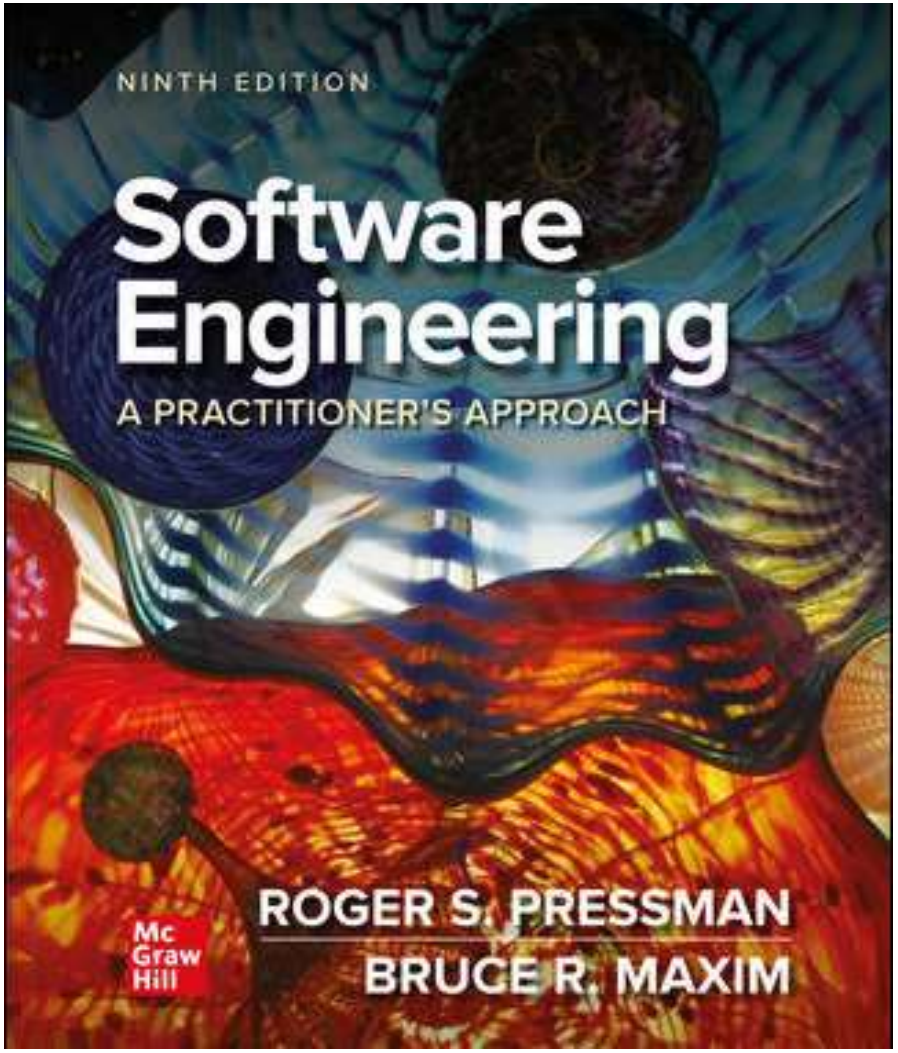
Mc Graw Hill Education

Because learning changes everything.®

Chapter 4

Recommended Process Model

Part One - The Software Process



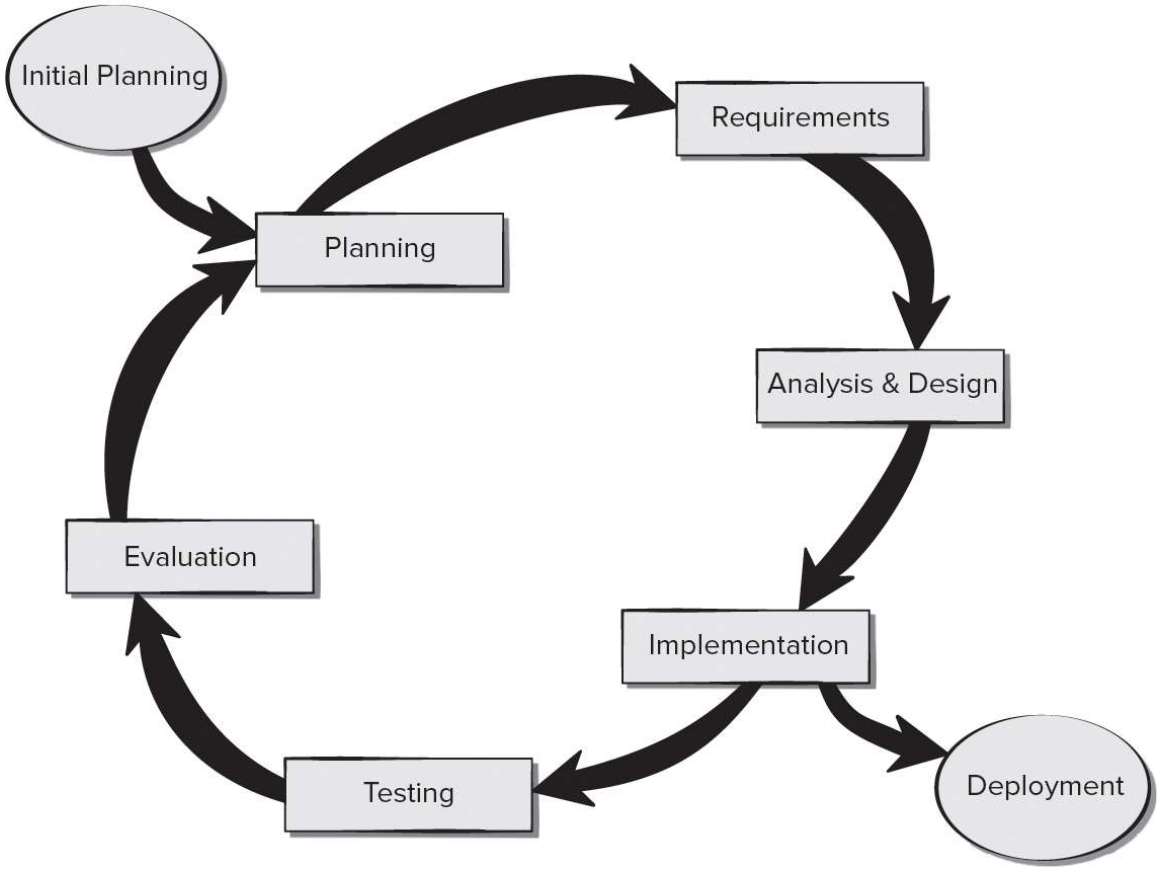
Adapting Process Models

- Every software project needs a “road map” or “generic software process” of some kind.
- Every project is different, and every team is different.
- No single software engineering framework is appropriate for every software product.
- Any road map or generic process should be based on best industry practices.
- Developers and stakeholders adapt generic process models and tailor them to fit the current project, the skills of the team members, and the user needs.

Principles for Organizing Software Projects

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

Incremental Model for Prototype Design



Access the text alternative for slide images.

