**Mid-Term Review**

**Topics**

**=======================================================================**

Process

software process有不同的结构，但是基本元素都是相同的

//phase 1 阶段

software elements analysis  
提取需求。客户通常知道他们想要什么，但不知道软件应该做什么。演示实时代码可能有助于降低需求不正确的风险。

Scope analysis 范围分析  
应当确定和明确发展范围的分析。某些功能可能超出了开发项目的范围，这可能是成本的作用，也可能是需求不明确的结果

Phase2

Specification

规范：准确描述要编写的软件的任务。

大多数成功的规范都是为了理解和调整已经开发得很好的应用程序而编写的，但是在应用程序开发之前也要仔细地指定安全关键的软件系统。

对于必须保持稳定的外部接口external interface来说，规范是最重要的。

Phase3 软件结构

结构:系统的抽象表示。

架构:

软件系统必须满足产品的要求，

它必须确保将来的需求能够得到满足。

处理软件系统和其他软件产品之间的接口

硬件或主机操作系统

Phase4

编码coding

软件工程工作中最明显的部分，但它不一定是最大的部分。

测试 testing

测试团队开发的软件的各个部分是软件工程师的职责。

实现 implementation

在对代码进行适当的测试和批准之后，它将被转移到生产环境中

Phase 5

文档documentation

记录软件的内部设计，以便将来进行维护和增强。文档对于外部接口来说是最重要的。

软件培训和支持software training and support

客户组织必须能够有效地使用系统。

Phase 6

维护maintenancce

维护和增强软件以处理新发现的问题或新需求(可能比软件的初始开发花费更多的时间)。

一小部分是修复bug。大多数维护都是扩展系统来做新事情

相比之下,大约⅔的土木工程,建筑和建筑也以类似的方式维护工作。

------------------------------------------------------------------------------------------------------------

两种process,waterfall,spiral

waterfall

requirement->design->implementation->verification->maintenance

需求，设计，实现，确认，维护

sequential:连续性， 一步搞定了就进入下一步，不需要进一步的交互。

但是事实上，作者并未排斥交互，他是有用的，他清楚描述了一系列事件，legally defensible，是法律上可辩护的

他在传统工程被使用，但在软件工程并未被很好理解，我们这个project就是waterfall，但是中间夹杂了agile与iterative，敏捷迭代开发

waterfall phases, requirements specification->design(一个给coder写的蓝图或者说Plan，当design完成时，coder会implemment这个design)->construction(coding/impelemntation)->integration集成（不同的team写的多个software component会被集合起来）->testing and debugging->installation安装->maintenance

