CIS 475/575: Topic # 4

Systems Development Life Cycle (SDLC)
A Modern Approach to Systems Analysis and Design

- **1950s:** focus on efficient automation of existing processes
- **1960s:** advent of procedural third generation languages (3GL) faster and more reliable computers
- **1970s:** system development becomes more like an engineering discipline
- **1980s:** major breakthrough with 4GL, CASE tools, object-oriented methods
- **1990s:** focus on system integration, GUI applications, client/server platforms, Internet
- **1995s:** Web application development, wireless PDAs and smart phones, component-based applications, application service providers (ASP)
- **2005s:** Mobile applications
- **2015s:** Smart applications (integrated application, cloud-based applications)

However, an approach to SAD is still the same: SDLC-based approach
System Development Methodology (SDLC)

- **System Development Methodology** is a standard well-structured process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

- **Phases in SDLC (Systems Development Life Cycle):**
  - Planning
  - Analysis
  - Design
  - Implementation
  - Maintenance

- **Different authors: from 3 to 20 various phases in SDLC**
Other Possible SDLC Phases
SDLC Model to be Used in this Course

- **Planning** – an organization’s total information system needs are identified, analyzed, prioritized, and arranged
- **Analysis** – system requirements are studied and structured
- **Design** – a description of the recommended solution is converted into logical and then physical system specifications
- **Logical design** – all functional features of the system chosen for development in analysis are described independently of any computer platform
- **Physical design** – the logical specifications of the system from logical design are transformed into the technology-specific details from which all programming and system construction can be accomplished
- **Implementation** – the information system is coded, tested, installed and supported in the organization
- **Maintenance** – an information system is systematically repaired and improved
### Products (outcomes, outputs, deliverables) of SDLC Phases

<table>
<thead>
<tr>
<th>Phase</th>
<th>Products, Outputs, or Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Priorities for systems and projects; an architecture for data, networks, and selection hardware, and information systems management are the result of associated systems. Detailed steps, or work plan, for project. Specification of system scope and planning and high-level system requirements or features. Assignment of team members and other resources. System justification or business case.</td>
</tr>
<tr>
<td>Analysis</td>
<td>Description of current system and where problems or opportunities are with a general recommendation on how to fix, enhance, or replace current system. Explanation of alternative systems and justification for chosen alternative.</td>
</tr>
<tr>
<td>Design</td>
<td>Functional, detailed specifications of all system elements (data, processes, inputs, and outputs). Technical, detailed specifications of all system elements (programs, files, network, system software, etc.). Acquisition plan for new technology.</td>
</tr>
<tr>
<td>Implementation</td>
<td>Code, documentation, training procedures, and support capabilities.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>New versions or releases of software with associated updates to documentation, training, and support.</td>
</tr>
</tbody>
</table>
CIS 475/575: Topic # 4

SDLC Models
1) SAD rests on a goal - an organizational commitment to produce top quality product (CIS)

2) Process layer (that uses formal process models) is the glue that holds the SAD technology layers together.

3) Methods to provide the concrete technical HOW-TO’s for building CIS.

4) Tools to provide automated or semi-automated support for CIS development.
SAD Processes: An Overview

- The roadmap to building high quality CIS is SAD process.
- Different types of CIS projects require different SAD processes.
5 SAD main (framework) activities (for large CIS projects, or in large-size companies).

But … in what order? Returns/iterations/loops? Feedback from users?
Main Types of Process Flow
(Various types will have significant impact on project time, cost, human resources, quality, etc.)

a) Linear process flow (no feedback, no iterations/loops)

b) Iterative process flow (a circular manner, with a feedback, with iterations)

c) Evolutionary process flow (with various versions or increments released)

d) Parallel process flow (no feedback, no iterations/loops, with parallel activities)

CIS development team (project manager, system analyst, etc.) **MUST** identify the best (optimal) process flow or, possibly, a combination of them for a specific CIS project.
Question: Why do we need to know all those types of SAD Process Models?

Answer: It is about Money, Time, People, and Quality. (Various types will have significant impact on project time, cost, human resources, quality, etc.)
CIS 475/575: Topic # 4

Waterfall SDLC (SAD) Model
The **Waterfall Model** suggest a systematic, linear, sequential approach to CIS SAD that begins at the system level and progresses through SAD main phases.