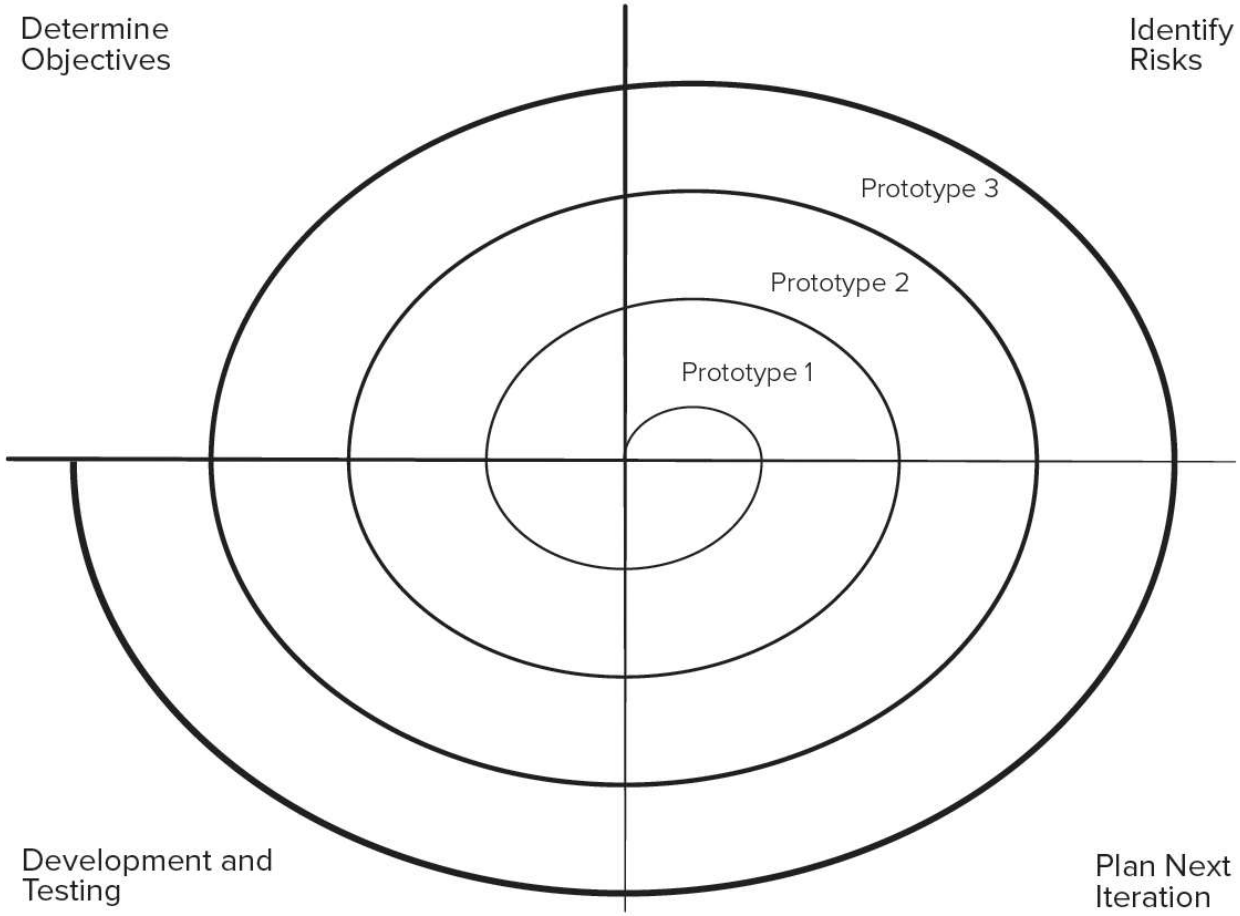
Characteristics of Agile Process Models

1. Not suitable for large high-risk or mission critical projects.
2. Minimal rules and minimal documentation.
3. Continuous involvement of testers.
4. Easy to accommodate product changes.
5. Depends heavily on stakeholder interaction.
6. Easy to manage.
7. Early delivery of partial solutions.
8. Informal risk management.
9. Built-in continuous process improvement.

Characteristics of Spiral Process Models

1. Not suitable for small, low-risk projects.
