2013—2014 现代软件工程考试要点

题型和考点

单项选择题(Choose the best answer)(20 分)

1 Software crisis(软件危机)

a set of problems present in developing software 开发软件中存在一系列问题
Over budget(超过预算)
Over schedule: delivered late(延迟交付)
Deficient in the stated requirements（不满足需求）
（Reasons for crisis（产生软件危机的原因）
1、complexity is inherent(essential) property 复杂性是固有的(基本的)属性
2、Intangible (creative effort) 无形的创造性（由于过分地依靠程序设计人员在软件开发过程中的技巧和创造性，加剧软件开发产品的个性化，也是发生软件开发危机的一个重要原因。）
3、Change 需求变更
4、No effective technology and management approach 没有有效的技术和管理方式）

2 iterative process（迭代过程）

A process for arriving at a decision or a desired result by repeating rounds of analysis or a cycle of operations. The objective is to bring the desired decision or result closer to discovery with each repetition (iteration). The iterative process can be used where the decision is not easily revocable (such as a marriage or war) or where the consequences of revocation could be costly.
一个为了得到决定或预期的结果而重复分析或操作的循环过程。每次迭代的目标就是得到所需的决策或与发现的结果更近似。迭代过程的决定不容易可撤销(如婚姻或战争)或者撤销的后果可能是昂贵的。
Iterative process models describe the software process as a cycle of activities(迭代过程模型描述软件过程作为一个周期的活动)
Specification, development and validation are interleaved.
（规格说明、开发及验证活动是迭代进行的）
Process activities are interleaved rather than separate, with rapid feedback across activities.（过程活动不是分开进行的，而是各项活动之间有快速的信息反馈）

3 The linear sequential model of software development（软件开发的线性顺序模型）瀑布模型

软件工程的线性顺序模型，有时也称“传统生命周期”或“瀑布模型”，线性顺序模型提出了软件开发的系统化的、顺序的方法（虽然由 Winston Royce［ROY70］提出的最早的瀑布模型支持带有反馈的循环，但大多数使用该过程模型的组织均把它视为是严格线性的），从系统级开始，随后是分析、设计、编码、测试和维护。借鉴了传统的工程周期，线性顺序模型包含以下活动：1.系统/信息工程和建模；2.软件需求分析

The linear sequential model of software development is Answer：a

a. A reasonable approach when requirements are well defined.

b. A good approach when a working program is required quickly.

c. The best approach to use for projects with large development teams.

d. An old fashioned model that cannot be used in a modern context.

4 How to manage complexity（如何管理复杂性）

降低复杂性：尽可能简化结构，而不用改变解决方案的本质

5 formal methods（形式化方法）

形式化方法定义为：“用于开发计算机系统的形式化方法是基于数学的用于描述系统性质的技术。这样的形式化方法提供了一个框架，人们可以在该框架中以系统的方式刻画、开发和验证系统”。

在软件开发的全过程中，凡是采用严格的数学语言，具有精确的数学语义的方法，都称为形式化方法。从广义角度，形式化方法是软件开发过程中分析、设计及实现的系统工程方法。狭义地，形式化方法是软件规格（specification）和验证（verification）的方法。

Formal specification is part of a more general collection of techniques that are known as ‘formal methods’. 正式规范是被称为“正规方法”的技术更一般的集合的一部分。

These are all based on mathematical representation and analysis of software. 这些都是基于数学表示及软件分析。

Formal methods include：

Formal specification; 正式的规范

Specification analysis and proof 规范分析和论证;

Transformational development 转型发展;

Program verification 程序验证.

Acceptance of formal methods 形式化方法的验收

Formal methods have not become mainstream software development techniques as was once predicted

Other software engineering techniques have been successful at increasing system quality. Hence the need for formal methods has been reduced; 形式化方法还没有成为主流的软件开发技术，如曾经预言其他的软件工程技术已成功地提高了系统的质量。因此，需要形式化方法已经减少

Market changes have made time-to-market rather than software with a low error count the key factor. Formal methods do not reduce time to market; 市场变化使软件上市的时间比低错误计算的软件更重要。形式化方法不缩短产品上市时间;

The scope of formal methods is limited. They are not well-suited to specifying and analysing user interfaces and user interaction 形式化方法的范围是有限的。它们并不适合于指定并分析用户界面和用户交互;

