Introduction

Introduction of the systems development life cycle (SDLC), that is common to all information system development projects

Four-phase model (planning, analysis, design, and implementation)

Evolution of system development methodologies.

Roles and skills necessary within a project team.

Key Ideas

• A 1996 survey by the Standish Group found that 42 percent of all corporate IS projects were abandoned before completion.
• A similar study done in 1996 by the General Accounting Office found 53 percent of all U.S. government IS projects were abandoned.
• Many of the systems that aren’t abandoned are delivered to the users significantly late, cost far more than planned, and have fewer features than originally planned.

Key Ideas

• Most of us would like to think that these problems only occur to “other” people or “other” organizations,
  – but they happen in most companies.
• Even Microsoft has a history of failures and overdue projects
  – e.g., Windows 1.0, Windows 95.

Some significant IT project failures

- http://it-project-failures.blogspot.com/
- http://www.colevconsulting.co.uk/failure.htm
- http://www.codinghorror.com/blog/2006/05/the-long-dismal-history-of-software-project-failure.html
- http://www.uml.edu~santier/analysis/0840_03_papers/frase/
- http://braender.tcu.edu/mit400/articles/ProjectManagement/ProjectManagementFailures.pdf

Some significant IT project failures

- http://it-project-failures.blogspot.com/
- http://www.colevconsulting.co.uk/failure.htm
- http://www.codinghorror.com/blog/2006/05/the-long-dismal-history-of-software-project-failure.html
- http://www.uml.edu~santier/analysis/0840_03_papers/frase/
- http://braender.tcu.edu/mit400/articles/ProjectManagement/ProjectManagementFailures.pdf
Key Ideas

• Many failed systems were abandoned because analysts tried to build wonderful systems!
  – without clearly understanding how the system would support the organization’s goals, current business processes, and other information systems.
• The primarily goal is to create value for the organization.
  – Increasing profit for most companies.
  • government agencies and not-for-profit organizations measure value differently

Key Ideas

• An investment in an information system is like any other investment, such as a new machine tool.
  – The goal is not to acquire the tool, because the tool is simply a means to an end;
  – The goal is to enable the organization to perform work better so it can earn greater profits or serve its constituents more effectively.
• There are some fundamental concepts and practical techniques that can be used to improve the probability of success.

Key Ideas

• The key person in the SDLC is the systems analyst
  – who analyzes the business situation, identifies opportunities for improvements, and designs an information system to implement them.
• A systems analyst will work with a variety of people and learn how they conduct business.
  – Specifically, (s)he will work with a team of systems analysts, programmers, and others on a common mission.
• It is important to understand and develop through practice the skills needed to successfully design and implement new information systems.

Participants to system development

• In a typical systems development project, there are the following major categories of players:
  – User
  – Management
  – Auditors, quality assurance people, and “standards bearers”
  – Systems analysts
  – Systems designers
  – Programmers
  – Operations personnel

User

• The most important player in the systems game,
  – the person (or group of people) for whom the system is being built. (S)he is the person whom will be interviewed, often in great detail, to learn what features the new system must have to be successful.
• The user is the “owner” in the sense that (s)he receives, or inherits-and thus owns- the system when it is finally built.

http://www.it-cortex.com/Stat_Failure_Cause.htm
• The user is the “customer” in at least two important respects:
  – As in so many other professions, “the customer is always right”, regardless of how demanding, unpleasant, or irrational (s)he may seem.
  – The customer is ultimately the person paying for the system and usually has the right and/or the ability to refuse to pay if (s)he is unhappy with the product received.
• In most cases, it is fairly easy to identify the user:
  – the user is typically the person who makes a formal request for a system.

• Whenever possible, the systems analyst should try to establish direct contact with the user.
  – Even if other people are involved as intermediaries, it is important to have regular, face-to-face meeting with the person who will ultimately inherit the system.
  – If it is not possible to communicate directly with the user, then the documentation produced by the systems analyst becomes even more crucial.

• To summarize, there are three different types, or levels, of users:
  • Operational:
    – Usually has local view. Carries out the function of the system. Has a physical view of the system.
  • Supervisory user:
    – May or may not have local view. Generally familiar with operation. Driven by budget considerations. Often acts as a middleman between users and higher levels of management.
  • Executive user:
    – Has a global view. Provides initiative for the project. No direct operating experience. Has a strategic concern.

