Software Engineering

HUANG Jie

School of Computer Science & Technology

Tongji University

2025

Lesson 5

Recommended Process Model

In the previous lessons, we have learned

- ✓ The traditional software process models.
- ✓ Agility and agile models.
- ✓ Relationship between traditional and agile models.

Lesson 5

Recommended Process Model

In this lesson, we will discuss

- ✓ How do we ensure that we've done the software process right ?
- ✓ The following issues should be discussed:
  - The timeliness
  - Levels of stakeholder satisfaction
  - Overall quality
  - Long-term viability of the product increments built are the best indicators that your process is working

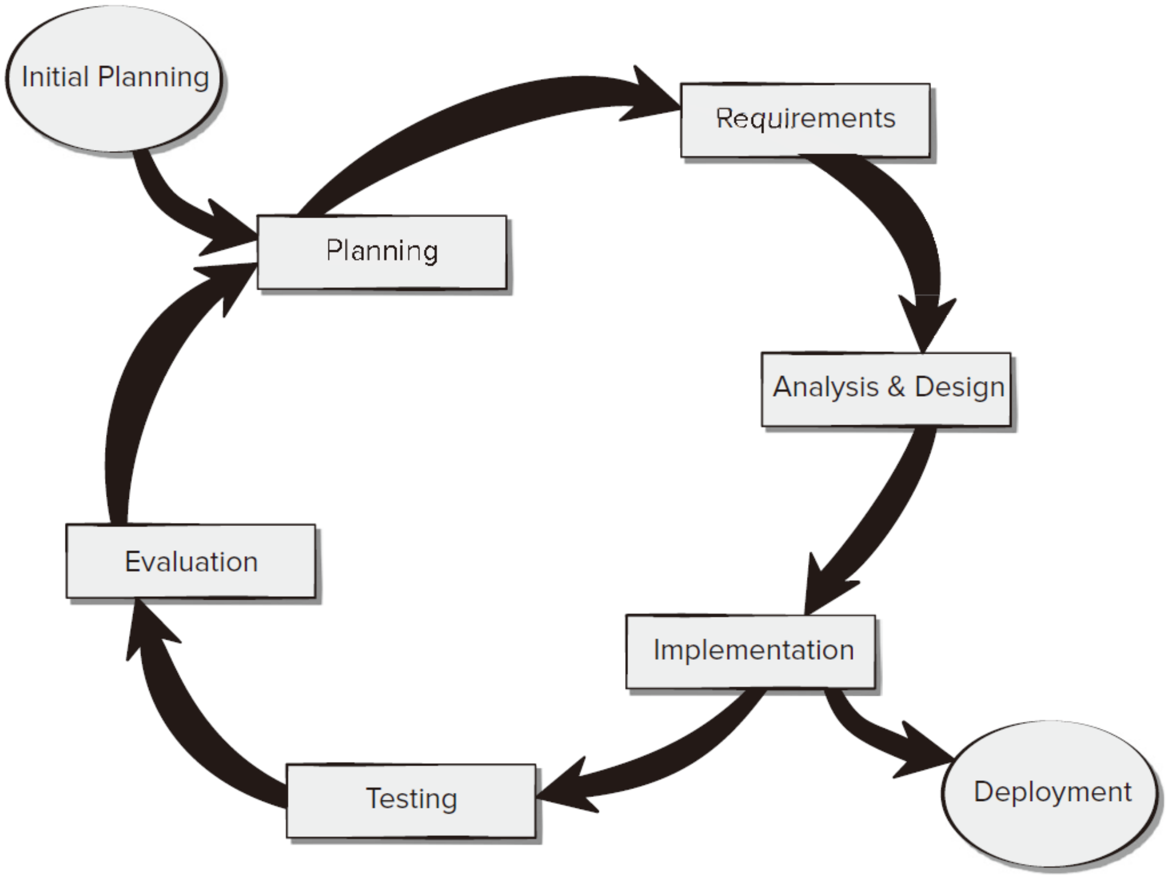
Weaknesses of prescriptive software life-cycle approaches

- It is risky to use a linear process model without ample feedback.
- It is never possible nor desirable to plan big up-front requirements gathering.
- Up-front requirements gathering may not reduce costs or prevent time slippage.
- Appropriate project management is integral to software development.
- Documents should evolve with the software and should not delay the start of construction.
- Involve stakeholders early and frequently in the development process.
- Testers need to become involved in the process prior to software construction.

