Systems Development Life Cycle

Contents

1 What's a System/Software Development Life Cycle? 1
- 1.1 Planning Phase . . . 2
- 1.2 Analysis Phase . . . 2
- 1.3 Design Phase . . . 2
- 1.4 Implementation/ Coding Phase: . . . 3
- 1.5 Maintenance Phase . . . 3

2 Systems Development Life Cycle Models 3
- 2.1 Waterfall Model . . . 3
- 2.2 Rapid Prototyping Model . . . 4
- 2.3 Incremental Model . . . 4
- 2.4 Spiral Model . . . 5
- 2.5 Agile process model . . . 6
- 2.6 Extreme Programming (*XP*) . . . 7
- 2.7 How to select the suitable *SDLC* model? . . . 7
  - 2.7.1 Criteria for deciding on a model . . . 8

1 What's a System/Software Development Life Cycle?

> The systems development life cycle (**SDLC**) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application. **SDLC** can apply to technical and non-technical systems

Consists of five consecutive phases:

- Planning Phase
- Analysis Phase
- Design Phase
- Implementation / coding phase
- Maintenance Phase

1.1 Planning Phase

- Answers why the information System should be built
- Determine how the project team will **go**
- Composed of:
  - Project initiation
  - Project management
- Includes:
  - Project initiation
  - ensuring feasibility
  - Project Plan & schedule
  - Obtain needed approvals

1.2 Analysis Phase

- Analysis Phase:
  - Answers the following:
    - Who will use the system
    - What hte system will do
    - Where it will be used
    - When It will be used
  - The Project team in this phase:
    - Investigate any current system
    - identify improvement
    - develop a concept for the new system
  - Composed of
    - Analysis Strategy
      - to guide project efforts
      - Analyze the current system if it exists
    - Requirements gathering
      - Leads the development of a concept for a new system
      - The concept is used to build a set of analysis models
    - System Proposal
      - Presented to Project sponsor
      - Sponsors decide if the project will continue Describes the basic business requirements the new system should meet.
    - Includes
      - Understanding business needs
      - obtaining functional requirements
    - Delivery:
      - Analysis
      - High-level initial design

1.3 Design Phase

- Define the solution system and how it will operate in terms of:

  - Hardware
  - Software
  - Networks
- Define the:
  - User interface
  - Forms used
  - Reports used
- Identify:
  - Needed files
  - Needed Programs
  - Needed databases and data

1.4 Implementation/ Coding Phase:

- The system is developed or purchased
- Is considered the longest phase.
- The most expensive phase
- Composed of:
  - System construction
    - Construction of the system
    - Testing of the system
  - Installation
    - Support installed systems
    - post-implementation review
  - Maintenance Plan
- Includes
  - Training the users
  - Installation of the new system

1.5 Maintenance Phase

- Is done periodically
- Done On hardware and software
- done to debug errors
- Keep improving and upgrading:
  - e.g. add new functionality

2 Systems Development Life Cycle Models

2.1 Waterfall Model

- Traditional sequential pioneer **SDLC**
- Became the defining engineering approach to *SW* development
- done in phases and each phase has various tasks and objectives
- Deliver complete product at the end.
- Characterized by:

  - Feedback Loops
  - Documentation-driven
- **Advantages** (pros):
  - Provides structured approach
  - Sets requirements early
  - Easy
  - Milestone are better understood
- **Disadvantages** (cons):
  - The working version will be available late.
  - has blocking states
  - specification are
    * Long
    * detailed
    * Written in a style unfamiliar to the client

2.2 Rapid Prototyping Model

- offers a small-scale replica working "prototype" of the product and then obtains customer feedback to refine the prototype.
- delivers a complete product at the end
- sub-phases for the prototype:
  - Establish objective → plan
  - define functionality → outline definition
  - develop → product
  - evaluate → evaluation report.
- Characteristics:
  - used in requirements phase
  - evaluated by the customer
- **Disadvantages** (cons)
  - could be discard-do not turn into a product
  - not proven
- *Recommendation* : use the **rapid prototyping model (RAD)** in defining requirements and **waterfall** in the rest of the phases.

