SDLC

Software Development Life Cycle

Trainer: Aleksandar Karamfilov
Email: alex@pragmatic.bg
Web: pragmatic.bg

Agenda

- Software Development Life Cycle
- Phases in Software Development
- SDLC and Models
- Legacy Models
- Challenges
- Iterative and Incremental Models
- Agile

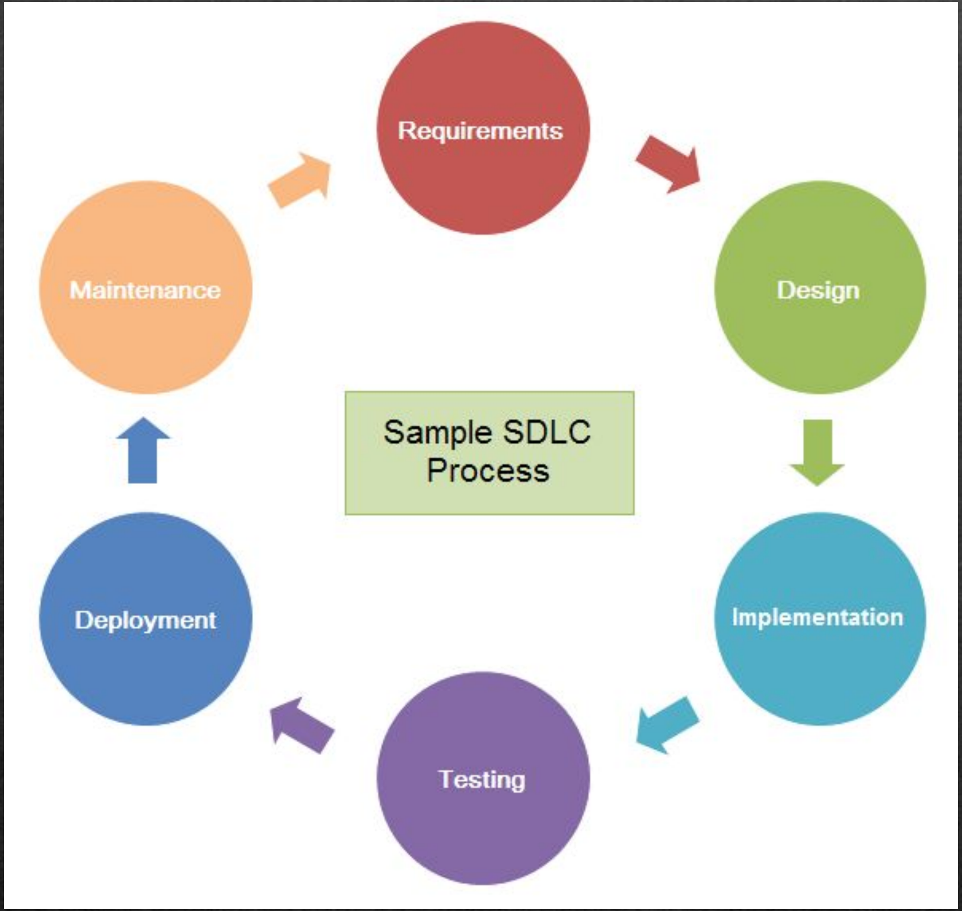
Software Development Life Cycle

“The software development life cycle(SDLC) sometimes referred to as the system development life cycle is the process of creating or altering software systems, and the methodologies that people use to develop these systems.”

Phases in SDLC

Phases in SDLC

- Requirements gathering (Analysis)
- Architecture (Design)
- Implementation and integration (Development)
- Validation (Testing)
- Installation (Deployment)
- Support (Maintenance)
- Retirement (may not exist)


Requirements
Design
Implementation
Testing
Deployment
Maintenance
Sample SDLC Process

Requirements Analysis

In the requirements analysis phase, the first step in the process, the requirements of the proposed system are collected by analyzing the needs of the user(s). This phase is concerned with establishing what the ideal system has to perform. However it does not determine how the software will be designed or built.

Design

System design is the phase where system engineers analyze and understand the business of the proposed system by studying the user requirements. The figure out possibilities and techniques by which the user requirements can be implemented. If any of the requirements are not feasible, user is informed of the issue.