同濟大學 TONGJI UNIVERSITY

Solution to the problem of waterfall

- Incremental model (e.g. Scrum)

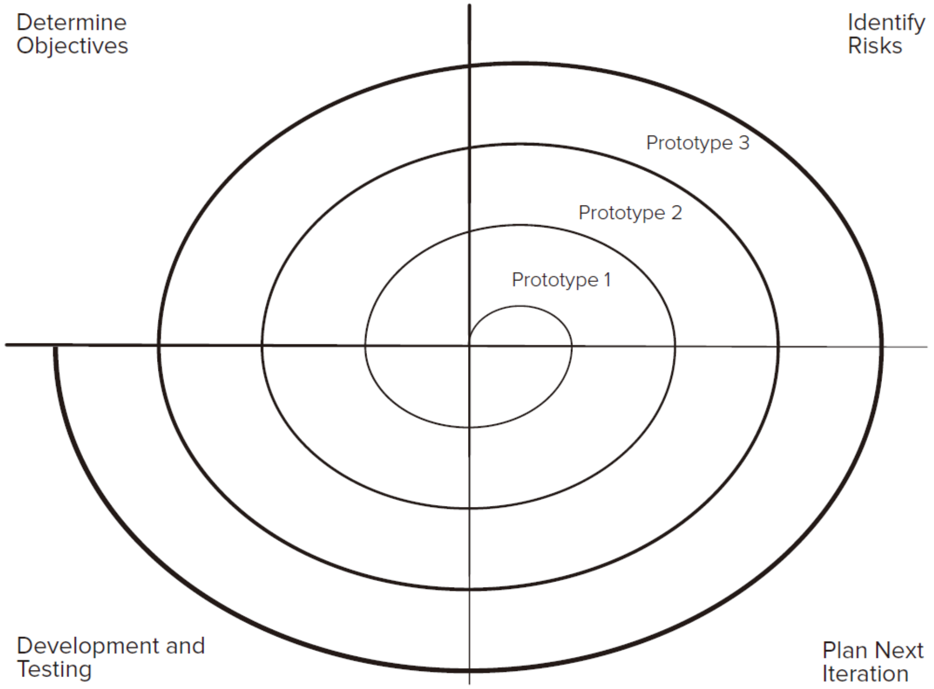


Key characteristics of agile process models

- Prototypes created are designed to be extended in future software increments.
- Stakeholders are involved throughout the development process.
- Documentation requirements are lightweight, and documentation should evolve along with the software.
- Testing is planned and executed early.
- Scrum and Kanban allow for controlled introduction of new requirements (user stories).

