Study Objectives

• Alternate Software Development Methodologies
• Alternate Development Methods
Alternate Software Development Methodologies

- Which ones of these are alternate software development methodologies
- A. Agile
- B. Scrum
- C. eXtreme Programming (XP)
- D. Prototyping
- E. Rapid Application Development
- F. Object-Oriented Programming
- G. Data-oriented systems development
- H. Component-based model
Water-Fall SDLC Model (week 4)

Barry Boehm’s water-fall model

Source: Adapted from Barry Boehm’s SDLC Model, IEEE, 1975
Waterfall Model - Pros and Cons (week 4)

• Structured
• Works well when requirements are well defined
• Better for relatively larger projects
• Better if there is time in hand

• Too much documentation
• Making changes becomes difficult during SDLC
• Speed to market
• Delay in implementation
Alternate Software Development Methodologies Characteristics

• Iterative
• Limited Documentation
• Closer collaboration between the development teams, and business partners and users team
Alternative Software Development Methodologies Advantages

• Attempt to addresses the “cons” of traditional water-fall methodologies
  – Many times the requirements are not well understood in the beginning and a prototype-based approach helps users and development teams define the requirements, as well as architecture and high-level design.
  – At times “speed-to-market” is important, and it’s more productive to have the “flexibilities of making changes” as time progresses
  – It’s also beneficial to get feedback from the end-users on the initial scope of the project, so that the project team can “fine tune” the project road-map. Interaction and collaboration across team members are the key to success.
  – Looking at the early success (or failure) of the “limited scope” in the beginning helps the project team and the business partners to determine the end-scope, financial spend, and cost-benefit.
Alternative Software Development Methodologies Disadvantages

• Alternate software methodologies is “not always a recipe of success”
  – Limited documentation (specifically around requirements and design), may not provide a good clarity on the functionality and traceability, resulting in poor testing quality, and even poor quality of the system.
  – It’s hard to duplicate the success of alternate software development methodologies in a large or distributed team, where “in the room” type interaction is not possible.
  – Alternative software development methodologies may cause high maintenance cost, as the functionality are not well defined or documented.
  – Alternative software development methodologies may cause rework as the focus is not on the “entire system” but just on the next iteration. Requirements baseline is missing
  – Process and structure in the organization is missing as the focus is on adaptability.
Spiral SDLC Model (week 4)

Spiral Model is a variant of water-fall model
Spiral SDLC Model (week 4)

- It’s iterative
- Each iteration goes through mini-waterfall
- Gives Business Team time to refine and prioritize requirements
- Helps assess Business Value before committing to entire project
- Helps course correct
- More common than traditional waterfall SDLC
Alternative Software Development Methodologies - Agile

Manifesto for Agile Software Development

“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.*”

Source: Adapted from [http://www.agilemanifesto.org/](http://www.agilemanifesto.org/)
Principles Behind Agile Manifesto

“We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. “

“The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development.

The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximizing the amount of work not done—is essential. The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”

Source: Adapted from [http://www.agilemanifesto.org/principles.html](http://www.agilemanifesto.org/principles.html)
Agile Development Framework

Source: Adapted from “CISA Review Manual 2013” ISACA, page 196”
Agile Development

• There are a number of flavors
  – Scrum, eXtreme Programming, Crystal Adaptive Software, Feature Drive Development

• Common theme
  – Small time-boxed iterations
  – Re-planning at the end of iterations or scrum
  – Self-organizing and collaborative team, often led by “scrum master”
  – Less documentation
  – Collaboration between Business and IT on daily basis
  – Project Manager’s role from “planning the project, allocating the tasks, and tracking and oversight to that of a facilitator
  – Focus is to quickly prove the architecture by building the actual functionality vs. spending in too much time in the detail analysis and approach

• Pros and Cons
  – Pretty much in line with what discussed for the Alternative Software Development Methodologies before
Alternative Software Development Methodologies - Prototyping

• Prototyping (Evolutionary Development) builds a working model that is used to elicit/verify requirements and come up with the design. Eventually, the prototype is hardened to make it production ready.