2. Several steps required, along with documentation done up front.
3. Early involvement of testers (might be done by outside team).
4. Hard to accommodate product changes until prototype completed.
5. Continuous stakeholder involvement in planning and risk assessment.
6. Requires formal project management and coordination.
7. Project end not always obvious.
8. Good risk management.
9. Process improvement handled at end of project.

Spiral Model for Prototype Design



Access the text alternative for slide images.

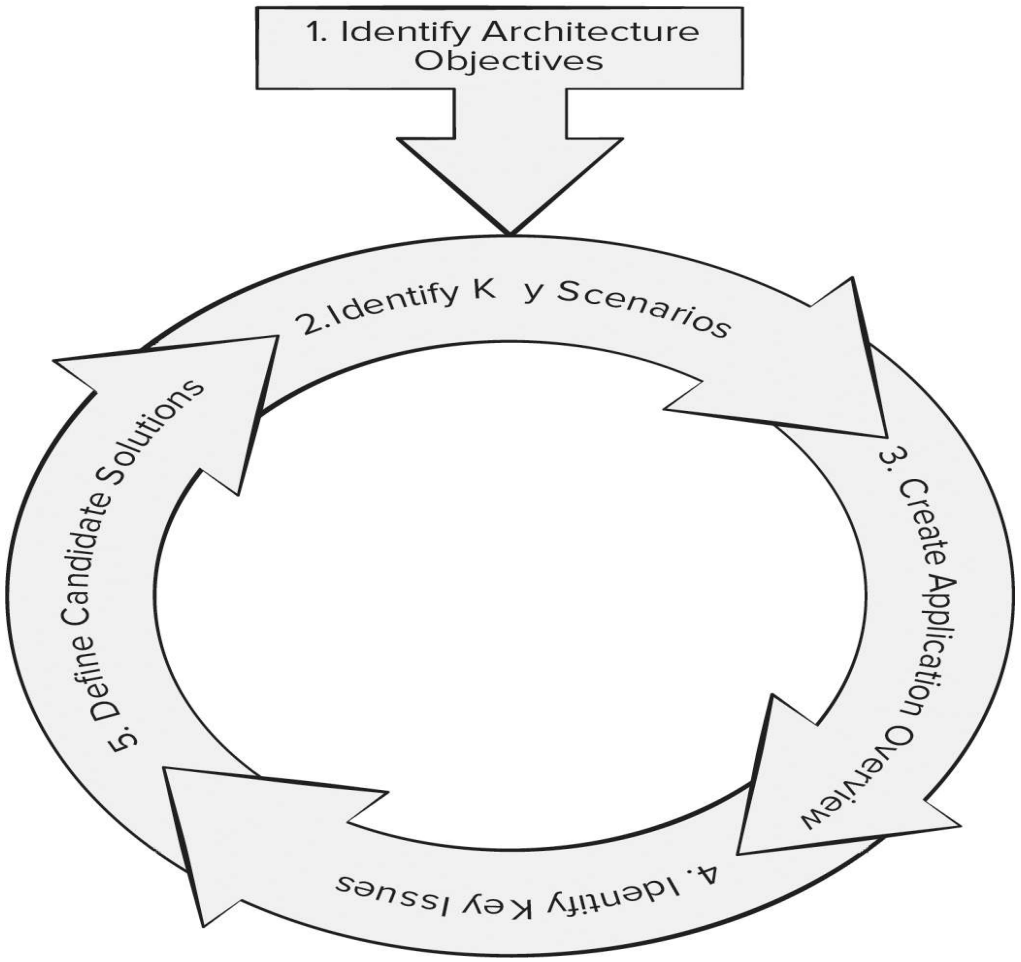
Agile Requirements Definition [1]

