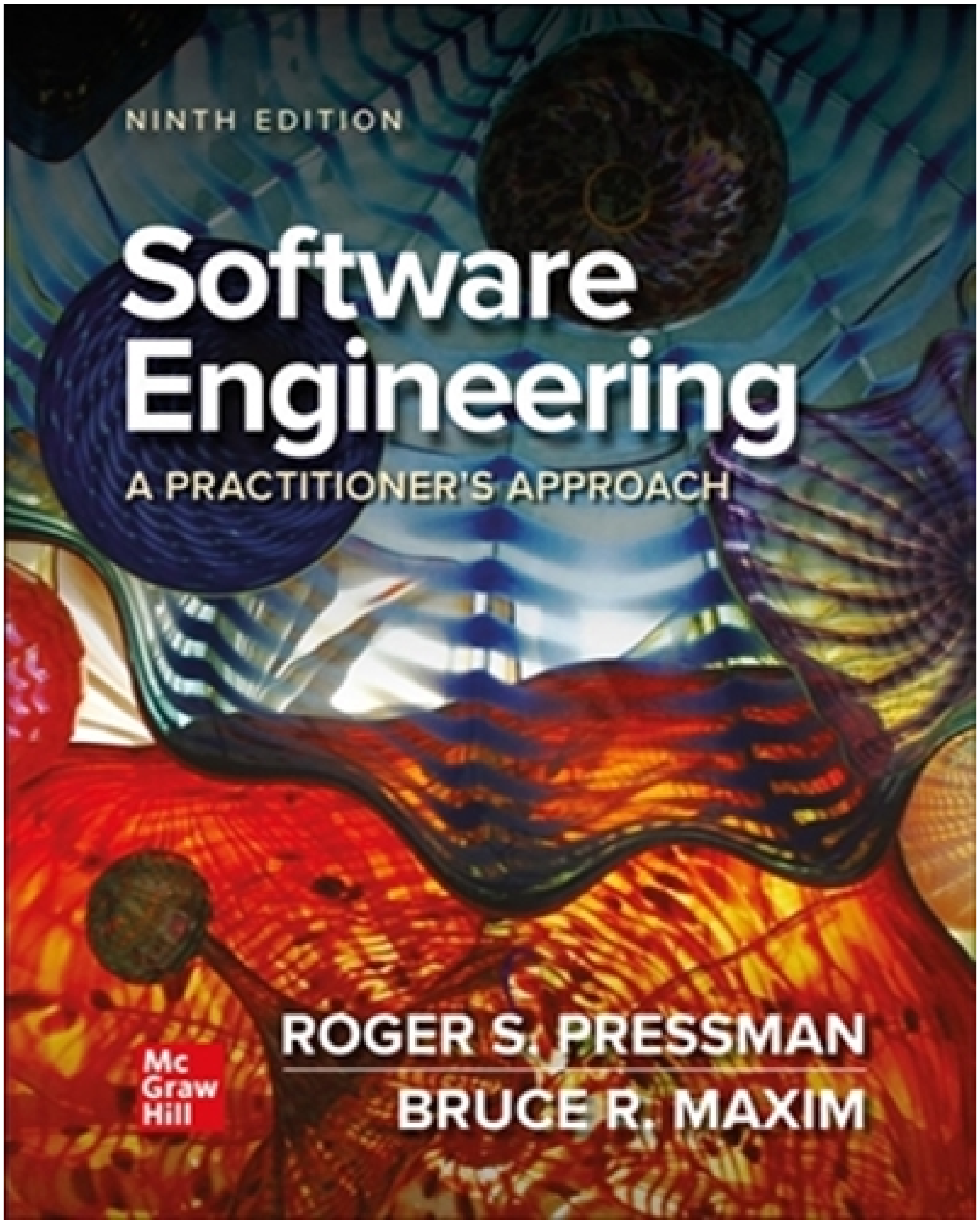
Solutions for Software Engineering: 9th Edition by Pressman



Solutions

Solutions: Chapter 2: Process Models.

2.1)

a) Designers should ask users:
   Is the product satisfactory, or does it require redesign or rework?
   Was user input solicited, to avoid the product being unsatisfactory and requiring rework?
   Is there a need for new requirements?
   Is the product larger than estimated?
   Do the modules require more testing, design and implementation work to correct than expected?

b) Users should ask as designers:
   Is the scope clear?
   Do we have the tools and people with skills required for the development?
   Are the requirements properly defined, are additional requirements needed.
   Are the specified areas of the product more time consuming than usual?
   Does the module require more testing, design?

c) Users should ask themselves about the software product that is to be built:
   What is the scope and purpose of the software product?
   Is the product larger than estimated?
   Are the best people available?
   Is the staff committed and possess skills required?
   Will the turnover among staff members be low enough to allow continuity?

d) Designers should ask themselves about software product that is to be built and the process that will used to build it:
   Scope and purpose of the document?
   What tools are to be used?
   What are the objectives and risk aversion priorities?
   What will be the steps for risk analysis, identification, estimation, and evaluation and management?

