Nickname: mike6606
Age: 8 years 10 months
Fans: 0
Followers: 5
+ Plus follow

< March 2023 >

day	One	Two	Three	Four	Five	Six
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Search

Look for it

Google search

Frequently used links

My essays
My comment
My involvement
Latest comments
My labels

Essay archives

January 2021 (1)
January 2018 (3)
January 2017 (8)

Read the leaderboard

1. Software Engineering Practitioners' Research Methods Chapter 13 Answers (1578)
2. Software Engineering Practitioners' Research Methods Chapter 1471 Answers (<>)
3. Software Engineering Research Methods for Practitioners Chapter 1290 Answers (<>)
4. Software Engineering Practitioners' Research Methods Chapter 34 Answers (1282)
5. Software Engineering Practitioners' Research Methods Chapter 1264 Answers (<>)

Recommended leaderboards

1. Software Engineering Practitioners' Research Methods Chapter 1 Answers (<>)

Software Engineering Practitioner Research Methods Chapter 7 Answers

Problem:
Since a focus on quality demands resources and time, is it possible to be agile and still maintain a quality focus?

Answer:

Yes, it is possible to maintain a quality focus while being agile. Agile methodologies are based on the iterative development. They involve a set of engineering best practices intended to allow for rapid delivery of high-quality software. Agile methodologies suggest to, keep the technical approach as simple as possible, keep the work products as concise as possible, and make decisions locally whenever possible. It means that unnecessary usage and wastage of resources must be reduced. So it does not affect the quality demands resources. Also by implementing agile methodologies, time can be saved because it suggests making most of the decisions locally.

Problem:Of the eight core principles that guide process (discussed in Section 7.2.1), which do you believe is most important?

Answer:

4633-4-2P SA: 4475

SR: 6376

Eight core principles that Guide Process are:

1. Be agile

2. Focus on quality at every step

3. Be ready to adapt

4. Build an effective team

5. Establish mechanisms for communications and coordination

6. Manage change

7. Assess risk

8. Create work products that provide value for others

Out of the eight core principles that guide process, we can consider that the fourth principle – **"building an effective team"** is most important, because the bottom line of a software process is people. Building a self-organizing team that has mutual trust and respect is needed. An effective team can only produce a quality product.

Problem:Describe the concept of *separation of concerns* in your own words.

Answer:

Separation of concerns:

• A Separation of concerns (SOC) is a large problem is easier to solve if it is subdivided into a collection of elements (or concerns).

• The Separation of concerns is states that a particular given problem involves different kinds of concerns, which should be identified and separated to deal with complexity and to achieve the required engineering quality factors such as robustness, flexibility, maintainability, reusability and different types of concerns may be relevant to different software engineers to different roles, and to achieve to different goals at different stages of the software lifecycle.

• Separation of concerns is a design concept that suggests that any complex problem can be more easily handled if it is subdivided into pieces that can each be solved and optimized independently.

• A concern is a behavior that is specified as part of the requirements model for the software. By separating concerns into smaller and therefore more manageable pieces, a problem takes less effort and time to solve.

• Separation of concerns is manifested in other related design concepts: modularity, aspects, functional independence and refinement.

you do? What work products might result as a consequence of early preparation?

Answer:

Before I communicate to others regarding my product, just I would under stand the total concept considering the do’ s & don’ ts. And , I identify my self as how that intends to ensure quality.

Planning is also an important role in the communication.

Planning – encompasses a set of management and technical practices that enable the product to define a road map as it travels toward its strategic goal and tactical objectives.

Planning Principles

1. Understand the scope of the product.

Scope provides the product team with a destination.

2. Involve the customer in the planning activity.

The customer defines priorities and established product constraints.

3. Recognize that planning is iterative.

The plan must be adjusted to accommodate changes.

In addition, iterative, incremental process models dictate replanting based on feedback received from users.

4. Estimate based on what you know.

The intent of estimation is to provide an indication of effort, cost, and task duration, based on the team’ s current understanding of the work to be done.

5. Be realistic.

Realities should be considered as a product plan is established.

People don’ t work 100 percent of every day.

Noise always enters into any human communication.

Omissions and ambiguity are facts of life.

Change will occur.

Even the best software engineers make mistakes.

6. Adjust granularity as you define the plan.

Granularity refers to the level of detail that is introduced as a product plan is developed.

A fine granularity plan provides significant work task detail that is planned over relatively short time increments.

A coarse granularity plan provides broader work tasks that are planned over longer time periods.

7. Define how you intent to ensure quality.

If formal technical reviews are to be conducted, they should be scheduled.