(Although the author – Winston Royce, 1970 – made provisions for “feedback loops”, the vast majority of CIS development companies treat it as **if it is was strictly linear**.)
Pros and Cons of Waterfall Model

+) Simple and clear
    • Well-formalized
    • Easy to monitor

-) Feedback ignored, milestones lock in design specs even when conditions change
    • Limited user involvement (only in requirements phase)
    • Too much focus on milestone deadlines of SDLC phases to the detriment of sound development practices
    • No flexibility in project activities

Bottom Line:
    • Real large-scale projects rarely follow this model because
        a) it requires ALL requirements to be known in advance;
        b) the working model will be available only at the very end;
        c) customers are not involved into process, etc.).
    • It works for small-size routine projects (nothing innovative in these projects)
    • It usually used by very experienced CIS developers
Examples of CIS Projects with Waterfall Model is SAD

1. A personal Web site or a Web site for an academic department
   - No feedback from end users is required
   - Nothing unique, revolutionary, and innovative

2. Embedded (built-in SW in a car, in a refrigerator, etc.)
   - No feedback from end users is required
   - Nothing unique, revolutionary, and innovative
   - Quality will be evaluated after completion of entire SW

SDLC Waterfall model is used when
- Requirements are stable and not changed frequently.
- Application is small.
- There is no requirement which is not understood or not very clear.
- The environment is stable
- The tools and technology used is stable and is not dynamic
- Resources are well trained and are available.

http://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model/
CIS 475/575: Topic # 4

Incremental SDLC (SAD) Model (with multiple sub-models)
The 1st increment = a core product.

Following increments are aimed to better meet customer requirements and deliver additional functionality.
The Incremental Model: Real World Example

- By September 1991, Linux version 0.01 was released. It had 10,239 lines of code.
- In October 1991, Linux version 0.02 was released.
- In December 1991, Linux 0.11 was released. This version was the first to be self-hosted - Linux 0.11 could be compiled by a computer running Linux 0.11.
- When he released version 0.12 in February 1992, Torvalds adopted the GNU General Public License (GPL) over his previous self-drafted license, which had not permitted commercial redistribution.
- In March 1992, Linux version 0.95 was the first to be capable of running X. This large version number jump (from 0.1x to 0.9x) was due to a feeling that a version 1.0 with no major missing pieces was imminent. However, this proved to be somewhat overoptimistic, and from 1993 to early 1994, 15 development versions of version 0.99 appeared.
- On 14 March 1994, Linux 1.0.0 was released, with 176,250 lines of code.
- In March 1995, Linux 1.2.0 was released (310,950 lines of code).
- Version 2 of Linux, released on 9 June 1996, was followed by additional major versions under the version 2 header, including the following ones:
  - 25 January 1999 - Linux 2.2.0 was released (1,800,847 lines of code).
  - 18 December 1999 - IBM mainframe patches for 2.2.13 were published, allowing Linux to be used on enterprise-class machines.
  - 4 January 2001 - Linux 2.4.0 was released (3,377,902 lines of code).
  - 17 December 2003 - Linux 2.6.0 was released (5,929,913 lines of code).
  - 9 June 2009 - Linux 2.6.30 was released (11,637,173 lines of code).
  - May 18, 2011 – Linux 2.6.39 was released.

Note: In July 2009 Microsoft contributed 20,000 lines of code to the Linux kernel. The contribution consisted of Hyper-V drivers, which improve the performance of virtual Linux guest systems in a Windows hosted environment. Microsoft licensed its Linux Hyper-V drivers under the GPL.
The Incremental Model: Real World Example (linux)
Examples of CIS Projects with Incremental Model is SAD

1. **Sakai LMS**
   - 1st increment = main increment was about 50-60% of overall functionality of LMS system
   - Several additional increments to populate it with courses and additional functions

2. **Webster system** (online academic services for Bradley students)
   - 1st increment = main increment was about 60-80% of overall functionality of the system

3. **Microsoft Internet Explorer (search engine) v.10.0.6, 10.0.7, 10.08, 10.0.9, ...**
   - 1st increment = main increment was about 60% of functionality
   - Several additional increments to add multiple additional valuable functions

