蓝色的是不确定的= =

* 'Find Product Help' with underline in a use case text means emphasis（突出、强调）F

当一个用例引用另一个用例，被引用的那个用例会被加以下划线

补充：用例是文本文档，而非图形；用例建模主要是编写文本的活动而非制图。

编写用例文本比画用例图更重要

* In practical OOD, we draw interactive diagram firstly, and then draw Design Class Diagram T

OOD（面向对象设计）的大致步骤：定义用例→定义领域模型（domain model）→定义交互图（interactive）→定义设计类图

* Software Engineering are the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software T

IEEE对于SE的定义：将系统化的、严格约束的、可量化的方法应用于软件的开发、运行和维护，即将工程化应用于软件

* Any UML diagram in OOA must describe the System as Black Box F

黑盒用例是最常用和推荐使用的类型，不代表一定要使用。

* XP 'story card' practice is acceptable for catching requirements in UP. T

UP：unified process统一过程，UP是灵活开放的，鼓励引进其他迭代方法中有用的时间，比如XP极限编程、Scrum等。UP项目甚至可以接受简单的XP素材卡片实践

迭代开发是UP和大多数其他现代方法中的关键实践

* An object in domain layer can send messages to an UI object F

领域模型是对现实世界中对象的概念透视图，而不是软件透视图。模型（领域）对象不应该直接与UI对象连接。只能通过接口向UI对象发送消息

* Interface is a UML classifier T

UML类元（classifier）是描述行为和结构特性的模型元素，他们是对众多UML元素的繁华，这些元素包括类、接口、用例和参与者。其中类和接口最为常用。

* Primary actors always be on the left and supporting actors on the right in UML Use Case Diagram T

表示法建议：主要参与者置于左侧，支持性参与者置于右侧。对于用例语境图，用例仅限于用户目标级别的用例。

* A timer may be a special actor of the system T

参与者是某些具有行为的事物，可以是人、计算机系统或组织。参与者可以是另一个计算机系统、一个计时器、一个传感器等。任何一个功能性需求，都有参与者启动。

* A domain model is a visual representation of conceptual classes or real-situation objects in a domain T

课本概念：领域模型是对领域内的概念类或者现实世界中对象的可视化表示。

* In Agile modeling, the purpose of modeling (sketching UML, …) is primarily to understand, not to document T

敏捷建模的目的：主要是用于理解和沟通，而不是构建文档

* Use-case-driven means the development team employs the use cases from requirements gathering through code and test T

用例驱动意味着系统用例是整个系统开发的基础，在以下四个方面用例都起到一个基础的作用：需求、分析与设计、实现、测试。

* The System Event handled by model layer in the MVC pattern F

系统事件由controller管理。

* Showing a report object in a domain model is always useful. F

报表对象（report objects）一般来说在领域模型中显示其他信息的报表并没有意义，因为其所有信息都是源于或复制于其他信息源的，这是排除的理由；另一方面，就业务规则而言，他有特殊的作用。所以要根据具体实例来分析，并非总是有用的。

* It is more than a few weeks long for most projects in Inception. F

初始阶段主要是为项目目标建立一些初始的共同构想，确定项目是否可行，并决定是否值得进入细化阶段加以认真研究。当认为大部分项目的初始阶段后持续几周或更长时间，说明自己并不了解初始阶段。

* A use case named 'Manage <X>', such as manage user, means that the user goal is to collapse(集中) CRUD on object X T

一般来说，为每个用户目标分别定义用例，用力的名称应该和用户目标类似，但有例外，就是将分散的CRUD（创建、提取、更新、删除）目标合并成一个CRUD用例，并习惯性称之为Manage<X>

* Finding stakeholders of the system is the most important thing for catching requirements T

无解……

* In domian model, only show some meaningful and interesting associations among classes T

关联：是类之间的关系，表示有意义和值得关注的链接。

* The message 'd = getProductDescription(id)' in sequence diagram is wrong syntax for missing a type of id F

类型信息并非必要，如果明显或者不重要，完全可以省略类型信息。

* In UP, a suitable iteration length is two to six weeks timebox T

P17大部分迭代方法建议迭代时间在2-6周之间

* The candidate use case 'Log In' can pass usefull Test, for we use it every day F

EXM???? PML的PPT上原话：“Log In：Boss not happy if this is all you do all day!”

* Composition in the UML loosely suggests whole-part relationships F

aggregation聚合不精确地描述了整体-部分关系；composition组合，是一种很强的整体-部分聚合关系

* The same UML diagram can be used in multiple perspectives T

同一种UML图可以用于多种透视图perspectives

* Incremental development for the same use case across iterations T

跨越多次迭代对用例进行增量式开发

* An end user can find the operation process in UML Use Case Diagram F

用例图能够展示系统边界、位于系统边界之外的事物以及系统如何被使用，不包括这些操作的具体过程。

* A '/' symbol before the attribute name is wrong in domain model F

UML约定，属性名称前加“/”符号，表示该属性重要，且可导出。

* Inception in one sentence: Envision the product scope, vision, and business case T

用一句话概括初始阶段，预见项目的范围、设想和业务案例。

* In Agile Principles, our highest priority is early and continuously deliver valuable software to customer T

敏捷原则，优先级最高的是，通过早期和持续交付有价值的软件来满足客户

* In UP, A Kickoff meeting should be hold to clarify goals in days at the beginning of iteration T

在项目开始为期三周的迭代中，召开启动会议，明确本次迭代的任务和目标

* Logical Architecture divides the software classes into packages, subsystems, and layers T

逻辑架构是软件类的宏观组织结构，他将软件类组织为：包、子系统和层。