Development

In engineering, system integration is the bringing together of the component subsystems into one system and ensuring that the subsystems function together as a system. In information technology, systems integration is the process of linking together different computing systems and software application physically or functionally, to act as a coordinated whole.

Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

Deployment

Software deployment is all of the activities that make a software system available for use.

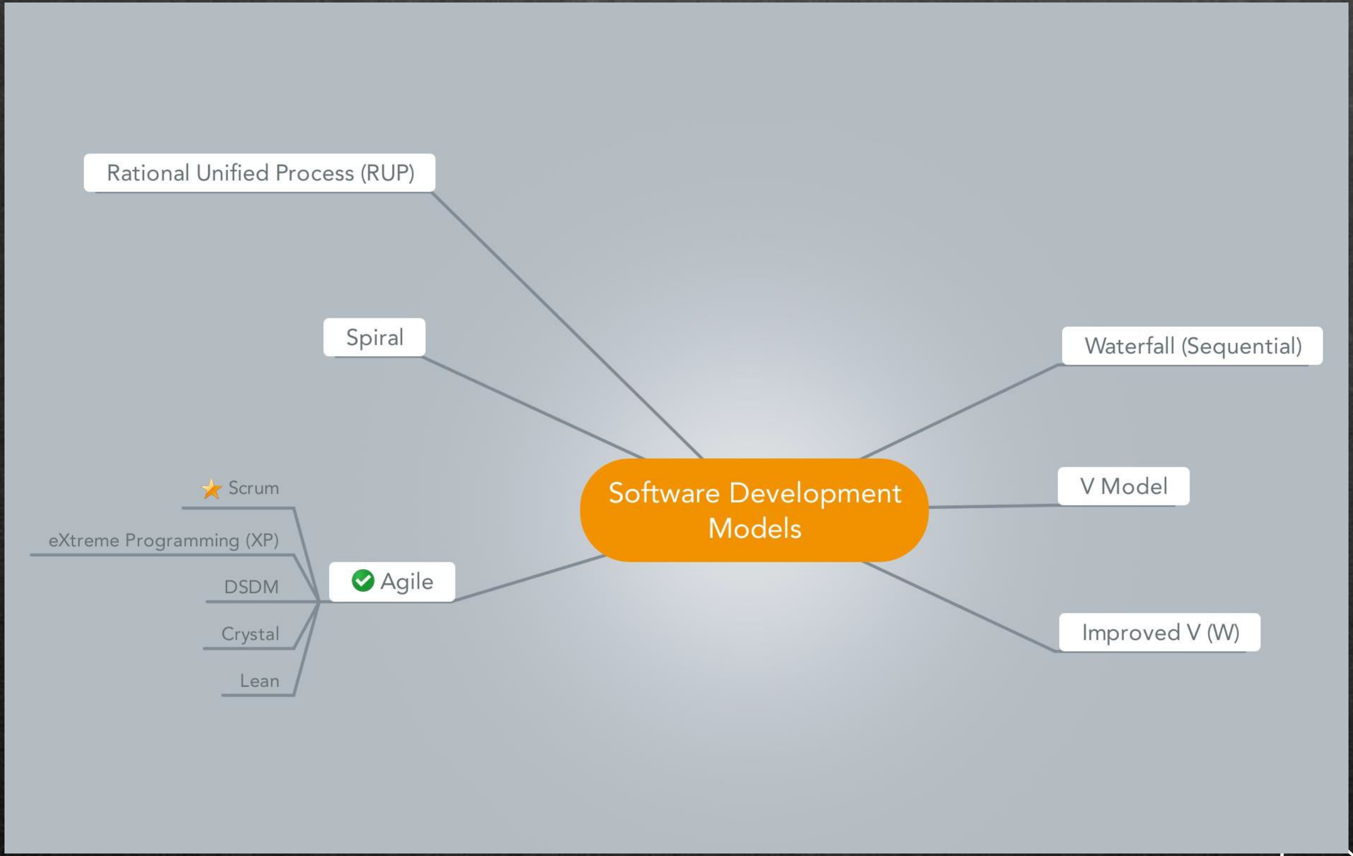
Maintenance

Software maintenance in software engineering is the modification of a software product after delivery to correct the faults, to improve performance or other attributes.

