1. 填空
2. 三个一般化的软件过程模型？

What are the three generic software process model? （reference: Book P29）

A software process model is a simplified representation of a software process. Each process model represents a process from a particular perspective, and thus provides only partial information about that process. These generic models are not definitive descriptions of software process. Rather, they are abstractions of the process that can be used to explain different approaches to software development.

The waterfall model（瀑布模型）: this takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on.

Incremental development（增量式开发）: This approach interleaves the activities of specification, development, and validation. The system is developed as a series of versions (increments), with each version adding functionality to the previous version.

Reuse-oriented software engineering（面向重用软件工程）: This approach is based on the existence of a significant number of reusable components. The system development process focuses on integrating these components into a system rather than developing them from scratch.

fundamental development activities（开发过程中的基本活动有哪些）:

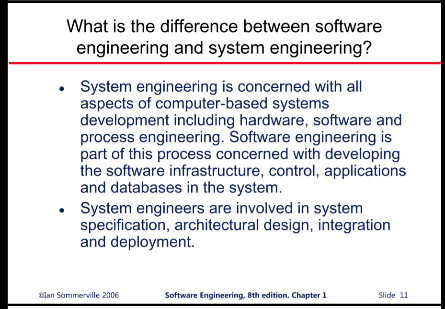
requirements analysis and definition, system and software design, implementation and unit testing, Integration and system testing, operation and maintenance.

1. 软件工程与其他工程学科的差异性？

What sets software engineering apart from other engineering discipline？

(reference: Book P7）

In general, software engineers adopt a systematic and organized approach to their work, as this is often the most effective way to produce high-quality software. However, engineering is all about selecting the most appropriate method for a set of circumstances so a more creative, less formal approach to development may be effective in some circumstances.

(What is Software engineering?) Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use. Software engineering is not just concerned with the technical process of software development. It also includes activities such as software project management and the development and the development of tools, methods, and theories to support software production.

(What is engineering?) Engineering discipline: Engineers make things work. They apply theories, methods, and tools where these are appropriate. However, they use them selectively and always try to discover solution to problems even when there are no applicable theories and method. Engineers also recognize that they must work to organizational and financial constraints, so they look for solutions within these constraints.

1. 对风险的分析过程主要有哪些方面，分为哪几个等级

What are the main aspects considered in the process of risk analysis and how can we evaluate the risk?

(reference: Book P599）

two main aspects are: the probability and effects (seriousness of each risk)

The probability of the risk might be assessed as very low(＜10), low (10-25), moderate(25-50), high(50-75), or very high(>75)

The effects of the risk might be assessed as catastrophic (threaten the survival of the project), serious (would cause major delays), tolerable (delays are within allowed contingency), or insignificant. (认识五个等级并且正确的排序)

1. 什么是风险策略计划

What is risk strategy planning?

(reference: Book P600）

（what is risk planning?）The risk planning process considers each of the key risks that have been identified and develops strategies to manage these risks. There are mainly three categories:

Avoidance strategies（回避策略，降低风险发生的可能性）: following these strategies means that the probability that the risk will arise will be reduced. An example of risk avoidance strategy is the strategy for dealing with defective components (replace potentially defective components with bought-in components of known reliability)

minimization strategies（最小化损失策略，降低风险可能带来的损失）: following these strategies means that the impact of the risk will be reduced. An example of a risk minimization strategy is the strategy for staff illness (Reorganize team so that there is more overlap of work and people therefore understand each other’s jobs)

contingency plans（应急计划，对最坏情况做好打算）: following these strategies means that you are prepared for the worst and have a strategy in place to deal with it. An example of a contingency strategy is the strategy for organizational financial problems (prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective)

hints提示：

you can see a clear analogy类比 here with the strategies used in critical systems to ensure reliability, security, and safety, where you must avoid, tolerate, or recover from failures. Obviously, it is the best to use a strategy that avoids the risk. If this is not possible, you should use a strategy that reduces the chances that the risk will have serious effects. Finally, you should have strategies in place to cope with the risk if it arises. These should reduce the overall impact of a risk on the project or product.

1. 需求工程过程包含哪些活动，请按顺序列出来

what activities are included in the requirement engineering process, please list them orderly.

(reference: Book P37）

(what is requirement engineering process?) the requirements engineering process aims to produce an agreed requirements document that specifies a system satisfying stakeholder requirements. Requirements are usually presented at two levels of detail. End-users and customers need a high-level statement of the requirements; system developers need a more detailed system specification.

There are four main activities in the requirements engineering process:

The order is Important顺序是很重要的！

1. Feasibility study

an estimate is made of whether the identifies user needs may be satisfied using current software and hardware technologies.

1. Requirements elicitation and analysis

This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers(购买方), task analysis, and so on.

1. Requirements specification

requirements specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements.

1. Requirements validation

This activity checks the requirements for realism, consistency, and completeness.

1. 软件产品有什么重要的属性？

What important attributes does a software product possess?

(reference: Book P8)

Maintainability: software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment

Dependability and security: Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.

Efficiency: Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.

Acceptability: software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable, and compatible with other systems that they use.

1. what is a software system?

