	17 software developers met at the Snowbird, Utah resort, to discuss lightweight development methods. Agile Alliance was formed

The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Principles of agile methods

- Practice:
    - Customer involvement
    - Incremental delivery
    - People not process
    - Embrace change
    - Maintain simplicity

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

When to apply agile methods

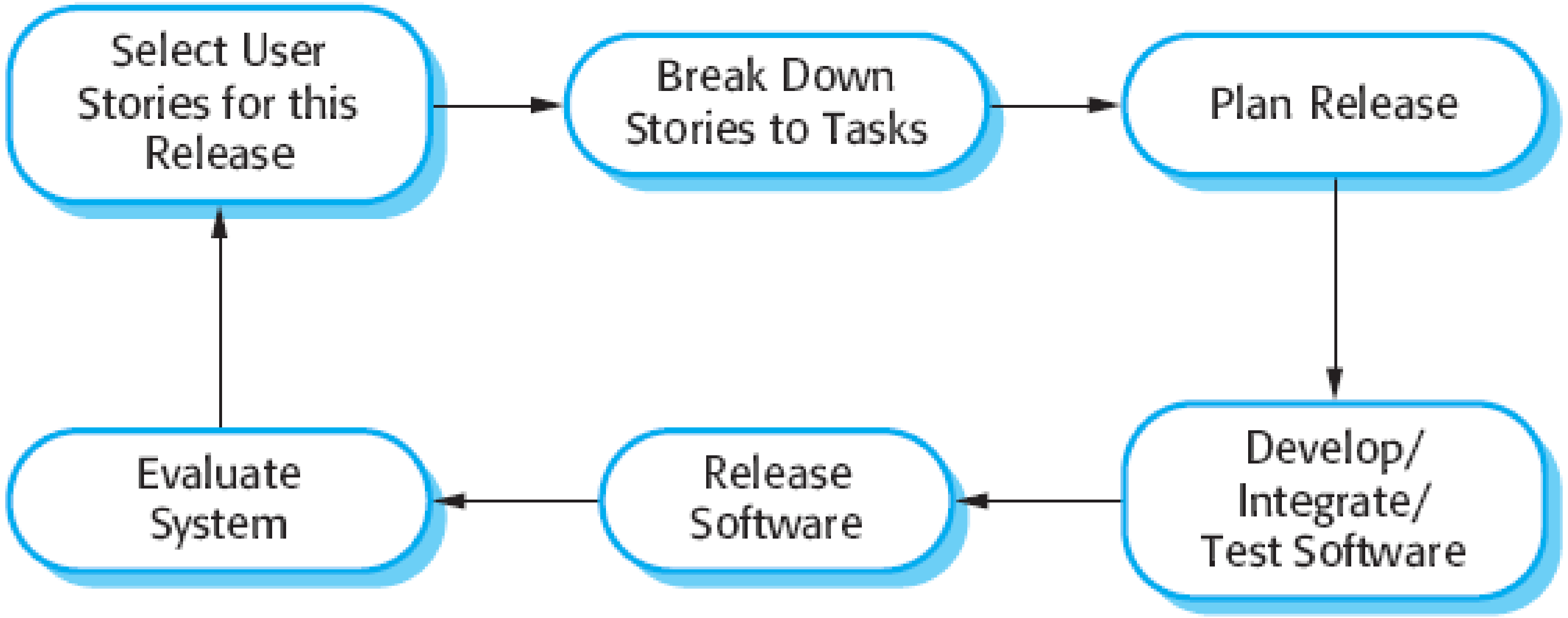
1. Product development where a software company is developing a small or medium-sized product for sale.

2. Custom system development within an organization.

UNIVERSITY *of* SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Extreme Programming

- The name was coined by Kent Beck (2000) because the approach was developed by pushing recognized good practice, such as iterative development, to ‘extreme’ levels.
- In extreme programming, requirements are expressed as **user stories**.
- Programmers **work in pairs** and develop tests for each task before writing the code.

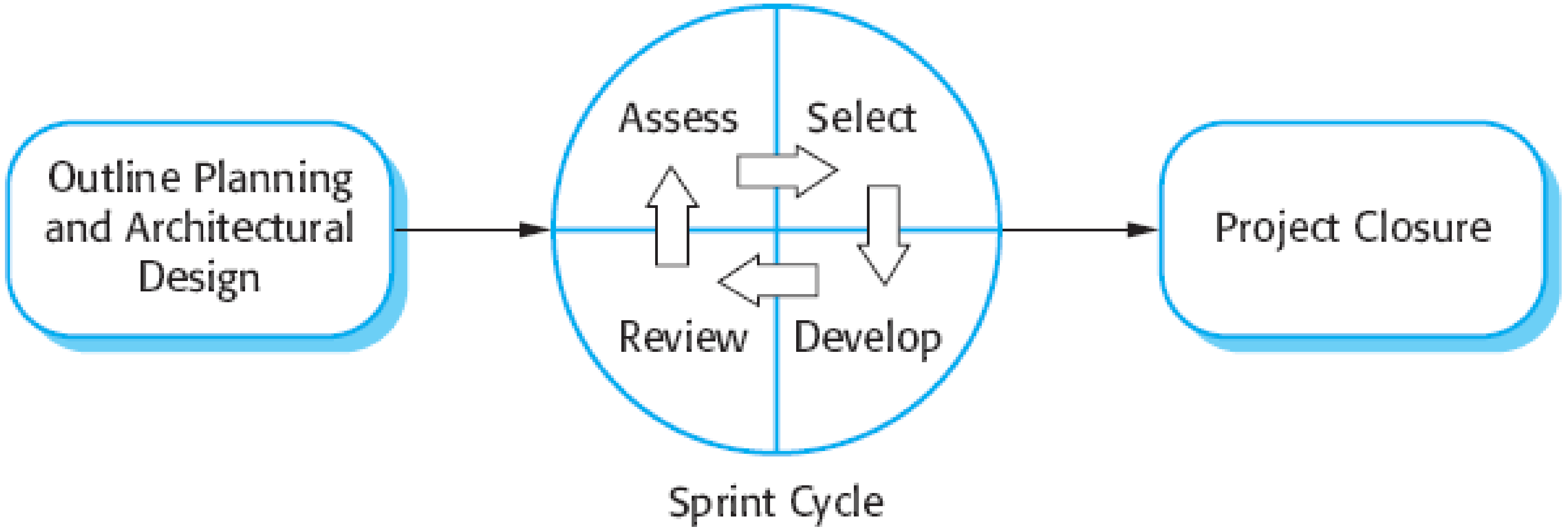

Select User Stories for this Release
Break Down Stories to Tasks
Plan Release
Develop/ Integrate/ Test Software
Release Software
Evaluate System

- Practice:
  - Incremental planning
  - Small releases
  - Simple design
  - Test-first development
  - Refactoring
  - Pair programming
  - Collective ownership
  - Continuous integration
  - Sustainable pace
  - On-site customer

Scrum Approach

- The Scrum approach is a general agile method, but its focus is on **managing iterative development** rather than specific technical approaches to agile software engineering.
- The '**Scrum master**' is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog, and communicates with customers and management outside of the team.
- Doing 'stand-up' meetings

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO


Outline Planning and Architectural Design
Assess
Select
Review
Develop
Sprint Cycle
Project Closure

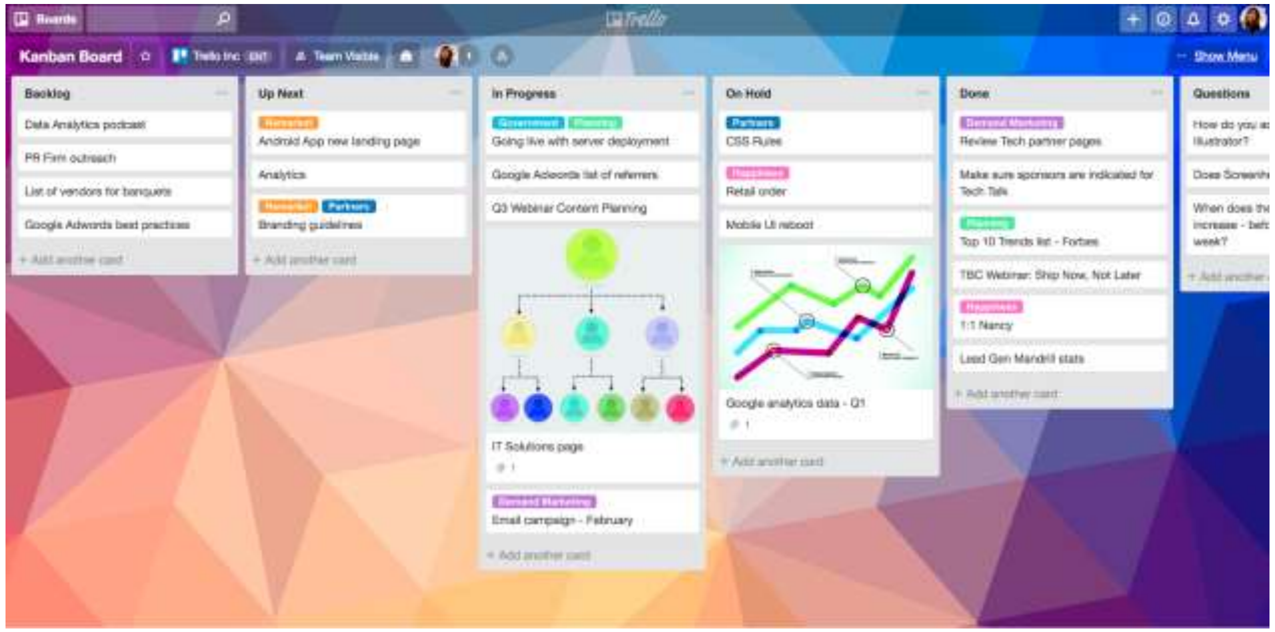
Scrum Process

1. Sprints are fixed length (2 – 4 weeks). They correspond to the development of a release of the system in XP.
2. The starting point for planning is the product backlog, which is the **list of work to be done** on the project. During the assessment phase of the sprint, this is reviewed, and priorities and risks are assigned.
3. The selection phase involves all of the project team who work with the customer to select the features and functionality to be developed during the sprint.

- Once these are agreed, the team organizes themselves to develop the software. Short daily meetings involving all team members are held to review progress and if necessary, reprioritize work.
- At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

Kanban

- A kanban board is an agile project management tool designed to help visualize work, limit work-in-progress, and maximize efficiency (or flow).
- Best example: Trello



Backlog
- Data Analytics podcast
- PR Firm outreach
- List of vendors for banquets
- Google Adwords best practices
- + Add another card

Up Next
- Android App new landing page
- Analytics
- Branding guidelines
- + Add another card

In Progress
- Going live with server deployment
- Google Adwords list of referrers
- Q3 Webinar Content Planning
- IT Solutions page
- Email campaign - February
- + Add another card

On Hold
- CSS Rules
- Retail order
- Mobile UI reboot
- Google analytics data - Q1
- + Add another card

Done
- Review Tech partner pages
- Make sure sponsors are indicated for Tech Talk
- Top 10 Trends list - Forbes
- TBC Webinar: Ship Now, Not Later
- 1:1 Nancy
- Lead Gen Mandrill stats
- + Add another card

Questions
- How do you a Illustrator?
- Does Screensh
- When does th increase - befo week?
- + Add another

Scrum Advantages

- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team has visibility of everything and consequently team communication is improved.

- Customers see on-time delivery of increments and gain feedback on how the product works.

- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

Agile Problems

- Success depends on **having a customer who is willing and able to spend time with the development team** and who can represent all system stakeholders.
- Individual team members may not have **suitable personalities** for the intense involvement that is typical of agile methods.
- **Prioritizing changes** can be extremely difficult, especially in systems for which there are many stakeholders. Typically, each stakeholder gives different priorities to different changes.

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

- Maintaining simplicity requires **extra work**.
- Many organizations, especially large companies, have spent **years changing their culture** so that processes are defined and followed.

Plan-driven vs Agile Method

- Is it important to have a **very detailed specification and design** before moving to implementation?
- Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic?
- How **large is the system** that is being developed?
- **What type of system** is being developed?
- What is the expected system **lifetime**?

- What **technologies** are available to support system development?
- How is the development **team organized**?
- Are there **cultural issues** that may affect the system development?
- How **good are the designers and programmers** in the development team?