If pair programming is to be used during construction, it should be explicitly defined within the plan.

8. Describe how you intend to accommodate change.

Example situations:

Can the customer request a change at any time?

If a change is requested, is the team obliged to implement it immediately?

How is the impact and cost of the change assessed?

9. Track the plan frequently and make adjustments as required.

It makes sense to track progress on a daily basis, looking for problem areas and situations in which scheduled work dons not conform to actual work conducted.

When slippage is encountered, the plan is adjusted accordingly.

Problem:Do some research on “facilitation” for the communication activity (use the references provided or others) and prepare a set of guidelines that focus solely

Answer:

Communication is the exchange of thoughts, messages, or information, as by speech, signals, writing, or behavior.

The process of communication is one that is deeply affected by following factors:

• culture (corporate and societal),

• personal character traits,

• and the environmental setting.

• Good communication is essential for productive design and sound facilitation is essential for encouraging attendees to communicate during the work.

Some one should facilitate the activity. Every communication meeting should have a leader (facilitator) to keep

(1) Conversation moving in a productive direction

(2) To mediate any conflict that does occur

(3) To ensure than other principles.

(4) Provide training to staff for effective communication

(5) Ask everyone to be a good listener.

(6) Provide guidance about maintaining a healthy relationship among team members.

(7) Provide feedback at the end of the communication.

Problem:How does agile communication differ from traditional software engineering communication? How is it similar?

Answer:

592-5-6P SA Code: 0560

SR Code: 4578

The agile view of customer communication and collaboration emphasizes the need for all stakeholders to work (continuously) as a team, be co-located, and recognize the change is part of the process. However, the basic agile philosophy of communication is applicable to all software engineering practice and like all software engineering practices, communication and collaboration are iterative. Each iteration divulges important new information and enhances the overall understanding of the software to be built.

Communication is one of the fundamental activities of software development. Project initiation and requirements gathering takes place in communication activity, no matter whether it is an agile communication or ordinary software communication.

Agility is dynamic, content specific, aggressively change embracing and growth oriented. It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as project proceeds.

The planning activity in agile process begins with creation of set of stories that describe required features of software to be built. Each story is written by the customer and is placed on an index card. The customer assigns a value (priority to it). The software team will look after these stories and make changes.

Where as traditional software engineering communication is simpler job than agile communication. It has more time to analyze and implement the system.

Both agile communication & traditional software engineering communication are having the same background idea. They both are externally similar but coming to internal point of view i.e. implementation, they differ from each other.

Problem:Why is it necessary to “move on”?

Answer:

Move on

Communication is one of the major software engineering activities. In this “moving on”

Is the one of the concept of communication practices and sometimes it is the best way to achieve communication agility.

It is necessary to implement in software engineering practice. It works from essence towards the implementation.

Move on is consists of following principle.

1. After agree to something, move on for that task.

Example:

The analysis is a task that should "move on" from essential information toward to implement as detail.

If it is needed to spend the time to analyze the problem before meeting with other, then if necessary, perform some research to analyze business logic. So, it takes long time to understand a complex problem from different kinds of users.

Hence in this scenario, "move on" is the one of the major principle in communication activity.

Problem:Do some research on "negotiation" for the communication activity and prepare a set of guidelines that focus solely on negotiation.

Answer:

592-5-8P SA Code: 0560

SR Code: 4578

Negotiation is the key decision-making approach used to reach consensus whenever a person, organization or another entity cannot achieve its goals unilaterally. Negotiations appear in a multitude of forms, take place in very different situations and are influenced by ethical, cultural and social circum-stances.

The negotiation process as the process of interpersonal communication for the purpose of forming and modifying perceptions and attitudes. Negotiation is every process of social interaction and communication involving distributio and redistribution of power, resources, and commitments. A significant contribution of behavioral research is in numerous heuristics and qualitative models that have been shown to be useful in negotiation practice.

Most of the existing research concentrates on two directions: (1) traditional, face-to-face negotiations that rely on human expertise but little, if at all, on the information systems, and (2) formal models of idealized negotiators involved in complex strategic encounters. Electronic negotiations and processes that involve information systems as actively participants, negotiations among both human and artificial agents who use distributed knowledge and information require consideration of approaches from areas that traditionally were "foreign to each other." We nee to continue integrating concepts and approaches proposed in law and social science with those proposed in economic science and management.

In order to establish a common terminology and to facilitate multidisciplinary research, we propose to define negotiations in such a way that it is possible to include the various existing negotiation situations and approaches.