软件系统是什么

(reference: Book P5)

A professionally developed software system is often more than a single program. The system usually consists of a number of separate programs and configuration files that are used to set up these programs. It may include system documentation, which describes the structure of the system; user documentation, which explains how to use the system, and websites for users to download recent product information. 软件系统并不是软件本身，还包括配置文件，系统文档，用户文档和网站等。软件系统的范畴要比软件更广

what is software: computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.

what are the attributes of good software? (reference: Book P6) Good software should deliver the required functionality and performance to the user and should be maintainable, dependable, and usable.

1. What is defect testing and what’s validation testing?

什么是缺陷测试 以及有效性测试？

(reference: Book P206)

what are the goals of the testing process? the testing process has two distinct goals.

1. to demonstrate to the developer and the customer that the software meets its requirements. (validation testing, where you expect the system to perform correctly using a given set of test cases that reflect the system’s expected use.) For custom software, this means that there should be at least one test for every requirement in the requirements document. For generic software products, it means that there should be tests for all of the system features, plus combinations of these features, that will be incorporated in the product release.
2. to discover situations in which the behavior of the software is incorrect, undesirable, or does not conform to its specification. These are a consequence of software defects. The second goal leads to defect testing, where the test cases are designed to expose defects. The test cases in defect testing can be deliberately obscure and need not reflect how the system is normally used. Defect testing is concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations, and data corruption.

There is no definite boundary between these two approaches to testing. During validation testing, you will find defects in the system; during defect testing. some of the tests will show that the program meets its requirements.

V & V

validation: are we building the right product

The aim of validation is to ensure that the software meets the customer’s expectations. It goes beyond simply checking conformance with the specification to demonstrating that the software does what the customer expects it to do.

verification: are we building the product right

The aim of verification is to check that the software meets its stated functional and non-functional requirements.

1. Please list 4-5 functionalities of CASE Tools/methods.

请列举CASE工具的几个功能

(reference: Book P37)

what is CASE tool: software development tools (sometimes called computer-aided software Engineering or CASE tools) are programs that are used to support software engineering process activities. These Tools therefore include design editors, data dictionaries, compilers, debuggers, system building tools, etc.

functionalities

software tools provide process support by automating some process activities and by providing information about the software that is being developed. Examples of activities that can be automated include:

1. the development of graphical system models as part of the requirements specification or the software design
2. the generation of code from these graphical models
3. the generation of user interfaces from a graphical interface description that is created interactively by the user
4. program debugging through the provision of information about an executing program
5. the automated translation of programs written using an old version of a programming language to a more recent version.
6. What is component interface?

组件接口的概念

(reference: Book P457)

Components have two related interfaces. These interfaces reflect the services that the component provides and the services that the component requires to operate correctly:

requires interface 需求接口: defines the services that are needed and should be provided by other components. The ‘requires’ interface specifies what services must be provided by other components in the system if a component is to operate correctly. If these are not available, then the component will not work.

provides interface 提供接口: defines the services that are provided by the component to other components. the ‘provides’ interface defines the services provided by the component. It defines the methods that can be called by a user of the component.

1. 业务流程重组中有哪些活动

what’s done during re-engineering (the activities in reengineering process)

(reference: Book P249)

a general model of the reengineering process. the input to the process is a legacy program and the output is an improved and restructured version of the same program. The activities in this reengineering process are as follows.

1. source code translation: Using a translation tool, the program is converted from an old programming language to a more modern version of the same language or to a different language.
2. Reverse engineering: the program is analyzed, and information extracted from it. This helps to document its organization and functionality. Again, this process is usually completely automated.
3. Program structure improvement: The control structure of the program is analyzed and modified to make it easier to read and understand. This can be partially automated but some manual intervention is usually required.
4. Program modularization: Related parts of the program are grouped together and, where appropriate, redundancy is removed. In some cases, this stage may involve architectural refactoring (e.g., a system that uses several different data stores may be refactored to use a single repository) This is a manual process.
5. Data reengineering: The data processed by the program is changed to reflect program changes. This may mean redefining database schemas and converting existing databases to the new structure. You should usually also clean up the data. This involves finding and correcting mistakes, removing duplicate record, etc. Tools are available to support data reengineering.

program reengineering may not necessarily require all of the steps above.

the costs of reengineering obviously depend on the extent of the work that is carried out.

The problem with software reengineering is that there are practical limits to how much you can improve a system by reengineering.

1. 项目管理是如何显化软件产品的工作结果的？

How can project management evaluate intangible software product

(reference: eBook P676)

Milestones and deliverables

Milestones are points in the schedule against which you can assess progress, for example, the handover of the system for testing.

Deliverables are project results delivered to customers, for example a requirements document for the system.

1. Corba 标准

Corba Standard

(reference: Book P482)

Common Object Request Broker Architecture

CORBA is a well-known specification for middleware system that was developed in the 1990s by the Object Management Group. It was intended as an open standard that would allow the development of middleware to support distributed component communications and execution, plus provide a set of standard services that could be used by these components.

Several implementations of CORBA were produced but the system never achieved critical mass. It was put forward quite early but is not widely used.

1. 在人员管理中，人员分成三大类

In people management, staff are divided by Bass and Dunteman based on personality types into three categories?