Agile Tools

- Trello: https://trello.com/en

Design Tools

- WireMock – WireMock
- Figma: the collaborative interface design tool.
- Intelligent Diagramming | Lucidchart

Development Tools

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Requirements

IT 3105

Topics

1. Requirements engineering
2. Levels of Requirements
3. Classification of Requirements
4. Requirements Elicitation
5. SRS
6. Requirements Validation

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Requirements engineering

- The process of finding out, analyzing, documenting and checking the services and constraints needed for a project

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Levels of Requirements

- **User requirements** are statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and the constraints under which it must operate.

- **System requirements** are more detailed descriptions of the software system's functions and operational constraints.

Classification of Requirements

- **Functional requirements.** These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations.

- **Non-functional requirements.** These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards.

Functional requirement

- Sample:
  - *A user shall be able to search the appointments lists for all clinics.*
  - *The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.*
  - *Each staff member using the system shall be uniquely identified by his or her eight-digit employee number.*

- In principle, the functional requirements specification of a system should be both complete and consistent.

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Non-functional Requirement

- Non-functional requirements, as the name suggests, are requirements that are not directly concerned with the specific services delivered by the system to its users.

- NFR's are further subdivided into two: **Technical NFR**s and **Non-Technical NFR**s.

Technical NFRs Sample

- Maintainability
- Portability
- Reliability
- Scalability
- Performance
- Usability
- Operational
- Robustness
- Safety
- Security

Non-Technical NFR Sample

- Cultural and Political
- Legal and Regulatory
- Conformity to Standards

Requirements Engineering Process

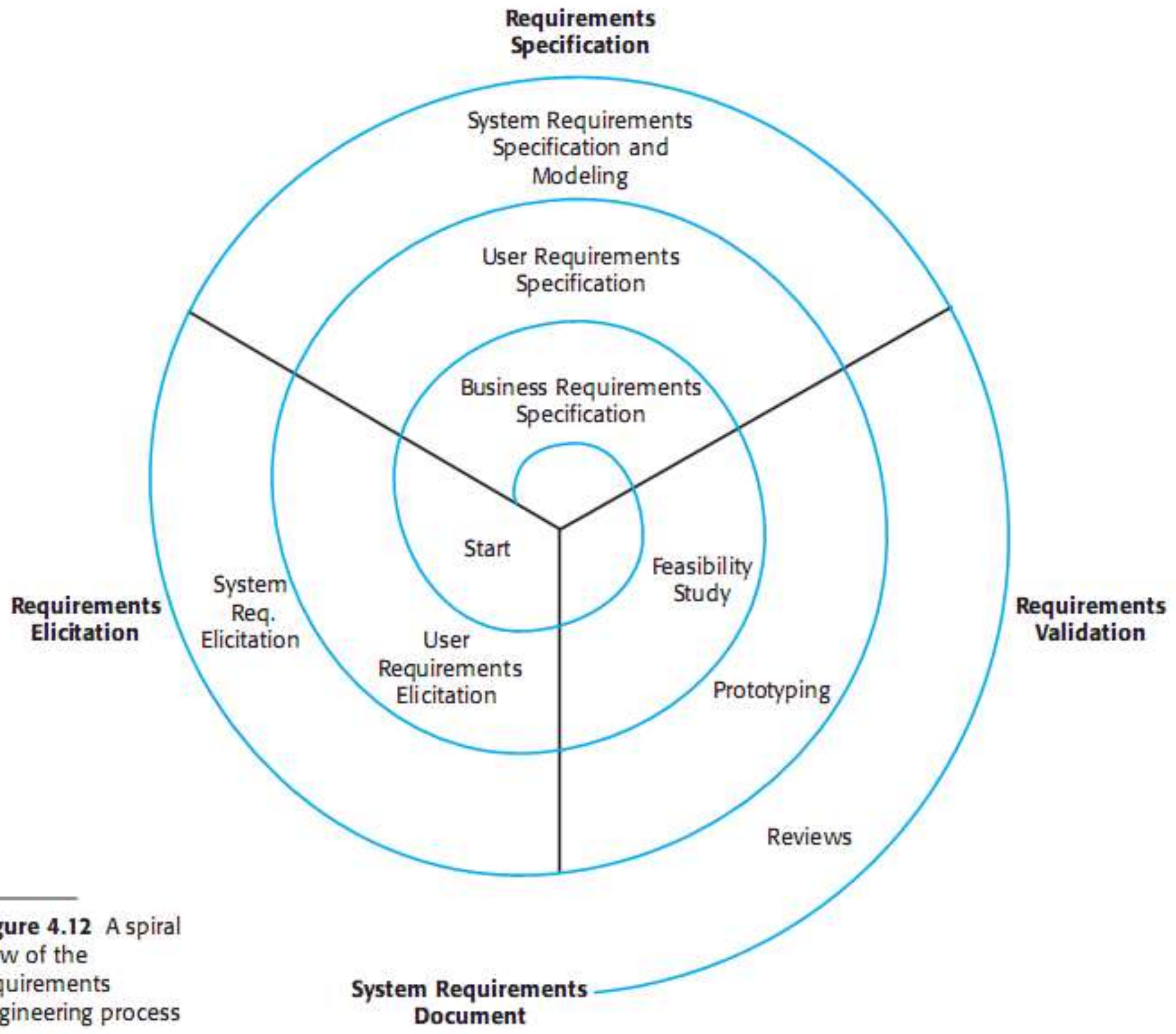
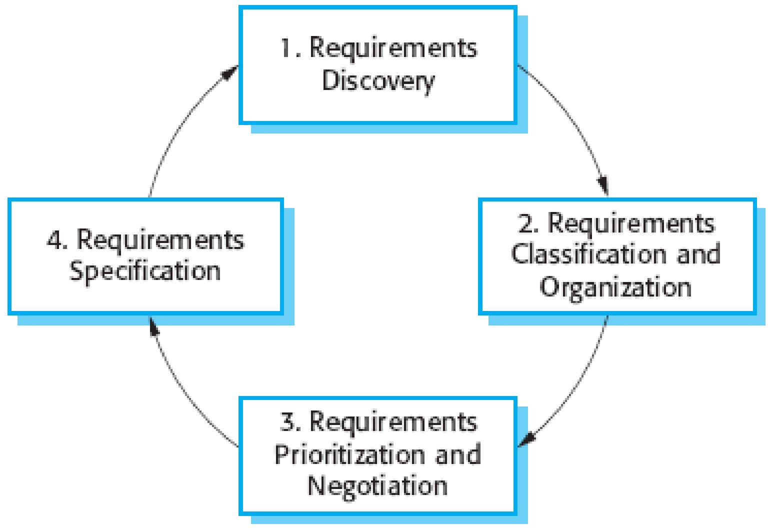

Requirements Specification
System Requirements Specification and Modeling
User Requirements Specification
Business Requirements Specification
Start
Feasibility Study
Prototyping
Reviews
Requirements Validation
Requirements Elicitation
System Req. Elicitation
User Requirements Elicitation
System Requirements Document

Figure 4.12 A spiral view of the requirements engineering process

Requirements Elicitation and Analysis


1. Requirements Discovery
2. Requirements Classification and Organization
3. Requirements Prioritization and Negotiation
4. Requirements Specification

Requirements Elicitation Techniques

- Interview
- Questionnaire
- Ethnography
- Scenario
- Use Case
- Prototype

Interview

- **Closed interviews**, where the stakeholder answers a pre-defined set of questions.

- **Open interviews**, in which there is no pre-defined agenda. The requirements engineering team explores a range of issues with system stakeholders and hence develop a better understanding of their needs.

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Questionnaire

- The questionnaire technique is used at the beginning of the elicitation process to poll the participants and stakeholders and to obtain their thoughts on issues related to the system.

Ethnography

- Ethnography is an **observational technique** that can be used to understand operational processes and help derive support requirements for these processes. An analyst immerses himself or herself in the working environment where the system will be used.

Scenario

- A description of what the system and users expects when the scenario starts.
- A description of the normal flow of events in the scenario.
- A description of what can go wrong and how this is handled.
- Information about other activities that might be going on at the same time.
- A description of the system state when the scenario finishes.

Use Case

- Use case identifies the actors involved in an interaction and names the type of interaction. This is then supplemented by additional information describing the interaction with the system.

- **Actors** in the process, who may be human or other systems, are represented as **stick figures**.
- Each class of **interaction** is represented as a named **ellipse**.
- **Lines link** the actors with the interaction. Optionally, arrowheads may be added to lines to show how the interaction is initiated.

Prototype

- Prototyping graphical user interface is considered an effective tool for eliciting and validating requirements.

- A prototype should show the main functionalities and should be easy to develop and modify. The prototype must be evaluated and modified interactively with the users' input.

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Software Requirements Specifications (SRS)

- The SRS will serve as a contract between the proponent(s)/stakeholder(s) and project. This part of the document will describe what the software will do (functional) as well as what it is not expected to do (non-functional requirements). This portion of the document discusses in detailed manner the minimum requirements that may not be possibly covered in the Scope and Limitation section of a project document

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

- The requirements document has a diverse set of users, ranging from the senior management of the organization that is paying for the system to the engineers responsible for developing the software.

Ways of writing a system requirements specification

- **Natural language sentences** - The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
- **Structured natural language** - The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
- **Graphical notations** - graphical models, supplemented by text annotations, are used to define the functional requirements for the system.

Users of a requirements document

- System Engineers
- System Test Engineers
- System Maintenance Engineers

Requirements validation checks

1. **Validity checks** - a user may think that a system is needed to perform certain functions.

2. **Consistency checks** - requirements in the document should not conflict.

3. **Completeness checks** - the requirements document should include requirements that define all functions and the constraints intended by the system user.

4. **Realism checks** - using knowledge of existing technology, the requirements should be checked to ensure that they can actually be implemented.

5. **Verifiability checks** - to reduce the potential for dispute between customer and contractor, system requirements should always be written so that they are verifiable.

Requirements validation techniques

1. **Requirements reviews** - The requirements are analyzed systematically by a team of **reviewers** who check for errors and inconsistencies.
2. **Prototyping** - In this approach to validation, an executable model of the system in question is demonstrated to end-users and customers. They can experiment with this model to see if it meets their real needs.
3. **Test-case generation** - Developing tests from the user requirements before any code is written is an integral part of extreme programming.

Design Principles

I. Introduction

We will concentrate on how design practice addresses the critical feature of an interactive system – usability from the human perspective. Interaction design is about creating interventions in often complex situations using technology of many kinds including PC software, the web, and physical devices. In this chapter we will think about interaction design. Note that we are not just thinking about the design of interactive systems, but about the interaction itself. An office has just got a new electric stapler. It is connected to the mains electricity and is hard to move around, so when you want to staple papers together you go to the stapler. In the past when someone wanted to staple things, they would take the stapler to their desk and keep it until someone else wanted it. You might write a letter, print it, staple it, write the next letter, staple it, and so on. Now you must take the letters to be stapled across the office, so instead you write–print, write–print until you have a pile of things to staple and then take them across. The stapler influences the whole pattern of interaction.

What is Design?

So what is design? A simple definition is:

"Achieving goals within constraints."

- **Goals** - What is the purpose of the design we are intending to produce? Who is it for? Why do they want it?
- **Constraints -** What materials must we use? What standards must we adopt? How much can it cost? How much time do we have to develop it? Are there health and safety issues?
- **Trade-off** - Choosing which goals or constraints can be relaxed so that others can be met.

Often the most exciting moments in design are when you get a radically different idea that allows you to satisfy several apparently contradictory constraints. However, the more common skill needed in design is to accept the conflict and choose the most appropriate trade-off. The temptation is to focus on one or other goal and optimize for this, then tweak the design to make it just satisfy the constraints and other goals

The golden rule of design

"Understand your materials."

Part of the understanding we need is about the circumstances and context of the particular design problem. However, there are also more generic concepts to understand. The designs we produce may be different, but often the raw materials are the same.

For Human–Computer Interaction the obvious materials are the human and the computer. That is we must:

Understand people
psychological, social aspects, human error.

Understand computers
limitations, capacities, tools, platforms

We must understand the fundamental materials of human–computer interaction in order to design.

To Err is human

It might sound demeaning to regard people as 'materials', possibly even dehumanizing. In fact, the opposite is the case: physical materials are treated better in most designs than people. This is particularly obvious when it comes to failures.