1. Encourage active stakeholder participation by matching their availability and valuing their input.
2. Use simple models (for example, Post-it notes, fast sketches, user stories) to reduce barriers to participation.
3. Take time to explain your requirement representation techniques before using them.
4. Adopt stakeholder terminology and avoid technical jargon whenever possible.
5. Use a breadth-first approach to get the big picture of the project done before getting bogged down in details.

Agile Requirements Definition 2

6. Developer and stakeholders refine requirements "just in time" as user stories are ready to be implemented.
7. Treat list of features like a prioritized list and implement the most important user stories first.
8. Collaborate closely with stakeholders and document requirements so they ares useful to all when creating the next prototype.
9. Question the need to maintain models and documents not referred to in the future.
10. Ensure management support for stakeholder and resource availability during requirements definition.

Protype Architectural Design



Access the text alternative for slide images.

Elements of Agile Architectural Design

1. Focus on key quality attributes and incorporate them into prototypes as they are constructed.
2. Keep in mind that successful software products combine customer-visible features and the infrastructure needed to enable them.
3. Agile architectures enable code maintainability and evolvability if attention is paid to architectural decisions and quality issues.
4. Managing and synchronizing dependencies among functional and architectural requirements is needed to ensure evolving architecture will be ready for future increments.

Resource Estimation for Agile Spiral Model

1. Team should use historic data to develop an estimate of number of days needed to complete each of user stories known at the start of the project.
2. Loosely organize the user stories into sets that will make up each sprint planned to complete a prototype.
3. Sum the number of days to complete each sprint to provide an estimate for the duration of the total project.
4. Revise the estimate as requirements are added to the project or prototypes are delivered and accepted by the stakeholders.

First Prototype Guidelines

1. Transition from paper prototype to software design.
2. Prototype a user interface.
3. Create a virtual prototype.
4. Add input and output to your prototype.
5. Engineer your algorithms.
6. Test your prototype.
7. Prototype with deployment in mind.