(reference: Book P606)

personality type influences motivation

1. task-oriented people. who are motivated by the work they do. In software engineering, these are people who are motivated by the intellectual challenge of software development.
2. Self-oriented people: who are principally motivated by personal success and recognition. They are interested in software development as a means of achieving their own goals. This does not mean that these people are selfish and think only of their own concerns. Rather, they often have longer-term goals, such as career progression, that motivate them and they wish to be successful in their work to help realize these goals.
3. interaction-oriented people: who are motivated by the presence and actions of co-workers. As software development becomes more user-centered, interaction-oriented individuals are becoming more involved in software engineering.
4. 为什么要在软件开发过程中使用 基于复用的开发， 复用的概念

Why implements reuse-based development in software development?

(reference: Book P426)

What is Reuse-based software engineering? Reuse-based software engineering is a software engineering strategy where the development process is geared to reusing existing software.

Reuse-based software engineering is an approach to development that tries to maximize the reuse of existing software.

The software units that are reused may be of radically different sizes.

Application system reuse, component reuse, Object and function reuse

Benefits (reference: Book P427)

1. Increased dependability可靠性: reused software, which has been tried and tested in working systems, should be more dependable than new software. Its design and implementation faults should have been found and fixed.
2. Reduced process risk 更小的损失: The cost of existing software is already known, whereas the costs of development are always a matter of judgment. This is an important factor for project management because it reduces the margin of error in project cost estimation. This is particularly true when relatively large software components such as subsystems are reused.
3. Effective use of specialists 对专家的有效利用: Instead of doing the same work over and over again, application specialists can develop reusable software that encapsulates their knowledge.
4. Standards compliance 统一服从的标准: Some standards, such as user interface standards, can be implemented as a set of reusable components. For example, if menus in a user interface are implemented using reusable components, all applications present the same menu formats to users. The use of standard user interfaces improves dependability because users make fewer mistakes when presented with a familiar interface.
5. Accelerated development 更快速的开发: Bringing a system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time may be reduced.

Problems. (reference: Book P427)

increased maintenance costs, lack of tool support, not-invented-here syndrome, creating, maintaining, and using a component library, Finding, understanding, and adapting reusable components.

1. 严格定义两种进化式开发的方法？

define two types of evolutionary development.

(reference week1-3 p9)

evolutionary development

Exploratory development

objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements and add new features as proposed by the customer.

Throw-away prototyping

objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

17-18 是相互关联的两个概念

* Over the years many methods have been proposed for analysis modeling. However, two now dominate. The first, structured analysis, is a classic modeling method and is described in 17, the other approach, object-oriented analysis, is considered in detail in 18.

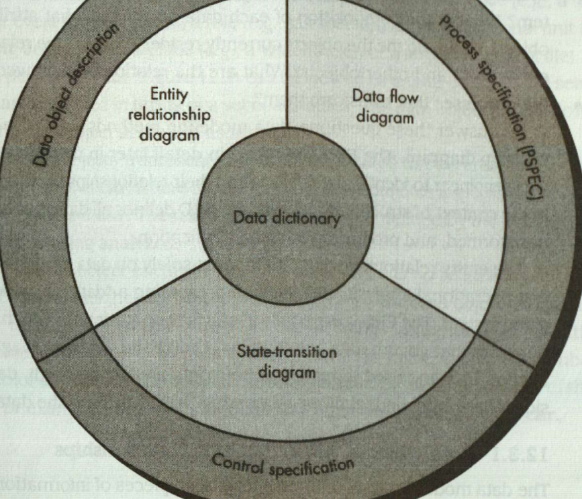
1. 在传统的结构设计中要建哪些模型？

what models are involved in a conventional structure design?

analysis model

the analysis model must achieve three primary objectives:

1. to describe what customer requires
2. to establish a basis for the creation of a software design
3. to define a set of requirements that can be validated once the software is build to accomplish



the structure of analysis model

data object description ER diagram

Process specification

PSPEC data flow diagram

Control specification sate-transition diagram

data modeling:

data modeling answers a set of specific questions

Entity Relation diagram

functional modeling and information flow:

information is transformed as it flows through a computer-based system. the system accepts input in a variety of forms, applies hardware, software, and human elements to transform it and produces output in a variety of forms.

behavioral modeling: is an operational principle for all requirements analysis methods.

control flow model

1. 面向对象的设计中，有哪些模型是静态的，有哪些模型是动态的？

What are the static system models and dynamic system models in OO design process.

(reference PPT week5-1)

Static models describe the static structure of the system in terms of object classes and relationships.

sub-system models

class diagram (book 129)

class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.

generalization (book 131)

Generalization is an everyday technique that we use to manage complexity.

Dynamic models describe the dynamic interactions between objects.

sequence models

state machine models

behavioral models (book P133)

Behavioral models are models of the dynamic behavior of the system as it is executing. They show what happens or what is supposed to happen when a system responds to a stimulus form its environment. (stimulus刺激，激励， can be Data, or events)

1. 两种迭代过程模型

two types of iterative process models 没找到 答案不确定

what is iterative process model? iterative process models describe the software process as a cycle of activities

book P47

Incremental delivery, Boehm’s spiral model

1. validation 和 verification 是什么？

validation: are we building the right product

The aim of validation is to ensure that the software meets the customer’s expectations. It goes beyond simply checking conformance with the specification to demonstrating that the software does what the customer expects it to do.

verification: are we building the product right

The aim of verification is to check that the software meets its stated functional and non-functional requirements.

1. P-CMM 模型的5 个level

what are the 5 levels of P-CMM People capability maturity model.