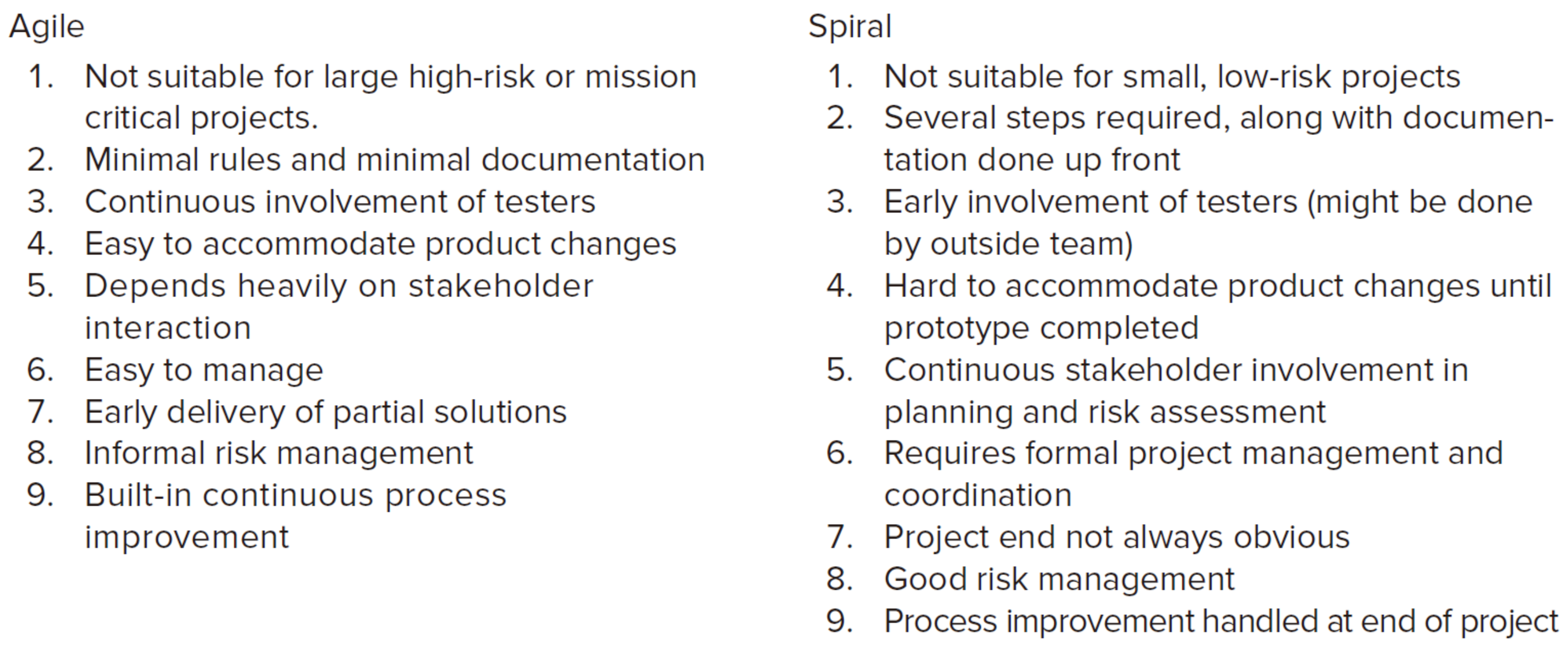
Key characteristics of spiral model

- The spiral model can be thought of as an evolutionary prototyping model with a risk assessment element.
- The spiral model is somewhat unique in that formal risk assessment is built in and used as the basis for deciding whether to invest the resources needed to create the next prototype.



Agile vs Evolutionary models

- Developers learn many things as they proceed in the development process. That is why it is important for developers to be able to adapt their process as quickly as practical to accommodate this new knowledge.



Agile	Spiral
1. Not suitable for large high-risk or mission critical projects.	1. Not suitable for small, low-risk projects
2. Minimal rules and minimal documentation	2. Several steps required, along with documentation done up front
3. Continuous involvement of testers	3. Early involvement of testers (might be done by outside team)
4. Easy to accommodate product changes	4. Hard to accommodate product changes until prototype completed
5. Depends heavily on stakeholder interaction	5. Continuous stakeholder involvement in planning and risk assessment
6. Easy to manage	6. Requires formal project management and coordination
7. Early delivery of partial solutions	7. Project end not always obvious
8. Informal risk management	8. Good risk management
9. Built-in continuous process improvement	9. Process improvement handled at end of project

Requirements definition

- What is requirements engineering?
- Scott Ambler [Amb12] suggests several best practices for agile requirements definition:
  - Encourage active stakeholder participation by matching their availability and valuing their input.
  - Use simple models (e.g., Post-it notes, fast sketches, user stories) to reduce barriers to participation.
  - Take time to explain your requirement representation techniques before using them.
  - Adopt stakeholder terminology, and avoid technical jargon whenever possible.
  - Use a breadth-first approach to get the big picture of the project done before getting bogged down in details.