3. **Linux (operation systems) v. 3.0, 3.1., 3.2., 3.3., 3.4., 3.5., 3.6....**

---

### Release versions of Internet Explorer for Windows

<table>
<thead>
<tr>
<th>Version</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0.303</td>
<td>Internet Explorer 1.0 (Build for Windows 95)</td>
</tr>
<tr>
<td>4.0.320</td>
<td>Internet Explorer 2.0</td>
</tr>
<tr>
<td>4.0.115</td>
<td>Internet Explorer 3.0</td>
</tr>
<tr>
<td>4.0.115</td>
<td>Internet Explorer 3.0 (Windows 95-OSR2)</td>
</tr>
<tr>
<td>4.0.1215</td>
<td>Internet Explorer 3.0.01</td>
</tr>
<tr>
<td>4.0.1300</td>
<td>Internet Explorer 3.0.02 and 3.02a</td>
</tr>
<tr>
<td>4.0.1544</td>
<td>Internet Explorer 4.0 Platform Preview 1.0 (PP1)</td>
</tr>
<tr>
<td>4.0.1530</td>
<td>Internet Explorer 4.0 Platform Preview 2.0 (PP2)</td>
</tr>
<tr>
<td>4.0.17126</td>
<td>Internet Explorer 4.0</td>
</tr>
<tr>
<td>4.0.2068</td>
<td>Internet Explorer 4.0.01</td>
</tr>
<tr>
<td>4.0.24076</td>
<td>Internet Explorer 4.01 Service Pack 1 (Windows 98)</td>
</tr>
<tr>
<td>4.0.2412</td>
<td>Internet Explorer 4.01 Service Pack 2</td>
</tr>
<tr>
<td>5.0.0510.10</td>
<td>Internet Explorer 5 Developer Review (Beta 1)</td>
</tr>
<tr>
<td>5.0.0510.339</td>
<td>Internet Explorer 5 Beta (Beta 2)</td>
</tr>
<tr>
<td>5.0.0.14.0216</td>
<td>Internet Explorer 5</td>
</tr>
</tbody>
</table>
Incremental Models: AGILE Model
(Motto: “The most important functions first”)
Iterative, Incremental Final Product Development

Every sprint (iteration)
is about 2 weeks long

New product:

- Most important 5…10..20% of functionality
- Most important 5…10..20% of functionality
  + ADDITIONAL about 10% of functions
- Most important 5…10..20% of functionality
  + ADDITIONAL about 10% of functions
  + 10% of functions, and so on
Incremental Models: The RAD (Rapid Application Dev.) Model
RAD Model: A Historic Excursus

- RAD Model was oriented to revolutionary CIS SAD projects.

- For example, Evolutionary Rapid Development (ERD) technology was developed by the Software Productivity Consortium, a technology development and integration agent for the Information Technology Office of the Defense Advanced Research Projects Agency (DARPA).

- Orientation to revolutionary breakthroughs in a connected approach: DARPA historically has focused on not incremental but radical innovation. It emphasizes high-risk investment, moves from fundamental technological advances to prototypes and system development.

- These days, RAD Model is actively used in various CIS projects due to RAD tools.
Desktop RAD Model Tools (examples)

- Apple Xcode
- **C++Builder**
- Clarion is a data-centric Advanced Rapid Application Development tool
- Code::Blocks
- Delphi for PHP
- **Gambas** Basic, Open source, Linux
- Gupta Team Developer / SQLWindows
- Habanero
- Microsoft Visual Basic
- Lazarus Pascal, Open Source, Multi-platform
- Panther
- RADvolution Designer
- Runtime Revolution
- REAL software **REALbasic**
- Softwell Maker is a desktop IDE with a cross-platform deployment component allowing publish application into almost any Java enable system.
- Thoroughbred OPENworkshop is a RAD for Windows, UNIX, Linux, and OpenVMS
- The Virtual Enterprise is an Interactive Voice Response (IVR) toolkit developed specifically for telephony and speech inside Microsoft Visual Studio.NET.
- wxDev-C++
- Microsoft **Visual Foxpro**
- WinDev
- XVT
- MX-Frame - Business Application Framework
- **GNAVI**
Database RAD Tools (examples)

- **Base One Foundation Component Library (BFC)** is a RAD framework for building .NET applications using SQL Server, Oracle, DB2, Sybase, and MySQL databases.
- **Clarion** is a data-centric Advanced Rapid Application Development (ARAD) tool featuring roundtrip code generation. **uniPaaS** (by Magic Software) is a database independent Rapid Application Tool for building traditional GUI applications as well as scaling enterprise-level websites. **IBM Rational Business Developer Extension** supports database application development for IBM DB2, IBM Informix, Oracle database, Microsoft SQL Server and other JDBC compliant relational databases.
- **IBM Rational Application Developer** supports database application development for IBM DB2, IBM Informix, Oracle database, Microsoft SQL Server and other JDBC compliant relational databases.
- **IBM Lotus Notes** is a RAD environment for collaboration and document management tasks.
- **is** a code generator that builds database-driven web **Web 2.0** applications for .NET. It generates application Web pages ASPX, user interface code and data access logic (C#, Visual Basic .NET; and SQL queries) without hand-coding.
- **FileMaker** is a cross-platform database application from FileMaker Inc. (a subsidiary of Apple Inc.).
- **Sybase PowerBuilder** is data-driven development tool for creating client/server, distributed, Web and Smart Clients applications for JEE, Win32, and .NET platforms.
- **is** an **open source** database-driven RAD development environment for building client (desktop) based applications.
- **Kexi** is an **open source** database-driven RAD development environment for building desktop applications. It is considered an alternative to Open Office Base in the Open Source environment and provides similar features to commercially available RAD development environments such as FileMaker, Alpha Five and Microsoft Access.
- **Oracle Forms**
- **Oracle Application Express** (Oracle APEX) is software development environment based on the Oracle database. It allows a very fast development cycle to be achieved to create web based applications.
- **Panther** (and its **open source** version POSSL) is a cross-platform (Windows, Unix, Linux; TUI, GUI, Web), cross-database RAD toolset for development of C/S and n-tier database oriented applications.
- **NConstruct** is Windows and Web rapid enterprise application development tool and environment for .NET framework. It supports Oracle database, Microsoft SQL Server databases and Microsoft Access.
- **Softwell Maker** is a ultra RAP data-centric IDE with a cross-platform deployment component allowing publish application into almost any Java enable system.
- **WinDev**
Web-Based RAD Tools (examples)