(reference book 606 PPT week4-2 p41)

what is P-CMM model? The people capability maturity model is a framework for assessing how well organizations manage the development of their staff. It highlights best practice in people management and provides a basis for organizations to improve their people management processes.

five stage model

initial. Ad-hoc people management

repeatable. Policies developed for capability improvement

defined. Standardized people management across the organization.

managed. Quantitative goals for people management in place

Optimizing. Continuous focus on improving individual competence and workforce motivation

1. 判断题
2. 概念题
3. 功能性需求和非功能性需求是指？

what are functional and non-functional requirements

(reference book p84)

Software system requirements are often classified as functional requirements or non-functional requirements:

Functional requirements

These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do.

Functional system requirements vary from general requirements covering what the system should do to very specific requirements reflecting local ways of working or an organization’s existing systems.

因为功能性需求在明确的时候，往往会因为粒度（需求具体程度）不同导致产生歧义性。那么在实现的过程中，设计师往往会按照更易于实现的理解方式去实现功能，这有可能会导致无法满足后期要求，所以需要进行额外的变更和修改，带来不必要的产品损失

（two criterion for functional requirements specification 功能性需求明确的两条准则）in principle, the functional requirements specification of a system should be both complete and consistent. Completeness means that all services required by the user should be defined. Consistency means that requirements should not have contradictory definitions.

Non-functional requirements p87

These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards. Non-functional requirements often apply to the system as a whole, rather than individual system features or services.

Non-functional requirements are requirements that are not directly concerned with the specific services delivered by the system to its users. They may relate to emergent system properties such as reliability, response time, and store occupancy. Or they may define constraints on the system implementation. Non-functional requirements, such as performance, security, or availability, usually specify or constrain characteristics of the system as a whole. Non-functional requirements are often more critical than individual functional requirements.

why implementation of these requirements may be diffused throughout the system?

1. non-functional requirements may affect the overall architecture of a system rather than the individual components.
2. A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define new system services that are required. In addition, it may also generate requirements that restrict existing requirements.

Where do all these non-function requirements stem from? 这些非功能性需求从哪来？

through user needs, organizational policies, or external factors

P88

·in reality, the distinction between different types of requirement is not as clear-cut as these simple definitions suggest. Requirements are not independent, and that one requirement often generates or constrains other requirements.

1. 如何具体化产品线？4大类具体化的举动

How to specialize software product line? 在面向对象的软件工程开发，产品线是公用的，对于某一个具体的产品要把产品线进行具体化，有四大类的举动，分别是 平台具体化，环境具体化，功能具体化，过程具体化

(reference book 434)

software product line: One of the most effective approaches to reuse is to create software product lines or application families. A software product line is a set of applications with a common architecture and shared components, with each application specialized to reflect different requirements. The core system is designed to be configured and adapted to suit the needs of different system customers. This may involve the configuration of some components, implementing additional components, and modifying some of the components to reflect new requirements.

P436

Various types of specialization of a software product line may be developed:

1. Platform specialization Versions of the application are developed for different platforms. For example, versions of the application may exist for Windows, Mac OS, and Linux platforms. In this case, the functionality of the application is normally unchanged; only those components that interface with the hardware and operating system are modified.
2. Environment specialization Versions of the application are created to handle particular operating environments and peripheral devices. For example, a system for the emergency services may exist in different versions, depending on the vehicle communications system. In this case, they system components are changed to reflect the functionality of the communications equipment used.
3. Functional specialization Versions of the application are created for specific customers who have different requirements. For example, a library automation system may be modified depending on whether it is used in a public library, a reference library, or a university library. In this case, components that implement functionality may be modified and new components added to the system.
4. Process specialization They system is adapted to cope with specific business processes. For example, an ordering system may be adapted to cope with a centralized ordering process in one company and a distributed process in another.
5. 设计模板是指？

What is a design pattern?

reference book p189

The pattern is a description of the problem and the essence of its solution, so that the solution may be reused in different settings. The pattern is not a detailed specification. Rather, you can think of it as a description of accumulated wisdom and experience, a well-tried solution to a common problem.

Patterns are a way of reusing the knowledge and experience of other designers. there are four essential elements of design patterns.

1. a name that is a meaningful reference to the pattern.
2. a description of the problem area that explains when the pattern may be applied.
3. a solution description of the parts of the design solution, their relationships, and their responsibilities. This is not a concrete design description. It is a template for a design solution that can be instantiated in different ways. This is often expressed graphically and shows the relationships between the objects and object classes in the solution.
4. a statement of the consequences-the results and trade-offs of applying the pattern. This can help designers understand whether or not a pattern can be used in a particular situation.

Some benefits of Pattern design

Patterns support high-level, concept reuse. using patterns means that you reuse the ideas but can adapt the implementation to suit the system that you are developing. Using patterns in a design process often involves developing a design, experiencing a problem, and then recognizing that a pattern can be used.

downsides

pattern design is a great idea but you need experience of software design to use them effectively. You have to recognize situations where a pattern can be applied.

1. 增量式开发 Incremental development

reference book 32

Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed. Specification, development, and validation activities are interleaved rather than separate, with rapid feedback across activities.

Incremental development reflects they way that we solve problems, which is a fundamental part of agile approaches.

Each increment or version of the system incorporates(使实体化) some of the functionality that is needed by the customer. Generally, the early increments of the system include the most important or most urgently required functionality. This means that the customer can evaluate the system at a relatively early stage in the development to see if it delivers what is required. If not, the only the current increment has to be changed and, possibly, new functionality defined for later increments.