SDLC and Models

Model(Methodology)

A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system.


Software Development Models
Rational Unified Process (RUP)
Spiral
Agile
Scrum
eXtreme Programming (XP)
DSDM
Crystal
Lean
Waterfall (Sequential)
V Model
Improved V (W)

SDLC and Models(Methodologies)

- They aren’t the same thing.
- We can use different models to create software.
- Models can be considered as part of SDLC.
- We usually use one specific model to create software at a time.
- Migration between different models during SDLC is possible.

Validation

- Confirmation by examination and through provision of objective evidence that the requirements for specific intended use or application have been fulfilled.
- “Did we build the right system?”
- Check the correctness of the output from each stage of the software life-cycle against the user requirements.

Verification

- Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled.
- “Did we build the system right?”
- Check the correctness of the output from each stage of the software life-cycle against the input to that stage.

Legacy Models

History

- When talking about software development models, most people involved with software development are familiar with the traditional models known as Waterfall and V-model
- These two models have been the mainstay of most software development for at least 30 years or more

Assumptions in Legacy Models

- The customer knows exactly and everybody can clearly understand their requirements.
- The project will have a set of well defined, clear, concise requirements.
- We are able to determine everything we need to know about building the new system: estimate how long it will take to build and therefore how much it will cost.

Assumptions in Legacy Models

- The impact of change to the project and it's requirements will small enough that we can manage the changes without the need to re-plan.
- Once we define what it is we are building, nothing will need to change.
- System integration is predictable as long as we have the required architecture and docs.

Problems in Traditional Models

- The project runs over the allocated budget.
- The project is cancelled or postponed.
- The project schedule is squeezed so much that inadequate levels of testing and quality. assurance are performed to the project “over the line”.

Problems in Traditional Models

- The project is delivered with reduced scope - as result of not being able to deliver all the original scope within the specified project schedule
- Considered to be heavy and disciplined with a large layer of bureaucracy surrounding the model.

PRAGMATIC

Challenges

Software Development Challenges

- Reducing time to market.
- Becoming more responsive to changing business and customer needs.
- Delivering higher quality services to customers.

Software Development Challenges

- Reducing costs.
- Becoming more efficient in the way they transact business through automating end to end business process.
- Greater visibility of what is being developed and when it will be available for use.

Iterative and Incremental Models

Iterative and Incremental Models

A development life cycle where a project is broken into a series of increments, each of which delivers a portion of the functionality in the overall project requirements. The requirements are prioritized and delivered in priority order in the appropriate increment.

Iterative and Incremental Specifics

- Incremental development splits the development into a number of increments based on either functionality or some other criteria.
- Typically, each increment is delivered to the client after it is finished.
- Each increment consists of design, coding and maintenance phases.

Iterative and Incremental Specifics

- Risks are analyzed and eliminated in the early stages of the project. Then revised in each increment.
- Tight controls of the development process - plans are updated after every increment.

Iterative and Incremental Specifics

- Special quality increments can be inserted as needed. These usually have stabilizing purpose.
- Team motivation is usually higher as the visibility of the project status and progress is higher.

Agile

Agile

A methodology for the creative process that anticipates the need for flexibility and applies a level of pragmatism into the delivery of the finished product. Agile software development focuses on keeping code simple, testing often, and delivering functional bits of the application as soon as they're ready.

Agile Manifesto

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Testing in Agile Environment

- It is common to use a less formalized process and a much closer working relationship that allows changes to occur more easily within the project.
- Because Agile is a “light weight” process, there is less comprehensive test documentation in favor of having a more rapid method of communication such as daily “stand up” meetings.

Testing in Agile Environment

- The QA should expect to be involved from the initiation of the project, working with the developers as they do their initial architecture and design work.
- Reviews may not be formalized but are continuous as the software evolves.

PRAGMATIC

Testing in Agile Environment

- Involvement is expected to be throughout the project and the QA should be available to the team.
- Because of this immersion, members of Agile teams are usually dedicated to single project and are fully involved in all aspects of the project.

Questions?