• Two approaches
  1. Build the model to bolster requirement and design
  2. Build the actual system with limited functionality and deploy in production
Alternative Software Development Methodologies - Prototyping

• Pros
  – Iterative framework while leveraging traditional SDLC
  – Reduces time to deployment using 4th generation tools

• Cons
  – The model may not be easy to create, and building production-ready prototype is not easy for a large system
  – Poor scope management
  – Poor control

Note: The overall pros and cons of alternative software development methodologies apply in this case too
Alternative Software Development Methodologies - RAD

• RAD (Rapid Application Development) helps develop some of the strategic important systems quickly
• Leverages
  – Small well-trained teams (such as “tiger team”, “seal team”)
  – Prototypes
  – Time-boxed iterations
• Focus on individual application systems rather than enterprise systems as a whole
• RAD methodology has four major stages
  – Concept Definition stage: for scope definition
  – Functional Design stage: uses workshops to model system’s data and processes
  – Development stage: for construction
  – Deployment stage: UAT, training, and implementation
Alternative Software Development Methodologies - RAD

• Pros
  – Can be applied to “some” strategic projects
  – Reduces cost, time-to-market, and maintains quality

• Cons
  – May not scale well to all the IT projects at the same time in the enterprise
  – May not scale well to distributed systems

Note: The overall pros and cons of alternative software development methodologies apply in this case too
Alternate Software Development Methods

- What are alternative software development methods

  A. These are software programming or software engineering methods other than Structured methods
  
  B. These are independent of Software Development methodologies
  
  C. Object-Oriented Programming
  
  D. Data-oriented systems development
  
  E. Component-based model
  
  F. All of the above
Alternate Development Method – Object-Oriented System Development

- Object-Oriented Programming Languages (OOP Language)
  - Simula67 (1967); Smalltalk (1970s)
  - C++, Object Pascal, Ada95 (1980s – 1990s)
  - Java (1990s); C# (2000s)

- Objects and Class
  - Objects are constructs with “attributes” and the “methods” that acted on it
  - Before OOP, “Structured” programming was used, where a program was written as a “function” that had a number of instructions that did the things like adding numbers, sort words, retrieve message from email etc.
  - In OOP the same problem is looked at differently. For example, Numbers, Words, or emails can be thought as objects. The objects have attributes like value of the numbers, the words, or the emails. There are “methods” associated with the objects, which act upon the attributes of the objects.
  - Objects can interact with each-other by sending messages.
  - Similar objects with different values can be created by the template of the object, call “class,” which is similar to cookie-cutter
Object-Oriented System Development

• Advantages of OOP
  – Data Encapsulation:
    • allows private attributes of the object not visible outside the object (abstraction)
    • allows implementation of the method to be separated from the signature of the method
  – Inheritance: allows reuse by creating super class and subclass (hierarchy)
Example of OOP using Java

```java
public class Human {
    private String name = "no name"; // the name of this human
    private Human friend = null; // the human's friend

    //This "creates" a new Human
    public Human(String name, Human friend) {
        this.name = name;
        this.friend = friend;
    }
    public Human(String name) {
        this.name = name;
        this.friend = null;
    }
    public Human() {
        this.name = "no name";
        this.friend = null;
    }
    public void sayName() {
        System.out.println("My name is " + this.name);
    }
    public void sayGoodnight() {
        if (friend == null)
            System.out.println("Good night nobody.");
        else
            System.out.println("Good night " + friend.name);
    }
}

public class Main {
    public static void main(String[] args) {
        //create a new human object john
        Human john = new Human("John");
        //create a human object named jane with john as a friend
        Human jane = new Human("Jane", john);
        john.sayName(); //shows 'My name is John'
        john.sayGoodnight(); //shows 'Good night nobody.'
        jane.sayName(); // shows 'My name is Jane'
        jane.sayGoodnight(); //shows 'Good night John'
    }
}
```

Upcoming Assignments/Tests

1. Group Case Study -3 (Testing): Mon 11/25 before the class

Questions?
Summary of Today’s Class

• Agile Development Methodology
• Retyping
• RAD
• OOP
• Focus of the Next Class and Reading
• Questions