2.3 Incremental Model

- requirements are broken down into multiple independent modules
- each iteration passes through 4 phases.
- deliver a portion of the product at each stage
- takes from 5 → 25 iteration to build a product
- **Advantages** (pros)
  - **SW** is generated quickly
  - flexible
  - less costly
  - less expensive to change requirements and scope
  - customers can respond to each building module

  - errors are easy to be identified
- **Disadvantages** (cons)
  - requires good planning and designing
  - system architecture causes problems
  - the iteration phase is rigid
  - time-consuming in debugging
- *When to use*
  - clear requirements
  - need for early releases.
  - no enough skilled engineers
  - There are hight risk features and goals
  - e.g. web applications and product based companies

2.4 Spiral Model

- is risk-driven **SDLC** model
- is based on risk patterns that push the team to adopt one or more process models
- each phase is:
  - proceeded by
    - alternative
    - *risk analysis*: identification of potential risk is done while risk mitigation strategy is planned and finalized
  - followed by
    - evaluation
    - planning for the next phase
- steps:
  - objectives and alternative solutions determination
  - identify/resolve risks
  - develop next version
  - review and plan for the next phase
- *when to use*
  - for a project that is/has :
    - large
    - requires frequent releases
    - prototypes applicable
    - medium to hight-risk projects
    - unclear and complex requirements
  - When
    - risk and cost evaluation is important
    - changes may occur at any time
    - long-term project commitment is not feasible due to changes in economic priorities
- **Advantages** (pros)
  - give space for customer feedback
  - fast development

  - features are added systematically
  - repeated development helps in risk management
  - cost analysis is easier due to prototype fragmentation
  - additional functionality can be done at a later stage
- **Disadvantages** (cons)
  - risk of not meeting budget/schedule
  - demands risk assessments expertise
  - should be followed strictly
  - needs more documentation
  - not good for small projects

2.5 Agile process model

- combines:
  - philosophy
    - encourages
      - customer satisfaction
      - early incremental deliver of the software
    - small highly motivated team
    - informal methods
    - minimal software engineering work
    - development simplicity
    - people and interaction are emphasized rather than processes and tools
  - Development guidelines
    - stresses *deliver* over *analysis* and *design*
    - active/continuous communication between developers and customers
- Steps
  - Exploration
  - planning
  - first release (repeat) - involve customer before release
  - Producterizing (Repeat) - Increase the pace of iterations after the product is released
  - Maintenance then planning
- **Advantages** (pros)
  - High customer satisfaction
  - better at adapting to late changes in requirements
  - supports face-to-face conversation
  - better at adapting to changing circumstances
  - **SW** is delivered frequently i.e. in weeks
  - customers, developers and tester constantly interact with each other
  - could continue to move forward without fear of reaching a sudden standstill
  - gives more freedom of options and time for developers and stakeholders.

        * freedom of option given them the ability to leave important decisions until more data is available
- **Disadvantages** (cons)
  - only senior programmers can take decision in the **SDLC**
  - the project can easily get taken off track if the customer is not clear about what outcome they want
  - less emphasis on designing and documentation
  - in large project, difficult ot assess the errors
- *When to use*
  - When new changes need to be implemented

2.6 Extreme Programming (*XP*)

- description
  - variated from the incremental model
  - controversial new approach
  - based on pair programming
  - based on the continuous integration of tasks
- steps
  - project requirements
  - stories - features that client wants to estimate cost and time
  - test cases - client select stories each build (tasks)
  - tasks
  - completion
- features
  - computers are put in the center of a large room lined with cubicles
  - client representatives works with the **XP** team at all times.
  - There is no specialization
  - There is no overall design phase - refactoring
  - After completion, the **XP** team has iteration planning meetings with the customers
  - Tasks are accepted based on customer testing

2.7 How to select the suitable *SDLC* model?

- Each of these approaches is suitable for different projects, environment, and requirements
- e.g. **Waterfall** is good for simple projects. However, **Spiral** or **iterative** are better for large-scale projects
- **No single model is the best of all the *SDLC* models discussed**
- To select the best model
  - know all the types of **SDLC** models
  - Assess the requirements of all the stakeholders
  - Decide on a method that best fits your needs

2.7.1 Criteria for deciding on a model

- Product complexity
- Product size
- Magnitude of changes
- Frequency of changes
- Skills of the development team
- Time constraints
- Access to users