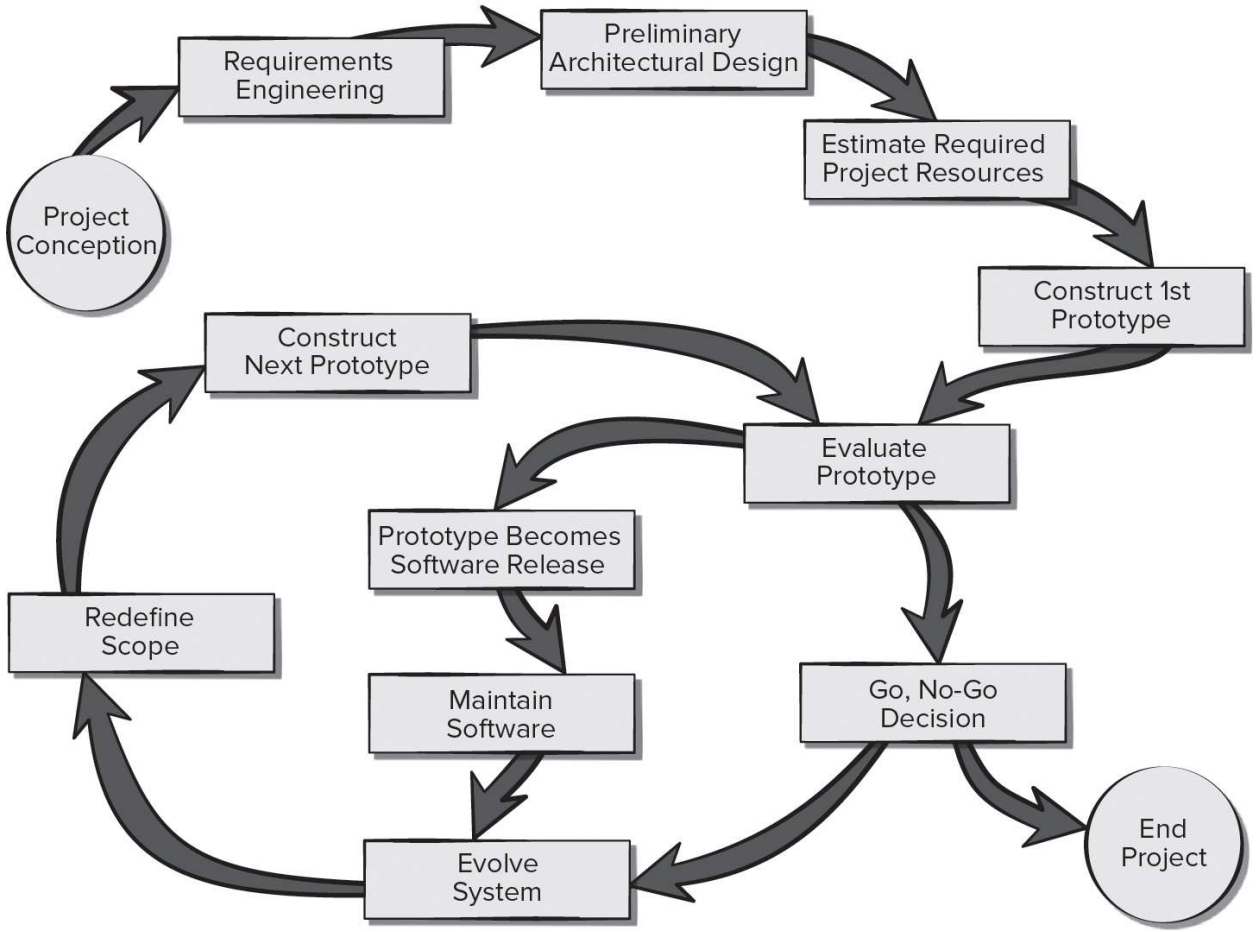
Prototype Evaluation

1. Provide scaffolding when asking for prototype feedback.
2. Test your prototype on the right people.
3. Ask the right questions.
4. Be neutral when presenting alternatives to users.
5. Adapt while testing.
6. Allow the user to contribute ideas.

Go No Go Decision

- A pass through the planning region follows the evaluation process.
- Revised cost estimates and schedule changes are proposed based on changes were requested when evaluating the current prototype.
- Risk of exceeding the budget and missing the project delivery date is assessed.
- Risk of failing to satisfy user expectations is also considered and discussed with the stakeholders and sometimes senior management.
- Goal of risk assessment is to get commitment from stakeholders and management to provide the resources needed to create the next prototype.

Recommended Prototype Evolutionary Process



Access the text alternative for slide images.

Recommended Process Steps 1

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 Process Steps 2

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 Process Steps 3

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 Process Steps 4

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.

Testing New Prototypes

- Testing should be performed by developers using test cases created during the design process before programming was completed.
- Each user story has an acceptance criteria attached to it and it should guide the creation of the test cases to ensure the prototype meets customer needs.
- Prototypes need to be tested for defects and performance issues.
- Ensure that adding new features to evolutionary prototypes does not accidentally break features working correctly in the previous prototype (***regression testing***).

Release Candidates 1

- A prototype considered as a release candidate is subjected to user acceptance testing in addition to testing conducted during prototype construction.
- User acceptance tests are based on acceptance criteria that were recorded as each user story was created and added to the product backlog.
- User feedback during acceptance testing should be organized by user-visible functions as portrayed via the user interface.
- Developers should make changes only if these changes will not delay the release of the prototype.