Negotiation as an iterative communication and decision making process between two or more agents (parties or thei representatives) who:

1. Cannot achieve their objectives through unilateral actions;

2. Exchange information comprising offers, counter-offers and arguments;

3. Deal with interdependent tasks; and

4. Search for a consensus which is a compromise decision.

The outcome of a negotiation can be a compromise (an allocation) or a disagreement. Negotiation arena is the accepted place where the negotiators communicate. The agenda specifies the negotiation framework, including the specification of the negotiated issues and format in which they are presented (e.g., sequentially or simultaneously). Decision-making rules are used to determine, analyze and select decision alternatives and concessions. Rules of communication determine the way offers and messages (arguments) are exchanged. Most negotiations follow certair rules; in negotiations in which software carries out some tasks, rules need to be explicitly specified allowing, among others, a distinction between tasks undertaken by a system and by people.

Problem:Describe what *granularity* means in the context of a project schedule.

Answer:

Granularity

The term granularity refers to detail with which some element of planning is represented or conducted. It is a type of communication activity.

It refers to the level of detail that is introduced in a project plan and development.

1. A "fine granularity" plan provides significant work task detail that is planned over relatively short time increments.

2. A "coarse granularity" plan provides broader work tasks that are planned over longer time periods.

The communication activity helps a software team to define overall goals and objectives. And these goals and objectives are not the same as defining a plan for getting. Planning activity is encompasses a set of management and

So, granularity moves from fine to course as the project timeline moves away from the current data. After that the project can be planned in significant detail. Activities won't occur for much time to do not require fine granularity.

Problem:Why are models important in software engineering work? Are they always necessary? Are there qualifiers to your answer about necessity?

Answer:

Models are important in software engineering work.

• They give a prototype of the actual product to be designed.

• They give a better understanding of the product to be designed.

• They help in reducing the complexity of the product to be designed.

• They act as a guide on how the software product can be built efficiently.

• They provide a technical guidance to the users who implement the software.

Models are important as well as necessary in software engineering work.

• They assist in developing the final product.

• Models usually help in getting the product to the customer faster.

• If a model slows down the process of getting the product to the customer, then such models can be avoided and ar not necessary.

Time taken to construct the model and complexity of the model can be considered as qualifiers for the necessity of the model in software engineering work.

• If a model slows down the process of getting the product to the customer, then such models can be avoided and ar not necessary.

• If a model is taking a lot of time to be built, then such models can be avoided and are not necessary.

Problem:What three "domains" are considered during requirements modeling?

Answer:

Analysis modeling represents the customer requirements by depicting the software in three different domains:

1. The information domain,

2. The functional domain, and

3. The behavioral domain.

Design models represent the characteristics of the software that help practitioners to construct it effectively: the architecture, the user interface, and component-level detail.

Problem:Try to add one additional principle to those stated for coding in Section 7.3.4.

Answer:

Additional principal about coding principal is '**secure coding principle**'.

Many security vulnerabilities could easily be prevented if security were taken into consideration at the beginning of the development process. Coding weaknesses degrade the security. So by using the security principles, we can avoid some of the most common security problems like

1. Weak file and group permissions

2. Race conditions

3. Buffer overflows

4. Problems with temporary files

5. Overly complex and unnecessary code

6. Insecure system calls and switches

7. Hard-coding passwords

Problem:What is a successful test?

Answer:

Thoroughly inspect the results of each test. Examining a program to see if it does not do what it is supposed to do is only half of the battle. The other half is seeing whether the program does what it is not supposed to do. Do not plan testing effort under the tacit assumption that no errors will be found. Testing is an extremity creative and intellectuall challenging task. Testing is the process of executing a program with the intent of finding errors. A good test case is one that has a high probability of detecting an as – yet undiscovered error. Testing is a never – ending process and Testing is not debugging. Test can never find 100% of the included errors. Test has got goal to find errors and not their reasons.

Problem:Do you agree or disagree with the following statement: "Since we deliver multiple increments to the customer, why should we be concerned about quality in the early increments—we can fix problems in later iterations." Explain your answer

Answer:

Consider the statement:

"Since we deliver multiple increments to the customer, why should we be concerned about quality in the early increments--we can fix problems in later iterations".

Strongly disagree with above statement.

Explanation:

In this statement, each increment builds on the next level and did not want to build a low quality foundation.

And from the early increments point of view, if the first increment is low in quality, then customers and users may become concerned for total team. Because each increment builds on the next and no one wants to build on a low quality foundation. If the first increments are low in quality then unnecessary tensions may develop. At this time, communication suffers follows.