• Management is a rather loose term. The systems analyst is likely to come into contact with several different kinds of managers:
  • User managers:
    – managers in charge of several people in the operational area where the new system will be used. These are usually middle-level managers who want systems that will produce a variety of internal reports and short-term trend analyses.
  • Executive development project (EDP)/MIS managers:
    – the person in charge of the systems development project itself, and the higher-level managers who are concerned with the overall management and allocation of resources of all the technical staff in the systems development organization.
  • General management:
    – top-level managers who are not directly involved in the EDP organization or in the user organization. This might include the president and/or chairman of the organization.

• The primary interaction between the systems analyst and all these managers has to do with the resources that will be assigned to the project.
  • It is the systems analyst’s job to identify and document the user’s requirements and the constraints within which the system must be built.
Auditors

- Depending on the size of the project and the nature of the organization you work in, you may or may not have auditors.
  - An auditor is an individual qualified at the state level to perform financial and accounting audits.
  - An auditor examines, corrects and verifies the accuracy of financial accounts related to businesses, non-profit organizations and government agencies. The auditor is expected to perform an unbiased evaluation. An auditor can be an internal employee or an external consultant.

Systems analysts

- The system analyst is a key member of any systems development project.
- The systems analyst plays several roles:
  - Archaeologist and scribe:
    - As a systems analyst, one of the main jobs is to uncover detail and to document business policy that may exist only as “tribal folklore”, passed down from generation to generation of users.
  - Innovator:
    - The systems analyst must separate the symptoms of the user’s problem from the true causes. With his or her knowledge of computer technology, the analyst must help the user explore useful, new applications of computers.

Systems designers

- The systems designer is the person (or group of people) who will receive the output of the systems analysis work.
- His or her job is to transform a technology-free statement of user requirements into a high-level architectural design that will provide the framework within which the programmer can work.
- In many cases, the systems analyst and the systems designer are the same person, or member of the same unified group of people.
- It is important for the systems analyst and systems designer to stay in close touch throughout the project.

Programmers

- Particularly on large systems development projects, the systems designers are likely to be a “buffer” between the systems analysts and the programmers.
- The systems analysts deliver their product to the system designers, and the system designers deliver their product to the programmer.
- There is another reason why the systems analyst and the programmer may have little or no contact with each other:
  - Work is often performed in a strictly serial sequence in many systems development projects. Thus, the work of systems analysis takes place first and is completely finished before the work of programming begins.

Operations personnel

- The operations personnel are responsible for the computer center, telecommunications network, security of the computer hardware and data, as well as the actual running of computer programs, mounting of disk packs, and handling of output from computer printers.
- All this happens after a new system has not only been analyzed and designed, but has also been programmed and tested.
• In many ways, building an information system is similar to building a house.
  – 1st, the house starts with a basic idea.
  – 2nd, this idea is transformed into a simple drawing that is shown to the customer and refined until the customer agrees that the picture depicts what(s)he wants.
  – 3rd, a set of blueprints is designed that presents much more detailed information about the house.
  – Finally, the house is built following the blueprints • and often with some changes and decisions made by the customer as the house is erected.

• Different projects may emphasize different parts of the SDLC or approach the SDLC phases in different ways, • but all projects have elements of these four phases.
• Each phase is itself composed of a series of steps,
  – which rely on techniques that produce deliverables • specific documents and files that provide understanding about the project

• Following two slides suggest that the SDLC phases and steps proceed in a logical path from start to finish.
• In some projects, this is true, but in many projects, the project teams move through the steps consecutively, incrementally, iteratively, or in other patterns.
• In practice, an organization may follow one of many variations on the overall SDLC.
Planning Phase

- The IS department works together with the person or department that generated the request (called the project sponsor) to conduct a feasibility analysis.
- The feasibility analysis examines key aspects of the proposed project:
  - The technical feasibility (Can we build it?)
  - The economic feasibility (Will it provide business value?)
  - The organizational feasibility (If we build it, will it be used?)
- The system request and feasibility analysis are presented to an information systems approval committee (sometimes called a steering committee), which decides whether the project should be undertaken.

Planning Phase

1. During project initiation, the system’s business value to the organization is identified:
   - how will it lower costs or increase revenues?
   - Most ideas for new systems come from outside the IS area (from the marketing department, accounting department, etc.) in the form of a system request.
   - A system request presents a brief summary of a business need, and it explains how a system that supports the need will create business value.

