Software Development Life Cycle

The SDLC

- The Systems Development Life Cycle (SDLC) is the process of understanding how an information system (IS) can support the business needs, designing the system, building it and delivering it to the users."

  System Analysis Design, Dennis, Wixon & Roth

The System Analyst

- The System Analyst analyzes the business situation, identifies opportunities for improvements and designs an information system to implement them.”

  System Analysis Design, Dennis, Wixon & Roth

The Primary Goal

- The primary goal is creating value to the organization.
- Creating a wonderful system is not the primary goal!

SDLC Fundamental Phases

- The SDLC includes the following four fundamental phases.

  Planning => Project Plan

  Why the system is needed? How to plan the project? Identify opportunities and analyze feasibilities. Creating the project plan. Study.

  Analysis => System Proposal

  Choose an analysis strategy, Use cases, Model processes, Model data.

  Design => System Specification

  How the system will work? System Architecture, Software & Hardware Spec, Interface Design, Process Model, Program Design, Database and File Spec.

  Implementation => Installed System

  Testing plan, Code, Migration plan, Documentation, Support plan, Maintaining.

Methodology & Methods

- Methodology is defined as "the analysis of the principles of methods, rules, and postulates employed by a discipline", "the systematic study of methods that are, can be, or have been applied within a discipline" or "a particular procedure or set of procedures". (Wikipedia)

System Development Methodology

- System development methodology is a formalized approach to implementing the SDLC.

  The software development methodology can be described using a list of steps and deliverables.

Structured Methodologies

- Formal step by step approach. Moving logically from one phase the next.

Waterfall Development

Moving forward from phase to phase...

Parallel Development

Instead of doing the the design and the implementation in a sequence, a general partial design for the whole project is implemented... the project is divided into small projects...

Rapid Development Methodologies

- Rapidly getting some part of the system up and running.

  Getting users feedback to ensure the whole system progresses in the right direction.

  ### Phased Development

  The system is sequentially developed in separated versions.

  ### Prototype Development

  A prototype is first developed.

  ### Throwaway Prototype Development

  Throwaway prototypes are done at various points in the SDLC.

Agile Methodologies

- Programming centric methodologies.
- Few simple rules to follow.
- Eliminating much of the modeling and documentation overhead.
- Simple iterative development.

The well known agile methodologies include the following:

Lean, Scrum & Extreme Programming.

Iterative & Incremental Development

- The main problems the Waterfall methodology has include the following:

  ### No Feedback Between The Phases

  Once Analysis is done everyone should be working on Design. Once Design is done everybody should be working on Implementation. Each phases, when it is done, it is done.

  ### Phases Have Rigid Dates

  The dates set for completing the Analysis and the Design phases are major milestones Developers should meet. This is the common criteria for measuring them.

  ### Analysis & Design Phases Have No Completion Criteria

  There is no completion criteria neither for the analysis or the design phases.

Iterative & Incremental Development

- The iterative and incremental development (IID) approach starts with subdividing our project into small binary deliverables.

  Each binary deliverable is defined as an executing segment of the overall project.

Iterative & Incremental Development

- Subdividing our project into small binary deliverables should be completed in accordance with the following rules:

  ### They should be vertical

  They cannot be subsystems and they should cut across as little as possible of the system functionality (e.g. I/O module cannot be a valid slice).

  ### They should represent features

  This way, if later we choose to eliminate a feature it would be as simply as eliminating the slice of that feature.

Iterative & Incremental Development

They should be executable and demonstrable

Each slice should have a completion and an acceptance criteria. This way it would be easy to evaluate whether a slice has successfully completed or not.

They should be simple to complete

Complete the development of each slice should be relatively easy and should take days or weeks. Having a slice that its completion takes months would be very bad.

They should be roughly the same in terms of manpower

Completing each one of the slices should take roughly the same amount of man power. This way we will have similar human risk involved with the development of each one of the slices.

Iterative & Incremental Development

- Once the slice partitioning is completed and the acceptance criteria for each one of them is set we can start developing the first slice.
- The first slice we develop better be the one with the highest development risk, error prone and complex one.

  Starting with the riskiest slice we will know the earliest whether or not the project will succeed. In addition, that will assist us to be conservative as possible with our project development estimation.

Iterative & Incremental Development

- When the development of the first slice is completed we can get the approximate end date of the project.

  Multiplying the duration it took us to develop that slice by the number of remaining slices will get the approximate end date of the project.

- Once the development of the first slice is completed we can move forward with the second slice or even to increase the manpower on the project and start two or three slices simultaneously.

Object Modeling with Use Cases

- Using UML Use Case diagrams can be a structured way for capturing the behavioral requirements of the system.
- The UML Use Case diagrams help us answering fundamental basic questions such as:

  Who are the users of the system?

  What are the users trying to do?

  What is their experience working with the system?

Object Modeling with Use Cases

- The UML Use Case diagrams would be the first to use. Based on them it would be possible to proceed with the system development.

  It is highly common to include Use Case diagrams in the System Requirements Specification (SRC) document.

Software Development Life Cycle

The SDLC

- The Systems Development Life Cycle (SDLC) is the process of understanding how an information system (IS) can support the business needs, designing the system, building it and delivering it to the users."

  System Analysis Design, Dennis, Wixon & Roth

The System Analyst

- The System Analyst analyzes the business situation, identifies opportunities for improvements and designs an information system to implement them."