Incremental development has three important benefits, compared to the waterfall model:

1. The cost of accommodating changing customer requirements is reduced. The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
2. It is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the software and see how much has been implemented. Customers find it difficult to judge progress from software design documents.
3. More rapid delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development in some form is now the most common approach for the development of application systems. this approach can be either plan-driven, agile, or, more usually, a mixture of these approaches. In a plan-driven approach, the system increments are identified in advance; if an agile approach is adopted, the early increments are identified but the development of later increments depends on progress and customer priorities.

From a management perspective, the incremental approach has two problems:

1. The process is not visible. Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
2. system structure tends to degrade as new increments are added. Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

The problems of incremental development become particularly acute for large complex, long-lifetime systems, where different teams develop different parts of the system. Large systems need a stable framework or architecture and the responsibilities of the different teams working on parts of the system need to be clearly defined with respect to that architecture. This has to be planned in advance rather than developed incrementally.

1. distributed Object Architecture 分布式对象体系结构

(在书上找到的 是 distributed component architectures，老师上课提到的是object，不太清楚这两个的区别，可以按照 component去理解)

（reference Book P495）

what is a distributed object Architecture?

a more general approach to distributed system design is to design the system as a set of services, without attempting to allocate these services to layers in the system. Each service, or group of related services, is implemented using a separate component. In a distributed component architecture the system is organized as a set of interacting components or objects. These components provide an interface to a set of services that they provide.

The benefits of using a distributed component model for implementing distributed systems are the following:

1. it allows the system designer to delay decisions on where and how services should be provided. Service-providing components may execute on any node of the network. There is no need to decide in advance whether a service is part of a data management layer, an application layer, etc.
2. It is a very open system architecture that allows new resources to be added as required. New system services can be added easily without major disruption to the existing system.
3. The system is flexible and scalable. New components or replicated components can be added as the load on the system increases, without disrupting other parts of the system.
4. It is possible to reconfigure the system dynamically with components migrating across the network as required. This may be important where there are fluctuating patterns of demand on services. A service-providing component can migrate to the same processor as service-requesting objects, thus improving the performance f the system.

A distributed component architecture can be used as a logical model that allows you to structure and organize the system. In this case, you think about how to provide application functionality solely in terms of services and combinations of services. You then work out how to provide these services using a set of distributed components.

a distributed component architecture can be used as a logical model that allows you to structure and organize the system.

distributed component architectures suffer from two major disadvantages: P498

1. they are more complex to design than client-server systems. Multi-layer client-server systems appear to be a fairly intuitive way to think about systems. They reflect many human transactions where people request and receive services from other people who specialize in providing these services. By contrast, distributed component architectures are more difficult for people to visualize and understand.
2. standardized middleware for distributed component systems has never been accepted by the community. Rather different vendors, such as Microsoft and Sun, have developed different, incompatible middleware. This middleware is complex and reliance on it increases the overall complexity of distributed component systems.
3. what is a checklist? checklist 是什么

reference book P666

During an inspection, a checklist of common programming errors is often used to focus the search for bugs. This checklist may be based on examples from books or from knowledge of defects that are common in a particular application domain. Each organization should develop its own inspection checklist based on local standards and practices. These checklists should be regularly updated, as new types of defects are found. The items in the checklist vary according to programming language because of the different levels of checking that are possible at compile-time.

checklist example

|  |  |
| --- | --- |
| Faults class | Inspection check |
| Data faults | Are all program variables initialized before their values are used? |
| Control faults | For each conditional statement, is the condition correct? |
| Input/output faults | Are all input variables used? |
| Interface faults | Do all function and method calls have the correct number of parameters? |
| Storage management faults | If a linked structure is modified, have all links been correctly reassigned? |
| Exception management faults | Have all possible error conditions been taken into account? |

1. UML sequence diagram UML 的时序图

reference book P126

Sequence diagrams in the UML are primarily used to model the interactions between the actors and the objects in a system and the interactions between the objects themselves. The UML has a rich syntax for sequence diagrams, which allows many different kinds of interaction to be modeled.

A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.

如何理解时序图 读图 P127 和 P128

The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these. Interactions between objects are indicated by annotated arrows. The rectangle on the dotted lines indicates the lifeline of the object concerned. You read the sequence of interactions form top to bottom. The annotations on the arrows indicate the calls to the objects, their parameters, and the return values.

图表上层是所有的object或actors，竖着的方块表示的是对应物体的声明周期（也就是存在的时间），箭头指示出objects之间的交互，并且有对应的参数或是返回参数，表示的是物体之间的调用calls。alt 方块表示不同返回值对应不同的情况（a box named alt is used with the conditions indicated in square brackets.方括号中表明条件）

对照127中的图表看和理解

1. Middleware services 中间件的一些服务

reference book P 487

why do we need middleware services? 我们为什么需要中间件

The components in a distributed system may be implemented in different programming languages and may execute on completely different types of processor. Models of data, information representation, and protocols for communication may all be different. A distributed system therefore requires software that can mange these diverse parts, and ensure that they can communicate and exchange data.

what is middleware?

The term ‘middleware’ is used to refer to this software, it sits in the middle between the distributed components of the system. Middleware is general-purpose software that is usually bought off the shelf rather than written specially by application developers. Examples of middleware include software for managing communications with databases, transaction managers, data converters, and communication controllers.