Formal methods are still hard to scale up to large systems. 形式化方法仍然很难扩展到大型系统

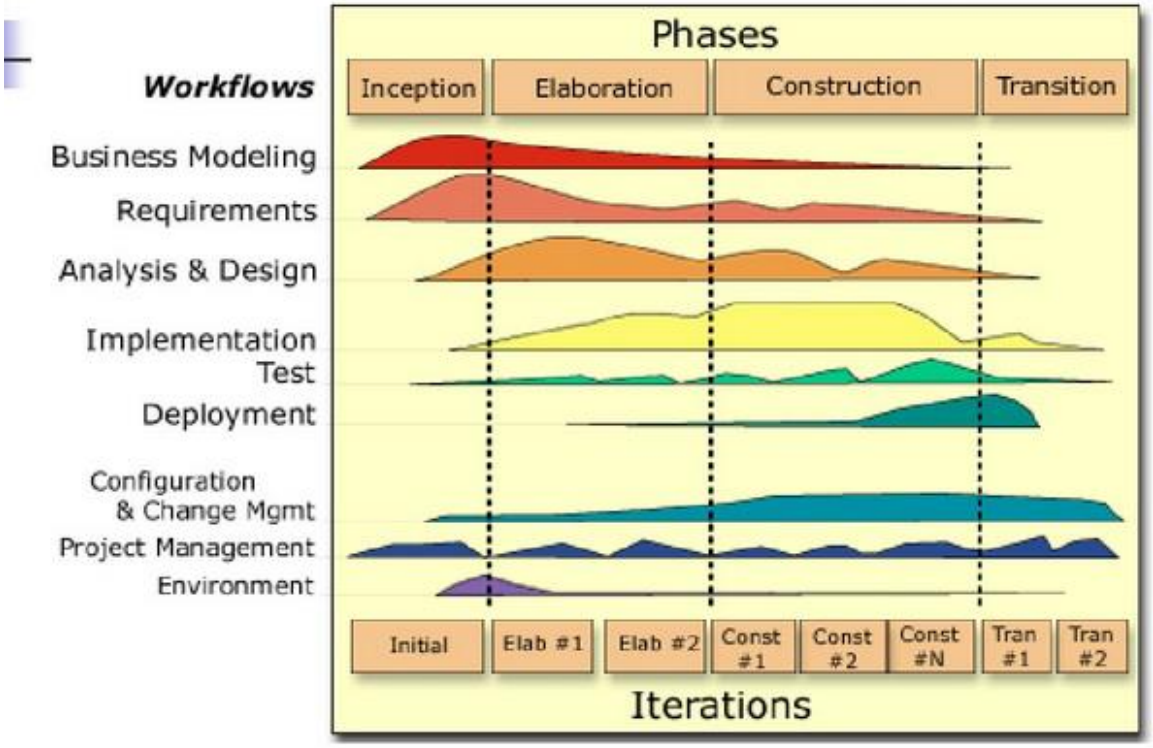
Use of formal methods

The principal benefits of formal methods are in reducing the number of faults in systems 形式化方法的主要好处是减少故障的系统的数量.

Consequently, their main area of applicability is in critical systems engineering. There have been several successful projects where formal methods have been used in this area 因此，适用性他们的主要领域是至关重要的系统工程。已经有地方形式化方法已经在这一领域已经使用了几个成功的项目

In this area, the use of formal methods is most likely to be cost-effective because high system failure costs must be avoided. 在这个领域，利用形式化方法是最有可能符合成本效益，因为较高的系统故障成本必须加以避免。

6 phase of Unified Process（阶段的统一过程）



Inception: focus on feasibility 成立：注重可行性

Establish the business case for the system.为系统建立业务案例。

Elaboration: focus on requirements 详细说明：重点要求

Develop an understanding of the problem domain and the system architecture.发展问题域和系统体系结构的理解。

Construction: focus on design 建设：注重设计

System design, programming and testing. 系统设计，编程和测试。

Transition 过渡

Deploy the system in its operating environment.在其操作环境中部署系统。

Which of these is not one of the phase names defined by the Unified Process model for software development? Answer：d

a. Inception phase

b. Elaboration phase

c. Construction phase

d. Validation phase

统一过程主要分五个阶段：起始阶段，细化阶段，构建阶段，转化阶段，生产阶段。

7 agile software processes（敏捷软件过程）

软件过程={活动，前置条件，后置条件，产品，资源，地点，过程目标和度量指标}

敏捷性度量指标：时间，生产率，健壮性，自适应和改善能力。

敏捷软件过程模型的结构：

敏捷方法普遍地依赖于迭代方法来完成软件描述、开发和移交，主要用于支持业务应用的开发，这里的系统需求总是在开发过程中快速的变化着。软件开发人员打算用它们来迅速完成和移交可用软件给客户，客户提出新的变化了的需求，由软件开发人员在下一个循环中实现。

