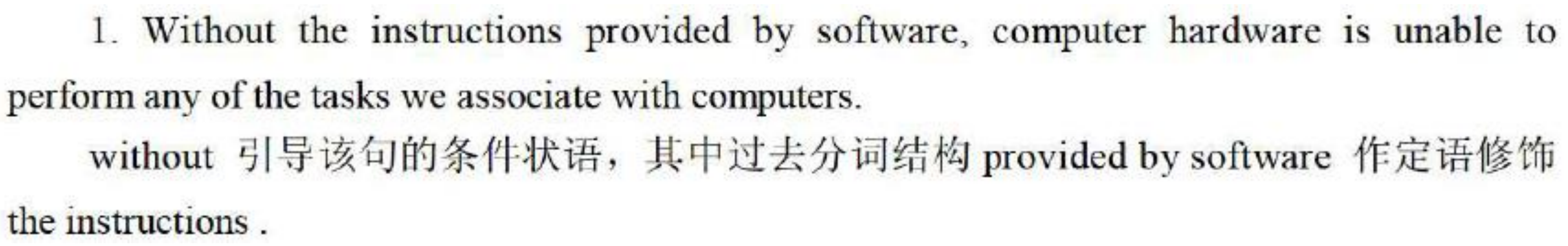
Chapter 9
Software Engineering

9.1 What is Software Engineering



1. Without the instructions provided by software, computer hardware is unable to perform any of the tasks we associate with computers.

without 引导该句的条件状语，其中过去分词结构 provided by software 作定语修饰 the instructions .

- 没有软件提供的指令，计算机硬件无法执行我们与计算机相关联的任何任务。
- That year NATO convened a workshop by that name to assess the state and prospects of software production.
- 那年北约以这个名字召开了一个研讨会，评估软件生产的现状和前景。

intermediary	中间媒介	**coin v.**	创造
aspiration	愿望	**rallying cry**	号召

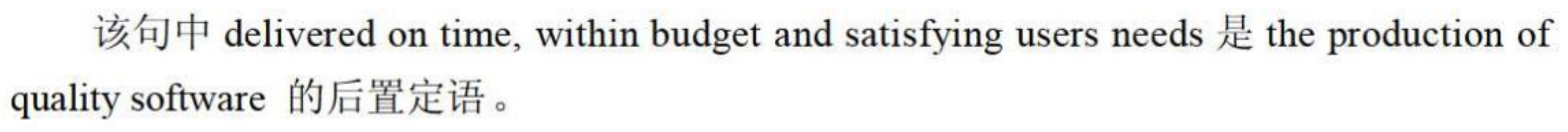
9.1 What is Software Engineering

- The resulting practice differs significantly from the practice of older forms of engineering.
- 由此产生的实践与旧形式的工程实践有很大不同。
- Software engineering is the application of a systematic, disciplined, quantifiable approach to development, operation, and maintenance of software; that is, the application of engineering to software.
- 软件工程是应用系统的、纪律性强的、可量化的方法于软件的开发、操作和维护，即工程在软件中的应用。

quantifiable 可量化的	**capture the imagination** 启发想象力
IEEE, Institute of Electrical and Electronics Engineers 电气与电子工程师学会	

9.1 What is Software Engineering

- The aim of software engineering is the production of quality software, delivered on time, within budget, and satisfying user’s needs.


该句中 delivered on time, within budget and satisfying users needs 是 the production of quality software 的后置定语。

- 软件工程的目标是生产高质量的软件，这些软件能按时交付，不超支，满足用户的需求。

quality a.	高质量的	**employ v.**	运用
paradigm	范例; 范式		

9.2 Key Issues of Software Engineering

- There are eight fundamental notions in software engineering that form the basis for an effective discipline of software engineering.
- 软件工程有八个基本概念，它们构成了一个有效的软件工程学科的基础。

daunting	令人畏惧的	tackle v.	解决

9.2 Key Issues of Software Engineering

- An abstraction is a description of the problem at some level of generalization that allows us to concentrate on the key aspects of the problem without mired in the details.