People make mistakes. This is not 'human error', an excuse to hide behind in accident reports, it is human nature. We are not infallible consistent creatures, but often make slips, errors, and omissions. A concrete lintel breaks and a building collapses. Do the headlines read 'lintel error'? No. It is the nature of concrete lintels to break if they are put under stress and it is the responsibility of architect and engineer to ensure that a building only puts acceptable stress on the lintel.

Similarly, it is the nature of humans to make mistakes, and systems should be designed to reduce the likelihood of those mistakes and to minimize the consequences when mistakes happen.

The central message – the user

This is the core of interaction design: *put the user first, keep the user in the center and remember the user at the end.*

Process of design

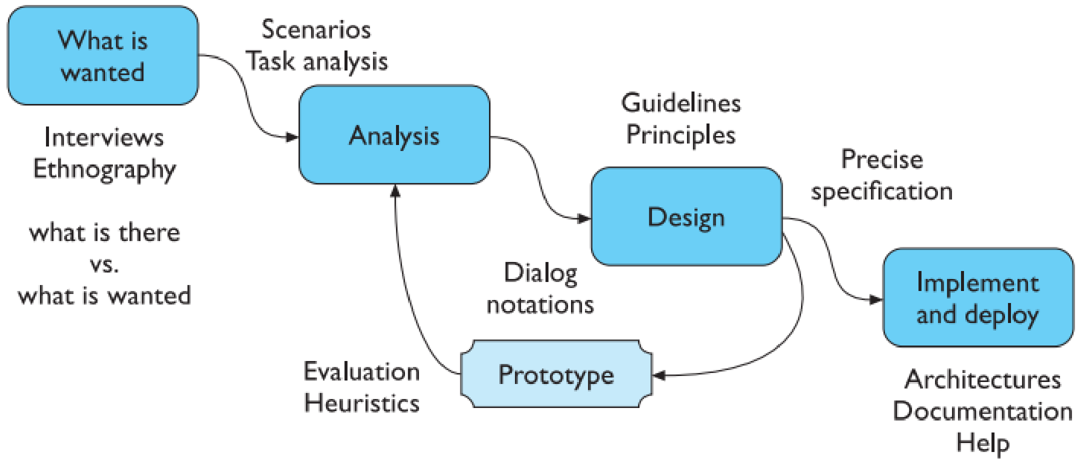


Figure 5.1 Interaction design process

Often HCI professionals complain that they are called in too late. A system has been designed and built, and only when it proves unusable do, they think to ask how to do it right! In other companies' usability is seen as equivalent to testing – checking whether people can use it and fixing problems, rather than making sure they can from the beginning. In the best companies, however, usability is designed in from the start.

- **Requirements** – what is wanted. The first stage is establishing what exactly is needed. As a precursor to this it is usually necessary to find out what is currently happening. For example, how do people currently watch movies? What sort of personal appliances do they currently use? There are several techniques used for this in HCI: interviewing people, videotaping them, looking at the documents and objects that they work with, observing them directly

- **Analysis** - The results of observation and interview need to be ordered in some way to bring out key issues and communicate with later stages of design.

- **Design** - Well, this is all about design, but there is a central stage when you move from what you want, to how to do it. There are numerous rules, guidelines and design principles that can be used to help with this
- **Iteration and prototyping** - Humans are complex and we cannot expect to get designs right first time. We therefore need to evaluate a design to see how well it is working and where there can be improvements.

User Focus

"Know your users."

As we've already said, the start of any interaction design exercise must be the intended user or users.

Over time many people are affected directly or indirectly by a system and these people are called *stakeholders*. Obviously, tracing the tenuous links between people could go on forever and you need to draw boundaries as to whom you should consider. This depends very much on the nature of the systems being designed, but largely requires plain common sense.

So, how do you get to know your users?

- Who are they?
    - Of course, the first thing to find out is who your users are. Are they young or old, experienced computer users or novices?
    - it may not be obvious who the users are, so you may need to ask this question again as you find out more about the system and its context.

- Probably not like you!
    - When designing a system it is easy to design it as if you were the main user: you assume your own interests and abilities.
    - So often you hear a designer say 'but it's obvious what to do'. It may be obvious for her.

- Talk to them.
    - It is hard to get yourself inside someone else's head, so the best thing is usually to ask them.
    - This can take many forms: structured interviews about their job or life, open-ended discussions, or bringing the potential users fully into the design process.

Global Structure: hierarchical organization

We will now look at the overall structure of an application. This is the way the various screens, pages or device states link to one another. One way to organize a system is in some form of hierarchy.

This is typically organized along functional boundaries (that is, different kinds of things), but may be organized by roles, user type, or some more esoteric breakdown such as modules in an educational system.

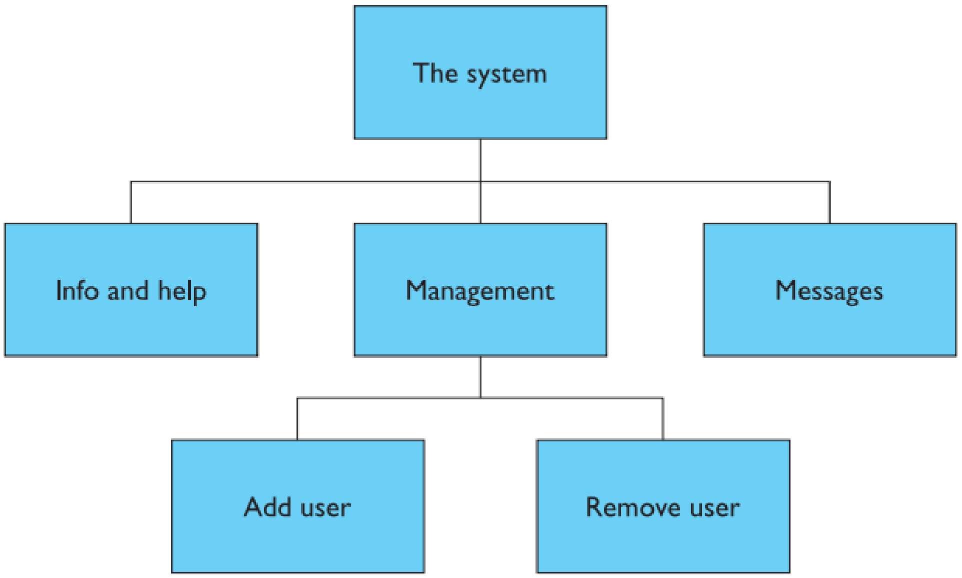


Figure 5.6 Application functional hierarchy

The hierarchy links screens, pages or states in logical groupings. For example, Figure 5.6 gives a high-level breakdown of some sort of messaging system.

Project Wireframe

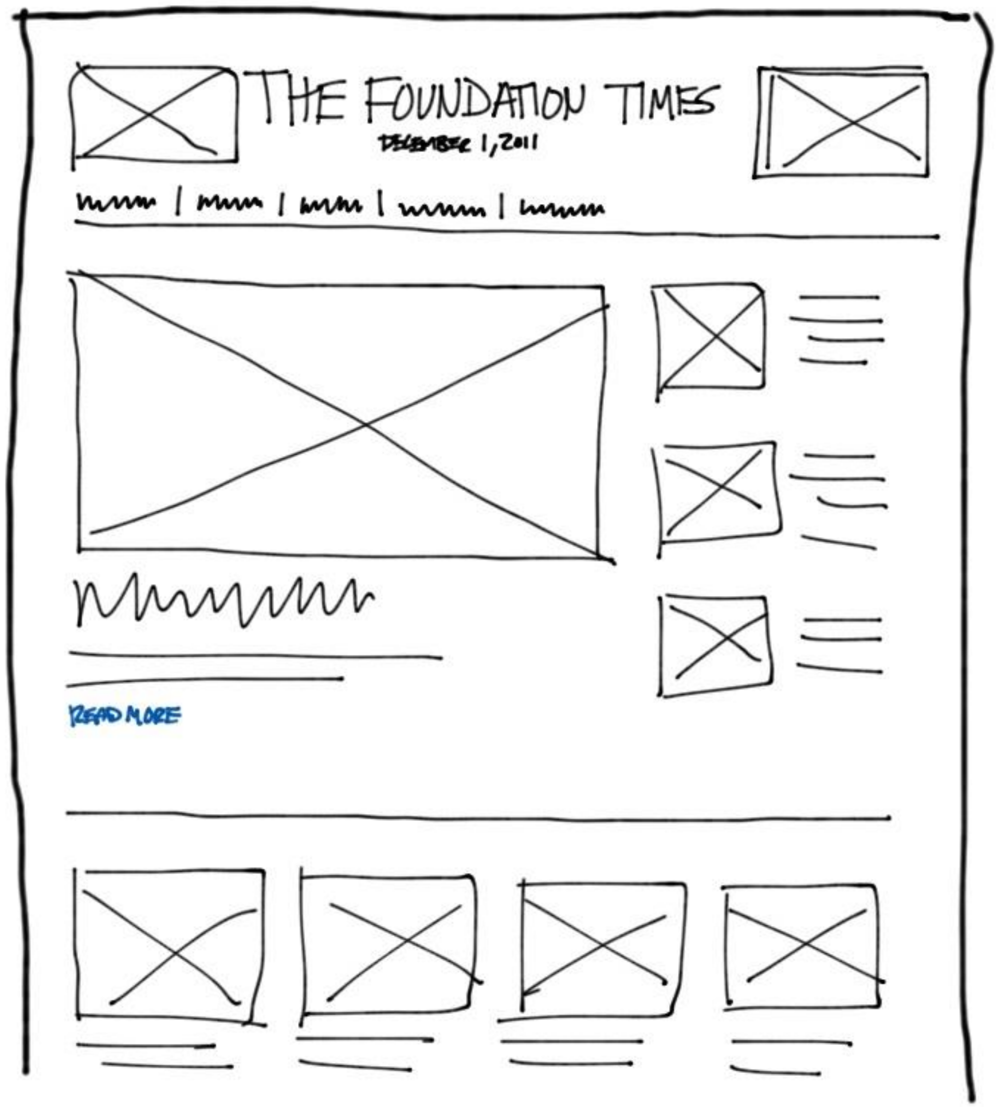
ANGIE M. CENIZA-CANILLO, PHD

DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS, SCHOOL OF ARTS AND SCIENCES, UNIVERSITY OF SAN CARLOS

1. SKETCH > WIREFRAME > VISUAL > CODE
2. SKETCH > WIREFRAME > HI-DEF WIREFRAME > VISUAL > CODE
3. WIFEFRAME > HI-DEF WIREFRAME > VISUAL > CODE
4. SKETCH > CODE
5. WIFEFRAME > INTERACTIVE PROTOTYPE > VISUAL > CODE

