software project management

Chenqi Yu 13211141
zhejiang normal university
software project management

abstract

In software engineering, a software development methodology (also known as a system development methodology, software development life cycle, software development process, software process) is a splitting of software development work into distinct phases (or stages) containing activities with the intent of better planning and management. It is often considered a subset of the systems development life cycle. The methodology may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application.

1. introduction

Common methodologies include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, extreme programming and various types of agile methodology. Some people consider a life-cycle "model" a more general term for a category of methodologies and a software development "process" a more specific term to refer to a specific process chosen by a specific organization. For example, there are many specific software development processes that fit the spiral life-cycle model.

2. history of software project management

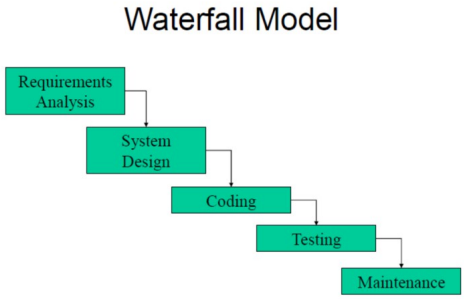
The software development methodology (also known as SDM) framework didn't emerge until the 1960s. According to Elliott (2004) the systems development life cycle (SDLC) can be considered to be the oldest formalized methodology framework for building information systems. The main idea of the SDLC has been "to pursue the development of information systems in a very deliberate, structured and methodical way, requiring each stage of the life cycle - - from inception of the idea to delivery of the final system - - to be carried out rigidly and sequentially" within the context of the framework being applied. The main target of this methodology framework in the 1960s was "to develop large scale functional business systems in an age of large scale business conglomerates. Information systems activities revolved around heavy data processing and number crunching routines".
Methodologies, processes, and frameworks range from specific proscriptive steps that can be used directly by an organization in day-to-day work, to flexible frameworks that an organization uses to generate a custom set of steps tailored to the needs of a specific project or group. In some cases a "sponsor" or "maintenance" organization distributes an official set of documents that describe the process.

3. overview

1. Waterfall development
2. Prototyping
3. Incremental development
4. Iterative and incremental development
5. Spiral development
6. Rapid application development
7. Agile development
8. Lightweight methodologies

4. methods/techniques

1.

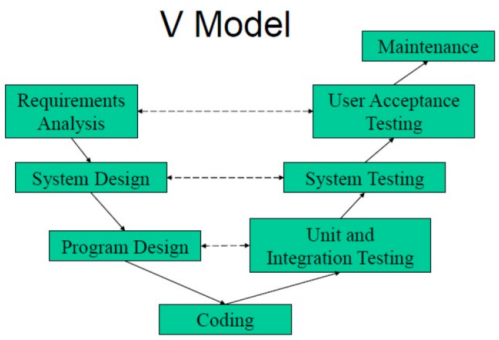


The waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases, typically:

Requirements analysis resulting in a software requirements specification
Software design
Implementation
Testing
Integration, if there are multiple subsystems
Deployment (or Installation)
Maintenance

Sashimi waterfall: activities are allowed to overlap
Waterfall with sub projects: implementation ofdifferent components proceeds in parallel
Waterfall with risk reduction: an initial risk analysishelps mitigate risks in later phases of implementation

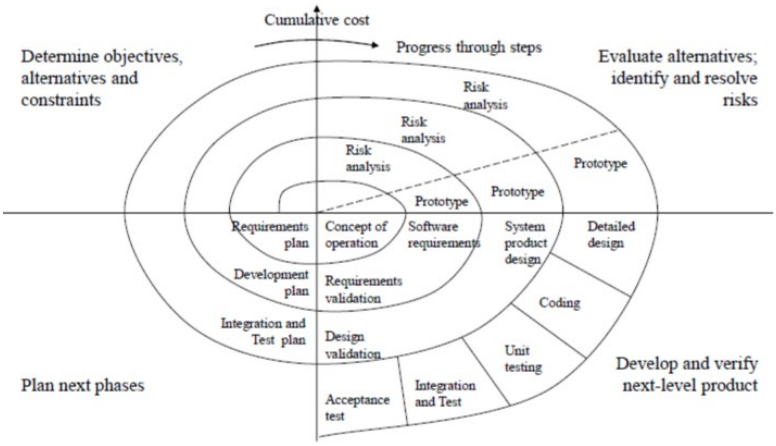
2.