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple 'software increments'
- Adapts as changes occur

How do you create agile processes to manage unpredictability?

a. Requirements gathering must be conducted very carefully

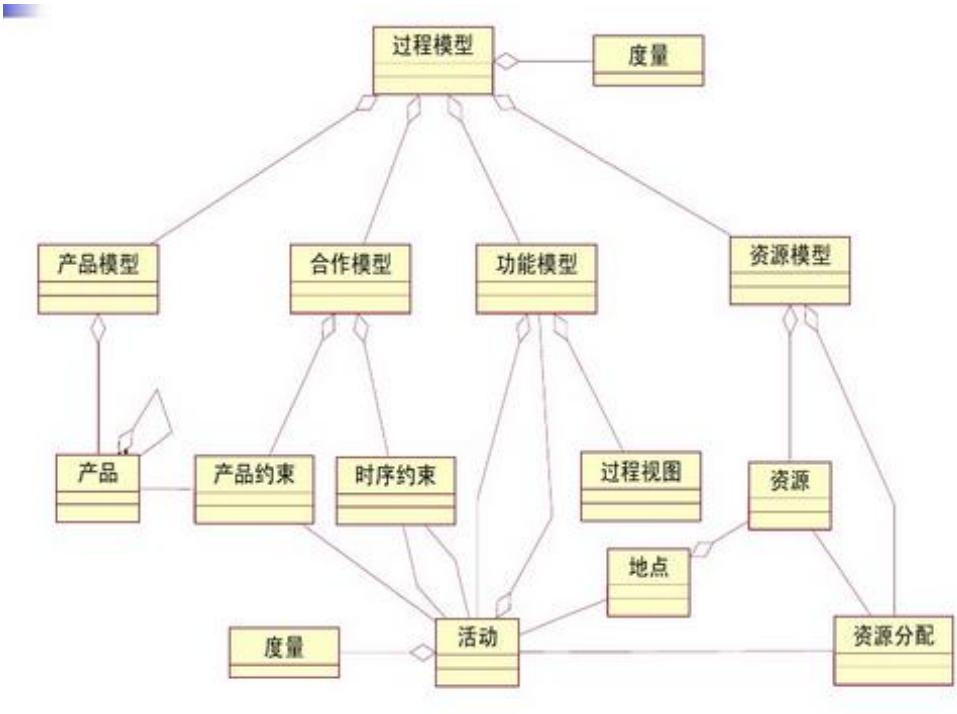
b. Risk analysis must be conducted before planning takes place

c. Software increments must be delivered in short time periods

d. Software processes must adapt to changes incrementally

e. Both c and d

Answer: e



8 requirements elicitation difficulty（需求获取困难）

1.需求获取范围大（人太多、类型太多、分布太广）

2.需求方表达能力有限（说不清楚、想不明白、各种信息噪音混淆在一起）

3.需求获取方能力（因为前两点，所以发展出的方法很多，要熟练掌握和运用不容易）

9 conflicting requirements（相互冲突的需求）

不同的涉众，要优先考虑最大的利益相关者

Different stakeholders have different viewpoints:

- Viewpoints are a way of structuring the requirements to represent the perspectives of different stakeholders. Stakeholders may be classified under different viewpoints.

- This multi-perspective analysis is important as there is no single correct way to analyse system requirements.

10 objective for building an analysis mode（构建一个分析模式的目的）

The purpose of analysis is to understand the problem so that a correct design can be constructed.

A good analysis captures the essential features of the problem without introducing implementation artifacts that prematurely restrict design decisions.