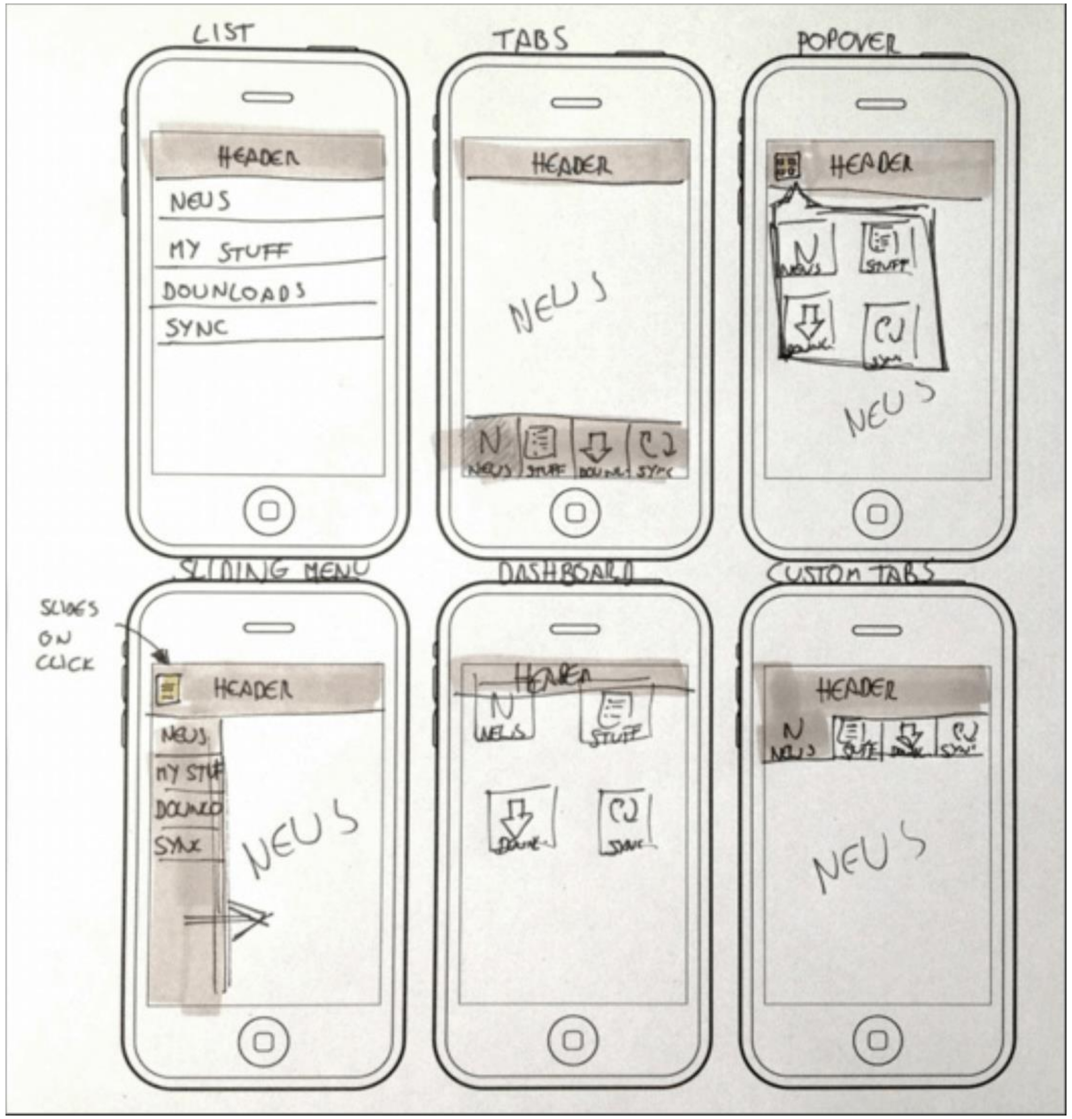
What to do today..

SKETCH a **wireframe** of your system.

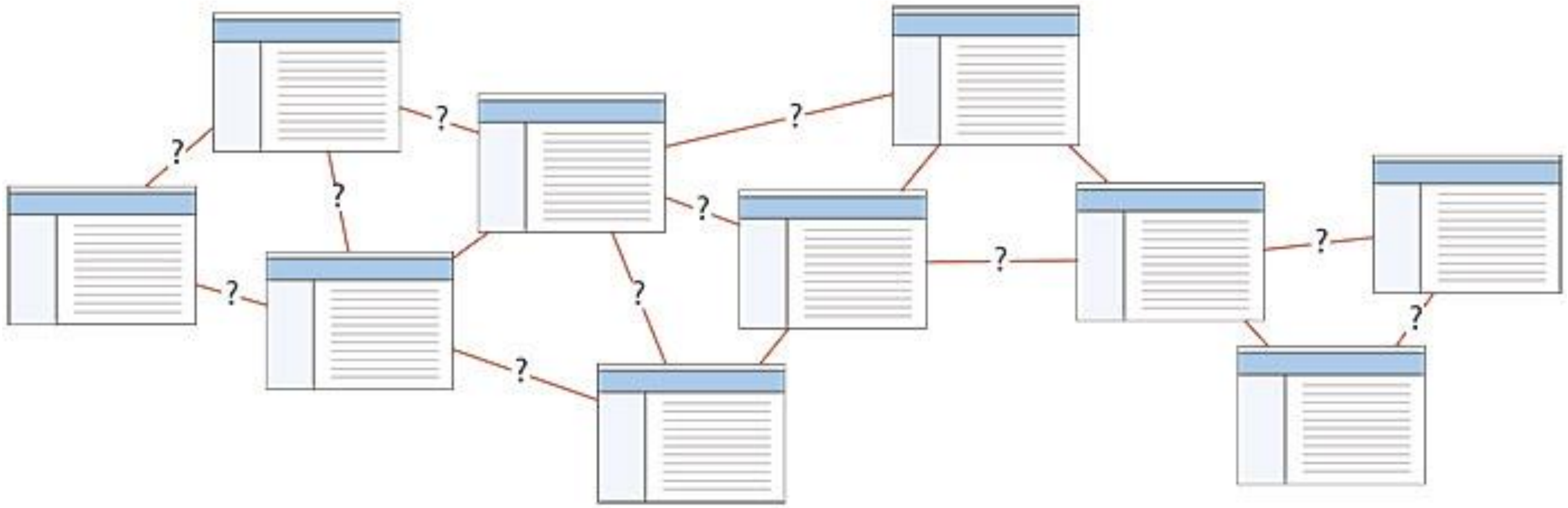


WIREFRAME

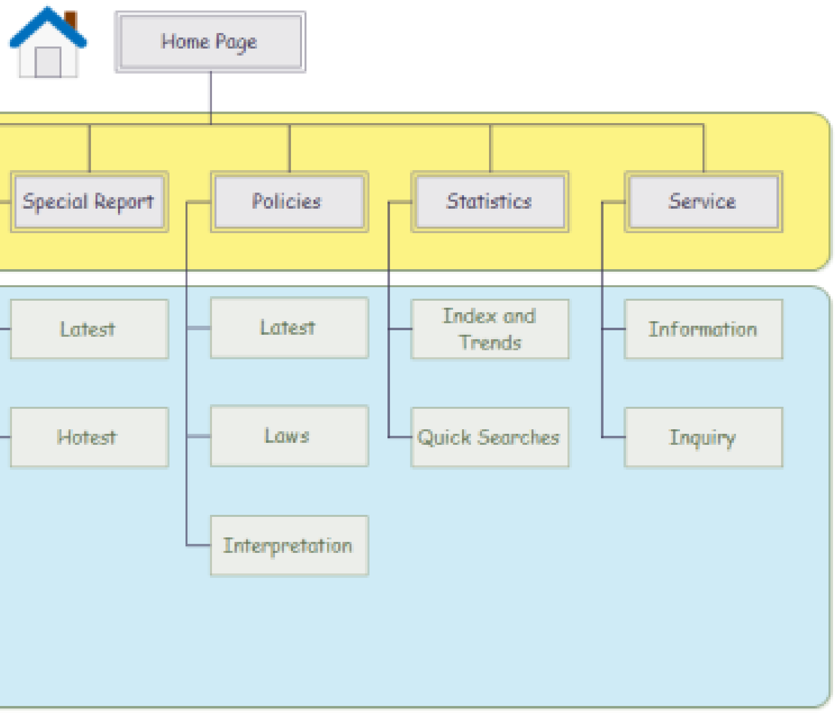
"An image or set of images which displays the functional elements of a website or page, typically used for planning a site's structure and functionality."



So what first?



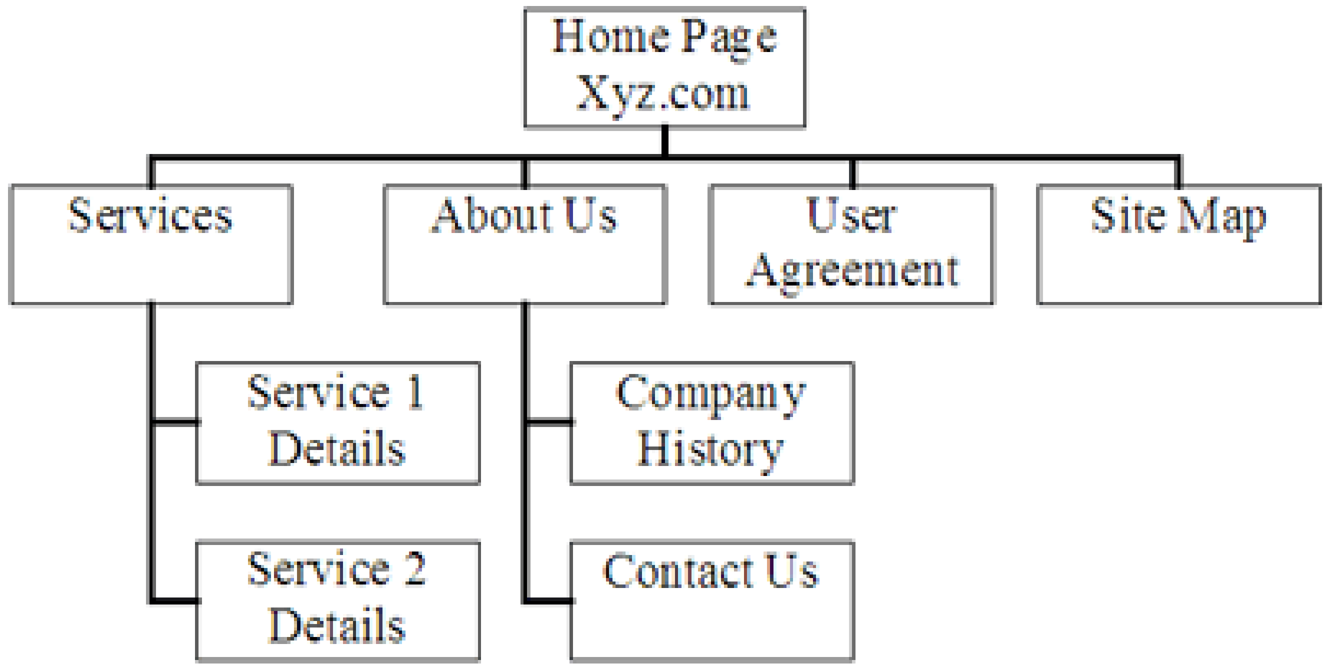
Structure and Scope!



Structure and Scope!

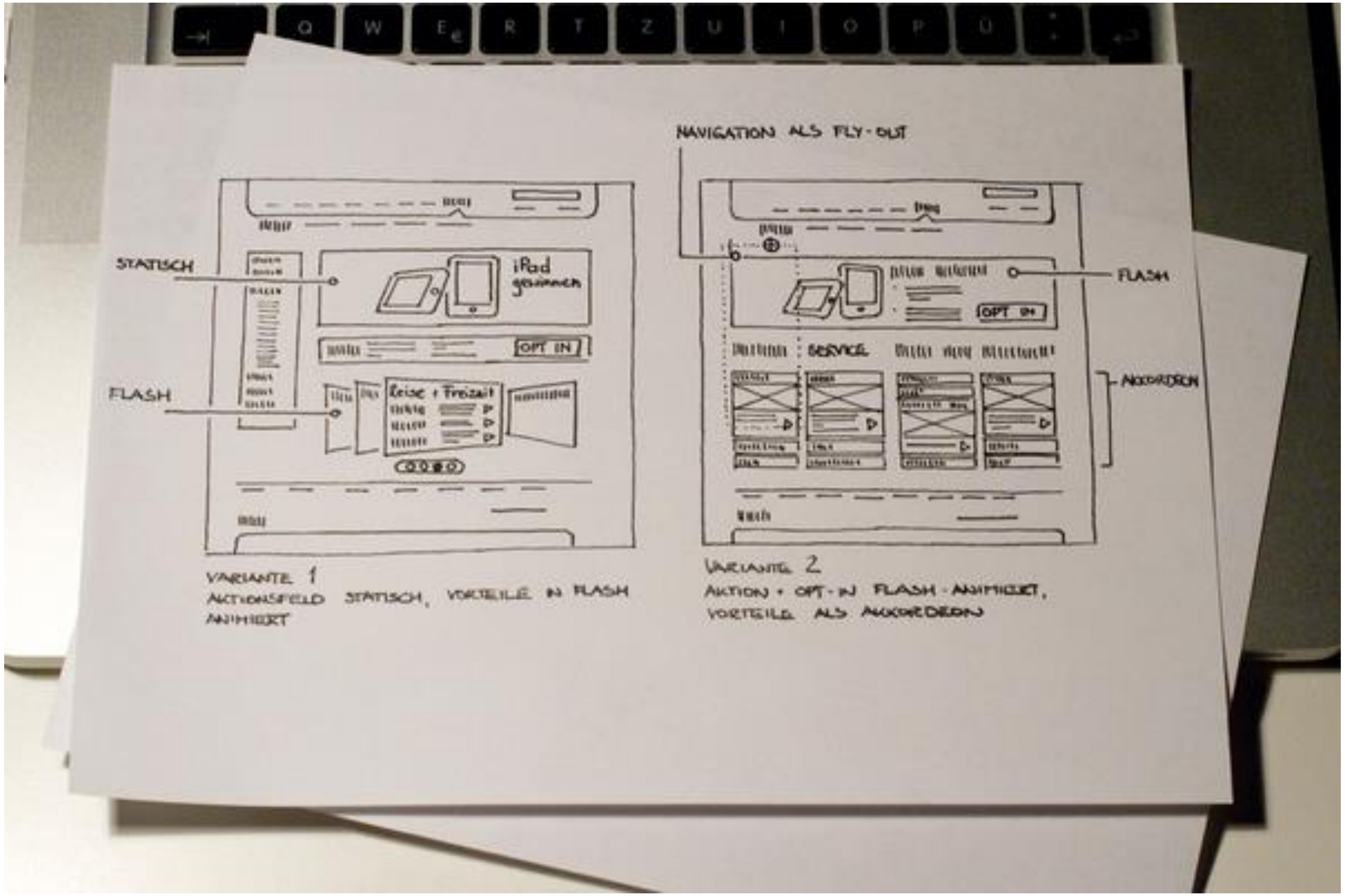
Think about your application's structure.