V Model (cont' d)
Additional validation process introduced
Relate testing to analysis and design
Loop back in case of discrepancy

3.

Spiral Model (adapted from Boehm 1987)

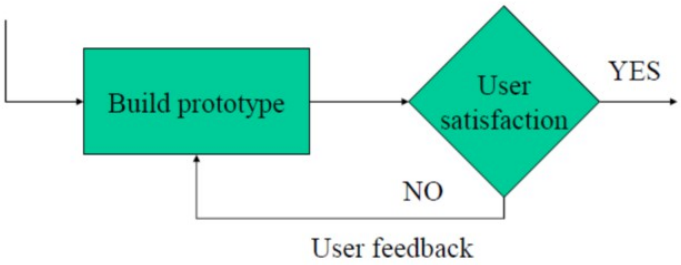


Spiral Model (cont' d)
Evolutionary approach
Iterative development combined with risk management
Risk analysis results in "go, re-do, no-go" decision

Four major activities
- Planning
- Risk analysis
- Engineering
- Evaluation

The basic principles are:
Focus is on risk assessment and on minimizing project risk by breaking a project into smaller segments and providing more ease-of-change during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle.
"Each cycle involves a progression through the same sequence of steps, for each part of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program."
Each trip around the spiral traverses four basic quadrants: (1) determine objectives, alternatives, and constraints of the iteration; (2) evaluate alternatives; Identify and resolve risks; (3) develop and verify deliverables from the iteration; and (4) plan the next iteration.
Begin each cycle with an identification of stakeholders and their "win conditions", and end each cycle with review and commitment.

Prototyping Model



Benefits of Prototyping
Learning by doing
Improved communication
Improved user involvement
Clarification of partially-known requirements
Demonstration of the consistency and completeness of a specification
Reduced need for documentation
Reduced maintenance costs
Feature constraint
Production of expected results for testing real system

Prototyping Sequences
Requirements gathering
Quick design
Prototype construction
Customer evaluation
Refinement
Loop back to quick design for fine tuning
Product engineering

Drawbacks of Prototyping
Users sometimes misunderstand the role of the prototype
Lack of project standards possible
Lack of control
Additional expense
Machine efficiency
Close proximity of developers

4. Prototyping Model
Goals
- meet (some) user requirements at an early stage
- reduce risk and uncertainty
- verify a design or implementation approach

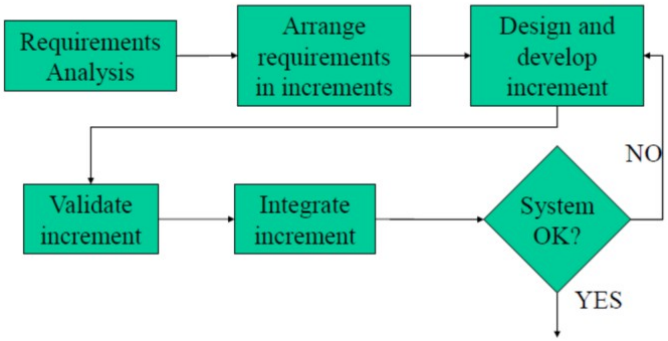
Should always answer specific questions; goals must be identified
Throw-away
- After users agree the requirements of the system, the prototype will be discarded.