Planning Phase

2. Once the project is approved, it enters project management.
   - During project management, the project manager creates a workplan, staffs the projects, and puts techniques in place to help the project team control and direct the project through the entire SDLC.
   - The deliverable for project management is a project plan that describes how the project team will go about developing the system.
Analysis Phase

• answers the questions of
  – who will use the system,
  – what the system will do, and
  – where and when it will be used.

• During this phase, the project team
  – investigates any current system(s),
  – identifies improvement opportunities, and
  – develops a concept for the new system.

Analysis Phase

• has three steps:
  1. An analysis strategy is developed to guide the project team’s efforts.
     – Such a strategy usually includes an analysis of the current system and its problems, and then ways to design a new system.
  2. The next step is requirements gathering through interviews or questionnaires.
     – The analysis of this information leads to the development of a concept for a new system.

Analysis Phase

• The system concept is then used as a basis to develop a set of business analysis models that describes how the business will operate if the new system were developed.

3. The analyses, system concept, and models are combined into a document called the system proposal, which is presented to the project sponsor and other key decision makers that decide whether the project should continue to move forward.

Design Phase

• decides how the system will operate, in terms of the
  – hardware, software, and network infrastructure;
  – the user interface, forms, and reports that will be used;
  – the specific programs, databases, and files that will be needed.

• Although most of the strategic decisions about the system were made in the development of the system concept during the analysis phase, the steps in the design phase determine exactly how the system will operate.

Design Phase

• The design phase has four steps:
  1. The design strategy must be developed.
     – This clarifies whether the system will be developed by the company’s own programmers, whether it will be outsourced to another firm or whether the company will buy an existing software package.
  2. This leads to the development of the basic architecture design for the system that describes the hardware, software, and network infrastructure that will be used.
     – The interface design specifies how the users will move through the system and the forms and reports that the system will use.
  3. The database and file specifications are developed.
     – These define exactly what data will be stored and where they will be stored.
  4. The analyst team develops the program design, which defines the programs that need to be written and exactly what each program will do.

• This collection of deliverables (architecture design, interface design, database and file specifications, and program design) is the system specification that is handed to the programming team for implementation.

• At the end of the design phase, the feasibility analysis and project plan are reexamined and revised, and another decision is made by the project sponsor and approval committee about whether to terminate the project or continue.
Implementation Phase

- is the final phase during which the system is actually built (or purchased, in the case of a packaged software design).
- This phase has three steps:
  1. System construction is the first step.
     - The system is built and tested to ensure it performs as designed.
     - Since the cost of bugs can be immense, testing is one of the most critical steps in implementation.
     - Most organizations spend more time and attention on testing than on writing the programs in the first place.

Implementation Phase

2. The system is installed.
   - Installation is the process by which the old system is turned off and the new one is turned on.
3. The analyst team establishes a support plan for the system.
   - This plan usually includes a formal or informal post-implementation review, as well as a systematic way for identifying major and minor changes needed for the system.

Processes and Deliverables

<table>
<thead>
<tr>
<th>Process</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>System Request</td>
</tr>
<tr>
<td></td>
<td>Feasibility Analysis</td>
</tr>
<tr>
<td></td>
<td>Workplan</td>
</tr>
<tr>
<td>Analysis</td>
<td>System Proposal</td>
</tr>
<tr>
<td>Design</td>
<td>System Specification</td>
</tr>
<tr>
<td>Implementation</td>
<td>New System and Maintenance Plan</td>
</tr>
</tbody>
</table>

SYSTEM DEVELOPMENT METHODOLOGIES

A methodology is a formalized approach to implementing the SDLC.

It is a list of steps and deliverables.

There are many different systems development methodologies and each one is unique because of its emphasis on processes versus data and the order and focus it places on each SDLC phase.

- Some methodologies are formal standards used by government agencies.
- Others have been developed by consulting firms to sell to clients.
- Many organizations have their own internal methodologies that have been refined over the years, and they explain exactly how each phase of the SDLC is to be performed in that company.

- All system development methodologies lead to a representation of the system concept in terms of processes and data;
  - they vary in terms of whether the methodology places primary emphasis on business processes or on the data that supports the business.
- As an illustration, refer to the diagram shown in the next slide, depicting the activities and information used in producing the payroll for an organization.
A Simple Model for Employee Payroll

- The open-ended rectangles represent data storage containers;
- The rounded rectangles represent activities performed;
- The arrows represent activity inputs and outputs.