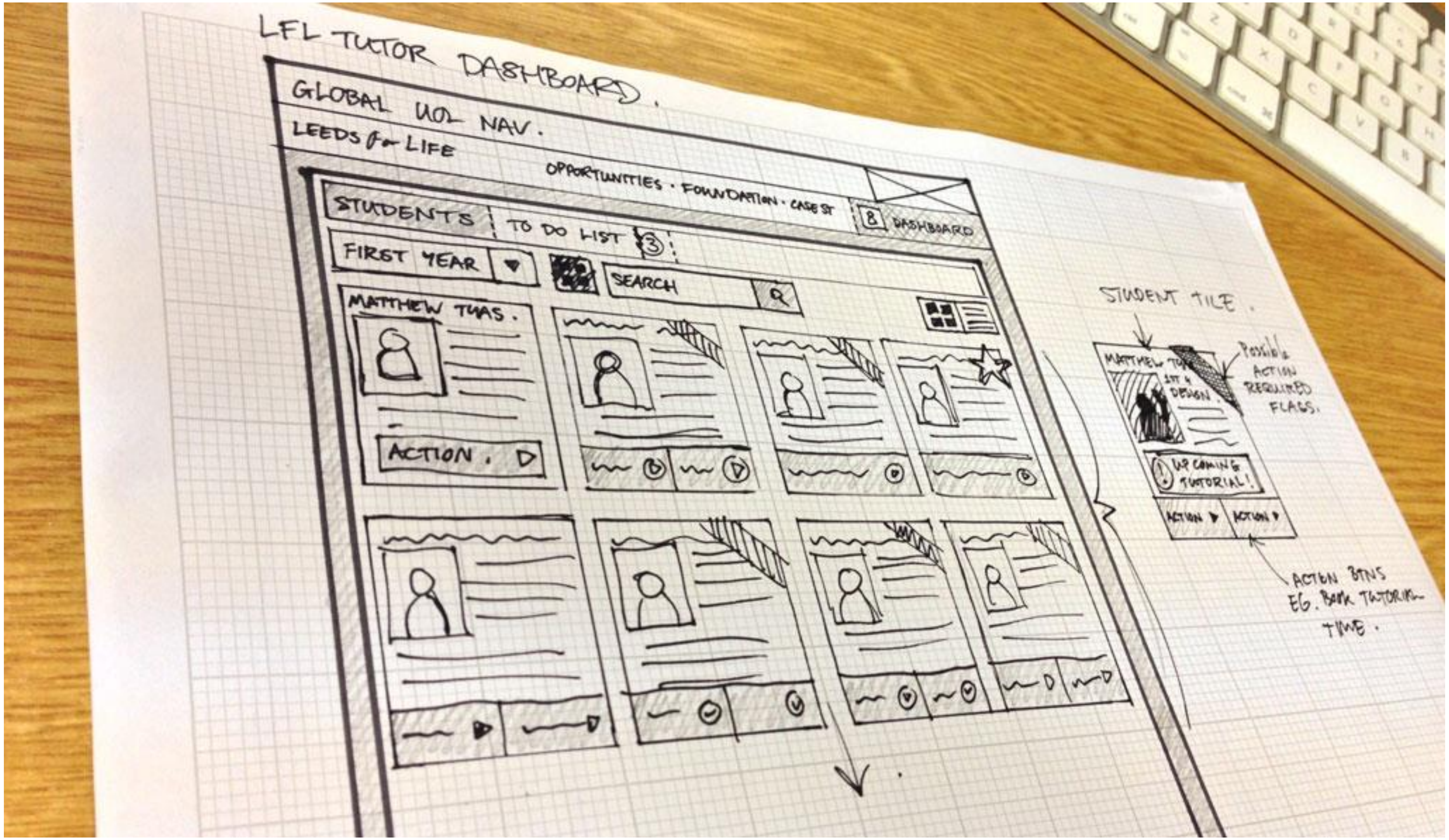
- What pages/screens should there be?
- Aim for a complete structure.
- Think of all possible pages

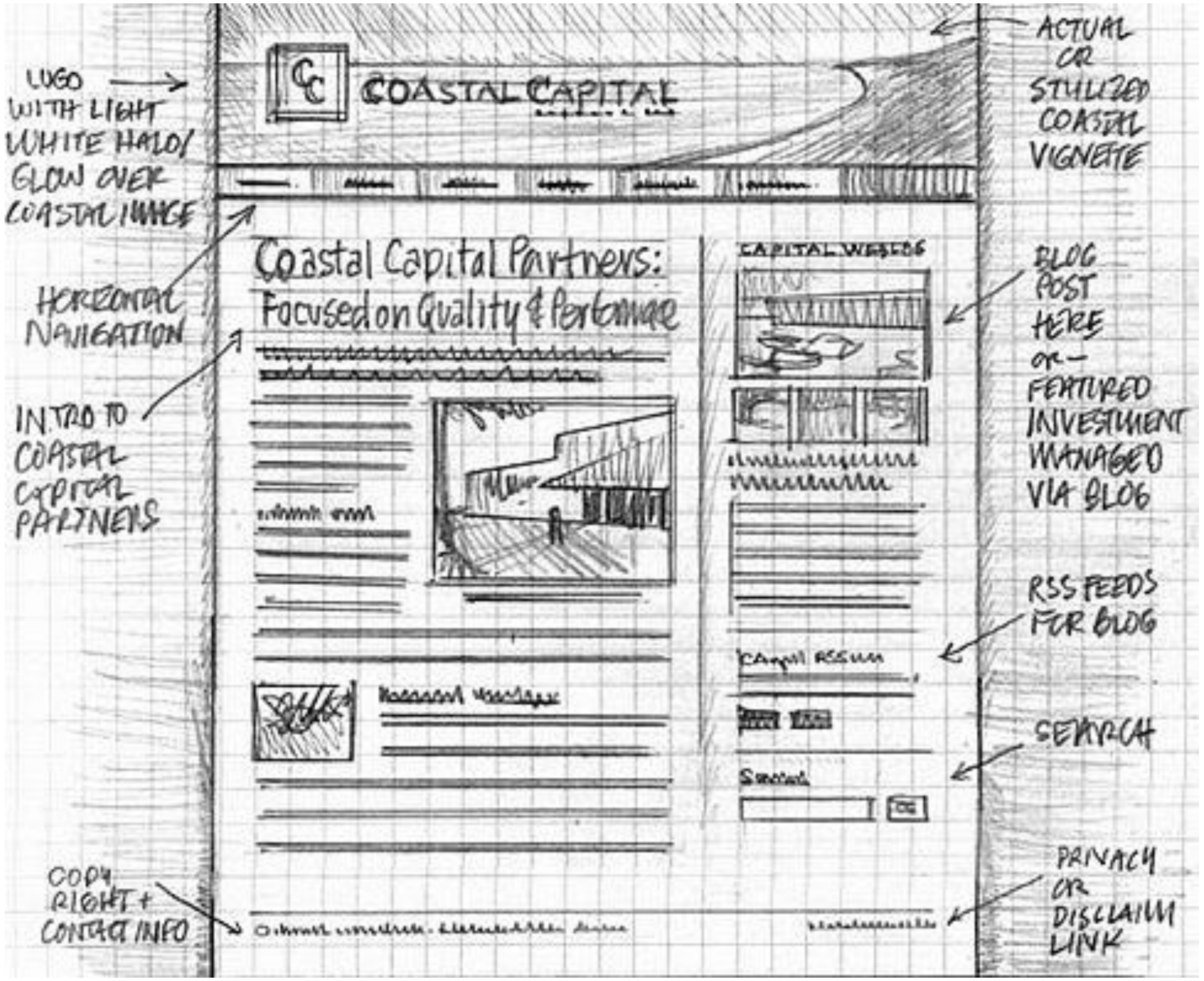


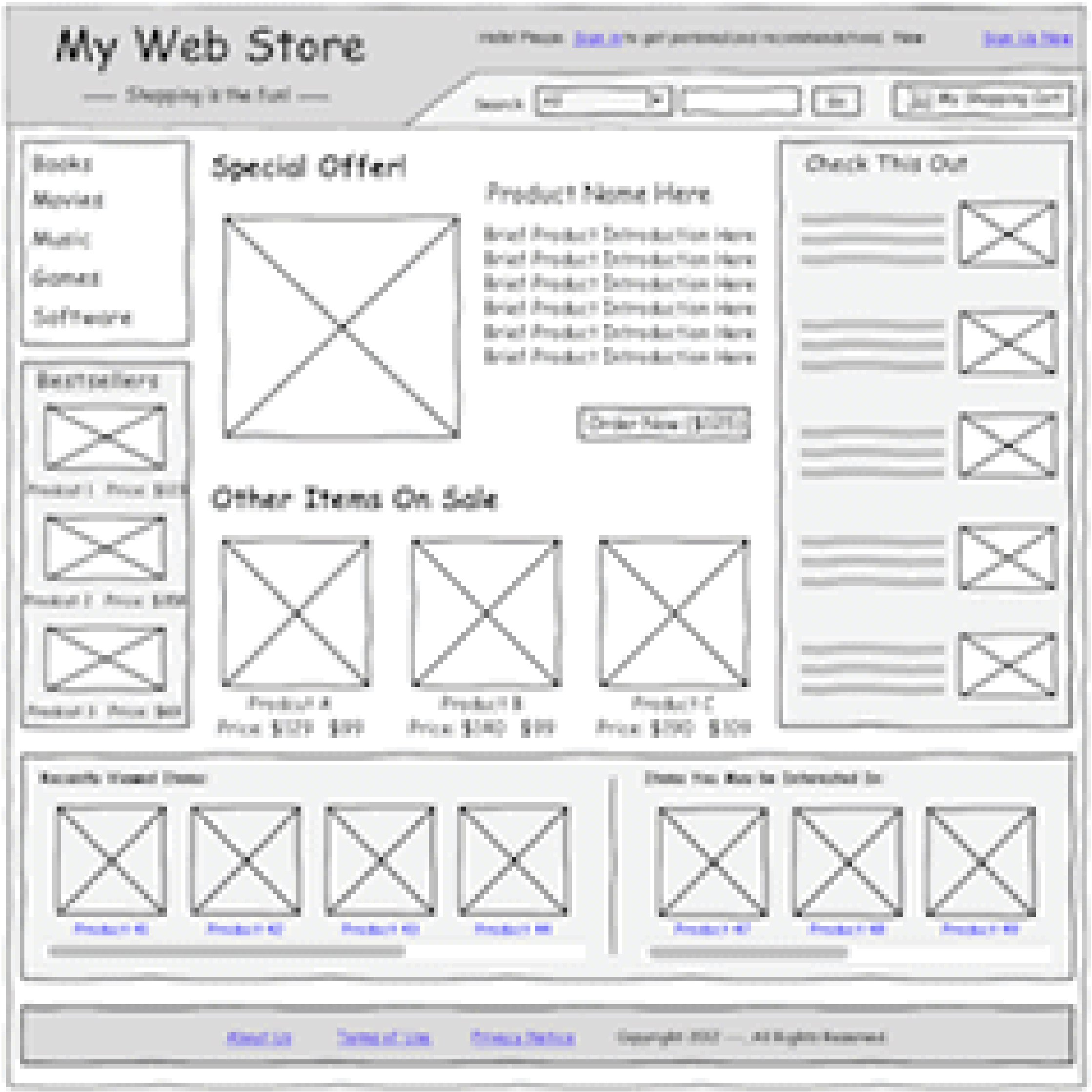
Next?