该句包含一个限制性定语从句 that allows us to …，它修饰的是 description，介词短语 without getting mired in the details 作状语表结果。

- 抽象是对问题在某种程度上的概括描述，它允许我们专注于问题的关键方面而不陷入细节的困境之中。

get mired in	陷入困境	**notion**	概念
transformation	变换		

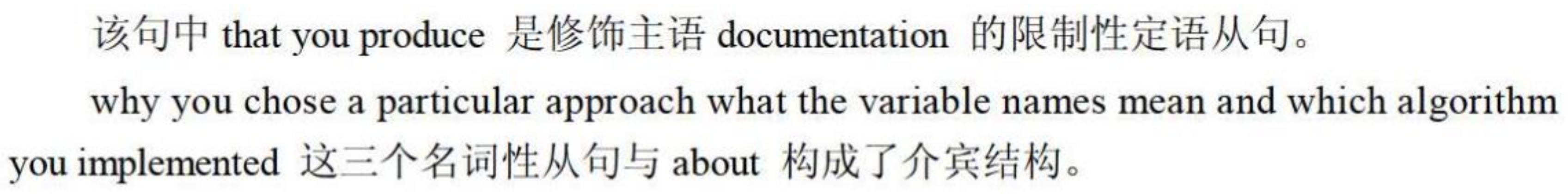
9.2 Key Issues of Software Engineering

- Typically, abstraction involves identifying classes of objects that allow us to group items together; this way, we can deal with fewer things and concentrate on the commonalities of the items in each class.
- 通常，抽象涉及到标识对象类，这使得我们可以将多个对象组合在一起；这样，我们可以处理更少的事情，并集中注意力于每个类中对象的共性。

commonality	共同特征;共性		

9.2 Key Issues of Software Engineering

- The documentation that you produce is a formal description of your notes to yourself about why you chose a particular approach, what the variable names mean, and which algorithm you implemented.



该句中 that you produce 是修饰主语 documentation 的限制性定语从句。

why you chose a particular approach what the variable names mean and which algorithm you implemented 这三个名词性从句与 about 构成了介宾结构。

- 你生成的文档是对你自己记录的正式描述，包括选择特定方法的原因、变量名的含义以及实现的算法。

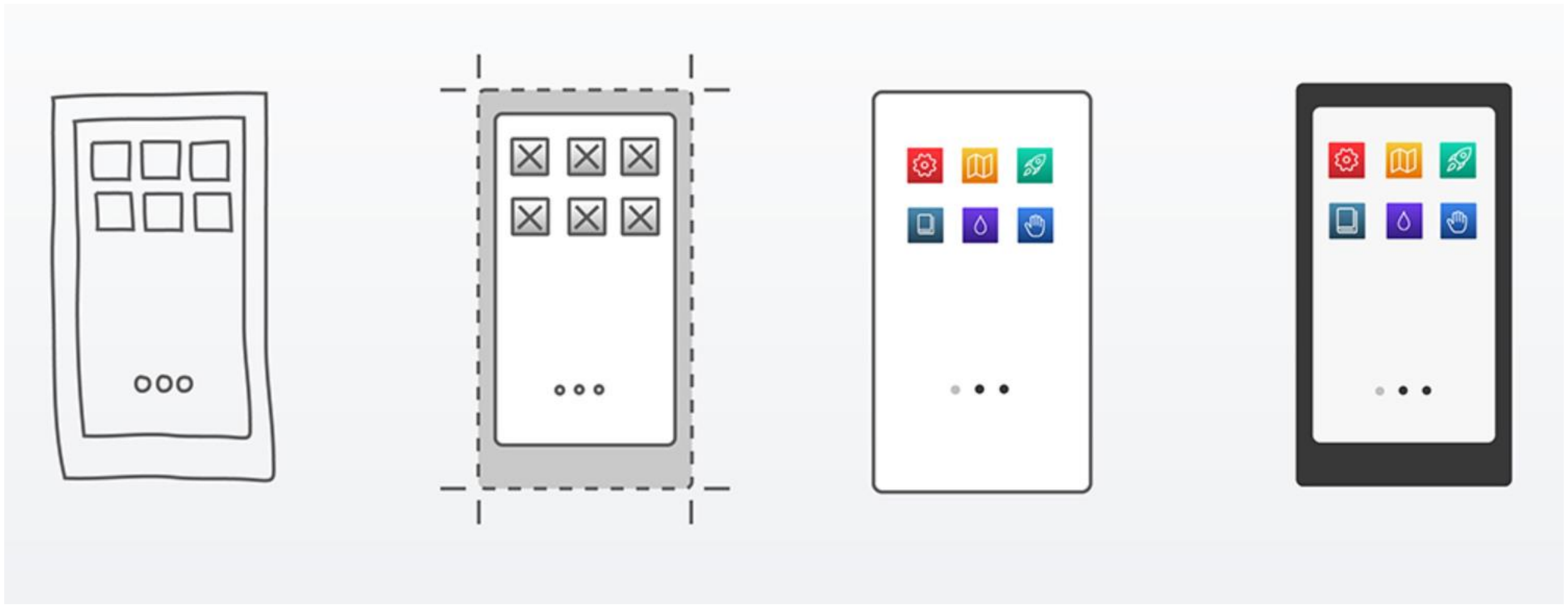
note	记录	**notation**	符号
blueprint	蓝图	**contractor**	承包商
medium	媒介	**readily**	轻而易举地