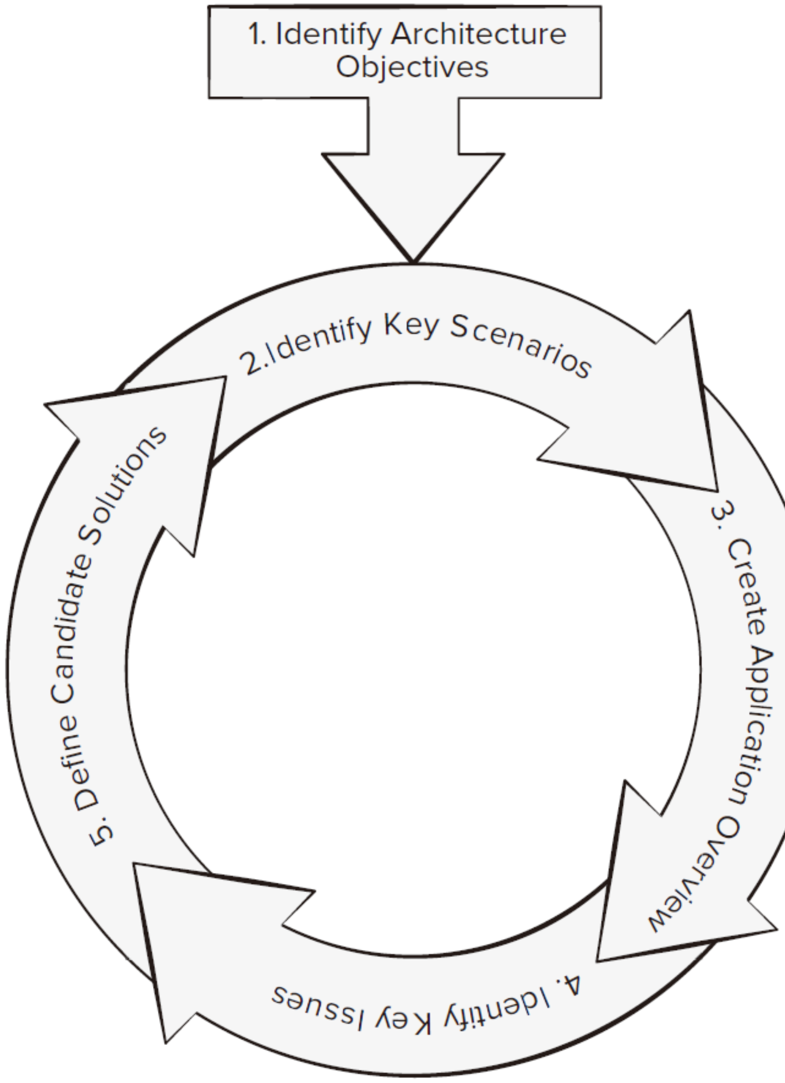
Requirements definition (Cont.)

- Scott Ambler [Amb12] suggests several best practices for agile requirements definition:
  - Allow the development team to refine (with stakeholder input) requirement details "just in time" as user stories are scheduled to be implemented.
  - Treat the list of features to be implemented like a prioritized list, and implement the most important user stories first.
  - Collaborate closely with your stakeholders and only document requirements at a level that is useful to all when creating the next prototype.
  - Question the need to maintain models and documents that will not be referred to in the future.
  - Make sure you have management support to ensure stakeholder and resource availability during requirements definition.

Requirements definition (Cont.)

- It is important to recognize two Realities:
  - It is impossible for stakeholders to describe an entire system before seeing the working software.
  - It is difficult for stakeholders to describe quality requirements needed for the software before seeing it in action.
- Features of prototype model:
  - Tangible realizations of project.
  - Motivate stakeholders discuss requirements changes in more concrete terms.
  - Increase the volatility(波动性) of the requirements.
  - Prototypes be discarded, wasting time and resources.

Preliminary architectural design


1. Identify Architecture Objectives
2.Identify Key Scenarios
3. Create Application Overview
4. Identify Key Issues
5. Define Candidate Solutions

TONGJI UNIVERSITY

Preliminary architectural design (Cont.)

