CS 410/560 - Software Engineering

Assignment 2 G C

Group members: Yi Ren (002269013), Wentao Lu (002276355)

Configuration Management

1. What are the main tasks of configuration management?

The major task of configuration management is to keep track of all artifacts during the software development life cycle, it entails procedures and work flows on how to deal with changes and maintain system integrity across different versions. Besides, it also has a set of support tools to manage changes, control releases and trace version history to guarantee a stable system.

[1, page 81-86]

2. Describe the role of the Configuration Control Board.

A Configuration Control Board (CCB) is responsible for monitoring changes to the baseline. It decides whether to approve, reject or defer a change request, which aims to ensure that any change is well scrutinized, properly handled and conducted in the right manner.

[1, page 82-83]

3. What is a configuration item?

A configuration item is a stand-alone entity intended to support the process of configuration management. It is a conglomerate of hardware, software or possibly both. Possible configuration items include but are not limited to:

- source code components
- the requirements specification
- the design documentation
- the test plan
- test cases
- test results
- the user manual

[1, page 81]

4. What is a baseline?

A baseline is an officially reviewed version of a software specification or product, where the complete set of documents have been approved and recognized based on consensus. A baseline plays the role of a concrete ground on which further development is built, it can only be changed through formal change control procedures. In practice, this can be a large stable version of the software.

[1, page 81]

5. Explain the difference between version-oriented and change-oriented configuration management.

In a version-oriented configuration management model, a new version stems from physical changes in a component, and different versions are featured by their difference. In contrast, change-oriented configuration management employs logical changes rather than physical ones as a basic unit of work, in this way, configuration identification is more intuitive, which makes the work flow more straightforward and less error-prone.

[1, page 85]

6. Discuss the main contents of a configuration management plan.

A configuration management plan describes methods to identify configuration items, to control change requests, and to document the implementation of those change requests. There are mainly two components of this plan:

The **Management** section focuses on how to organize the project, particularly on how to split and assign duties in a proper manner. This typically includes managing the way of handling change requests, closing development phases, checking system status, identifying interfaces between components and quality assurance, to name just a few.

The **Activities** section describes the way of identifying and controlling a configuration, accounting and reporting its status as appropriate. A configuration is identified by its complete set of artifacts, so it must be formally approved by the parties involved.

[1, page 86-87]

7. Discuss differences and similarities between configuration management during development and maintenance.

Similarities: Configuration management plays the same role in both development and maintenance, which is to ensure that any change is inspected, identified and controlled and then properly carried over to the next version.

Differences: During development, change requests are submitted by both developers and users, most of which are identified in testing, and the current development progress can have a notable impact on how those change requests will be handled. During maintenance, however, change requests are primarily identified in production and issued by users, and the work must be done in a separate branch parallel to the baseline to ensure continuing business of the current released production version.

[1, page 81-86]

8. Configuration management at the implementation level is often supported by tools. Can you think of ways in which such tools can also support the control of other artifacts (design documents, test reports, etc.)?

It goes without saying that the idea of using tools to aid and facilitate human work not only happens at the implementation level, but also applies to the control of other artifacts like design documents and test reports. These artifacts per se are no different than pieces of code blocks.

For instance, there are many automated testing tools such as *selenium* that automates testing web applications in browsers, *Traves CI* that automatically detects and rebuilds a repository. There

are also many modules that deals with logging information, which helps us keep track of the program and find bugs more easily. When it comes to requirement engineering and design, there are still many software tools that assist in the drawing of UML diagrams.

[1, page 500-501]

Software Management

1. Explain Mintzberg's classification of organizational configurations and their associated coordination mechanisms.

According to Mintzberg, there are 5 distinct types of organizations, each calls for a different coordination mechanism. Therefore, we have to consider the influence of these external factors in software development.

- Simple structure: This simple configuration is common in relatively new and small organizations where a small number of people are supervised by one or few managers to do the work. This coordination mechanism is called direct supervision. In such a simple structure, specialization, training and formalization are not emphasized, coordination takes place only when individuals are responsible for the work of others.
- Machine bureaucracy: In this type of configuration, workers are given well-defined duties and tasks that follow a set of precise instructions. Therefore, it's much easier to carry out those tasks, and performance can be evaluated accordingly as well. The most common scenarios of this type are mass-production and assembly lines, where little training is required but specialization and formalization are heavily focused. In this case, coordination is achieved through standardization of work processes.
- Divisionalized form: In such a decentralized organization, a single division or project has the right to set their own goal and then move forward towards it. The operating details are also up to the division itself, with only limited supervision in the form of regular KPI assessment. This coordination mechanism is not feasible unless the end result has been well defined.
- Professional bureaucracy: If it's hard to precisely define either the end result or the work contents, then a professional bureaucracy comes into play. In this scenario, skilled individual practitioners are granted considerable freedom to do their work based on their own judgement, and coordination can be achieved through standardization of professionalism. Hospitals are typical examples of this type.
- Adhocracy: This happens when a group of people work together to carry out a large complex project where work is divided amongst many specialists. Neither the end result nor the work contents can be well specified, even professionalism can not guarantee to save the world. Given the complexity of context, it's hard to tell what each individual should do and how he does it. Therefore, the key to success is coordination via mutual adjustment, and it has to be an effort on everybody's part.