In a distributed system, middleware normally provides two distinct types of support:

1. Interaction support, where the middleware coordinates interactions between different components in the system. The middleware provides location transparency in that it isn’t necessary for components to know the physical locations of other components. It may also support parameter conversion if different programming languages are used to implement components, event detection, and communication, etc.
2. The provision of common services, where the middleware provides reusable implementations of services that may be required by several components in the distributed system. By using these common services, components can easily interoperate and provide user services in a consistent way.

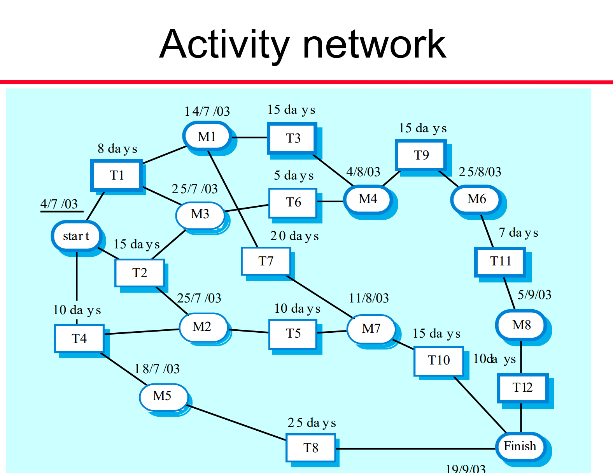
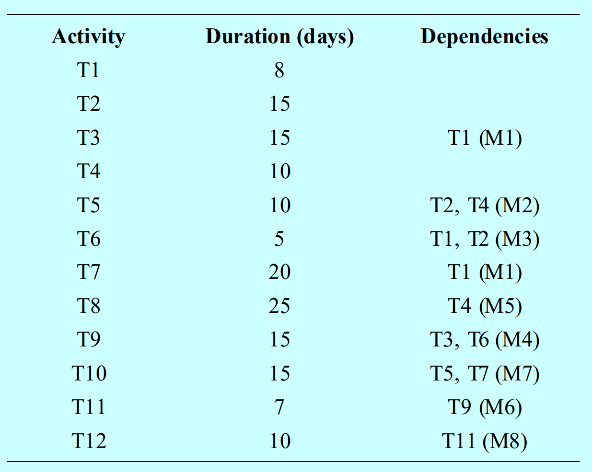
Common services are those services that may be required by different components irrespective of the functionality of these components. These may include security services (authentication and authorization), notification and naming services, and transaction management services, etc. These common services can be viewed as being provided by a middleware container. you then deploy your component in that container and it can access and use these common services.

1. 模型作图题

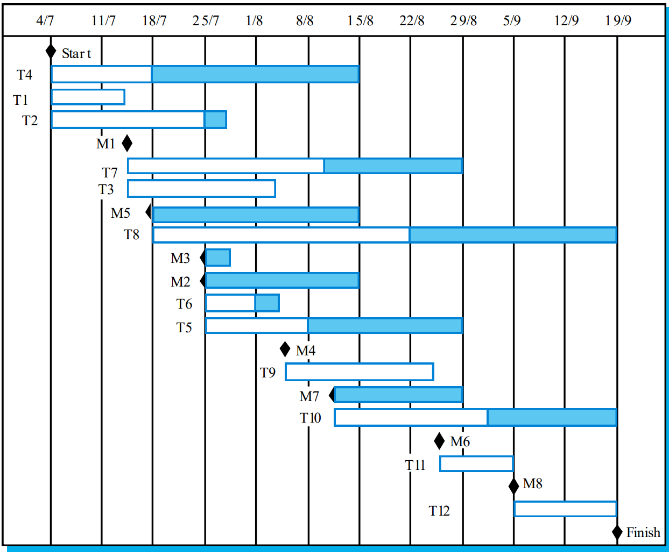
状态转化模型 state transition diagram BOOK P 188 545

标清楚各个状态（对应的名字和表达的含义），类似数电的有限状态机，标清各部分的转移，很好理解。 P136 微波炉的例子，P136是图，P137是对应的图例

活动网络图 activity network

在项目管理中的，列出每一个事件的持续时间和依赖（先行事件），然后可以精准的绘制网络图，然后计算总运行天数。依赖表示的是如果执行该事件，那么对应的几个事件必须先完成，对应的这些应该先完成的事件构成了一个个milestone， 用M来表示

P630 activity bar chart

Gantt chart 横道图，条状图，Bar chart （都是一个东西） 以图示的方式通过活动列表和时间刻度形象地表示出任何特定项目的活动顺序和持续事件（如下，表格和上面一样）

时序模型 sequence diagram P127 P128 P186 详见三-7，理解时序模型是怎么画的，含义是什么。就可以了

底下两个不重要

~~数据驱动模型 data-driven modeling P135 (and activity model~~

~~事件驱动模型 event-driven modeling P136~~

1. 问答题
2. middleware 的服务有哪些 （详见三-8 这一部分基本把书都抄下来了）
3. change management 变更管理 P685

Change is a fact of life for large software systems. Organizational needs and requirements change during the lifetime of a system, bugs have to be repaired, and systems have to adapt to changes in their environment. To ensure that the changes are applied to the system in a controlled way, you need a set of tool-supported, change management processes.