9.2 Key Issues of Software Engineering

- Different tools and techniques address different aspects of a problem, and we need to identify the modeling primitives that will allow us to capture all important aspects of a problem with a single technique.
- 不同的工具和技术处理问题的不同方面，我们需要识别建模原语，它使我们能够用一种技术采集问题的所有重要方面。

settle on	决定	**primitive n.**	原语
tailor v.	专门制作;订做		

9.2 Key Issues of Software Engineering



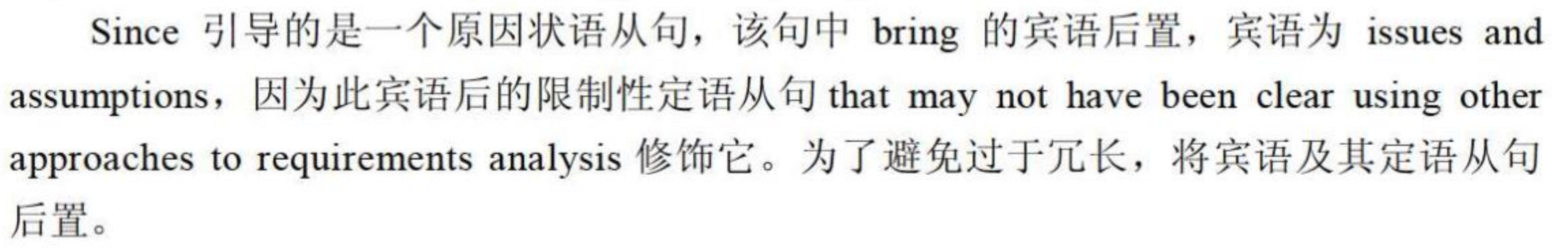
9.2 Key Issues of Software Engineering

- We build a prototype, evaluate it (with user and customer feedback), consider how changes might improve the product or design, and then build another prototype.
- 我们构建一个原型，对其进行评估（通过用户和客户的反馈），考虑如何改变来改进产品或设计，然后构建另一个原型。

prototyping	原型设计法	**iterative a.**	迭代的
feasibility	可行性	**embedded system**	嵌入式系统

9.2 Key Issues of Software Engineering

- Since the user interface is, in a sense, a bridge between the application domain and the software development team, prototyping can bring to the surface issues and assumptions that may not have been clear using other approaches to requirements analysis.



Since 引导的是一个原因状语从句，该句中 bring 的宾语后置，宾语为 issues and assumptions，因为此宾语后的限制性定语从句 that may not have been clear using other approaches to requirements analysis 修饰它。为了避免过于冗长，将宾语及其定语从句后置。

- 由于用户界面在某种意义上是应用领域和软件开发团队之间的桥梁，原型设计法可以让使用其他需求分析方法时并不清楚的问题和假设浮出水面。

bring … to the surface		

9.2 Key Issues of Software Engineering

- The overall architecture of a system is important not only to the ease of implementing and testing it, but also to the speed and effectiveness of maintaining and changing it.
- 系统的总体架构不仅对系统实现和测试的容易程度很重要，而且对维护和更改系统的速度和效率也很重要。

make or break	要么成功，要么失败		

9.2 Key Issues of Software Engineering

- A system's architecture describes the system in terms of a set of architectural units, and a map of how the units relate to one another.
- 系统的体系结构用一组体系结构单元以及这些单元之间如何相互关联的图来描述。
- The more independent the units, the more modular the architecture and the more easily we can design and develop the pieces separately.
- 单元越独立，体系结构越模块化，我们就越容易分别设计和开发各个部分。

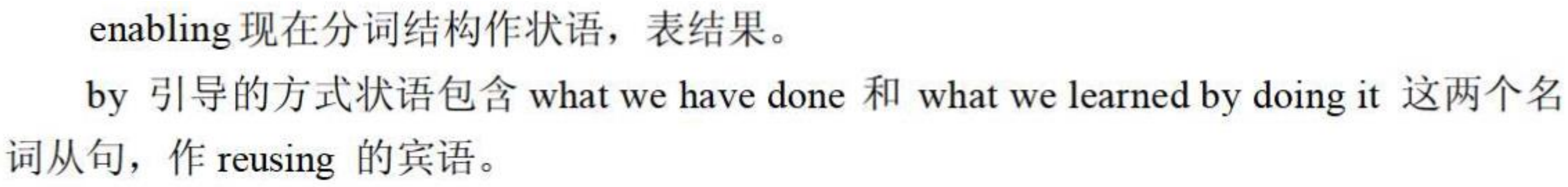
9.2 Key Issues of Software Engineering

- Outside-in design: start with the outputs of the system and investigate why they are needed and how the software can provide them.
- 从外到内设计：从系统的输出开始，研究为什么需要这些输出，以及软件如何提供这些输出。