[1, page 94-95]

2. Discuss Reddin's basic management styles.

Depend on whether attention is given to relation directedness or task directedness, there are four basic management styles.

Separation style mainly focuses on efficiency. The organization forms a top-down hierarchical structure where people on the top make decisions and give commands to people in the lower. This management style leads to a stable project organization, but the dictatorship tends to rule out the possibility of innovations. This style corresponds to Mintzberg's coordination through standardization of work processes.

Relation style is expected in situations where people have to be motivated, coordinated and trained. Each individual does his own work that requires much innovation, complexity and specification. When it comes to making decisions, a group needs to reach a consensus through negotiation. The potential downside is that compromises could possibly lead to long discussions that won't bear fruit. As a result, the manager must be able to harmonize the group and help achieve an agreement. This style fits Mintzberg's mutual adjustment coordination mechanism.

Commitment style is most expected when work involves much pressure. More attention is given to tasks rather than relations, and decisions are implied by the team's shared vision as to the goals of the project. However, once this vision has been established, the team is inclined to comply with this vision regardless of changes in the environment, as opposed to responding differently to these changes. This style best fits Mintzberg's professional bureaucracy.

Integration style puts emphasis on both tasks commitment and coordination relations. It happens when the result is hard to predict and work is highly complex, innovative and include several independent tasks. This management style encourages innovation and self-challenge for each worker, decision-making is bottom-up. However, an individual team member's goals may be detached from those of the project, and they are likely to compete with one another. This style also fits Mintzberg's mutual adjustment coordination mechanism.

[1, page 95-96]

3. What are the critical issues in a hierarchical team organization?

First, in a hierarchical organization structure, different divisions may adopt different management styles and coordination mechanisms to meet their own needs, which is likely to result in a conflict at some intermediate level. Another critical issue is concerned with the distance between top and bottom of the hierarchical pyramid. If this distance is too long, there is going to be a discrepancy of knowledge between the top and bottom, because information tends to be distorted and becomes less reliable as it travels upward through each level. Finally, it is inevitable that one is judged socially and financially by his level in the pyramid.

[1, page 98-99]

4. Highlight the differences between a chief programmer team, a SWAT team and an agile team.

In a Chief Programmer Team, the chief programmer as the team leader is not only in charge of design and implementation of key components, but also management and making decisions. Although the chief programmer has an assistant, he still has to be very competent in both technical and management roles, such demands are very rigid. In a SWAT team, a small number of skilled workers share a workspace. Instead of meetings, they only have short communication and brainstorming sessions, and they use tools to support their work and coordination. Leader of a SWAT

team is similar to a chief programmer, but other members are also competent in a variety of tasks. An agile team has a lot in common with the SWAT team, but people usually work in pairs. They emphasize more on self-motivation and self-discipline.

[1, page 100-102]

5. Which of Reddin's management styles fits in best with an agile team?

Relation style best fits an agile team, as it is most effective in situations where people have to be motivated and coordinated. Relation style treats decision-making as a group process, this is consistent with the people-oriented attitude in an agile team.

[1, page 96, 101]

6. What is the Peter Principle? Where does it crop up in software development?

The Peter Principle says that each employee in a hierarchical organization in general rises until reaching a level at which he is incompetent. This problem appears in team organization, particularly in hierarchical organizations.

[1, page 99]

7. Why would an agile team need better people than a team following a planning-based approach?

Because agile processes need more self-discipline and self-motivation. Agile team members must consider the system as a whole after incorporating a piece of work, and refactor if needed.

[1, page 101-102]

On Managing Software Quality

1. Define the term representation condition. Why is it important that a measure satisfies the representation condition?

The representation condition states that empirical relations between objects in the real world should be preserved in the numerical relation system that we use. That is, the measure must make sense in a real-world model. If a measure satisfies the representation condition, it is said to be a valid measure. Otherwise, a measurement is meaningless as it cannot be used to interpret real-world models.

[1, page 116]

2. What is the main difference between an ordinal scale and an interval scale? And between an interval scale and a ratio scale?

Interval scale is not only ordinal, but the distance between successive values of an attribute must also be the same. A ratio scale not only has equal distances between attributes, but there must also exists a value of 0 along the scale.

[1, page 114]

3. What are the main differences between the user-based and product-based definitions of quality?

The user-based quality is mostly concerned with the extent to which a system satisfies the user's needs, which is subjectively based on an individual user's perceived satisfaction. The product-based quality relies on attributes of a software, which is more objective in the sense that quality solely depends on the values of those attributes.

[1, page 125]

4. Which of Garvin's definitions of quality is mostly used by the software developer? And which one is mostly used by the user?

Software developers usually address the product-based and manufacturing-based definitions of quality, which is objective and quantifiable. Users prefer to adopt the user-based definition of quality since they have subjective evaluation.