对经典瀑布模型的普遍理解是，只有当前一个阶段完成并完善时，才应该移动到该阶段。瀑布模型中的开发阶段是离散的，它们之间没有跳跃或重叠

waterfall模型是document driven的，每一步都会产生documents

一共有两种模型

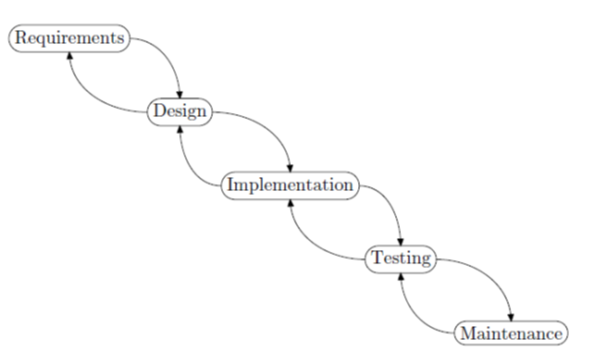
说明性模型描述了应该做什么 prescriptive model

描述性模型描述实际发生的事情 descriptive model

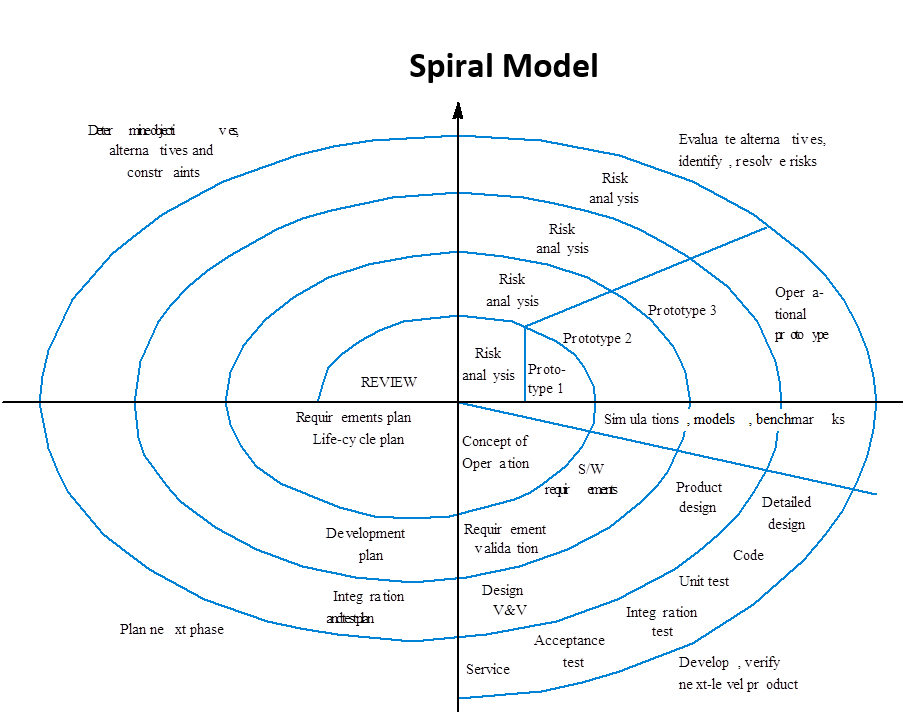
瀑布模型是作为描述性模型引入的:它的目的是描述工程项目中发生的事情，并将这些观察结果用于未来的项目。//曾经

瀑布模型成为了一种说明性模型:它提出了一种软件开发方法，这种方法很难精确地遵循，并且在遵循时存在一些问题。//现在

Royce模型



spiral



每个phase都从一个设计目标开始，并以让客户回顾到目前为止的progress结束。每个phase都要analysis以及engineering efforts，并且着眼于项目的最终目标

螺旋模型结合了原型模型和瀑布模型的特点。它适用于大型、昂贵和复杂的项目。

第一个prototype:

重复前面的步骤，直到客户确信改进的原型表示所需的最终产品。

新的系统需求应该被尽可能详细地定义。

对新系统进行了初步设计。

在初步设计的基础上构建了新系统的第一个原型。(一个缩小的系统，最终产品的近似)。

第二个prototype:

1)评估第一个原型的优点、缺点和风险;

(2)确定第二原型的要求;

(3)第二原型的规划设计;

(4)第二原型的构建与测试。

接下来的prototype：

新原型的评估方式与前一个原型相同。

如果有必要，根据上面概述的四重步骤，从它开发另一个原型。

重复这些步骤，直到客户满意为止。

最后在改进原型的基础上构建了最终的系统。对最终的系统进行了全面的评估和测试。

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

incremental development

增长软件的一个好听的名字是增量开发incremental development。

增量开发的优势:

任何时候都有一个可以运行的系统;

客户client可以看到系统并提供反馈

进步progress是看得见的，而不是埋没在文件中;

有些错误是可以避免的。

坏处：

设计的error部分也成了system的一部分，这样就很难移除

客户看见了可能性然后想更改需求requirements

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

iterative process

迭代开发：在那些可能导致灾难的在问题或错误的假设发生之前，构建软件项目最初较小但越来越大的部分（增长的部分），以帮助及早发现重要的问题。

迭代过程有可能达到客户的设计目标，即便这些客户嘴比较笨，不知道怎么描述

迭代开发有一个巨大的挑战：创建可靠的开发基础foundation，然后才能迭代扩展。

这样的基础通常需要大量的前期分析upfront analysis和原型porototypinng来构建开发模型。

开发模型通常依赖于特定的设计模式和实体关系图(ERD)。

如果没有这个预先的基础，迭代开发可能会产生长期的挑战，这些挑战在成本和质量方面都产生很大的浪费

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

agile process

敏捷软件开发过程建立在迭代开发的基础上。

他们使用一种更轻量lighter、更以人为中心people-centric的方法。他们使用反馈，而不是计划，作为他们的主要控制机制。反馈是由对软件的定期测试和不断发展的新版本驱动的。