Example: word processor

When word processor is purchased and just typing the word, then it works fine. It search the item and looking for. Bu the spellcheck doesn't find the misspelled word. Because, in paging it shows an error.

Software should not be released with any known bugs.

Problem:Why is feedback important to the software team?

Answer:

592-5-15P SA Code: 0560

SR Code: 4578

Feedback is important to the software team because it provides the information that can be used to correct modeling mistakes, change misinterpretations, and add features or functions that were inadvertently omitted.

The delivered software provides benefit for the end – user, but it also provides useful feedback for software team. As the delivered software application is implementing, the end – users should be encouraged to comment on features and functions, ease of use, reliability and characteristics that are appropriate.

Feedback must be collected from end-user and recorded by a software team. These are used to

• make immediate modifications to the delivered increment (if necessary)

• Define changes to be incorporated into the next planned increment.

• Make necessary design modifications to accommodate changes

• Revise the plan (including delivery schedule) for the next increment to reflect the changes.

Solutions Chapter 7: PRINCIPLES THAT GUIDE PRACTICE

7.1) It is possible to be agile and maintain a quality focus, if the team focus is to deliver the best possible increment t customer and to react to the customer feedback in a responsible manner.

7.2) Answers will vary

7.3) A large problem is easier to solve if it is subdivided into a collection of elements each of which delivers a distinct functionality that can be developed, and in some cases validated, independently of other concerns.

principles and concepts focus on the need to reduce noise and improve bandwidth as the conversation between developer and customer progresses. Both parties must collaborate for the best communication to occur.

7.5) "Facilitation" for the communication activity is very essential to carry out the plan Every communication meeting should have a leader (a facilitator) to keep the conversation moving in a productive direction; (2) to mediate any conflict that does occur; (3) to ensure than other principles are followed. The guidelines that focus on this are:

Understand the problem (communication and analysis).

Plan a solution (software design).

Carry out the plan (code generation).

Examine the result for accuracy (testing and quality assurance).

7.6) The agile view of customer communication and collaboration is applicable to all software engineering practice and like all software engineering practices, communication and collaboration are iterative, Each iteration divulges important new information and enhances the overall understanding of the software to be built.

7.7) It is necessary to "move on" because communication, like any software engineering activity, takes time. Rather than iterating endlessly, the people who participate should recognize that many topics require discussion and that "moving on" is sometimes the best way to achieve communication agility.

7.8) " Negotiation" for the communication activity works best when both parties win. Sets of guidelines that focus solely on negotiation are many instances in which the software engineer and the customer must negotiate functions and features, priorities, and delivery dates. If the team has collaborated well, all parties have a common goal. Therefore, negotiation will demand compromise from all parties.

7.9) Granularity refers to the level of detail that is introduced as a project plan is developed. A 'high granularity' plan provides significant work task detail that is planned over relatively short time increments (so that tracking and control occur frequently). A 'low granularity' plan provides broader work tasks that are planned over longer time periods. In general, granularity moves from high to low as the project timeline moves away from the current date.

7.10) The software design model is the equivalent of an architect' s plans for a house. It begins by representing the totality of the thing to be built (e.g., a three-dimensional rendering of the house) and slowly refines the thing to provide guidance for constructing each detail (e.g., the plumbing layout). Similarly, the design model that is created for software provides a variety of different views of the system, hence it is always necessary.

7.11) Analysis models represent the customer requirements by depicting the software in three different domains: the information domain, the functional domain, and the behavioral domain. Design models represent characteristics of the software that help practitioners to construct it effectively: the architecture, the user interface, and component-level detail.

7.12) One additional principle to those stated for coding would be: Code should be written in a manner it explains th logic even to the user or who is not a developer, self-explainatory and meets the logic of it' s working.

7.13) A successful test is one that uncovers an as-yet-undiscovered error.

7.14) It would be preferable to disagree with the statement, "Since we deliver multiple increments to the customer, why should we be concerned about quality in the early increments—we can fix problems in later iterations." It is always advisable to have a quality check throughout, as the customers will forget you delivered a high quality produc a few days late, but they will never forget the problems that a low quality product caused them.

7.15) Feedback is important to the software team, because it provides the information that can be used to correct modeling mistakes, change misinterpretations, and add features or functions that were inadvertently omitted.

mike6606
followers - 0 followers - 5

0 0

« Previous: Software Engineering Research Methods for Practitioners Chapter 6 Answers
» Next: Software Engineering Research Methods for Practitioners Chapter 8 Answers

posted @ 2021-01-20 14:51 mike6606 read (1295) Comments (0) Edit Favorite repor