modular	模块化的	**assign v.**	分配；分派
data-oriented	面向数据的	**event-oriented**	面向事件的
outside-in	从外到内	**interrelationship**	相互联系

9.2 Key Issues of Software Engineering

- The importance of these approaches is their capture of our design experience, enabling us to capitalize our past projects by reusing both <u>what</u> we have done and <u>what</u> we learned by doing it.



enabling 现在分词结构作状语，表结果。

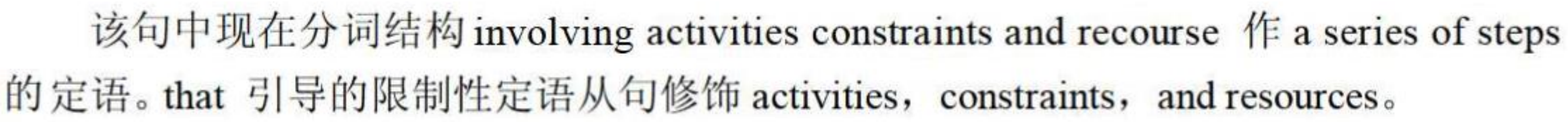
by 引导的方式状语包含 what we have done 和 what we learned by doing it 这两个名词从句，作 reusing 的宾语。

- 这些方法的重要性在于它们获得了我们的设计经验，使我们能够通过重用我们所做的和我们从中所学到的东西来利用我们过去的项目。

capitalize v.	利用		

9.3 Software Process

- A process is a series of steps involving activities, constraints, and resources that produce an intended output of some kind.



该句中现在分词结构 involving activities constraints and recourse 作 a series of steps 的定语。that 引导的限制性定语从句修饰 activities， constraints， and resources。

- 过程是一系列的步骤，这些步骤涉及到产生某种预期输出的活动、约束和资源。
- The process uses resources, subject to a set of constraints (such as schedule), and produces intermediate and final products.
- 该过程使用资源，受一组约束（如进度）约束，并生成中间产品和最终产品。

prescribe v.	规定	**subject to**	服从
intermediate	中间的		

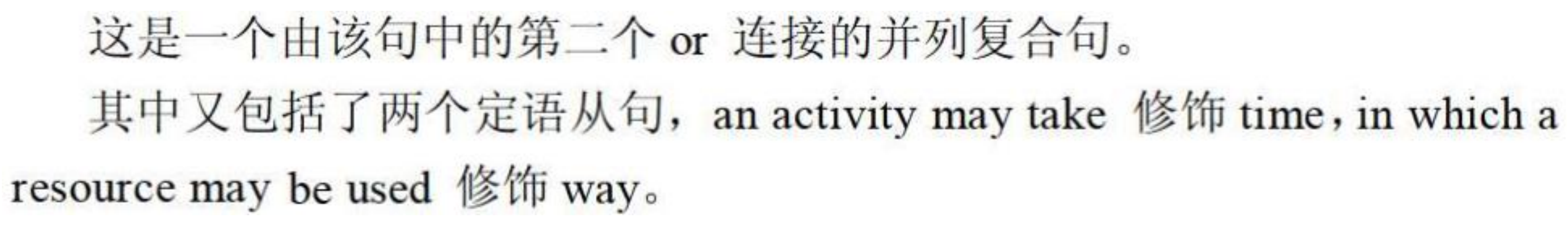
9.3 Software Process

- The process may be defined as a hierarchy of processes, organized so that each subprocess has its own process model.
- 过程可以定义为一个层次结构，这样组织起来使得每个子过程都有自己的过程模型。
- The activities are organized in a sequence, so that it is clear when one activity is performed relative to the other activities.
- 这些活动是按顺序组织的，以便明确相对于其他活动何时执行一项活动。

subprocess	子过程		

9.3 Software Process

- For example, the budget or schedule may constraint the length of time an activity may take or a tool may limit the way in which a resource may be used.