- Four key elements:
  - Focus on key quality attributes, and incorporate them into prototypes as they are constructed.
  - When planning prototypes, keep in mind that successful software products combine customer-visible features and the infrastructure to enable them.
  - Recognize that an agile architecture enables code maintainability and evolvability if sufficient attention is paid to architectural decisions and related Quality issues.
  - Continuously managing and synchronizing dependencies(关联性) between the functional and architectural requirements is needed to ensure the evolving architectural foundation will be ready just in time for future increments.

Resource estimation

- Pressman's words
  - Use historic data, and work as a team to develop an estimate of how many days it will take to complete each of the user stories known at the start of the project.
  - Loosely organize the user stories into the sets that will make up each sprint planned to complete a prototype.
  - Sum the number of days to complete each sprint to provide an estimate for the duration of the total project.
  - Revise the estimate as requirements are added to the project or prototypes are delivered and accepted by the stakeholders.
- Keep in mind that doubling the number of developers almost never cuts the development time in half. (Brooks)

First prototype construction

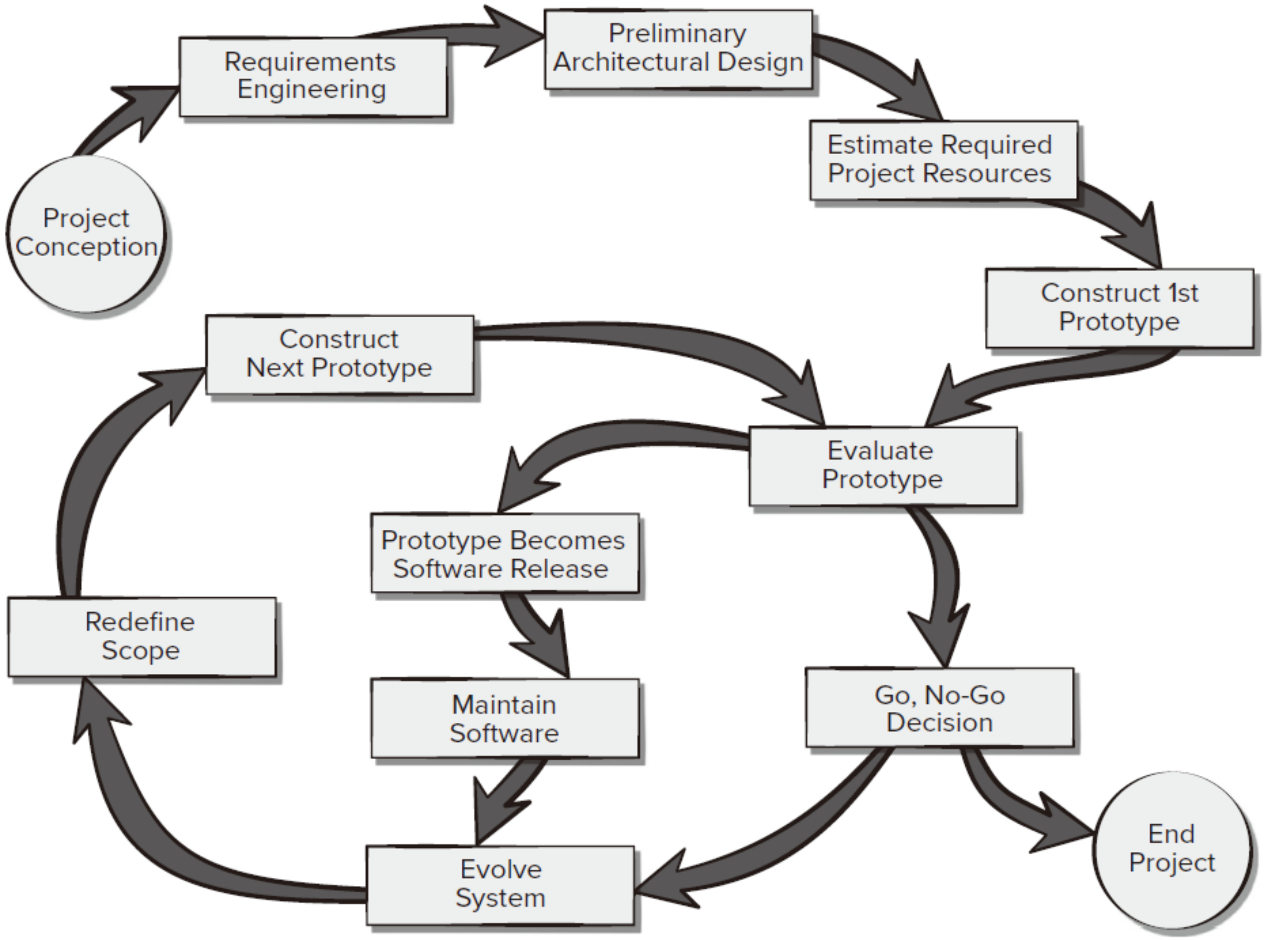
- Steps: given by National Instruments
  - Transition from paper prototype to software design
  - Prototype a user interface
  - Create a virtual prototype
  - Add input and output to your prototype
  - Engineer your algorithms
  - Test your prototype
  - Prototype with deployment in mind.