它们似乎比旧的方法更有效，

用更少的程序员时间来生产更多功能、更高质量的软件，

但是从业务角度来看，它们的缺点是不能提供长期规划能力 long-term planning capability

敏捷方法声称它会带来最大的效益provide the most bang for the buck，但没有说具体什么时候会带来效益，或者最终需要多大的效益

========================================================================

Activities

Business modeling Domain diagram

=================================================================

Requirements

=================================================================

Design

Testing

CMMI

5个level:

initiaL: processes are usually ad hoc and chaotic.

managed, the projects have ensured that processes are planned and executed in accordance with policy

defined: processes are well characterized and understood, and are described in standards, procedures, tools, and methods.

quantitively managed: the organization and projects establish quantitative objectives for quality and process performance and use them as criteria in managing projects.

optimizing:an organization continually improves its processes based on a quantitative understanding of its business objectives and performance needs.

初始:过程通常是特别的和混乱的。

在管理下，项目确保了流程是按照策略计划和执行的

定义的:过程具有良好的特征和理解，并在标准、过程、工具和方法中进行描述。

量化管理:组织和项目建立质量和过程绩效的量化目标，并将其作为管理项目的标准。

优化:组织基于对其业务目标和性能需求的定量理解，不断改进其流程。

Software Management

========================================================================

Domain Model：是一个针对特定问题的把所有topic联系在一起的conceptual model概念模型

四个要素

Domain Class: 他描述了不同的entities//(实体，意思就是COURSE,LEC,LAB这种大类)，每个CLASS表达一种object的大类

业务对象business objects——表示在业务中操作的东西，例如Order。

真实世界的对象real world objects——业务跟踪的对象，例如联系人、站点。

发生的事件events that transpire-例如销售和付款。

Attributes: 他们的attribute//参数

Associations:relationship关系，特指entities之间的relationship，

additional rules:无法用上面三要素表达的复杂rule。

domain model域模型应该用系统术语terminology of system记录关键概念document key

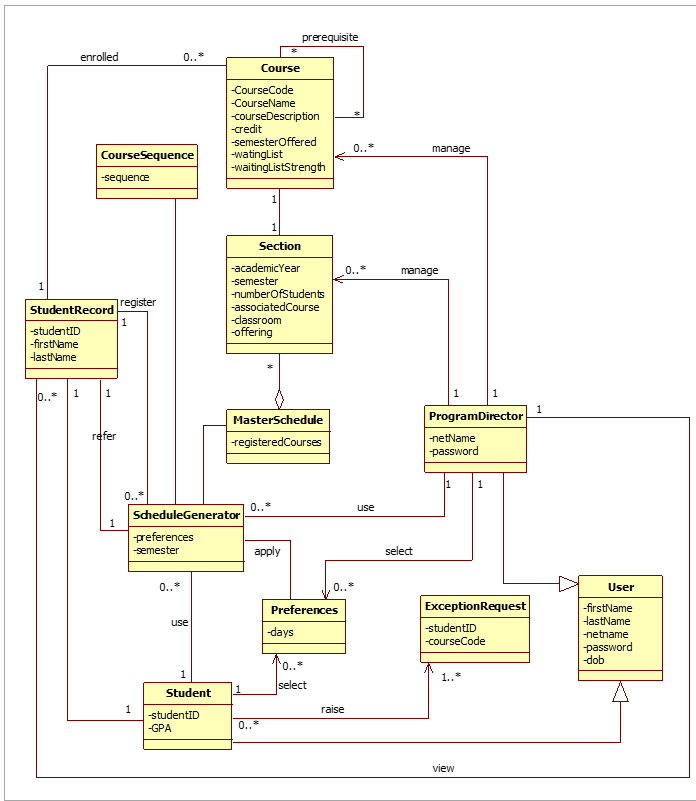
应该注意的事

Domain model是static model，静态模型，他不应该描述出time flow, sequence of events事件序列，和information flow