Which of the following is not an objective for building an analysis model? Answer: c

a. define set of software requirements that can be validated
b. describe customer requirements
c. develop an abbreviated solution for the problem
d. establish basis for software design

The analysis model is an abstraction, or generalization, of the design. It omits most of the details of how the system works and provides an overview of the system's functionality.
分析模型是一种抽象，或概括，设计的。它省略了大部分的系统是如何工作的细节，并提供了系统的功能的概述。

11 importance of software design（软件设计的重要性）

Design is the place where software quality is established.设计了软件的质量建立的地方。

Without design, we risk building an unstable, difficult test and maintainable system. 如果没有设计，我们就有可能建立一个不稳定的，难以测试和维护的系统。

Design sits at the technical kernel of SE and is applied regardless of the software process model that is used.设计位于 SE 的技术内核中心，不管所使用的软件过程模型如何都被应用。

Design is the place where creativity rules.设计是地方创造力的规则。

12 Cohesion（内聚）

内聚度是指构成构件的内部“粘合”程度。一个构件的内聚度越高，构件内部的各部分相互之间的相互联系就越紧密，与总体的目标就越相关。换而言之，如果一个构件所有的元素都是直接面向执行同一个任务的并且是必须的，那么该构件就是内聚的。

Is the measure of the strength of functional relatedness of elements within a module

1. Cohesion is a qualitative indication of the degree to which a module Answer: b
   a. can be written more compactly.
   b. focuses on just one thing.
   c. is able to complete its function in a timely manner.
   d. is connected to other modules and the outside world.

13 Architecture, Frameworks and design patterns（体系结构、框架和设计模式）

体系结构：系统的一个或者多个结构。结构中包括软件的构件，构件的外部可见属性以及他们之间的相互关系。外部可见属性则是指软件构件提供的服务，性能，使用特性，错误处理，共享资源使用等。
框架：特定应用领域问题的体系结构模式，定义了软件系统的元素和关系，创建了基本的模块，定义了涉及功能更

改和扩充的插件位置。包括基本构成元素和关系。

设计模式：为软件系统的子系统，构件或者构件之间的关系提供一个精炼之后的解决方案，描述了在特定环境下，用于解决通用软件设计问题的构件以及这些构件相互通信时的各种结构。

What is software architecture design?

An early stage of the system design process.

The design process for identifying the sub-systems making up a system and the framework for sub-system control and communication is architectural design.

The output of this design process is the software architecture.

What is software architecture?

Software architecture has emerged as an important sub-discipline of software engineering, particularly in the realm of large-system development.

It utilizes “divide and conquer” strategy.

It allows groups of to work cooperatively and productively together to solve a much larger problem than any of them would be capable of individually.

- Design Pattern:
  - A small set of classes that provide a template solution to a recurring design problem
  - Reusable design knowledge on a higher level than data structures (link lists, binary trees, etc)
- Framework:
  - A moderately large set of classes that collaborate to carry out a set of responsibilities in an application domain.
- Examples: User Interface Builder, MFC…

1Why Architecture ?

- Stakeholder communication
  - Architecture may be used as a focus of discussion by system stakeholders.
- System analysis
  - Means that analysis of whether the system can meet its non-functional requirements is possible.
- Large-scale reuse
  - The architecture may be reusable across a range of systems.

14 Actor（执行者）

业务执行者的定义：在组织之外和组织交互的人群或组织。业务执行者是一个组织（或人群），而不是系统。

系统执行者的定义：在所研究系统外，与该系统发生功能性交互的其他系统。要点：1.系统外，系统执行者不是所研究系统的一部分，是该系统边界外的一个系统。2.交互，系统执行者必须和系统有交互，不和系统交互的不算是系统的执行者。3.功能性交互，执行者和系统发生的交互是系统的功能需求。4.系统，系统可以是一个人肉系统，也可以是一个非人智能系统，甚至是一个特别的外系统—时间。

Use cases are always started by an actor

Use cases are always written from the point of view of an actor.

Use cases provide a means of expressing the problem in a way that is understandable to a wide range of stakeholders: users, developers, and customers business that the actors can do with the system

15 Stakeholder（涉众）

与系统有关系的所有人

end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc.

Stakeholder is any entity (not necessarily human) who either affects the system development

16 Software test（软件测试）

1. What is the normal order of activities in which traditional software testing is organized?Answer: c
    a. integration testing, unit testing, system testing, validation testing
    b. validation testing, unit testing, integration testing, system testing
    c. unit testing, integration testing, validation testing, system testing
    d. system testing, validation testing, integration testing, unit testing
2. Which of the following strategic issues needs to be addressed in a successful software testing process? Answer: e
    a. conduct formal technical reviews prior to testing
    b. specify requirements in a quantifiable manner
    c. use independent test teams
    d. wait till code is written prior to writing the test plan
    e. answers a and b

System testing ,Test case design ,Test automation, Component testing

Component testing 组件测试

Testing of individual program components;

Usually the responsibility of the component developer (except sometimes for critical systems) 通常情况下，组件开发人员（除了有时为关键系统）的责任;

Tests are derived from the developer's experience 测试来自开发者的经验.