Prototype evaluation

- Dam and Siang:
  - Provide scaffolding when asking for prototype feedback.
  - Test your prototype on the right people.
  - Ask the right questions.
  - Be neutral when presenting alternatives to users.
  - Adapt while testing.
  - Allow the user to contribute ideas.

Go or No-Go decision

Recommended software process model



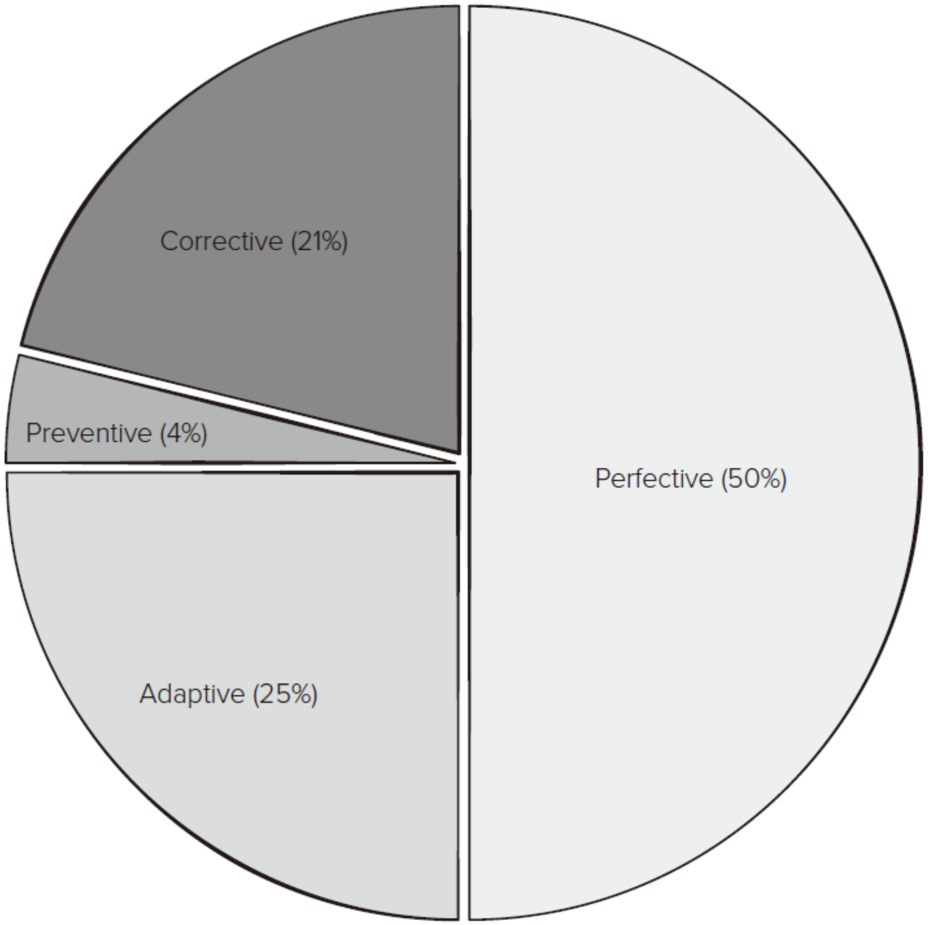
同濟大學 TONGJI UNIVERSITY

Prototype evolution

- New Prototype Scope
- Constructing New Prototypes
- Testing New Prototypes

Prototype release & maintenance

- Corrective maintenance
  纠正性维护
- Adaptive maintenance
  适应性维护
- Perfective maintenance
  完善性维护
- Preventive maintenance
  预防性维护



Recommended software process steps (page 71 in 9th edition)

- 1. Requirements engineering
  - Gather user stories from all stakeholders.
  - Have stakeholders describe acceptance criteria user stories.
- 2. Preliminary architectural design
  - Make use of paper prototypes and models.
  - Assess alternatives using nonfunctional requirements.
  - Document architecture design decisions.
- 3. Estimate required project resources
  - Use historic data to estimate time to complete each user story.
  - Organize the user stories into sprints.
  - Determine the number of sprints needed to complete the product.
  - Revise the time estimates as use stories are added or deleted.

Recommended software process steps (Cont.)

- 4. Construct first prototype
  - Select subset of user stories most important to stakeholders.
  - Create paper prototype as part of the design process.
  - Design a user interface prototype with inputs and outputs.
  - Engineer the algorithms needed for first prototypes.
  - Prototype with deployment in mind.
- 5. Evaluate prototype
  - Create test cases while prototype is being designed.
  - Test prototype using appropriate users.
  - Capture stakeholder feedback for use in revision process.

Recommended software process steps (Cont.)

- 6. Go, no-go decision
  - Determine the quality of the current prototype.
  - Revise time and cost estimates for completing development.
  - Determine the risk of failing to meet stakeholder expectations.
  - Get commitment to continue development.