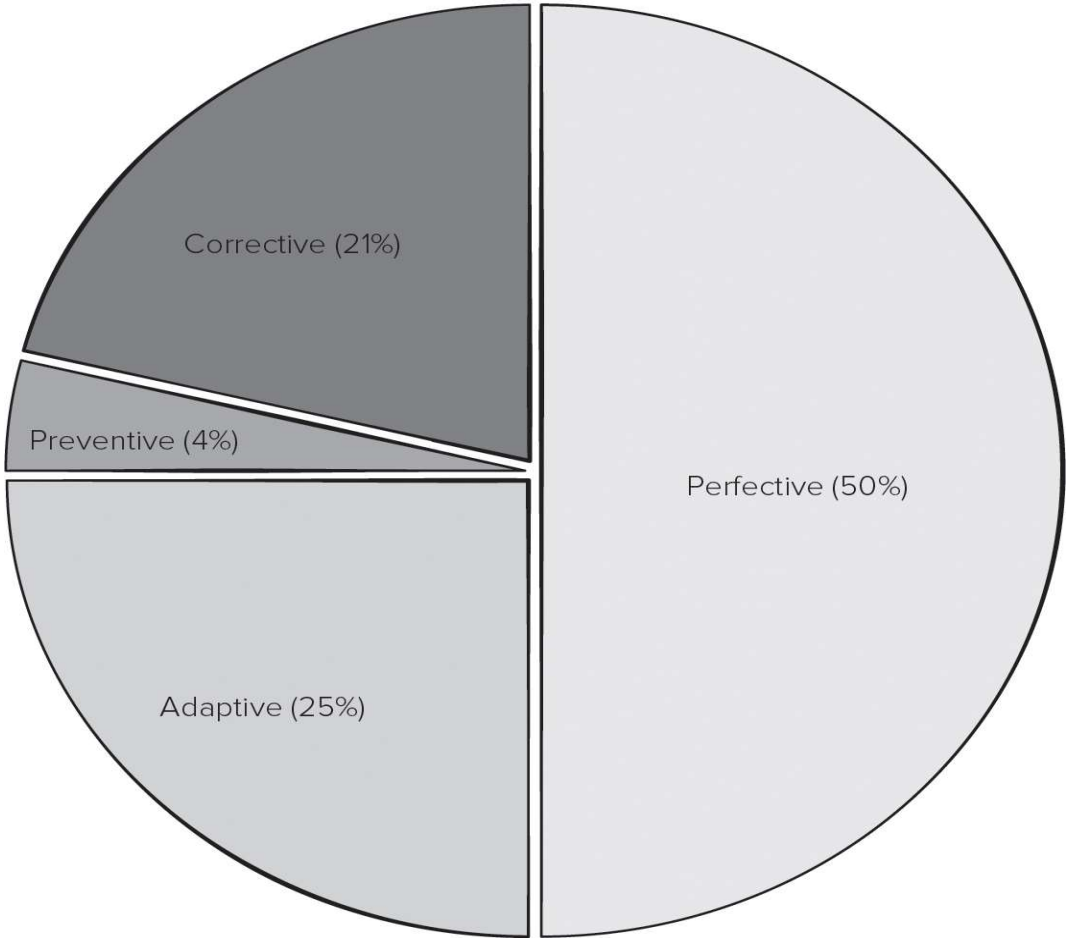
Release Candidates 2

- If changes are made, they need to be verified in a second round of acceptance testing before moving on.
- The issues and lessons learned from creating the release candidate should be documented and considered by the developers and stakeholders as part of the project postmortem.
- This information should be considered before deciding to undertake future development of a software product.
- Lessons learned from the current product can help developers make better cost and time estimates for similar projects in the future.

Software Release Maintenance

- ***Maintenance*** - activities needed to keep software operational after it has been accepted and released in the end-user environment.
- ***Corrective maintenance*** - reactive modification of software to repair problems discovered after the software has been delivered.
- ***Adaptive maintenance*** - reactive modification of software after delivery to keep the software usable in a changing environment.
- ***Perfective maintenance*** - proactive modification of the software after delivery to provide new user features, better program code structure, or improved documentation.
- ***Preventive maintenance*** – proactive modification software after delivery to correct product faults before discovery by users.
- In agile process models much (but not all) of the maintenance work is preventive or perfective as new features are added.

Maintenance Effort Distribution



Access the text alternative for slide images.

Because learning changes everything.®

www.mheducation.com