Methodology Categories

- Methodologies in terms of processes and data
  - Process-Centered methodologies
  - Data-Centered methodologies
  - Object-Oriented methodologies
- Methodologies in terms of the sequencing of the SDLC phases and the amount of time and effort devoted to each
  - Structured Design
  - Rapid Application Development
  - Agile Development

Process-centered methodologies

- focus first on defining the activities associated with the system, i.e., the processes.
- utilize process models as the core of the system concept.
- Analysts concentrate initially on representing the system concept as a set of processes with information flowing into and out of the processes

Data-centered methodologies

- focus first on defining the contents of the data storage containers and how the contents are organized.
- utilize data models as the core of the system concept.
- For example, analysts concentrate initially on identifying the data that must be available to produce the payroll and organizing that data into well-defined structures
  - e.g., employee work log, employee pay rates, payroll tax tables, employee pay history, etc.

Object-oriented methodologies

- attempt to balance the focus between processes and data.
- utilize the Unified Modeling Language (UML) to describe the system concept as a collection of objects incorporating both data and processes.
Another important factor in categorizing methodologies is the sequencing of the SDLC phases and the amount of time and effort devoted to each.

In the early days of computing, the need for formal and well-planned life cycle methodologies was not well understood.

Programmers tended to move directly from a very simple planning phase right into the construction step of the implementation phase; in other words, they moved directly from a very fuzzy, not-well-thought-out system request into writing code.

In the following few slides, three major categories of systems development methodologies that have evolved over time are described:

- Structured Design,
- Rapid Application Development (RAD),
- Agile Development.

Each category represents a collection of methodologies that attempts to improve on previous practice, and varies in terms of the progression through the SDLC phases and the emphasis placed on each phase.

Methodology Categories

Structured design methodologies

- became dominant in the 1980s, replacing the previous ad hoc and undisciplined approach.
- adopt a formal step-by-step approach to the SDLC that moves logically from one phase to the next.
- introduced the use of formal modeling or diagramming techniques to describe a system’s basic business processes and the data that support them.

Traditional structured design uses one set of diagrams to represent the processes (process models) and a separate set of diagrams to represent data (data models).

- Because two sets of models are used, the systems analyst must decide which set to develop first and use as the core of the system—process models or data models.
- Since each type of model is important to the system, there is much debate over whether to emphasize processes before data or vice versa.

Numerous process-centered and data-centered methodologies follow the basic approach of the two structured design categories outlined next.

Structured design methodologies

1. Waterfall Development Methodology

- is original structured design methodology that is still used today.
- With waterfall development-based methodologies, the analysts and users proceed sequentially from one phase to the next.
- The key deliverables for each phase are typically voluminous (often hundreds of pages in length) and are presented to the project sponsor for approval as the project moves from phase to phase.

Once the sponsor approves the work that was conducted for a phase, the phase ends and the next one begins.

This methodology is called waterfall development because it moves forward from phase to phase in the same manner as a waterfall.

Although it is possible to go backward in the SDLC (e.g., from design back to analysis), it is extremely difficult—imagine yourself as a salmon trying to swim upstream in a waterfall as shown in next slide.

1. Waterfall Development Methodology
1. Waterfall Development Methodology

Pros and Cons of the Waterfall Methodology

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifies systems requirements long</td>
<td>Design must be specified on paper before</td>
</tr>
<tr>
<td>before programming begins</td>
<td>programming begins</td>
</tr>
<tr>
<td>Minimizes changes to requirements as</td>
<td>Long time between system proposal and</td>
</tr>
<tr>
<td>project progresses</td>
<td>delivery of new system</td>
</tr>
</tbody>
</table>

2. Parallel Development Methodology

Pros and Cons of Parallel Development Methodology

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduces Schedule Time</td>
<td>Still Uses Paper Documents</td>
</tr>
<tr>
<td>Less Chance of Rework</td>
<td>Sub-projects May Be Difficult to Integrate</td>
</tr>
</tbody>
</table>

• Today’s dynamic business world often imposes rapid environmental change on businesses.
• A system that meets existing environmental conditions during the analysis phase may need considerable rework to match the environment when it is implemented.
• This rework requires going back to the initial phase and making needed changes through each of the subsequent phases in turn.