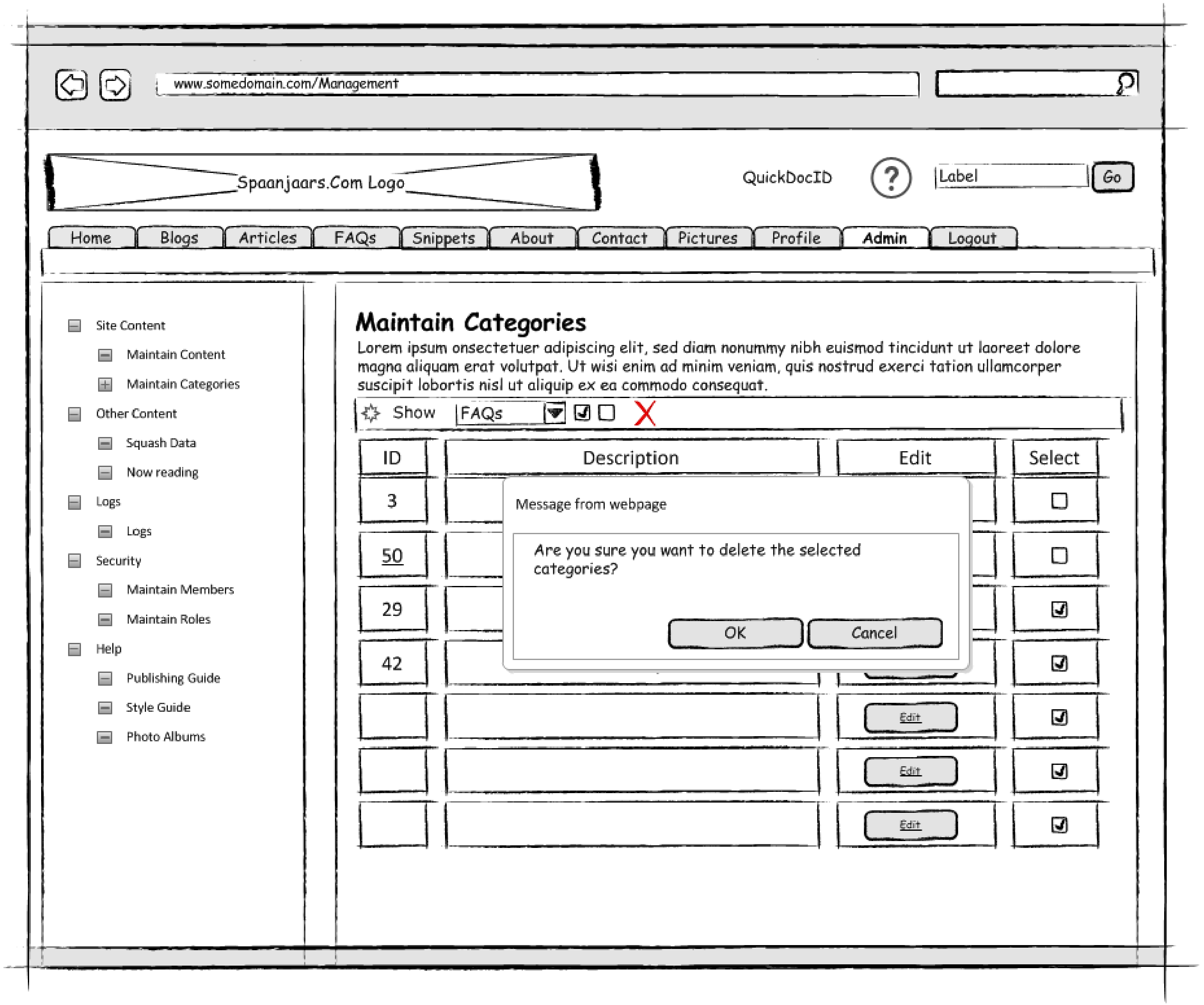
SKETCH IT!


NAVIGATION ALS FLY-OUT
STATISCH
iPad gewinnen
OPT IN
FLASH
Reise + Freizeit
FLASH
OPT IN
SERVICE
AKKORDEON
VARIANTE 1
AKTIONSFELD STATISCH, VORTEILE IN FLASH ANIMIERT
VARIANTE 2
AKTION + OPT-IN FLASH-ANIMIERT,
VORTEILE ALS AKKORDEON


LFL TUTOR DASHBOARD.
GLOBAL UOL NAV.
LEEDS for LIFE
OPPORTUNITIES · FOUNDATION · CASE ST
8 DASHBOARD
STUDENTS
TO DO LIST 3
FIRST YEAR
SEARCH
MATTHEW TUAS.
ACTION.
STUDENT TILE.
MATTHEW T
1ST A DESIGN.
Possible ACTION REQUIRED FLAGS.
UPCOMING TUTORIAL!
ACTION ▶
ACTION ▶
ACTION BTNS EG. BOOK TUTORIAL TIME.


LOGO WITH LIGHT WHITE HALO/ GLOW OVER COASTAL IMAGE
COASTAL CAPITAL
ACTUAL OR STYLIZED COASTAL VIGNETTE
HORIZONTAL NAVIGATION
Coastal Capital Partners: Focused on Quality & Performance
CAPITAL WEBLOG
BLOG POST HERE OR — FEATURED INVESTMENT MANAGED VIA BLOG
INTRO TO COASTAL CAPITAL PARTNERS
RSS FEEDS FOR BLOG
SEARCH
Search
GO
COPYRIGHT + CONTACT INFO
PRIVACY OR DISCLAIM LINK


My Web Store
Shopping is the Fun!
Sign Up Now
Search
Go
My Shopping Cart
Books
Movies
Music
Games
Software
Bestsellers
Special Offer!
Product Name Here
Brief Product Introduction Here
Brief Product Introduction Here
Brief Product Introduction Here
Brief Product Introduction Here
Brief Product Introduction Here
Brief Product Introduction Here
Other Items On Sale
Product A
Product B
Product C
Check This Out
Recently Viewed Items
Items You May be Interested In
About Us
Terms of Use
Privacy Notice



Maintain Categories

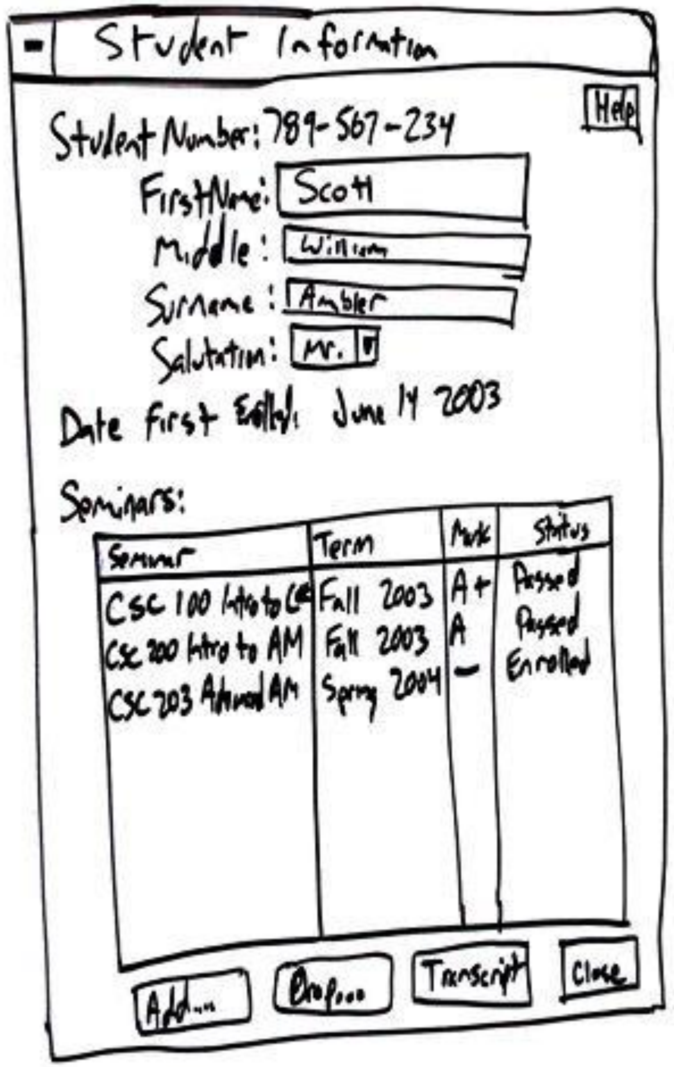
Lorem ipsum onsectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

ID	Description	Edit	Select
3			☐
50			☐
29			☑
42			☑
		Edit	☑
		Edit	☑
		Edit	☑

Message from webpage

Are you sure you want to delete the selected categories?

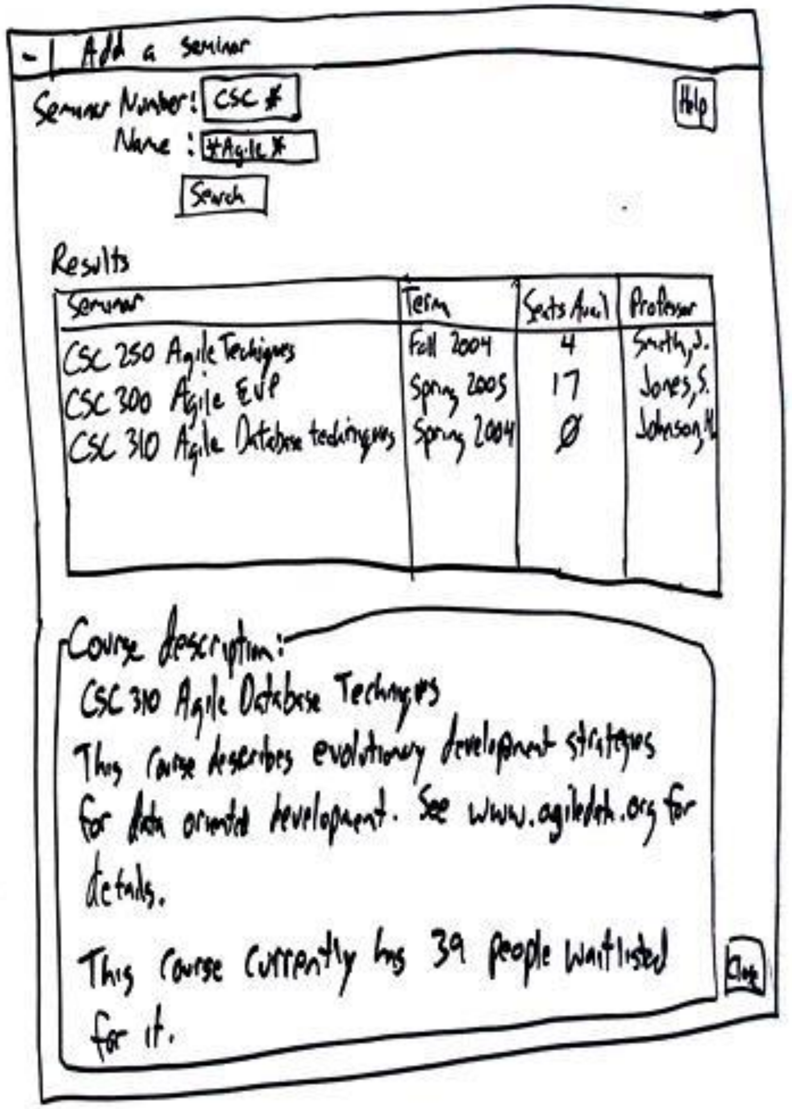
OK Cancel



Student Number: 789-567-234
FirstName: Scott
Middle: William
Surname: Ambler
Salutation: Mr.
Date First Enrolled: June 14 2003

Seminars:

Seminar	Term	Mark	Status
CSC 100 Intro to CS	Fall 2003	A+	Passed
CSC 200 Intro to AM	Fall 2003	A	Passed
CSC 203 Advanced AM	Spring 2004	—	Enrolled