这是一个由该句中的第二个 or 连接的并列复合句。

其中又包括了两个定语从句，an activity may take 修饰 time，in which a resource may be used 修饰 way。

- 例如，预算或计划可能会限制活动可能需要的时间长度，或者工具可能会限制资源的使用方式。

9.3 Software Process

- Thus, the software development process is sometimes called the software development life cycle (SDLC), because it describes the life of a software product from its conception to its implementation, delivery, use, and maintenance.
- 因此，软件开发过程有时被称为软件开发生命周期（SDLC），因为它描述了软件产品从构思到实现、交付、使用和维护的生命周期。

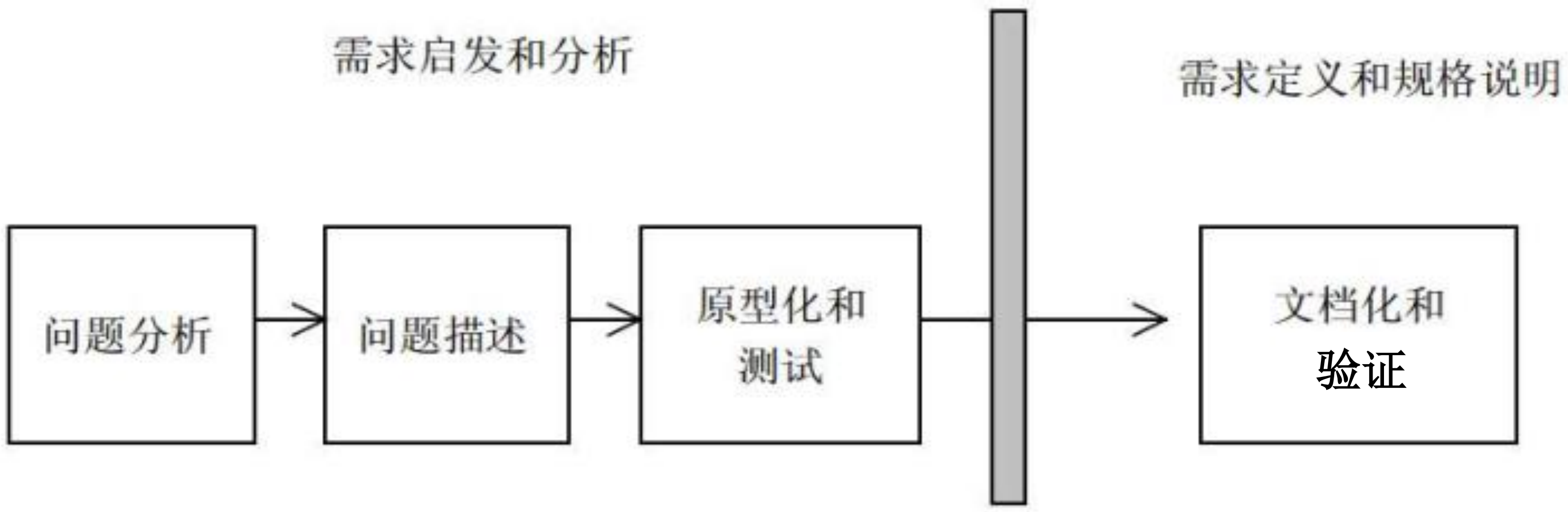
SDLC	软件开发生命周期	unit testing	单元测试
requirements analysis		需求分析	
integration testing		集成测试	

9.3 Software Process

- Each stage is itself a process (or collection processes) that can be described as a set of activities. And each activity involves constraints, outputs and resources.
- 每个阶段本身就是一个过程（或一组过程），可以将其描述为一组活动。每项活动都涉及约束、输出和资源。

9.3 Software Process

- A requirement is a feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose.
- 需求是系统的一个特征，或者是系统为了实现其目的而能够做的事情的描述。



9.3 Software Process

- We work with our customers to elicit the requirements, by asking questions, demonstrating similar systems, or even developing prototype of all or part of the proposed system.
- 我们与客户合作，通过提出问题、演示类似的系统、甚至开发整个或部分计划开发系统的原型来启发需求。
- Then, the requirements are often rewritten, usually in a more mathematical representation, so that the designers can transform the requirements into a good system design.
- 然后，需求经常被重写，通常以更数学的形式表示，这样设计者就可以将需求转化为一个好的系统设计。