• attempt to address the long time interval between the analysis phase and the delivery of the system.
• Instead of doing the design and implementation in sequence, a general design for the whole system is performed, then the project is divided into a series of distinct subprojects that can be designed and implemented in parallel.
• Once all subprojects are complete, there is a final integration of the separate pieces, and the system is delivered (next slide).
RAD-based methodologies are a newer class of system development methodologies that emerged in the 1990s in response to both structured design methodology weaknesses. RAD-based methodologies adjust the SDLC phases to get some part of the system developed quickly and into the hands of the users. In this way, the users can better understand the system and suggest revisions that bring the system close to what is needed. There are process-centered, data-centered, and object-oriented methodologies that follow the basic approaches of the three RAD categories (will be described in the coming slides).

Rapid Application Development (RAD)

- Rapid Application Development

Most RAD-based methodologies recommend that analysts use special techniques and computer tools to speed up the analysis, design, and implementation phases, such as:
  - CASE (computer-aided software engineering) tools
  - JAD (joint application design) sessions
  - Fourth generation/visualization programming languages
  - Code generators

It is the combination of the changed SDLC phases and the use of these tools and techniques that improves the speed and quality of systems development.

Rapid Application Development

Three RAD Categories

- Phased development
  - A series of versions developed sequentially
- Prototyping
  - System prototyping
- Throw-away prototyping
  - Design prototyping

Phased development-based methodologies

- Phased development-based methodologies

break the overall system into a series of versions that are developed sequentially.

The analysis phase identifies the overall system concept, and the project team, users, and system sponsor then categorize the requirements into a series of versions.

The most important and fundamental requirements are bundled into the first version of the system.

The analysis phase then leads into design and implementation, but only with the set of requirements identified for version 1 (see next figure).

Phased development-based methodologies

- Phased development-based methodologies

Once version 1 is implemented, work begins on version 2.

Additional analysis is performed on the basis of the previously identified requirements and combined with new ideas and issues that arose from users’ experience with version 1.

Version 2 then is designed and implemented, and work immediately begins on the next version.

This process continues until the system is complete or is no longer in use.
**Phased development-based methodologies**

- have the advantage of quickly getting a useful system into the hands of the users.
- begins to provide business value sooner than if the system were delivered only after all requirements are completed.
  - Likewise, because users begin to work with the system sooner, they are more likely to identify important additional requirements sooner than with structured design situations.
- The major drawback is that users begin to work with systems that are intentionally incomplete.
  - It is critical to identify the most important and useful features and include them in the first version while managing users' expectations along the way.

**Pros and Cons of Phased Development Methodology**

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users Get a System To Use Quickly</td>
<td>Users Work with a System that is Intentionally Incomplete</td>
</tr>
<tr>
<td>Users Can Identify Additional Needs For Later Versions</td>
<td></td>
</tr>
</tbody>
</table>

**Prototyping-based methodologies**

- perform the analysis, design, and implementation phases concurrently, and all three phases are performed repeatedly in a cycle until the system is completed.
- With these methodologies, a basic analysis and design are performed, and work immediately begins on a system prototype, a "quick-and-dirty" program that provides a minimal amount of features.
- The first prototype is usually the first part of the system that the user will use.
  - This is shown to the users and the project sponsor, who provide reaction and comments.
  - This feedback is used to reanalyze, redesign, and reimplement a second prototype that provides a few more features.
- This process continues in a cycle until the analysts, users, and sponsor agree that the prototype provides enough functionality to be installed and used in the organization.

**Prototyping-based methodologies**

- After the prototype (now called the "system") is installed, refinement occurs until it is accepted as the new system — as seen in the following figure.

**Advantages**

- very quickly provides a system for the users to interact with,
  - even if it is not initially ready for widespread organizational use.
- Prototyping reassures the users that the project team is working on the system, and the approach helps to more quickly refine real requirements.
- Rather than attempting to understand system specification materials, the users can interact with the prototype to better understand what it can and cannot do.

**Disadvantages**

- The major problem with prototyping is that its fast-paced system releases challenge attempts to conduct careful, methodical analysis.
- Often the prototype undergoes such significant changes that many initial design decisions prove to be poor ones.
  - This can cause problems in the development of complex systems because fundamental issues and problems are not recognized until well into the development process.
  - Imagine building a car and discovering late in the prototyping process that you have to take the whole engine out to change the oil (because no one thought about the need to change the oil until after the car had been driven 10,000 miles).
Pros and Cons of Prototyping Methodology

Pros
- Users Interact with Prototype Very Quickly
- Users Can Identify Needed Changes And Refine Real Requirements

Cons
- Tendency to do Superficial Analysis
- Initial Design Decisions May Be Poor

Throwaway prototyping-based methodologies

• are similar to the prototyping-based methodologies in that they include the development of prototypes;
• however, throwaway prototypes are done at a different point in the SDLC.
• These prototypes are used for a very different purpose than ones previously discussed, and they have a very different appearance (see next figure).