- **Active Agenda**'s code generator is a RAD development framework using **XML** specification files and the **PHP** development language.
- **Alpha Five** is a commercial RAD development environment for both client and web-server based database driven applications. This tool is typically classified with commercial packages such as **Microsoft Access** and **FileMaker**.
- **Axiom Stack** is an open source web application framework designed to foster rapid development through the use of **ECMAScript** (**JavaScript**) and **Java**. Tools such as the **Axiom CMS** and **Inspector** are written to aid in application development.
- **BFC** is a RAD framework for both client and server-side development in the .NET environment.
- **CakePHP** is a RAD development framework using the **PHP** development language.
- **is a visual RAD development environment for web-based database driven application development.** places emphasis on code generation technology to provide **ASP.NET**, **PHP**, **JSP**, Servlets, **ColdFusion** and **Perl** language support.
- **Zend Framework** is an open source, object-oriented web application framework licensed under the New BSD License.
- **Django** is an open source web application framework, written in Python, which loosely follows the model-view-controller design pattern.
- **IBM Rational Business Developer Extension** is a cross-platform, Rapid Application Development IDE for creating enterprise and web applications and services for Windows, Linux, Unix (Solaris, HPUX, AIX), System z and System i.
- **NConstruct** is a RAD application development tool and environment for **NET framework**.
- **nuBuilder** is an open source browser based database development tool which stores all forms, reports, data and any custom code in MySQL and displays the content dynamically.
- **Oracle Application Development Framework** uses Oracle's **JDeveloper** a FREE IDE that supports ADF's J2EE based framework.
- **Panther** (and its open source version **POSSL**) is a cross-platform (**Windows**, **Unix**, **Linux**; **TUI**, **GUI**, **Web**), cross-database RAD toolset for development of **C/S** and **n-tier** database oriented applications.
- **Pylons** is an open source web application framework, written in Python, which makes extensive use of the Web Server Gateway Interface (WSGI) standard to promote re-usability and to separate functionality into distinct modules.
- **Radicore** is a RAD development framework using the **PHP** development language. It is for building administrative web applications, not web sites, and includes a Role Based Access Control (RBAC) system, Audit Logging system (without database triggers), Data Dictionary and Workflow system.
- **Thoroughbred T-WEB** is a Web RAD tool.
- **Web2py** is a RAD framework for web-based database driven applications with key features including in-browser coding support, admin/design interface, DAL (database abstraction layer), and translation support.
- **WebDev**
- **Wavemaker** **Visual Ajax Studio** is an open-source, browser-based IDE based on Dojo, Spring and Hibernate.
- **Visual WebGui** **Visual WebGui (VWG)** is an open-source rapid application development (RAD) framework for AJAX & Silverlight GUIs. The platform presents a new approach to applying desktop usability to the web by viewing it as an extension to a desktop rather than web.
- **cakeApp** an online rapid development tool with WYSIWYG SQL editor and framework based on **CakePHP**.
- **Wavemaker** **Visual Ajax Studio** is an open-source, browser-based IDE based on Dojo, Spring and Hibernate.

is a web based development and design tool for designing forms and pages for mobile and hand-held devices as well as delivering cross platform web applications using a standard internet browser.
phpMyAdmin

Bringing MySQL to the web

About

phpMyAdmin is a free software tool written in PHP intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc.) can be performed via the user interface, while you still have the ability to directly execute any SQL statement.

phpMyAdmin comes with a wide range of documentation and users are welcome to update our wiki pages to share ideas and howto for various operations. The phpMyAdmin team will try to help you if you face any problem; you can use a variety of support channels to get help.

phpMyAdmin is also very deeply documented in a book written by one of the developers: Mastering phpMyAdmin for Effective MySQL Management, which is available in English and Spanish.