2.2)

a) Linear process flow does not accommodate change well, but can be good if a team is building a routine product similar to something they have done before
b) Iterative process flow handles change better by building in opportunities to reviews the intermediate work products as they are developed. Often used when building systems involving technologies that are new to the development team.
c) Evolutionary process models are often adopted for projects (e.g. WebApps) that need to be developed in a rapid, but controlled manner that avoids unnecessary rework.
d) Parallel process flow has the potential to allow self-contained work products to be developed simultaneously for systems that are composed of subsystems.

2.3) Task Set for Communication Activity: A task set would define the actual work to be done to accomplish the objectives of a software engineering action. For the communication activity these are:

Make a list of stakeholders for the project
Invite all the stakeholders to an informal meeting
Ask them to make a list of features and functions
Discuss requirements and build a final list
Prioritize requirements and note the areas that he is uncertain of
These tasks may be larger for a complex software project, they may then include
To conduct a series of specification meetings, build a preliminary list of functions and features based on stakeholder input.
To build a revised list of stake holder requirements
Use quality function deployment techniques to prioritize the requirements.
Note constraints and restrictions on the system.
Discuss methods for validating system.

2.4)

Pattern Name. *Conflicting Stakeholder Requirements*

Intent. This pattern describes an approach for resolving conflicts between stakeholders during the communication framework activity.

Type. Stage pattern

Initial context. (1) Stakeholders have been identified; (2) Stakeholders and software engineers have established a collaborative communication; (3) overriding software problem to be solved by the software teams has been established; (4) initial understanding of project scope, basic business requirements and project constraints has been developed.

Problem. Stakeholders request mutually conflicting features for the software product under development.

Solution. All stakeholders asked to prioritize all known system requirements, with resolution being to keep the stakeholder requirements with highest priorities and/or the most votes.

Resulting Context. A prioritized list of requirements approved by the stakeholders is established to guide the software team in the creation of an initial product prototype.

Related Patterns. Collaborative-guideline definition, Scope-isolation, Requirements gathering, Constraint Description, Requirements unclear

Known Uses/Examples. Communication is mandatory throughout the software project.

2.5)

The waterfall model is amenable to the projects that focus on the attributes such as the data structures, software architecture, and procedural detail and interface characterization of objects.

2.6)

Software applications that are relatively easy to prototype almost always involve human-machine interaction and/or heavy computer graphics. Other applications that are sometimes amenable to prototyping are certain classes of mathematical algorithms, subset of command driven systems and other applications where results can be easily examined without real-time interaction. Applications that are difficult to prototype include control and process control functions, many classes of real-time applications and embedded software.

2.7)

As work moves outward on the spiral, the product moves toward a more complete state and the level of abstraction at which work is performed is reduced (i.e., implementation specific work accelerates as we move further from the origin).

2.8)

The process models can be combined, each model suggests a somewhat different process flow, but all perform the same set of generic framework activities: communication, planning, modeling, construction, and delivery/feedback.

For example the linear sequential model can serve as a useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner. In cases, where the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human-machine interaction should take. In these, and many other situations, a prototyping model may offer the best approach. In other cases, an incremental approach may make sense and the flow of Spiral model may be efficient. Special process models take on many of the characteristics of one or more of the tradition.

2.9)

The advantages of developing software in which quality is ”good enough” is that the product or software will meet the deadline, it may however lead to the delivery of software that is low in quality and requires time to improper the quality. When speed is emphasized over the product quality it may lead to many flaws, the software may require more testing, design and implementation work then done. Requirements may be poorly defined and may need to continuously change. Half hearted and speed may cause the risk management to fail to detect major project risks .Too little quality may result in quality problems and later rework.

2.10)

It is possible to use mathematical techniques to prove the correctness of software components and even entire programs (see Chapter 28). However, for complex programs this is a very time consuming process. It is not possible to prove the correctness of any non-trivial program using exhaustive testing.

2.11)

UML provides the necessary technology to support object-oriented software engineering practice, but it does not provide the process framework to guide project teams in their application of the technology. Over the next few years, Jacobson, Rumbaugh and Booch developed the Unified Process, a framework for object-oriented software engineering using UML. Today, the Unified Process and UML are widely used on OO projects of all kinds.