Throwaway prototyping-based methodologies

• have a relatively thorough analysis phase that is used to gather information and to develop ideas for the system concept.
  – However, many of the features suggested by the users may not be well understood and there may be challenging technical issues to be solved.
  – Each of these issues is examined by analyzing, designing, and building a design prototype.
• A design prototype is not a working system;
  – it is a product that represents a part of the system that needs additional refinement, and it contains only enough detail to enable users to understand the issues under consideration.

Throwaway prototyping-based methodologies

• For example, suppose users are not completely clear on how an order entry system should work.
  – The analyst team might build a series of HTML pages viewed using a Web browser to help the users visualize such a system.
  – In this case, a series of mock-up screens appear to be a system, but they really do nothing.
• Or, suppose that the project team needs to develop a sophisticated graphics program in Java.
  – The team could write a portion of the program with artificial data to ensure that they could create a full-blown program successfully.

Throwaway prototyping-based methodologies

• A system that is developed using this type of methodology probably uses several design prototypes during the analysis and design phases.
  – Each of the prototypes is used to minimize the risk associated with the system by confirming that important issues are understood before the real system is built.
• Once the issues are resolved, the project moves into design and implementation.
  – At this point, the design prototypes are thrown away, which is an important difference between this approach and prototyping, in which the prototypes evolve into the final system.
Advantages-Disadvantages

- Throwaway prototyping-based methodologies balance the benefits of wellthought-out analysis and design phases with the advantages of using prototypes to refine key issues before a system is built.
- It may take longer to deliver the final system as compared with prototyping-based methodologies (because the prototypes do not become the final system), but the approach usually produces more stable and reliable systems.

Pros and Cons of Throwaway Prototyping Methodology

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risks are Minimized</td>
<td>May Take Longer Than Prototyping</td>
</tr>
<tr>
<td>Important Issues are Understood Before the Real System is Built</td>
<td></td>
</tr>
</tbody>
</table>

Agile Development methodologies

- These programming-centric methodologies have few rules and practices, all of which are fairly easy to follow.
  - They focus on streamlining the SDLC by eliminating much of the modeling and documentation overhead and the time spent on those tasks.
  - Instead, projects emphasize simple, iterative application development.
- Examples of Agile Development methodologies include extreme programming, Scrum, and the Dynamic Systems Development Method (DSDM).
  - To illustrate an agile development methodology, extreme programming will be described in the next slides.

Extreme programming (XP)

- is founded on four core values:
  - communication,
  - simplicity,
  - feedback,
  - courage.
- These four values provide a foundation XP developers use to create any system.

Extreme programming (XP)

- 1st, the developers must provide rapid feedback to the end users on a continuous basis.
- 2nd, XP requires developers to follow the KISS (Keep It Simple, Stupid) principle.
- 3rd, developers must make incremental changes to grow the system and they must embrace change, not merely accept it.
- 4th, developers must have a qualityfirst mentality.
- XP also supports team members in developing their own skills.
**Extreme programming (XP)**

- Testing and efficient coding practices are core to XP.
  - Each day code is tested and placed into an integrative testing environment.
  - If bugs exist, the code is backed out until it is completely free of errors.
- XP relies heavily on refactoring,
  - which is a disciplined way to restructure code to keep it simple.
- Standards are very important to minimize confusion,
  - so XP teams use a common set of names, descriptions, and coding practices.

**Extreme programming (XP)**

- For small projects with highly motivated, cohesive, stable, and experienced teams, XP should work just fine.
- However, if the project is not small then the likelihood of a successful XP project is reduced.
- Consequently, the use of XP in combination with outside contractors produces a highly questionable outcome, since the outside contractors may never “jell” with insiders.
- XP requires a great deal of discipline to prevent projects from becoming unfocused and chaotic.