[1, page 125]

5. Why should the Software Quality Assurance organization be independent of the development organization?

Because there are potential conflicts of interest between the SQA organization and the development organization, for example, in case some hard issues were detected just before the deadline, the development organization might want to ship the product, while SQA on its own standpoint would prefer to defer shipment. This is obvious, as a player on the court cannot be a referee at the same time.

[1, page 129-130]

6. Why should project members get feedback on the use of quality data they submit to the Quality Assurance Group?

Because if no feedback is received on the submitted data, project members will not have the incentive to take it seriously. As a result, data collection may be incomplete or imprecise, the reliability of data shrinks.

[1, page 130-131]

7. Describe the maturity levels of the Capability Maturity Model.

Initial: At the initial process level, no procedures, project plans, or cost estimates have been formalized since we are just starting off. At this level, we can improve performance by establishing basic project management controls:

- Requirement management
- Project planning
- Project monitoring and control
- Supplier agreement management
- Measurement and analysis
- Process and product quality assurance
- Configuration management

Repeatable: This level places control over how plans and commitments are established. Although some control have been achieved through prior experience, software process must be standardized across the projects of the organization to advance to the next level, this includes:

- Requirements development
- Technical solution
- Product integration
- Verification
- Validation
- Organization process focus
- Organization process definition
- Organizational training
- Integrated project management
- Risk management
- Decision analysis and resolution

Defined: This level now is equipped with a set of standard processes for the development and maintenance of software. Based on this solid foundation, the major tasks at this level is to improve the processes.

- Organizational process performance
- Quantitative project management

Quantitatively managed: At this level, everything is under control, data has been collected and analyzed on a regular basis. Now we will focus on the chances and possibilities for continuous improvement, which includes:

- Organizational innovation and deployment
- Causal analysis and resolution

Optimizing: At this final level, a stable baseline has been achieved, thus attention will shift from the product to the process. Collected data can then be used to further improve the software development process.

[1, page 132-136]

8. What is the major difference between level 2 and level 3 of the Capability Maturity Model?

At Level 2, the repeatable level, software processes have not yet been standardized across the projects of the organization. At level 3, the defined level, standardized processes for development and maintenance have already been established, so we start to look at performance, and the focus shifts to process improvement.

[1, page 134-135]

9. What is the difference between the staged and continuous versions of CMMI?

In a staged model CMMI, there are only five levels of maturity. In a continuous model CMMI, process improvement is done on every process area, which gives us much more information about the software quality.

[1, page 138]

10. Why is it important to quantify quality requirements?

Without a quantifiable measure of quality requirements, there will be no objective target that all parties can easily agree on. Since everyone can have a different understanding of that quality requirement, a lot of ambiguity is going to be introduced.

[1, page 128-129]

Cost Estimation

1. In which ways may political arguments influence cost estimates?

Since political arguments tend to influence people involved in the project and introduce biases, but cost estimates are usually given by the same people, so cost estimates will be biased as well.

[1, page 168]

2. How may early cost estimates influence the way in which a project is executed?

From the perspective of developers, if a very demanding deadline is given, they will have to sacrifice development quality in order to ship the product on time, but the maintenance will be much more expensive. In contrast, if the schedule is not tight, developers are more likely to devote themselves to high-quality development, and this will save money in the maintenance phase.

[1, page 170]

3. Why is it difficult to compare different cost estimation models?

Different models use different measures and cost drivers, this makes it very difficult to compare one to another. Unless there's a universal standard that specifies how to measure cost and effort, we cannot come to a reliable conclusion as to which one is better. In fact, estimate always comes with uncertainty, we can only guess at the best, so the answer of a better model may vary on a case-by-case basis depending on the context, there's no best model in the world. We should just take it with a grain of salt and use the one that fits us better.

[1, page 170-171]

4. Suppose you are involved in a project, which is estimated to take 100 man months. How would you estimate the nominal calendar time required for this project? Suppose the project is to be finished within six calendar months. Do you think such a schedule compression is feasible?

Without further background information and domain knowledge, we can use one of the cost estimation models to obtain an estimate of the development time. For example, according to the organic COCOMO model, the nominal calendar time would be 100 to the power of 0.38 times 2.5, which is roughly 14 calendar months. According to Boehm's rule of thumb, which says that compressing the development time by X% results in a cost increase of X% relative to the nominal cost estimate, if we compress such a 14-month project to be finished within 6 months, then the cost will increase by over 130%, obviously this is not feasible in the real world.

[1, page 173]

5. Suppose you have a LOC-based cost estimation model available whose parameters are based on projects from your own organization that used COBOL as the implementation language. Can you use this model to estimate the cost of a project whose implementation language is Pascal? What if the model is based on projects that used C?

No, a cost estimation model is environment-specific, it can only be used to estimate similar projects. If the implementation language has changed, then the project environment is completely new. For example, we may need another group of developers who are able to code in Pascal or C to do the work, we may need a different set of tools for debugging and testing, in some cases the system architecture also needs to be changed to accommodate a different language. Therefore, we must have the model recalibrated to fit Pascal or C before it can be used.

[1, page 164]

References

1. Hans van Vliet. Software Engineering: Principles and Practice. Wiley 2007.