System testing 系统测试

Testing of groups of components integrated to create a system or sub-system 组件的集成，创建一个系统或子系统组的检验;

The responsibility of an independent testing team; 一个独立的测试团队的责任

Tests are based on a system specification. 测试是基于系统规范

Defect testing 缺陷检测

; The goal of defect testing is to discover defects in programs

A successful defect test is a test which causes a program to behave in an anomalous way

Tests show the presence not the absence of defects

缺陷检测的目标是发现在程序的缺陷

一个成功的缺陷测试是一个测试这会导致程序的行为出现异常的方法

测试表明不存在的情况下的缺陷

Testing process goals

Validation testing

To demonstrate to the developer and the system customer that the software meets its requirements;

A successful test shows that the system operates as intended.

验证测试

为了证明给开发者和系统用户，该软件满足其需求;

一个成功的测试表明，该系统按计划运行。

缺陷检测

Defect testing

To discover faults or defects in the software where its behaviour is incorrect or not in conformance with its specification;

A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.

发现在它的行为是不正确或不符合其规范的软件故障或缺陷;

一个成功的测试是一个测试，使系统工作不正常，因此暴露在系统中的缺陷。

简答题(terse but to the point answers) (40 分)

业务建模 business modeling

业务建模是以软件模型方式描述企业管理和业务所涉及的对象和要素、以及它们的属性、行为和彼此关系，业务建模强调以体系的方式来理解、设计和构架企业信息系统。

~~业务建模（Business Modeling）是以软件模型方式描述企业管理和业务所涉及的对象和要素、以及它们的属性、行为和彼此关系，业务建模强调以体系的方式来理解、设计和构架企业信息系统。~~

业务建模（Business Modeling）是一种建模方法的集合，目的是对业务进行建模。这方面的工作可能包括了对业务流程建模，对业务组织建模，改进业务流程，领域建模等方面。

软件开发的工作流：业务建模、需求、分析和设计等 (business modeling, requirements, analysis and design)

软件对现状的改进

1. 物流变成信息流：物的流转成本会随着距离的增加而明显增加，信息流则不会。尽可能把物流变成信息流，在需要物的时候，才将信息变成物。这样，流转成本会大大降低。
2. 改善信息流转：信息系统越来越多以后，一个人为了达到目的可能需要和多个信息系统打交道，各个信息系统中的信息靠人来协调。可以引进新的信息系统改进。
3. 封装领域逻辑：这些领域逻辑原来可能封装在人脑中，通过提炼后封装到信息系统中，从而使人脑得到解放。

•

用例模型 use case model

Composed of different use case diagrams, each contains:

- Actors
- Use cases
- Relationships between actors and use cases

用例模型（Use-Case Model）是系统既定功能及系统环境的模型，它可以作为客户和开发人员之间的契约。用例是贯穿整个系统开发的一条主线。同一个用例模型即为需求工作流程的结果，可当作分析设计工作流程以及测试工作流程的输入使用。构建用例模型有三个主要的原因：

使用例更易于理解。2 将在许多用例内说明的公有行为分离出来。3 使用例模型更易于维护。

主、辅助执行者 actor

主执行者主动发起用例的交互，在用例图中，箭头从执行者指向用例。辅执行者在交互的过程中被动参与进来，在用例图中，箭头是从用例指向执行者，多数情况为非人智能系统。主辅执行者是针对某个用例来说的，一个系统在这个用例充当主执行者，也可以在另一个用例充当辅执行者。这两者都是达到用例的目标所需要的。

系统用例和业务用例 use case and business use case

系统用例的定义：系统能够为执行者提供的、涉众可以接受的价值。系统类协作完成系统用例。

业务用例的定义：业务执行者希望通过和组织交互达到的，而且组织能提供的价值。业务用例是为业务执行者服务，不是为业务工人服务。或者说，业务用例表达组织的本质价值。业务工人和业务实体协作完成业务用例。

建模题(Modeling) (40 分)

给定一个熟悉的实际应用，对应用中的一个软件系统建模，要求给出 vision、business model、requirement model and analysis model.

一．Vision 想实现的目标 （办公室主任）

二．业务建模：画序列图

三．需求，选一个用例，写出用例规约（取款）

四．分析三，找出类图