**Extreme programming (XP)**

- Furthermore, it is only recommended for small groups of developers (not more than ten).
  - Since little analysis and design documentation is produced with XP there is only code documentation; therefore, maintenance of large systems developed using XP may be impossible.
- It is not advised for mission-critical applications.
  - Since mission-critical business information systems tend to exist for a long time, the utility of XP as a business information system development methodology is in doubt.
- Finally, the methodology requires considerable on-site user input, something that is frequently difficult to obtain.

**Pros and Cons of Agile Methodologies**

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast Delivery of Results</td>
<td>Requires Discipline</td>
</tr>
<tr>
<td>Works Well in Projects With Undefined or</td>
<td>Works Best in Small Projects</td>
</tr>
<tr>
<td>Changing Requirements</td>
<td>Requires Much User Input</td>
</tr>
</tbody>
</table>

**Criteria for Selecting the Appropriate Methodology**

- Since there are many methodologies, the first challenge faced by analysts is to select which methodology to use.
- Choosing a methodology is not simple, because no one methodology is always best.
- Many organizations have standards and policies to guide the choice of methodology.
- You will find that organizations range from having one “approved” methodology to having several methodology options to having no formal policies at all.
Clarity of User Requirements

- When the user requirements for what the system should do are unclear, it is difficult to understand them by talking about them and explaining them with written reports.
- Users normally need to interact with technology to really understand what the new system can do and how to best apply it to their needs.
- The RAD methodologies of prototyping and throwaway prototyping are usually more appropriate when user requirements are unclear
  - because they provide prototypes for users to interact with early in the SDLC.
- Agile development may also be appropriate if on-site user input is available.

Familiarity with Technology

- When the system will use new technology with which the analysts and programmers are not familiar, applying the new technology early in the methodology will improve the chance of success.
- If the system is designed without some familiarity with the base technology, risks increase
  - because the tools may not be capable of doing what is needed.
- Throwaway prototyping-based methodologies are particularly appropriate for a lack of familiarity with technology
  - because they explicitly encourage the developers to create design prototypes for areas with high risks.

Familiarity with Technology

- Phased development-based methodologies are good as well
  - because they create opportunities to investigate the technology in some depth before the design is complete.
- While one might think prototyping-based methodologies would also be appropriate, they are much less so,
  - because the early prototypes that are built usually only scratch the surface of the new technology.
  - Usually, it is only after several prototypes and several months that the developers discover weaknesses or problems in the new technology.

System Complexity

- Complex systems require careful and detailed analysis and design.
- Throwaway prototyping-based methodologies are particularly well suited to such detailed analysis and design
- Prototyping-based methodologies are not suited.
- The traditional structured design-based methodologies can handle complex systems,
  - but without the ability to get the system or prototypes into users’ hands early on,
  - some key issues may be overlooked.

System Complexity

- The phased development based methodologies enable users to interact with the system early in the process
  - However, it is observed that project teams who follow these methodologies tend to devote less attention to the analysis of the complete problem domain than they might if they were using other methodologies.

System Reliability

- System reliability is usually an important factor in system development.
- However, reliability is just one factor among several.
  - For some applications reliability is truly critical
    • medical equipment, missile control systems
  - For other applications it is merely important
    • games, Internet video
System Reliability

• Throwaway prototyping-based methodologies are most appropriate
  – when system reliability is a high priority,
    • because they combine detailed analysis and design phases with the ability for the project team to test many different approaches through design prototypes before completing the design.
• Prototyping-based methodologies are generally not a good choice
  – when reliability is critical
    • because they lack the careful analysis and design phases that are essential for dependable systems.

Short Time Schedules

• Projects that have short time schedules are well suited for RAD-based methodologies
  – because those methodologies are designed to increase the speed of development.
• Prototyping and phased development-based methodologies are excellent choices
  – when timelines are short
    • because they best enable the project team to adjust the functionality in the system on the basis of a specific delivery date.
    • If the project schedule starts to slip, it can be readjusted by removing functionality from the version or prototype under development.

Criteria for Selecting a Methodology

TEAM ROLES AND SKILLS

The project team needs a variety of skills. Project members are change agents who identify ways to improve an organization, build an information system to support them, and train and motivate others to use the system. Leading a successful organizational change effort is one of the most difficult jobs that someone can do.
• Understanding what to change, how to change it, and convincing others of the need for change requires a wide range of skills.
• These skills can be broken down into six major categories:
  – technical,
  – business,
  – analytical,
  – interpersonal,
  – management,
  – ethical.