To ease usage to a wide range of people, phpMyAdmin is being translated into 74 languages and supports both LTR and RTL languages.

phpMyAdmin is a seventeen-year-old project with a stable and flexible code base; you can find out more about the project and its history and the awards it earned. When the project turned 15, we published a celebration page.

The phpMyAdmin project is a member of Software Freedom Conservancy. SFC is a non-profit organization that helps promote, improve, develop, and defend Free, Libre, and Open Source Software (FLOSS) projects.

Features

- Intuitive web interface
- Support for most MySQL features:
  - browse and drop databases, tables, views, fields and indexes
  - create, copy, drop, rename and alter databases, tables, fields and indexes
  - maintenance, server, databases and tables, with proposals or server configuration
  - execute, edit and bookmark any SQL statement, even batch-queries
  - manage MySQL user accounts and privileges
  - manage stored procedures and triggers

Latest News
Cross-Platform RAD Tools (examples)

- **Boa constructor** is a cross-platform,.wxPython based Python RAD IDE
- **Code::Blocks** is a cross-platform C/C++ RAD IDE using wxWidgets; the latest developmental builds have a built-in form designer wxSmith, so it's similar to Borland C++ Builder and Microsoft Visual C++/MFC now.
- **HyperNext** is a freeware cross-platform software development system for Macintosh OS X & OS 9, and Microsoft Windows XP & Vista. It has many similarities with HyperCard and can compile to both stand alone applications and stacks for the cross-platform HyperNext Player.
- **IBM Rational Business Developer Extension** is a cross-platform, Rapid Application Development IDE for creating enterprise and web applications and services for Windows, Linux, Unix (Solaris, HPUX, AIX), System z and System i
- **IBM Rational Application Developer** is a cross-platform, Rapid Application Development IDE for creating enterprise and web applications and services for Windows, Linux, and Unix (Solaris, HPUX, AIX)
- **LANSA** is a development environment for generating applications on multiple computer systems. The main feature of the LANSA environment is the RDML language. It is classified as a 4GL (4th generation computing language). It runs on many systems including MS Windows, Unix, and Linux. In its first release in 1987, the RDML language was known as lambda
- **Lazarus** is a cross-platform IDE similar to Borland Delphi.
- **m-Power** is a Software Development tool which automates application development and rapidly creates enterprise-class Web applications over any database or platform.
- **NetBeans** is a cross-platform, RAD IDE for creating visual desktop, mobile, web, and SOA applications for Linux, Windows and Mac OS X. The IDE officially supports Java, Ruby, PHP, JavaScript and C/C++ programming languages.
- **Omnis Studio** is a cross-platform, Rapid Application Development tool or IDE for creating enterprise and web applications for Windows, Linux, Solaris, and Mac OS X.
- **(OpenERP)** is a RAD framework in python.
- **OpenROAD** is a cross-platform IDE for Linux/Unix, Windows with embedded SQL support
- **Web Dynpro** is SAP’s RAD to create web applications connected to function modules in mySAP ERP.
- **RadRails** is a cross-platform IDE for creating Ruby on Rails web applications.
- **Servoy** is a cross-platform application development and deployment environment. Servoy consists of a GUI designer, is event-driven and runs scripts through JavaScript. Servoy allows applications to be deployed to both a native Smart client / Rich client and to a pure HTML Web client from the same codebase and user interface
- **WideStudio** is an open source integrated development environment for desktop applications purely
- **XVT** is a cross-platform, Rapid Application Development IDE for creating enterprise and desktop applications in C/C++ on Windows, Linux, Unix (Solaris, HPUX, AIX), and Mac
- **CA Plex**, a software development tool that combines the techniques of model-based development, patterns and code generation to accelerate the delivery and maintenance of multi-platform, distributed business applications
**RAD Model: Pros and Cons**

**Advantages of RAD Model:**
- Fast CIS SAD.
- Reduced (probably, significantly) time-to-market (or, development time).
- Low cost (overall)
- It helps to identify the most (probably, several) perspective way of future developments.

**Disadvantages of RAD Model:**
- In many cases, a low-cost SAD due to CASE tools ... but they usually generate not an optimal code.
- The process is too fast; as a result, proper Software Quality Assurance and Testing (especially security testing) may not be done.
- In many cases, software generated is of a "throw-away" type. In this case, if a new version of the software is needed, it is developed from scratch using the newest RAD techniques and tools.
- It requires extra resources (financial, human, technical, etc.)
<table>
<thead>
<tr>
<th>CIS 475/575: Topic # 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolutionary Prototyping SDLC (SAD) Model</td>
</tr>
</tbody>
</table>
Evolutionary Process Flow: A Prototyping Model

Prototyping Model (good first step when customer has a legitimate need, but is clueless about the details, developer needs to resist pressure to extend a rough prototype into a production product)
Types of Prototyping Model

- **Throwaway prototyping** – (also called *close Ended Prototyping, or Rapid Prototyping*) refers to the creation of a model that will eventually be discarded rather than becoming part of the final delivered software.