elicit v.	启发	**capture v.**	表达，描述

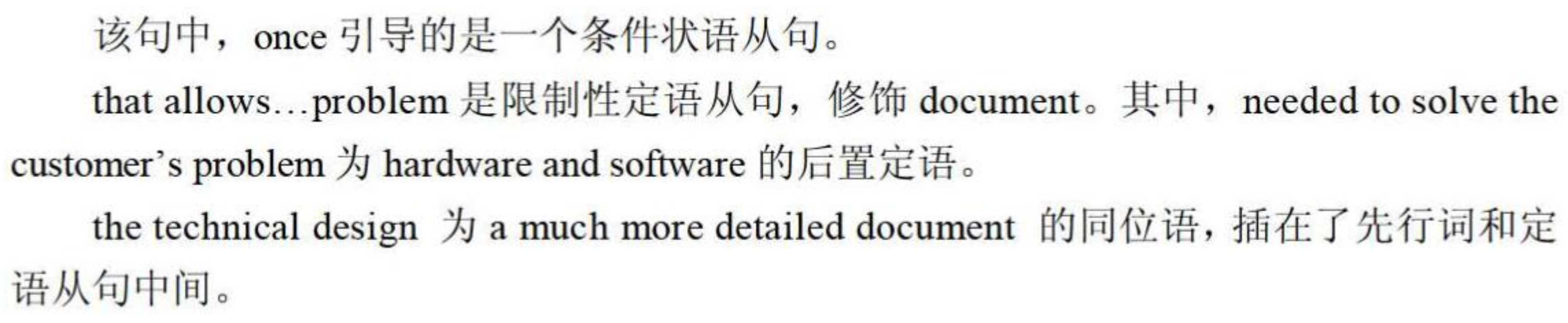
9.3 Software Process

- A verification step ensures that the requirements are complete, correct, and consistent, and a validation step makes sure that we have described what the customer intends to see in the final product.
- 检查步骤确保需求是完整、正确和一致的，验证步骤确保我们已经描述了客户在最终产品中想要看到的内容。
- To transform requirements into a working system, designers must satisfy both customers and the system builders on our development team.
- 要将需求转化为能工作的系统，设计人员必须同时满足客户和开发团队中的系统构建者。

verification	检查	**validation**	验证

9.3 Software Process

- Once the customer approves the conceptual design, we translate the conceptual design into a much more detailed document, the technical design, that allows system builders to understand the actual hardware and software needed to solve the customer's problem.



该句中，once 引导的是一个条件状语从句。

that allows...problem 是限制性定语从句，修饰 document。其中，needed to solve the customer's problem 为 hardware and software 的后置定语。

the technical design 为 a much more detailed document 的同位语，插在了先行词和定语从句中间。

- 一旦客户批准了概念设计，我们就将概念设计转化为更详细的文档，即技术设计，它使系统构建人员能够了解解决客户问题所需的实际硬件和软件。

iterative a.	**迭代的**	**conceptual**	**概念的**

9.3 Software Process

- The process is iterative because, in actuality, the designers move back and forth among activities involving understanding the requirements, proposing possible solutions, testing aspects of a solution for feasibility, presenting possibilities to the customers, and documenting the design for the programmers.
- 这个过程是迭代的，因为实际上，设计人员在各种活动中来回移动，这些活动涉及理解需求、提出可能的解决方案、测试解决方案的各个方面的可行性、向客户展示可能性以及为程序员编写设计文档。

feasibility	可行性		

9.3 Software Process

- Sometimes the choice is based on designer preferences; other times, the method is dictated by the system’s required structure or data.
- 有时选择是基于设计师的偏好；其他时候，方法由系统所需的结构或数据决定。。
- Starting with a high-level depiction of the system’s key elements and creating lower-level looks at how the system’s features and functions will fit together.
- 从系统关键元素的高层次描述开始，创建底层模型来说明系统的特征和功能将如何结合在一起。

decomposition	分解		

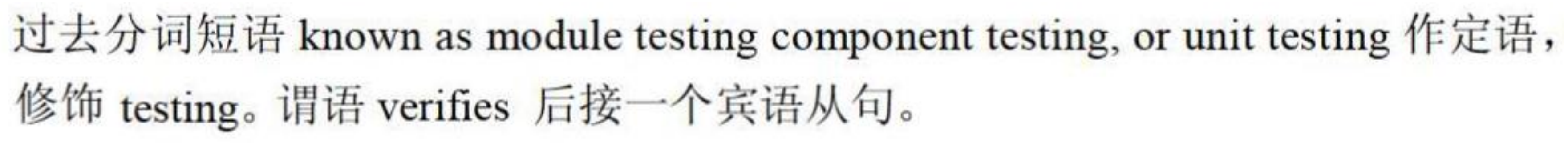
9.3 Software Process

- The design is a guide to the function or purpose of each component, but the programmer has great flexibility in implementing the design as code.
- 设计是对每个组件的功能或用途的指导，但程序员在以代码形式实现设计时具有很大的灵活性。
- No matter what language is used, each program component involves at least three major aspects: control structures, algorithms, and data structures.
- 无论使用何种编程语言，每个程序组件至少涉及三个主要方面：控制结构、算法和数据结构。