• Analysts must have the technical skills to understand
  – the organization’s existing technical environment,
  – the new system’s technology foundation,
  – the way in which both can be fit into an integrated technical solution.
• Business skills are required
  – to understand how IT can be applied to business situations
  – to ensure that the IT delivers real business value.
• Analysts are continuous problem solvers at both the project and the organizational level, and they put their analytical skills to the test regularly.

• Analysts need to communicate effectively with
  – users and business managers
    • who often have little experience with technology
  – programmers
    • who often have more technical expertise than the analyst.
• They must be able to give presentations to large and small groups and write reports.
• They need to have strong interpersonal abilities.
• They need to manage people with whom they work
• They must manage the pressure and risks associated with unclear situations.

• Finally, analysts must deal fairly, honestly, and ethically with
  – other project team members,
  – managers,
  – system users.
• Analysts often deal with confidential information or information that, if shared with others, could cause harm
  – e.g., dissent among employees.
• It is important to maintain confidence and trust with all people.

Information Systems Roles

• Business analyst
• Systems analyst
• Infrastructure analyst
• Change management analyst
• Project manager

Business analyst

• focuses on the business issues surrounding the system. These include
  – identifying the business value that the system will create,
  – developing ideas and suggestions for how the business processes can be improved,
  – designing the new processes and policies in conjunction with the systems analyst.
**Business analyst**

- This individual will likely have business experience and some type of professional training
  - For example, the business analyst for accounting systems will likely be a
    - CPA [certified public accountant in the United States]
    - CA [chartered accountant in Great Britain and Canada]
    - diplomalı muhasebeci, yetkili hesap uzmanı in TR
  - He or she represents the interests of the project sponsor and the ultimate users of the system.
  - The business analyst assists in the planning and design phases but is most active in the analysis phase.

**Systems analyst**

- focuses on the IS issues surrounding the system.
- This person
  - develops ideas and suggestions for how IT can improve business processes,
  - designs the new business processes with help from the business analyst,
  - designs the new information system, and ensures that all IS standards are maintained.

**Systems analyst**

- The systems analyst likely will have significant training and experience in
  - analysis and design
  - programming
  - areas of the business
- He or she represents the interests of the IS department and works intensively throughout the project
  - but perhaps less so during the implementation phase.

**Infrastructure analyst**

- focuses on the technical issues surrounding how the system will interact with the organization’s technical infrastructure e.g.,
  - hardware,
  - software,
  - networks,
  - databases.
  - The infrastructure analyst’s tasks include
    - ensuring that the new information system conforms to organizational standards
    - identifying infrastructure changes needed to support the system.

**Infrastructure analyst**

- This individual will likely have significant training and experience in
  - networking
  - database administration
  - various hardware and software products
- He or she represents the interests of the organization and IS group that ultimately will have to operate and support the new system once it has been installed.
- The infrastructure analyst works throughout the project
  - but perhaps less so during planning and analysis phases.

**Change management analyst**

- focuses on the people and management issues surrounding the system installation.
- The roles of this person include
  - ensuring that adequate documentation and support are available to users
  - providing user training on the new system,
  - developing strategies to overcome resistance to change
- This individual likely will have significant training and experience in organizational behavior in general and change management in particular.
Change management analyst

- He or she represents the interests of the project sponsor and users
  - for whom the system is being designed.
- The change management analyst works most actively during the implementation phase
  - but begins laying the groundwork for change during the analysis and design phases.

Project manager

- is responsible for ensuring that
  - the project is completed on time and within budget
  - the system delivers all benefits that were intended by the project sponsor.
- The role of the project manager includes
  - managing the team members,
  - developing the project plan,
  - assigning resources,
  - being the primary point of contact when people outside the team have questions about the project.

Project manager

- This individual likely will have significant experience in project management and likely has worked for many years as a systems analyst beforehand.
- He or she represents the interests of the IS department and the project sponsor.
- The project manager works intensely during all phases of the project.

Summary

- The Systems Development Lifecycle consists of four stages:
  - Planning,
  - Analysis,
  - Design,
  - Implementation

Summary

- There are six major development methodologies:
  - the waterfall method,
  - the parallel development method,
  - the phased development method,
  - system prototyping,
  - design prototyping,
  - agile development.
Summary

• There are five major team roles:
  – business analyst,
  – systems analyst,
  – infrastructure analyst,
  – change management analyst and project manager.