- **Evolutionary Prototyping** (also known as *Breadboard Prototyping*) is quite different from Throwaway Prototyping. The main goal when using Evolutionary Prototyping is to build a very robust prototype in a structured manner and constantly refine it.

- **Incremental prototyping** – In this case final product is built as separate prototypes. At the end the separate prototypes are merged in an overall design.

- **Extreme prototyping** -- Extreme Prototyping as a development process is used for developing especially web applications. Basically, it breaks down web development into three phases, each one based on the preceding one.
  1) The first phase is a static prototype that consists mainly of HTML pages.
  2) In the second phase, the screens are programmed and fully functional using a simulated services layer.
  3) In the third phase the services are implemented.

The process is called Extreme Prototyping to draw attention to the second phase of the process, where a fully-functional UI is developed with very little regard to the actual services to be used.
Examples of CIS Projects with Prototyping Model is SAD

1. **SW with heavy HCI (Human-Computer Interaction):**
   ADVANCED online shopping sites
   - Quality of images
   - Quality of interaction
   - Convenience for end users
   - End users are involved into SE process

2. **SW with heavy HCI (Human-Computer Interaction):**
   ADVANCED online games
   - Quality of images
   - Quality of interaction
   - End users are involved into SE process

3. **ADVANCED Online Ticket Reservation Systems**
   (airline companies, hotels, etc.)
Prototyping Model: Pros and Cons

Advantages of Prototyping Model:

- **Reduced time and costs**: Prototyping can improve the quality of requirements and specifications provided to developers. Because changes cost exponentially more to implement as they are detected later in development, the early determination of what the user really wants can result in faster and less expensive software.

- **Improved and increased user involvement**: Prototyping requires user involvement and allows them to see and interact with a prototype allowing them to provide better and more complete feedback and specifications. The final product is more likely to satisfy the users desire for look, feel and performance.

Disadvantages of Prototyping Model:

- **Insufficient analysis**: The focus on a limited prototype can distract developers from properly analyzing the complete project. This can lead to overlooking better solutions, preparation of incomplete specifications or the conversion of limited prototypes into poorly engineered final projects that are hard to maintain. Further, since a prototype is limited in functionality it may not scale well if the prototype is used as the basis of a final deliverable, which may not be noticed if developers are too focused on building a prototype as a model.

- **User confusion of prototype and finished system**: Users can begin to think that a prototype, intended to be thrown away, is actually a final system that merely needs to be finished or polished. (They are, for example, often unaware of the effort needed to add error-checking and security features which a prototype may not have.) This can lead them to expect the prototype to accurately model the performance of the final system when this is not the intent of the developers. Users can also become attached to features that were included in a prototype for consideration and then removed from the specification for a final system. If users are able to require all proposed features be included in the final system this can lead to conflict.
Prototyping Model: Pros and Cons

Disadvantages of Prototyping Model:

- **Developer misunderstanding of user objectives:** Developers may assume that users share their objectives (e.g. to deliver core functionality on time and within budget), without understanding wider commercial issues. For example, user representatives attending Enterprise software (e.g. PeopleSoft) events may have seen demonstrations of "transaction auditing" (where changes are logged and displayed in a difference grid view) without being told that this feature demands additional coding and often requires more hardware to handle extra database accesses. Users might believe they can demand auditing on every field, whereas developers might think this is feature creep because they have made assumptions about the extent of user requirements. If the solution provider has committed delivery before the user requirements were reviewed, developers are between a rock and a hard place, particularly if user management derives some advantage from their failure to implement requirements.

- **Developer attachment to prototype:** Developers can also become attached to prototypes they have spent a great deal of effort producing; this can lead to problems like attempting to convert a limited prototype into a final system when it does not have an appropriate underlying architecture. (This may suggest that throwaway prototyping, rather than evolutionary prototyping, should be used.)

- **Excessive development time of the prototype:** A key property to prototyping is the fact that it is supposed to be done quickly. If the developers lose sight of this fact, they very well may try to develop a prototype that is too complex. When the prototype is thrown away the precisely developed requirements that it provides may not yield a sufficient increase in productivity to make up for the time spent developing the prototype. Users can become stuck in debates over details of the prototype, holding up the development team and delaying the final product.