what does change management do? Change management is intended to ensure that the evolution of the system is a managed process and that priority is given to the most urgent and cost-effective changes, approving those changes that are worthwhile, and tracking which components in the system have been changed.

change management process: always have a means of checking, costing, and approving changes. This process should come into effect when the software is handed over for release to customers or for deployment within an organization.

process:

1. submit change requests
2. check to ensure it is valid
3. for valid changes, the next stage of the process is change assessment and costing.
4. decide if it is cost-effective form a business perspective to make the change to the software (P687 CCB or product development group 有几个factors)

变更管理是软件工程的灵魂部分，这部分大部分是CASE工具帮助实现的，可以认为这些工具是CASE工具中的一部分

change management is usually supported by specialized software tools. These may be relatively simple web-based tools such as Bugzilla, which is used to report problems with many open source systems. Alternatively, more complex tools may be used to automate the entire process of handling change requests from initial customer proposal to change approval.

工具有哪几种关键的特性，或是有哪些功能 CASE tools 有5个功能 P37页上

1. reuse benefits & problems

P427

P428 正好对应两个表格

一-15 对应的答案中已经给出 见上

1. OO Design 的一些关键活动 P178 五个关键活动

An object-oriented system is made up of interacting objects that maintain their own local state and provide operations on that state.

OO design processes involve designing object classes and the relationships between these classes. These classes define the objects in the system and their interactions.

To develop a system design from concept to detailed, object-oriented design, there are several things that you need to do

1. understand and define the context and the external interactions with the system
2. design the system architecture
3. identify the principal objects in the system
4. develop design models
5. specify interfaces

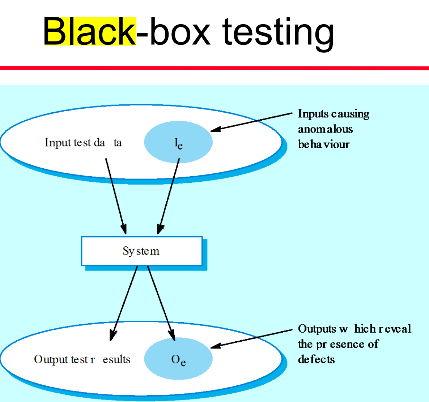
like all creative activities, design is not a clear-cut, sequential process. you develop a design by getting ideas, proposing solutions, and refining these solutions as information becomes available. you inevitably have to backtrack and retry when problems arise.

（答题的时候不能只写这五个活动，老师明确强调要展开写，虽然找不到其他更多的扩展内容了）

1. reengineering 的活动 P249 中间段落

与一-11 问题完全是重合的（见上）

1. black box testing and equivalence testing



P216 guideline

choose inputs that force the system to generate all error messages;

design inputs that cause input buffers to overflow

repeat the same input or series of inputs numerous times

force invalid outputs to be generated

force computation results to be too large or too small

equivalence testing

input data and output results often fall into different classes where all members of a class are related. Each of these classes is an equivalence partition or domain where the program behaves in an equivalent way for each class member.

如何用等价类划分的方法取构造测试用例Test cases should be chosen from each partition.

search routine – input partitions

1. inputs which conform to the pre-conditions
2. inputs where a pre-condition does not hold 假设条件不成立
3. inputs where the key element is a member of the array
4. inputs where the key element is not a member of the array

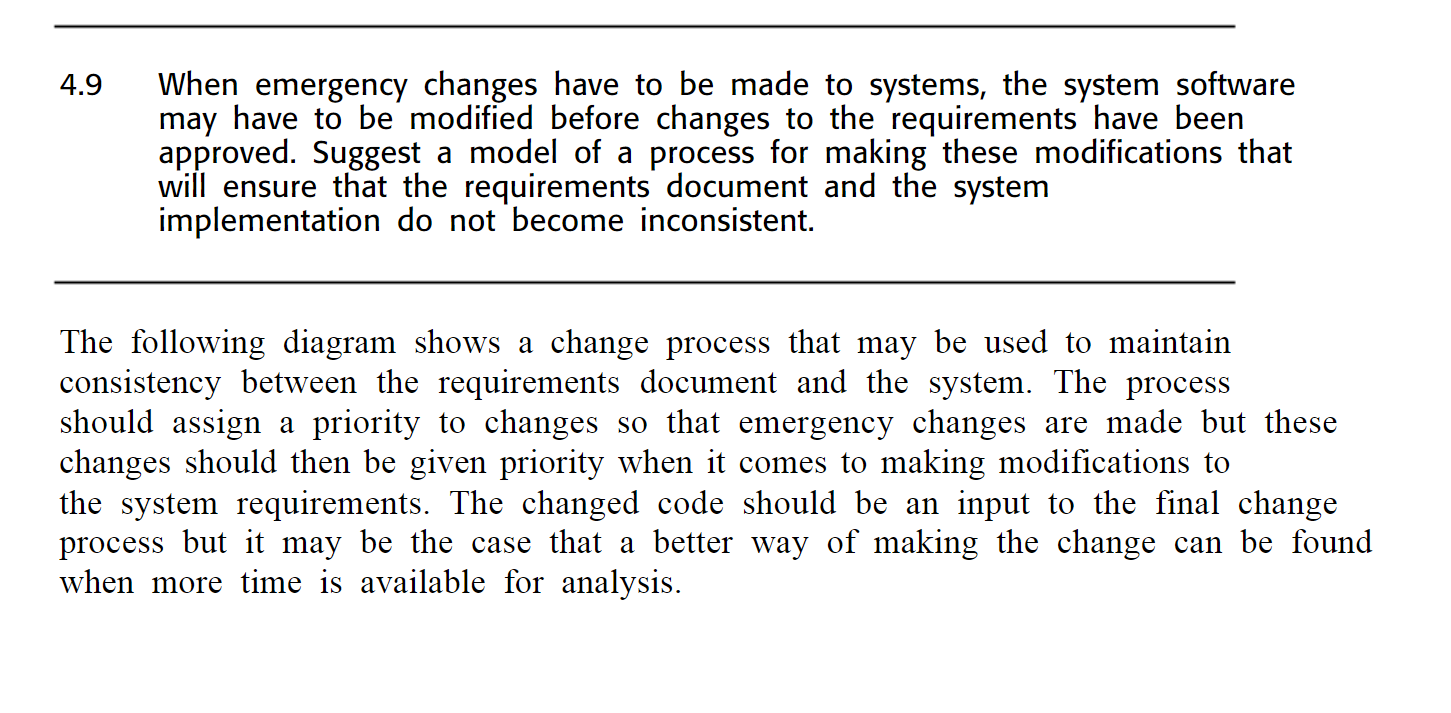
test software with sequences which have only a single value. Use sequences of different sizes in different tests. Derive tests so that the first, middle and last elements of the sequence are accessed. Test with sequences of zero length.