System Analysis Design, Dennis, Wixon & Roth

The Primary Goal

- The primary goal is creating value to the organization.
- Creating a wonderful system is not the primary goal!

SDLC Fundamental Phases

- The SDLC includes the following four fundamental phases.

  Planning => Project Plan

  Why the system is needed? How to plan the project? Identify opportunities and analyze feasibilities. Creating the project plan. Study.

  Analysis => System Proposal

  Choose an analysis strategy, Use cases, Model processes, Model data.

  Design => System Specification

  How the system will work? System Architecture, Software & Hardware Spec, Interface Design, Process Model, Program Design, Database and File Spec.

  Implementation => Installed System

  Testing plan, Code, Migration plan, Documentation, Support plan, Maintaining.

Methodology & Methods

- Methodology is defined as "the analysis of the principles of methods, rules, and postulates employed by a discipline", "the systematic study of methods that are, can be, or have been applied within a discipline" or "a particular procedure or set of procedures". (Wikipedia)

System Development Methodology

- System development methodology is a formalized approach to implementing the SDLC.

  The software development methodology can be described using a list of steps and deliverables.

Structured Methodologies

- Formal step by step approach. Moving logically from one phase the next.

Waterfall Development

Moving forward from phase to phase...

Parallel Development

Instead of doing the the design and the implementation in a sequence, a general partial design for the whole project is implemented... the project is divided into small projects...

Rapid Development Methodologies

- Rapidly getting some part of the system up and running.
  Getting users feedback to ensure the whole system progresses in the right direction.

  Phased Development

  The system is sequentially developed in separated versions.

  Prototype Development

  A prototype is first developed.

  Throwaway Prototype Development

  Throwaway prototypes are done at various points in the SDLC.

Agile Methodologies

- Programming centric methodologies.
- Few simple rules to follow.
- Eliminating much of the modeling and documentation overhead.
- Simple iterative development.

The well known agile methodologies include the following:

Lean, Scrum & Extreme Programming.

Iterative & Incremental Development

- The main problems the Waterfall methodology has include the following:

  No Feedback Between The Phases

  Once Analysis is done everyone should be working on Design. Once Design is done everybody should be working on Implementation. Each phases, when it is done, it is done.

  Phases Have Rigid Dates

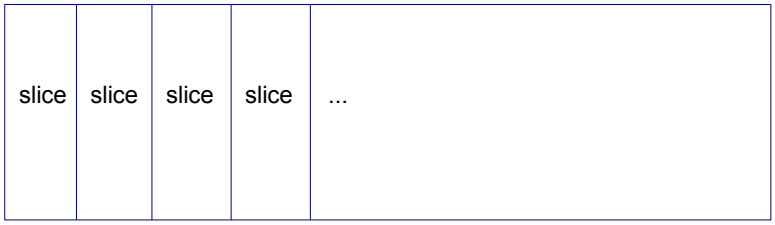
  The dates set for completing the Analysis and the Design phases are major milestones Developers should meet. This is the common criteria for measuring them.

  Analysis & Design Phases Have No Completion Criteria

  There is no completion criteria neither for the analysis or the design phases.

Iterative & Incremental Development

- The iterative and incremental development (IID) approach starts with subdividing our project into small binary deliverables.
  Each binary deliverable is defined as an executing segment of the overall project.



Iterative & Incremental Development

- Subdividing our project into small binary deliverables should be completed in accordance with the following rules:

  They should be vertical

  They cannot be subsystems and they should cut across as little as possible of the system functionality (e.g. I/O module cannot be a valid slice).

  They should represent features

  This way, if later we choose to eliminate a feature it would be as simply as eliminating the slice of that feature.

Iterative & Incremental Development

They should be executable and demonstrable

Each slice should have a completion and an acceptance criteria. This way it would be easy to evaluate whether a slice has successfully completed or not.

They should be simple to complete

Complete the development of each slice should be relatively easy and should take days or weeks. Having a slice that its completion takes months would be very bad.

They should be roughly the same in terms of manpower

Completing each one of the slices should take roughly the same amount of man power. This way we will have similar human risk involved with the development of each one of the slices.

Iterative & Incremental Development

- Once the slice partitioning is completed and the acceptance criteria for each one of them is set we can start developing the first slice.
- The first slice we develop better be the one with the highest development risk, error prone and complex one.

  Starting with the riskiest slice we will know the earliest whether or not the project will succeed. In addition, that will assist us to be conservative as possible with our project development estimation.

Iterative & Incremental Development

- When the development of the first slice is completed we can get the approximate end date of the project.
  Multiplying the duration it took us to develop that slice by the number of remaining slices will get the approximate end date of the project.
- Once the development of the first slice is completed we can move forward with the second slice or even to increase the manpower on the project and start two or three slices simultaneously.

Object Modeling with Use Cases

- Using UML Use Case diagrams can be a structured way for capturing the behavioral requirements of the system.
- The UML Use Case diagrams help us answering fundamental basic questions such as:
  - Who are the users of the system?
  - What are the users trying to do?
  - What is their experience working with the system?

Object Modeling with Use Cases

- The UML Use Case diagrams would be the first to use. Based on them it would be possible to proceed with the system development.

  It is highly common to include Use Case diagrams in the System Requirements Specification (SRC) document.