- **Expense of implementing prototyping:** The start up costs for building a development team focused on prototyping may be high. Many companies have development methodologies in place, and changing them can mean retraining, retooling, or both. Many companies tend to just jump into the prototyping without bothering to retrain their workers as much as they should.
Spiral Evolutionary SDLC (SAD) Model
Evolutionary Process Flow: A Spiral Model

The model combines the iterative nature of prototyping AND the controlled and systematic aspects of linear sequential model (inner: concept development, new product development, product enhancement, outer: product maintenance).
Examples of SD Projects with Spiral Evolutionary Model is SE

   - Versions (like 1.1, 1.2., 1.3, 1.4. …) of the same software system are inside a new system (1.00)
   - Usually a new version is created by a new SE/SD team
   - Corporate accumulated knowledge (“know-how”) is involved
   - Usually new generation of a software system is produced in 24-36 months

2. Oracle DBMS: 10i, 11i, 12i, …

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Release Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.0</td>
<td>June-1997</td>
<td>Oracle 8</td>
</tr>
<tr>
<td>8.1.5</td>
<td>February-1999</td>
<td>Oracle 8i Release 1</td>
</tr>
<tr>
<td>8.1.6</td>
<td>November-1999</td>
<td>Oracle 8i Release 2</td>
</tr>
<tr>
<td>8.1.7</td>
<td>August-2000</td>
<td>Oracle 8i Release 3</td>
</tr>
<tr>
<td>9.0</td>
<td>June-2001</td>
<td>Oracle 9i Release 1</td>
</tr>
<tr>
<td>9.0</td>
<td>May-2002</td>
<td>Oracle 9i Release 2</td>
</tr>
<tr>
<td>10.1</td>
<td>January-2004</td>
<td>Oracle 10g Release 1</td>
</tr>
<tr>
<td>10.2</td>
<td>July-2005</td>
<td>Oracle 10g Release 2</td>
</tr>
</tbody>
</table>
Spiral Model: Pros and Cons

**Advantages of the Spiral Model**
- **Evolution of SW** -- large-scale CIS evolve as the SAD process progresses (no need to develop final product from the very beginning)
- **Customer involvement** into SAD process.
- **Good risk management** (with a good risk reduction rate).
- **Good change management**.
- **Careful step-by-step** (“spiral-by-spiral”) **development** of CIS project. It maintains a systematic stepwise approach like the classic life cycle model, but also incorporates an interactivity in order to reflect the real world.

**Disadvantages of the Spiral Model**
- **Time consuming process**
- **Demands considerable risk-assessment expertise**.
CIS 475/575: Topic # 4

SDLC (SAD) Models: HW assignment
Topic # 4:

Homework Assignment

1) This HW is intended to help you to practice with a correspondence between SAD Process Models and specific well-known commercial CIS systems. A hint: create a list of about 15 well-known CIS applications that you actively use, and try to classify them in accordance with learned SAD Process Models.

2) There is NO NEED to email your results to instructor.

3) This HW is aimed at your preparation for In-Classroom Test next class.
Other Process Models

- **Agile methodology** — emphasizes the RAD approach to SAD

- **Component-based (re-usable) software development** — the process to apply when reuse (re-usability) is a development objective

- **Unified Process** — a “use-case driven, architecture-centric, iterative and incremental” software process closely aligned with the Unified Modeling Language (UML)

- **Aspect-Oriented Software Development (AOSD)** — provides a process and methodological approach for defining, specifying, designing, and constructing aspects
CIS 475/575: Topic # 4

Additional information.
SDLC (SAD) Models: Agile Methods
“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.”
Agile Principles

1. Customer Satisfaction
2. Changing Requirements
3. Frequent Delivery
4. Measure of Progress
5. Sustainable Development
6. Close Cooperation
7. Motivated Individuals
8. Face-to-face Conversations
9. Technical Excellence
10. Simplicity
11. Self-organizing teams
12. Regular Adaptation

Source: http://www.biggerplate.com/mapImages/xl/64fc52c5-ce73-4757-b208-b4f6463a5879.png
Main Agile Methods

- Scrum
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Lean Development (Lean Programming)
- Rapid Product Development (RPD)
- Dynamic Systems Development Method (DSDM)
- Crystal Clear Family of methods
- Rational Unified Process (RUP)

Source: http://www.codeproject.com/Articles/604417/Agile-software-development-methodologies-and-how-t
Agile development methodology is a conceptual framework for undertaking any software engineering project. There are a number of agile CIS development methods but the most popular are Extreme Programming (XP) and Scrum. In general, agile methods attempt to minimize risk and maximize productivity by developing software in short iterations and by deemphasizing work on interim work artifacts. Furthermore, it is people oriented and can be a great choice when starting a new project.