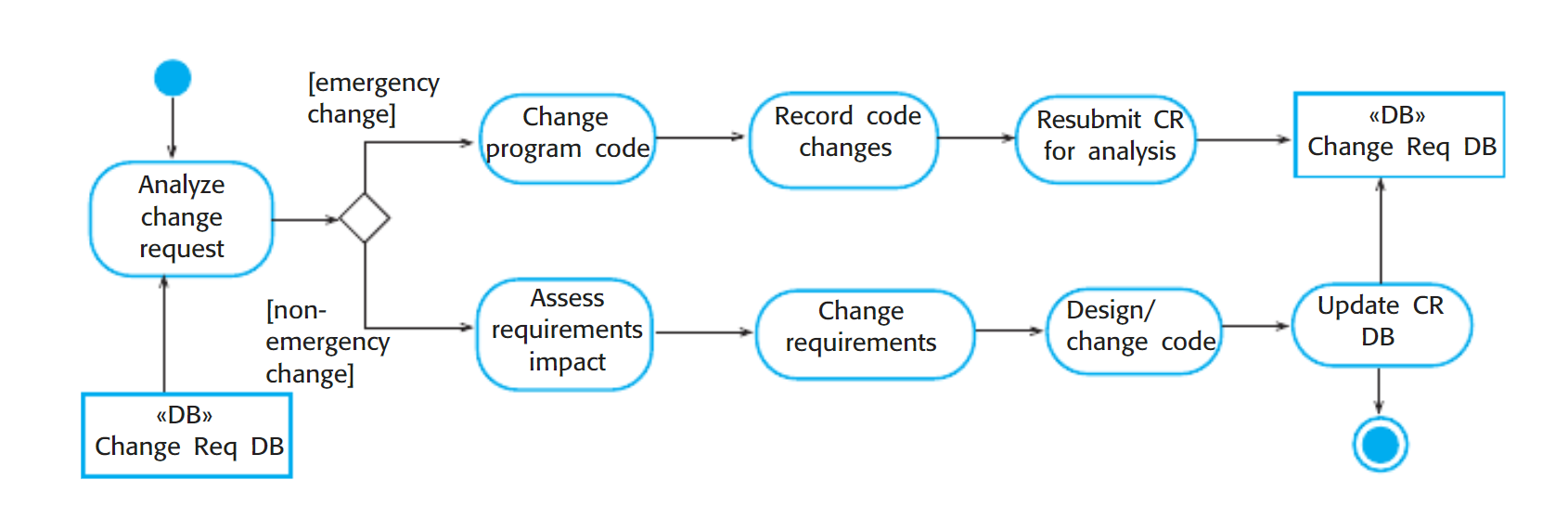
P213 possible strategies

1. partition testing where you identify groups of inputs that have common characteristics and should be processed in the same way. You should choose tests from within each of these groups
2. Guideline-based testing where you use testing guidelines to choose test cases. These guidelines reflect previous experience of the kinds of errors that programmers often make when developing components.
3. 当系统中出现紧急变更的时候，如果没有时间去进行change management 中的各个步骤，请设计一个过程模型，能够确保这个发生的变更不会让系统出现不一致性。

就是change在未被认可之前就已经修改了！

原问题：when emergency changes have to be made to the system, the system software may have to be modified before changes to the requirements have been approved. suggest a process for making these modifications that ensures that the requirement documents and system implementation do not become inconsistent.





1. 额外
2. independent parts

PPT

the objective of path testing is to ensure that the set of test cases is such that each path through the program is executed at least once. the starting point for path testing is a program flow graph that shows nodes representing program decisions and arcs representing the flow of the control. Statements with conditions are therefore nodes in the flow graph.

test cases should be derived so that all of these paths are executed, a dynamic program analyzer may be used to check that paths have been executed.