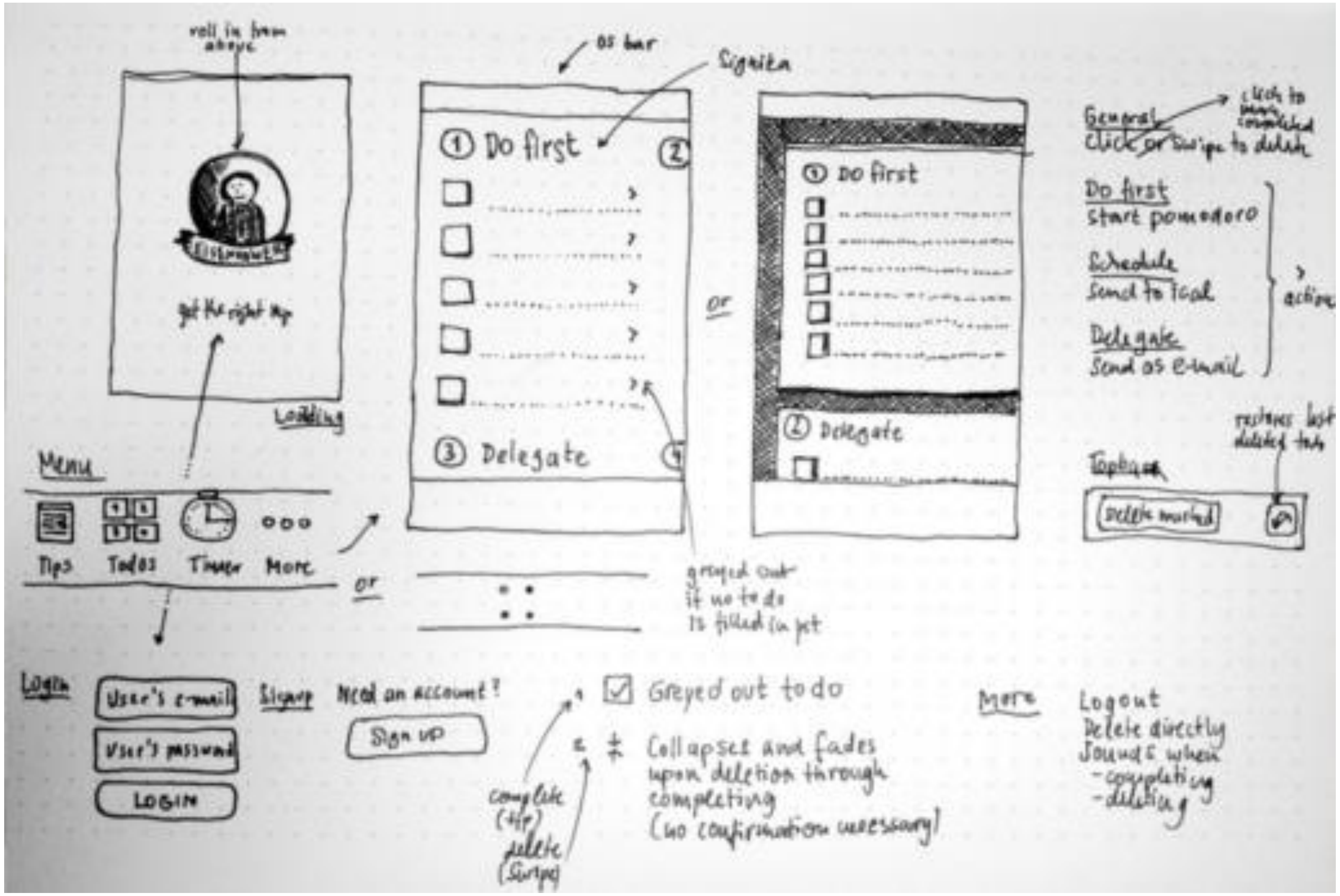


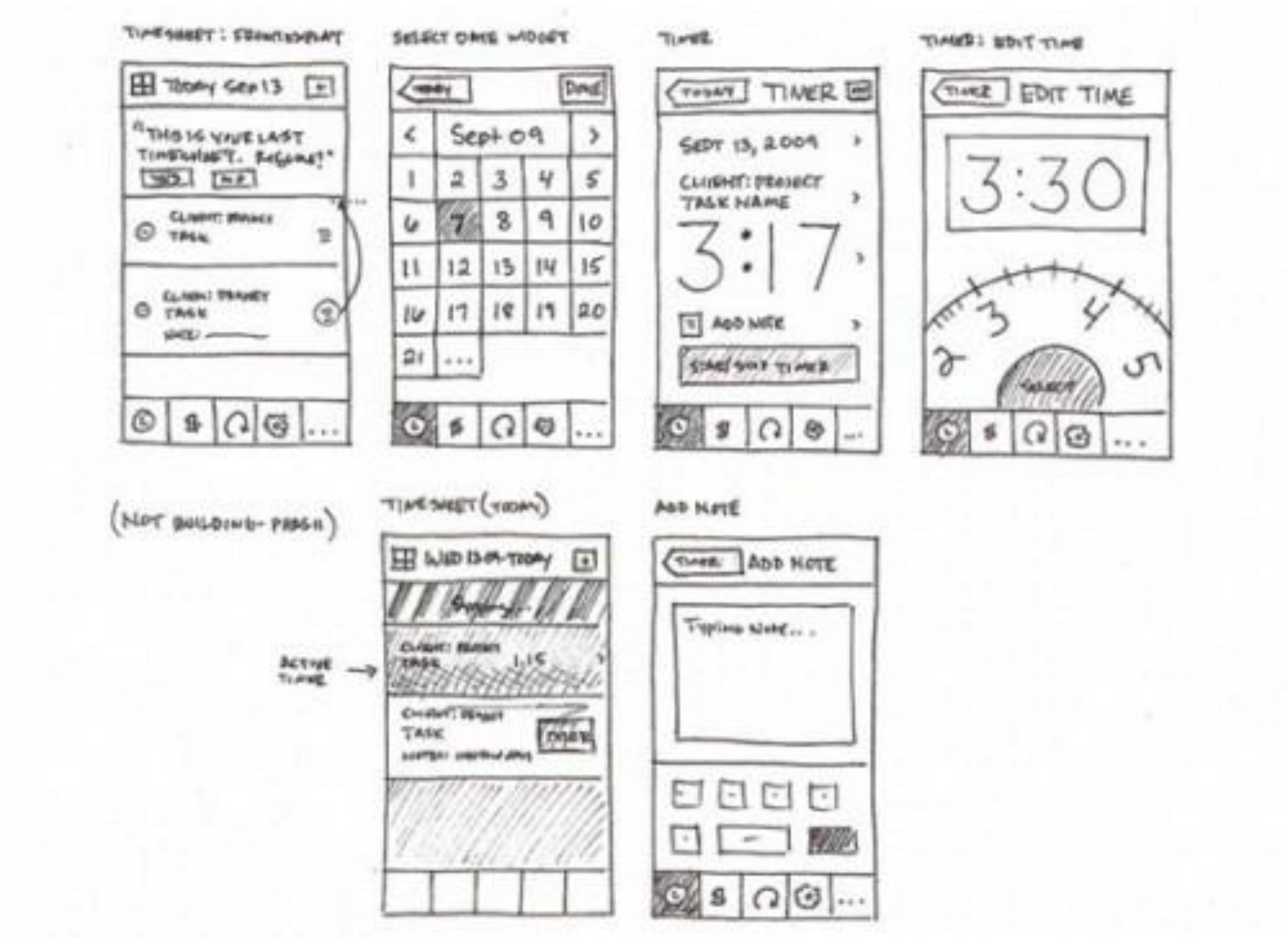
Seminar Number: CSC *
Name: *Agile *

Results

Seminar	Term	Seats Avail	Professor
CSC 250 Agile Techniques	Fall 2004	4	Smith, J.
CSC 300 Agile EUP	Spring 2005	17	Jones, S.
CSC 310 Agile Database techniques	Spring 2004	Ø	Johnson, M.

Course description:
CSC 310 Agile Database Techniques
This course describes evolutionary development strategies for data oriented development. See www.agiledata.org for details.
This course currently has 39 people waitlisted for it.


roll in from above
get the right tip
Loading
os bar
Signika
1 Do first
2
3 Delegate
4
or
1 Do first
2 Delegate
greyed out if no to do is filled in yet
General
Click to mark completed
Click or swipe to delete
Do first
Start pomodoro
Schedule
Send to iCal
Delegate
Send as e-mail
actions
restores last deleted task
Topbar
Delete marked
Menu
Tips
Todos
Timer
More
or
Login
User's e-mail
User's password
LOGIN
Signup
Need an account?
Sign up
Greyed out to do
Collapses and fades upon deletion through completing (no confirmation necessary)
complete (tip)
delete (swipe)
More
Logout
Delete directly
Sounds when
- completing
- deleting


SELECT DATE WIDGET
TIMER
TIMER: EDIT TIME
(NOT BUILDING PAGE!)
TIMESHEET (TODAY)
ADD NOTE
TODAY SEP 13
"THIS IS YOUR LAST TIMESHEET. RESUME?"
Sept 09
TIMER
SEPT 13, 2009
CLIENT: PROJECT
TASK NAME
3:17
ADD NOTE
START/STOP TIMER
EDIT TIME
3:30
SELECT
ACTIVE TIMER
Typing Note...

Testing

IT 3105

Topics

What is Software Testing?

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Software testing is a process of executing a program or application with the intent of finding evidence of software bugs.

-International Software Testing Qualifications Board

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

What is a bug?

Formally, we say that a s/w bug occurs when one or more of the following five rules is true.

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Rule 1

The software doesn't do something that the product specification says it should do.

Rule 2

The software does something that the product specification says it shouldn't do.

Rule 3

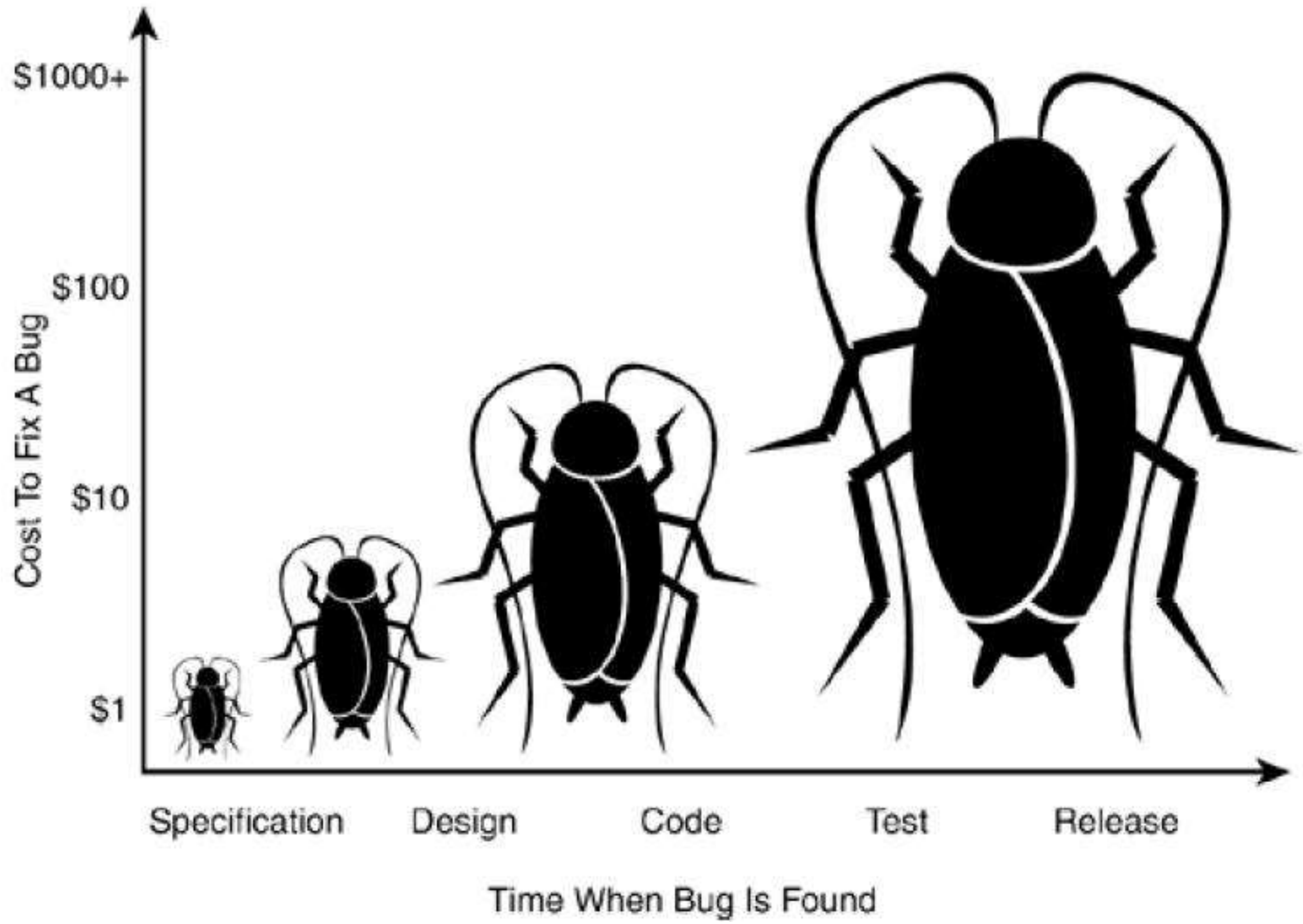
The software does something that the product specification doesn't mention.