These are key characteristics of the agile methodology:
- Deliver frequently
- More iterations
- Less defects
- Test frequently
- Collaborative approach
- Maximum ROI

Source: http://visual.ly/6-characteristics-good-agile-team-member
Scrum method
Main Idea: Iterative, Incremental Product Development

Most important 5…10..20% of functionality
+ ADDITIONAL about 10% of functions
+ 10% of functions, and so on
Scrum Method: Sprint

The sprint is an effort restricted to a specific duration

The duration is fixed in advance for each sprint and is normally between one week and one month, although two weeks is typical.

Scrum emphasizes working product at the end of the Sprint that is really "done". In the case of software, this means a system that is integrated, fully tested, end-user documented.

Most of today's agile teams use iterations to maintain a constant tempo.

The iteration begins when the team selects specific product features to be delivered at the end of the iteration, identifies the tasks needed to complete those features and commits to the iteration.

During the iteration the delivery team meets each day to plan that day.

At the end of the iteration the team produces a potentially shippable product increment.
Agile Development: An Overview

- Agile software engineering represents a **reasonable compromise** between conventional software engineering for certain classes of software and certain types of software projects.

- Agile development processes can **deliver successful systems quickly**.

- Agile development **stresses continuous communication and collaboration** among developers and customers.

- Agile software engineering embraces a **philosophy** that encourages customer satisfaction, incremental software delivery, small project teams (composed of software engineers and stakeholders), informal methods, and minimal software engineering work products.

- Agile software engineering development guidelines **stress on-time delivery** of an operational software increment over analysis and design (the main idea: “the only really important work product is an operational software increment”).
Agility in CIS SAD

- An agile team is able to respond to changes during project development.
- Agile development recognizes that project plans must be flexible.
- Agility encourages team structures and attitudes that make communication among developers and customers more facile.
- Eliminates the separation between customers and developers.
- Agility emphasizes the importance of rapid delivery of operational software and de-emphasizes importance of intermediate work products.
- Agility can be applied to any software process as long as the project team is allowed to streamline tasks and conduct planning in way that eliminate non-essential work products.
- The costs of change increase rapidly as a project proceeds to completion, the earlier a change is made the less costly it will be.
- Agile processes may flatten the cost of change curve by allowing a project team to make changes late in the project at much lower costs.
Agile Process Models

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Scrum
- Dynamic Systems Development Method (DSDM)
- Crystal
- Feature Driven Development (FDD)
- Lean Software Development (LSD)
- Agile Modeling (AM)
- Agile Unified Process (AUP)
<table>
<thead>
<tr>
<th>Factor</th>
<th>Agile Methods</th>
<th>Traditional Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Well matched to small products and teams. Reliance on tacit knowledge limits scalability.</td>
<td>Methods evolved to handle large products and teams. Hard to tailor down to small projects.</td>
</tr>
<tr>
<td>Criticality</td>
<td>Untested on safety-critical products. Potential difficulties with simple design and lack of documentation.</td>
<td>Methods evolved to handle highly critical products. Hard to tailor down to products that are not critical.</td>
</tr>
<tr>
<td>Dynamism</td>
<td>Simple design and continuous refactoring are excellent for highly dynamic environments but a source of potentially expensive rework for highly stable environments.</td>
<td>Detailed plans and Big Design Up Front, excellent for highly stable environment but a source of expensive rework for highly dynamic environments.</td>
</tr>
<tr>
<td>Personnel</td>
<td>Requires continuous presence of a critical mass of scarce experts. Risky to use no-agile people.</td>
<td>Needs a critical mass of scarce experts during project definition but can work with fewer later in the project, unless the environment is highly dynamic.</td>
</tr>
<tr>
<td>Culture</td>
<td>Thrives in a culture where people feel comfortable and empowered by having many degrees of freedom (thriving on chaos).</td>
<td>Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear practices and procedures (thriving on order).</td>
</tr>
</tbody>
</table>
Agile Methodology in Action: real-world examples
Research by Microsoft: Usage of Agile Practices

CIS 475/575: Topic # 4

SDLC (SAD) Models: Other Methods
Rational Unified Process (RUP)

- An object-oriented systems development methodology
- Establishes four phases of development: inception, elaboration, construction, and transition
  - Each phase is organized into a number of separate iterations.
FIGURE 1-13
Phases of OOSAD-based development
eXtreme Programming in SAD

- Short, incremental development cycles
- Automated tests
- Two-person programming teams
- Coding, testing, listening, designing
- Coding and testing operate together
- Advantages:
  - Communication between developers
  - High level of productivity
  - High-quality code