devise v.	**设计**	**specification**	**规格说明**

9.3 Software Process

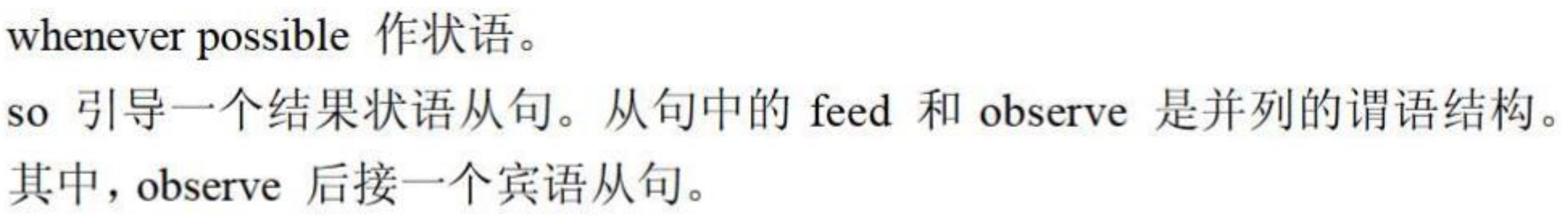
- Such testing, known as module testing, component testing, or unit testing, verifies that the component functions properly with the types of input expected from studying the component's design.


过去分词短语 known as module testing component testing, or unit testing 作定语，修饰 testing。谓语 verifies 后接一个宾语从句。

- 这种测试称为模块测试、组件测试或单元测试，它验证组件是否能根据研究组件设计所需的输入类型正常工作。

9.3 Software Process

- Unit testing is done in a controlled environment whenever possible, so the test team can feed a predetermined set of data to the component being tested and observe what output actions and data are produced.



whenever possible 作状语。

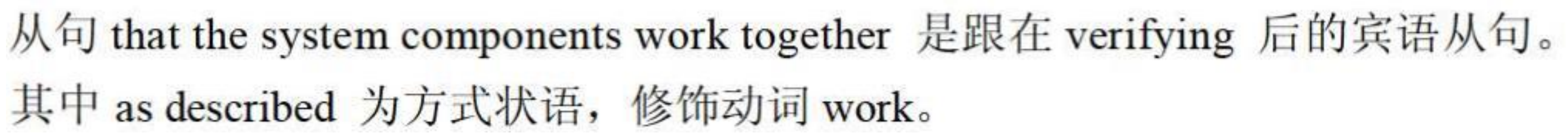
so 引导一个结果状语从句。从句中的 feed 和 observe 是并列的谓语结构。其中，observe 后接一个宾语从句。

- 单元测试尽可能在受控环境中进行，因此测试团队可以向被测试的组件提供一组预先确定的数据，并观察产生了哪些输出动作和数据。

boundary condition	边界条件	

9.3 Software Process

- Integration testing is the process of verifying that the system components work together as described in the system and program design specifications.


从句 that the system components work together 是跟在 verifying 后的宾语从句。其中 as described 为方式状语，修饰动词 work。