- 7. Evolve system
  - Define new prototype scope.
  - Construct new prototype.
  - Evaluate new prototype and include regression testing.
  - Assess risks associated with continuing evolution.

Recommended software process steps (Cont.)

- 8. Release prototype
  - Perform acceptance testing.
  - Document defects identified.
  - Share quality risks with management.
- 9. Maintain software
  - Understand code before making changes.
  - Test software after making changes.
  - Document changes.
  - Communicate known defects and risks to all stakeholders.

Summary

- In this chapter, we suggested the use of a highly interactive, incremental prototyping process. We think this is better than producing rigid product plans and large documents prior to doing any programming. We suggested the use of a highly interactive, incremental prototyping process. We think this is better than producing rigid product plans and large documents prior to doing any programming.
- We suggest the use of an evolutionary process model that emphasizes frequent stakeholder involvement in the creation and evaluation of incremental software prototypes.
- Planning is important but should be done expeditiously to avoid delaying the start of development.
- Risk assessment and acceptance testing are an important part of the prototype assessment process.
- The biggest challenges developers have with evolutionary process models is managing scope creep while delivering a product that meets customer expectations and doing all this while delivering the product on time and within budget. That's what makes software engineering so challenging and rewarding.

Assignment

- Chapter 4: Problems to ponder

  4.7. What data points are needed to make the go, no-go decision during the assessment of an evolutionary prototype ?

- Reading

  《The Mythical Man-Month: Essays on Software Engineering》

  Chapter 7 Why Did the Tower of Babel Fail? by Frederick P.Brooks

- Preview

  《Software Engineering》 (8th Edition) by R.S.Pressman

  Chapter 6 Human Aspects of Software Engineering

  《Software Engineering》 (9th Edition) by R.S.Pressman

  Chapter 5 Human Aspects of Software Engineering