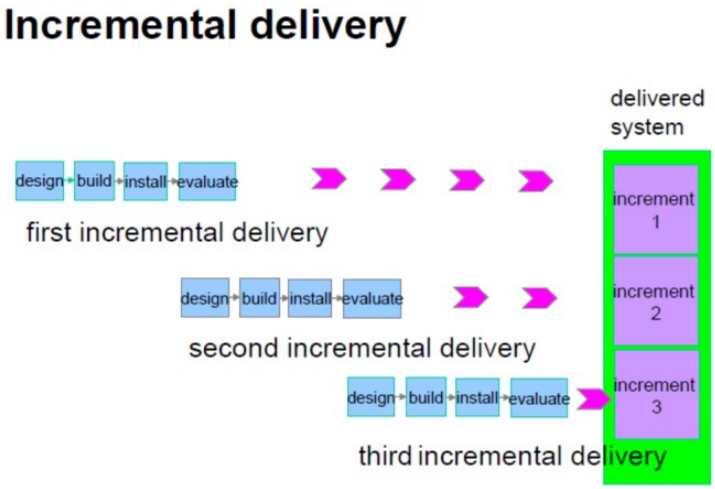
Evolutionary
- Modifications are based on the existing prototype.

Incremental
- Functions will be arranged and built accordingly.

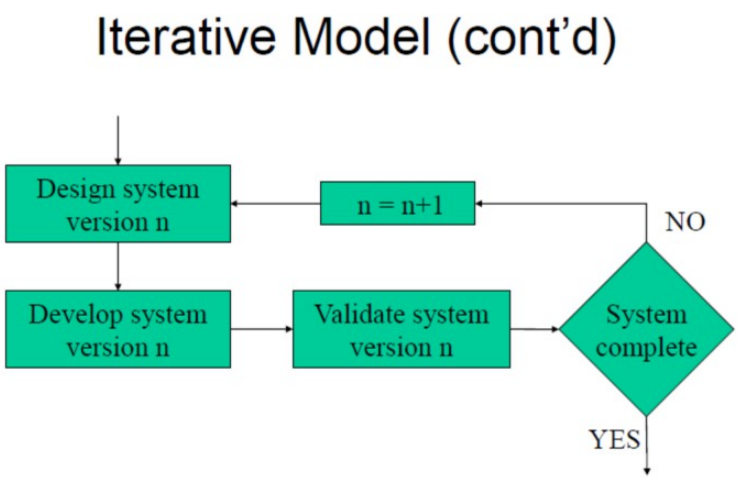
5. Incremental Model
Break system into small components
Implement and deliver small components in sequence
Every delivered component provides extra functionality to user

Incremental Model (cont'd)





6. Iterative Model
Deliver full system in the beginning
Enhance functionality in new releases



Combined Incremental and Iterative Model
Every new release includes
  extra functionality
  enhancement of existing functionality
Popularly used in software industry
Ranking the Increments
Rank by value to cost ratio
V = value to customer (score 1 -10)
C = cost (score 1-10)
Value to cost ratio = V/C

7. Advantages of Phased Development
Early feedback
Less possible requirement changes
Early benefits for users
Improved cash flow
Easier to control and manage
Capture early market
Facilitate early training
Can be temporarily abandoned
Increase job satisfaction

Disadvantages of Phased Development
'Software breakage'
Reduced productivity

8. Operational Specification Model
Executable or translatable specification
Use executables to demonstrate system behaviour
Resolve requirement uncertainties in early stage
Merging functionality and system design

9. Transformational Model
Transform a specification into delivered system
Requires automated support
Relies on formal specification method

reference

1. a b Whitten, Jeffrey L.; Lonnie D. Bentley, Kevin C. Dittman. (2003). Systems Analysis and Design Methods. 6th edition. ISBN 0-256-19906-X.
2. Kent Beck, Extreme Programming, 2000.
3. McConnell, Steve. "7: Lifecycle Planning". Rapid Development. Redmond, Washington: Microsoft Press. p. 140.
4. wikipedia.