- 集成测试是验证系统组件是否按照系统和程序设计规范所述协同工作的过程。

in accordance with		符合；依照	
functionality	功能		

9.3 Software Process

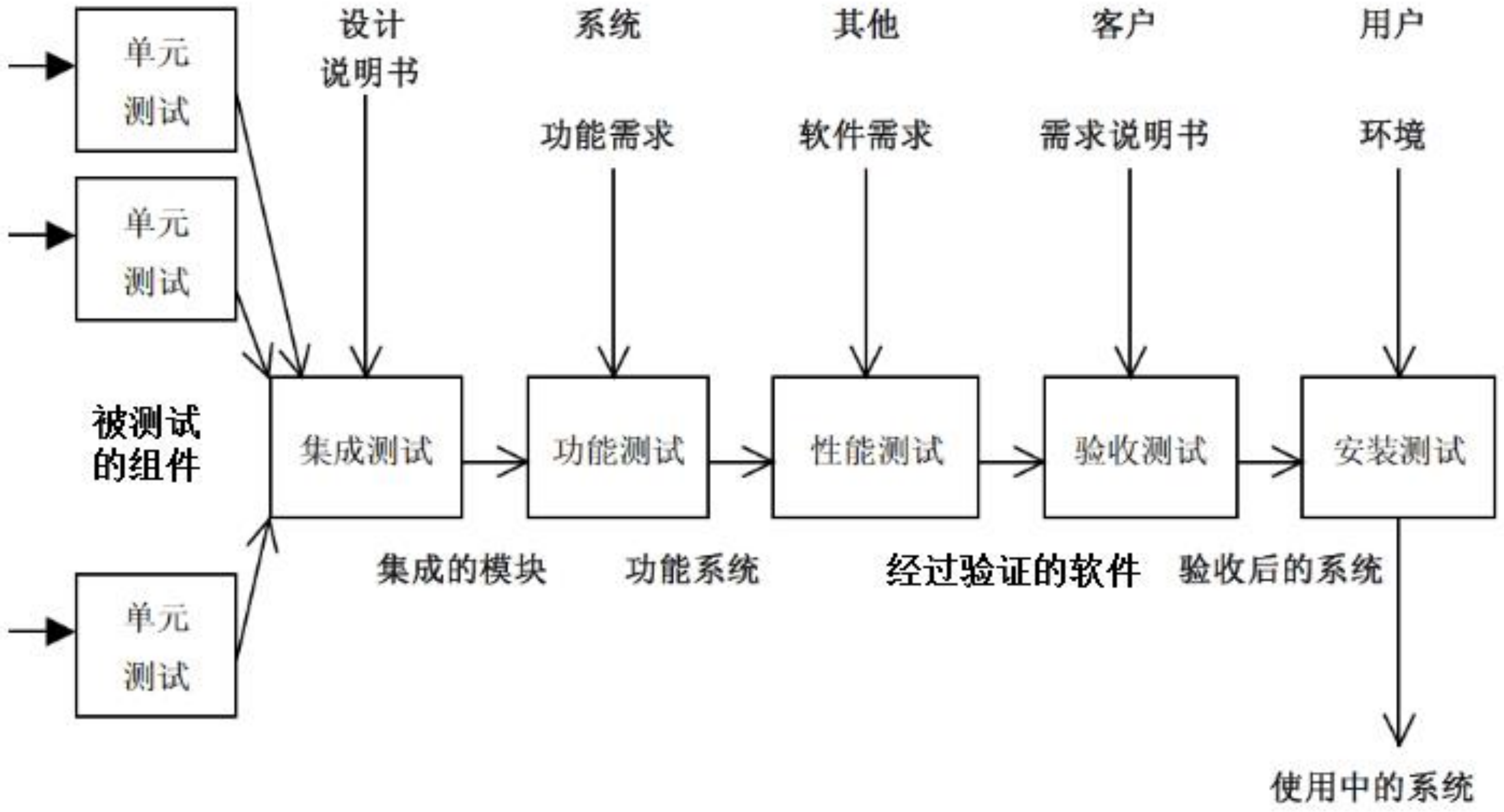
- A function test evaluates the system to determine if the functions described by requirements specification are actually performed by the integrated system.
- 功能测试评估系统以确定需求规格说明中描述的功能是否由集成后的系统实际执行。
- Recall that the requirements are documented in two ways: first in the customer’s terminology and again as a set of the software and hardware requirements the developers could use.
- 回想一下，需求是以两种方式记录的：首先是以客户的术语，其次是作为开发人员可以使用的一组软件和硬件需求。

9.3 Software Process

- Upon completion of acceptance testing, the accepted system is installed in the environment in which it will be used; a final installation test is run to make sure that the system still functions as it should.
- 验收测试完成后，将被认可的系统安装在将要使用的环境中；运行最终安装测试，以确保系统仍能正常工作。

function test	**功能测试**	**performance test**	**性能测试**
confer	**协商;交换意见**	**acceptance test**	**验收测试**
installation test	**安装测试**	**validated**	**有效的**

9.3 Software Process



9.3 Software Process

- System development is complete when the system is operational, that is, when the system is being used by users in an actual production environment.
- 当系统运行时，即当用户在实际生产环境中使用系统时，系统开发即告完成。
- Any work done to change the system after it is in operation is considered to be maintenance.
- 在系统运行后，为改变系统而进行的任何工作均视为维护。

9.3 Software Process

- Maintenance activities are similar to those of development: analyzing requirements, evaluating system and program design, writing and reviewing code, testing changes, and updating documentation.
- 维护活动类似于开发活动：分析需求、评估系统和程序设计、编写和检查代码、测试更改和更新文档。

role	角色	**intimate a.**	密切的
control over	控制	**day-to-day**	日常的
perfect v.	使完善	**degrade v.**	降低，削弱
corrective	纠正性的	**adaptive**	适应性的
perfective	完善性的	**preventive**	预防性的