Rule 4

The software doesn't do something that the product specification doesn't mention but should.

Rule 5

The software is difficult to understand, hard to use, slow, or in the software tester's eyes will be viewed by the end user as just plain not right.



UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Software Quality

"Software quality is the degree of conformance to **explicit** or **implicit requirements** and **expectations**."

-International Software Testing Qualifications Board

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Software Quality has two main divisions:

1) Software Quality Assurance (SQA)
2) Software Quality Control (SQC)

SQA

- is a set of activities for ensuring **quality in software engineering processes** that ultimately result in quality in software products.

- Also includes the following:
  - Process definition and implementation
  - Auditing
  - Training

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

- Once the processes have been defined and implemented, Quality Assurance has the following responsibilities:
  - identify weaknesses in the processes
  - correct those weaknesses to continually improve the process

SQC

- Software Quality Control (SQC) is a set of activities for ensuring quality in software products.

- SQC includes the following activities:

Reviews	Testing
Requirement Review Design Review Code Review Deployment Plan Review Test Plan Review Test Cases Review	Unit Testing Integration Testing System Testing Acceptance Testing

SQA vs. SQC

- SQA is oriented towards prevention, SQC is oriented towards detection.

Effects of Testing to Quality

- Measures Quality
- Gives Quality Confidence
- Increases Software Quality

Software Testing Life Cycle

STLC Phases	Deliverables
Requirements/ Design Review	‘Review Defect’ Reports
Test Planning	Test Plan / Test Estimation/ Test Schedule
Test Designing	Test Cases / Test Scripts /Test Data
Test Environment Setup	Test Environment
Test Execution	Test Results (Incremental) / Bug Reports
Test Reporting	Test Results (Final) Test/ Defect Metrics Test Closure Report

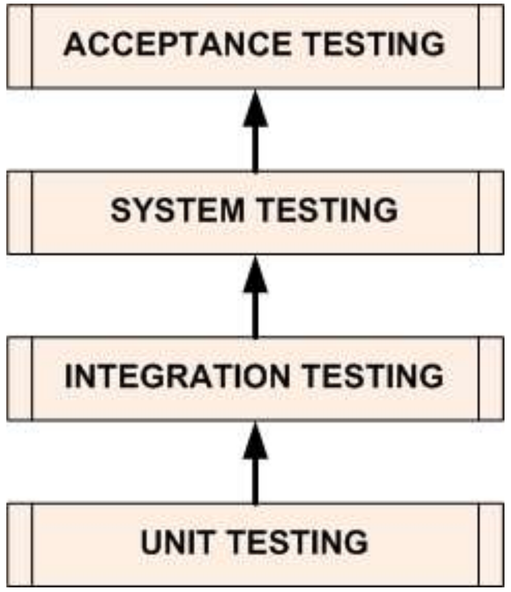
Verification vs. Validation

It is entirely possible that a product passes when verified but fails when validated. This can happen when, say, a product is built as per the specifications but the specifications themselves fail to address the user’s needs.

Verification	Validation
The process of evaluating work-products (not the actual final product) of a development phase to determine whether they **meet the specified requirements** for that phase.	The process of evaluating software during or at the end of the development process to determine **whether it satisfies specified business requirements**.
Plans, Requirement Specs, Design Specs, Code, Test Cases	The actual product/software

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

Software Testing Levels



Unit Testing

Unit Testing is a level of the software testing process where individual parts of a software are tested separately.

Integration Testing

- Integration Testing is a level of the software testing process where individual units are combined and tested as a group.

System Testing

- System Testing is a level of the software testing process where a complete and fully integrated software is tested.

Acceptance Testing

- Acceptance Testing is a level of the software testing process where system is tested for acceptability & validates the end-to-end business flow. Such testing executed by client in separate environment (similar to production environment).

Software Testing Methodologies

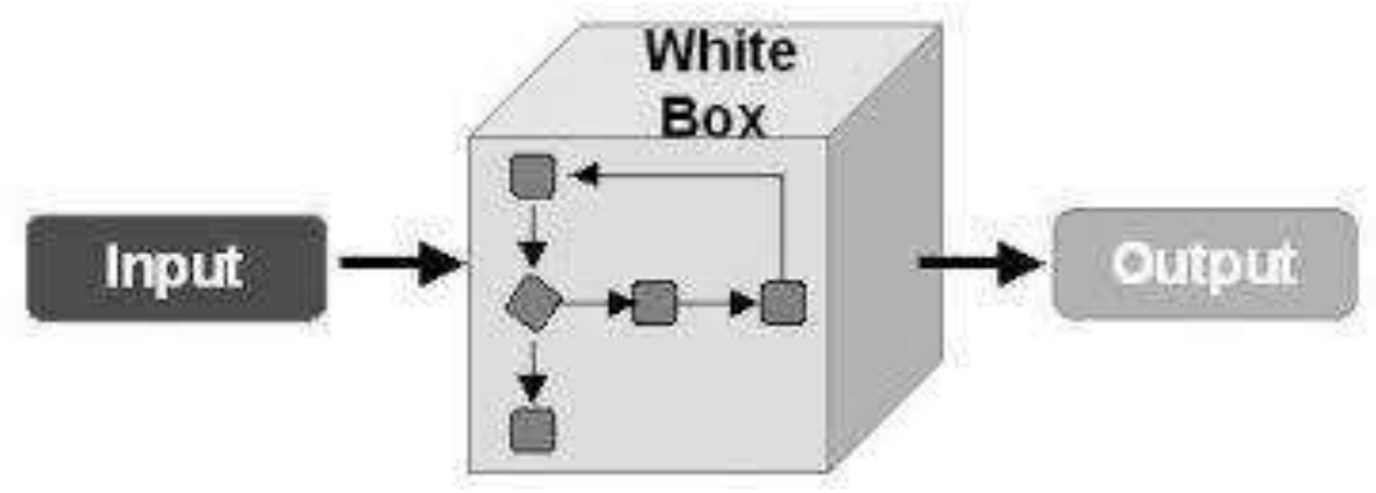
- Methods of testing according to the visibility of the software’s internal structure are as follows:

- White Box Testing
- Black Box Testing
- Gray Box Testing

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

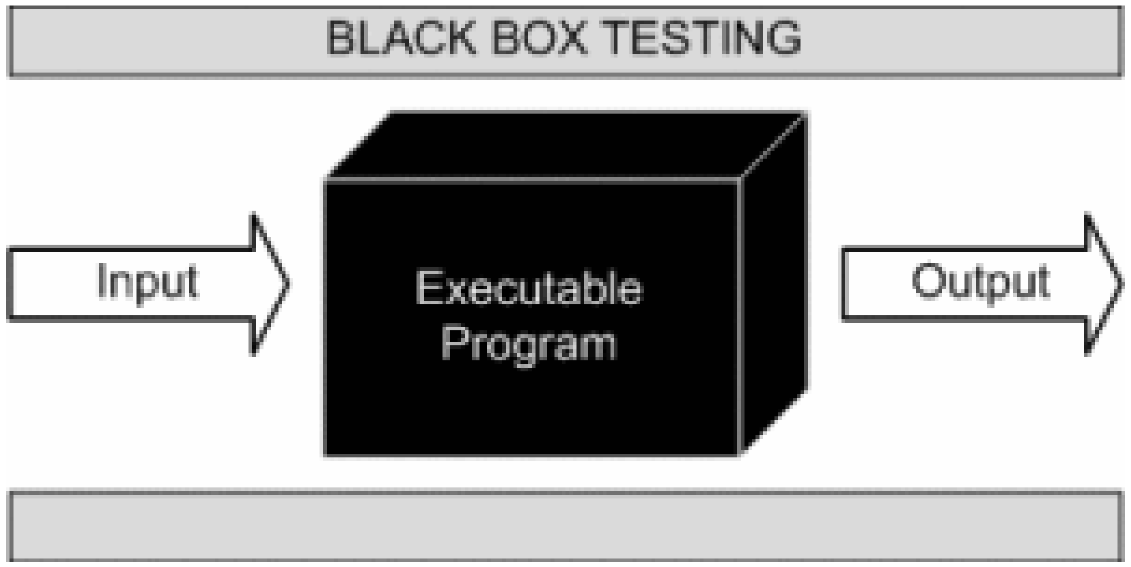
White Box Testing

- White Box Testing is a software testing method in which the internal structure of the item being tested is known to the tester.



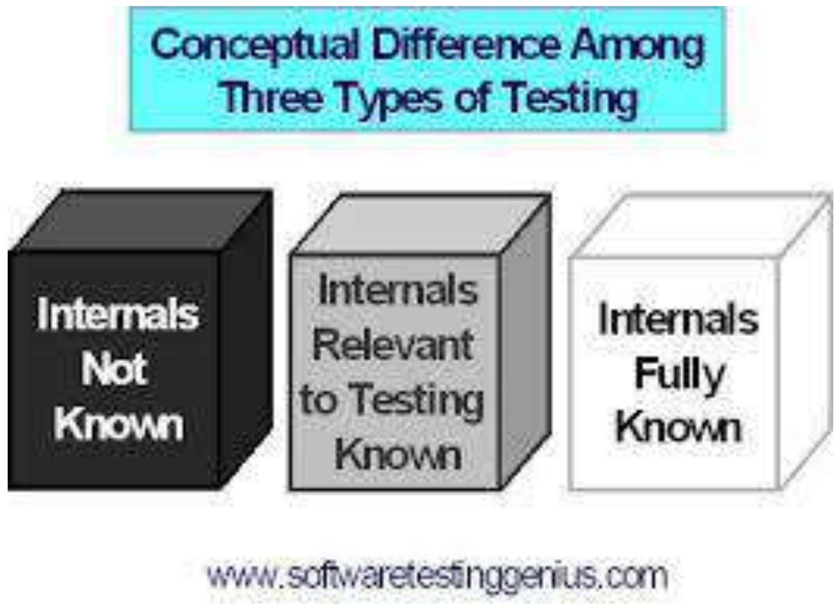
Black Box Testing

- Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure of the item being tested is not known to the tester.



Gray Box Testing

- In Gray Box Testing, the internal structure is partially known. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level.


Conceptual Difference Among Three Types of Testing
Internals Not Known
Internals Relevant to Testing Known
Internals Fully Known
www.softwaretestinggenius.com

Types of Testing

Smoke Testing

- Smoke Testing, also known as “Build Verification Testing”, is a type of software testing that comprises of a non-exhaustive set of tests that aim at ensuring that the most important functions work.
- Smoke testing covers most of the major functions of the software but none of them in depth.

Compatibility Testing

- Compatibility testing is used to determine if your software application has issues related to how it functions in concert with the operating system and different types of system hardware and software.

Regression Testing

- Regression testing is a type of software testing that intends to ensure that changes (enhancements or defect fixes) to the software have not adversely affected it.

Performance Testing

- Performance Testing is a type of software testing that intends to determine how a system performs in terms of responsiveness and stability under a certain load.

Stress Testing

- Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements to determine the load under which it fails and how.

Pilot Testing

- Testing which involves the users just before actual release to ensure that users become familiar with the release contents and ultimately accept it. Often is considered a Move-to-Production activity for ERP releases or a beta test for commercial products.

Concurrent Testing

- Concurrent testing is a multi-user testing used to determine the effects of accessing the same application code, module or database records.

Security Testing

- Security Testing is the process to determine that an IS (Information System) protects data and maintains functionality as intended.

- Security testing should be able to cover 6 aspects of application security.

- Confidentiality
- Integrity
- Authentication
- Authorization
- Availability
- Non-repudiation