避免展示程序关系procedural relationship。这个模型不包括软件。域模型中的对象是编程对象的候选对象。

另外UML 模型中只有参数attribute， 没有method

终极目标是，写出一个time invariant static domain model，不随时间改变的域模型



注意这个数字，代表数量，星就是无穷的意思，..代表范围，relationship则是动词

=================================================================

Team -Formation , Assignment of artifacts

========================================================================

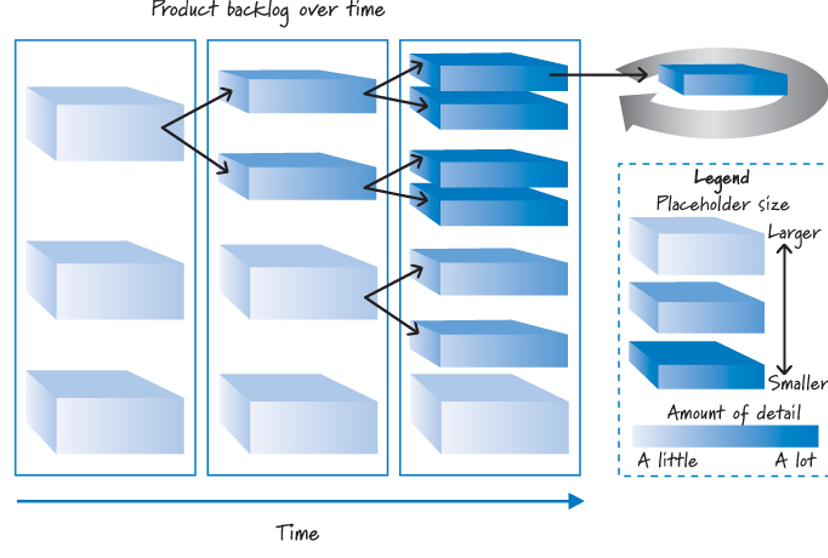
User stories



以这个形式写出的，user或designer的想法

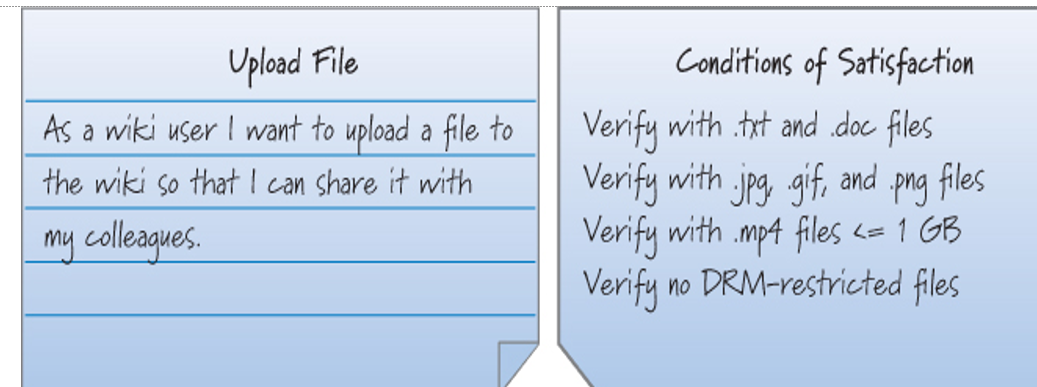
应该从概括的user story 一点点精炼成细节的多个user story，progressively refine

Product Backlog and Stories,产品待办列表product backlog与用户故事



随着时间渐进，product backlog 被逐渐细化，然后循环，每个product backlog所占的存储空间越来越小，包含的细节越来越多

User story还可以以conditions of satisfaction的形式描述确认信息，他们描述了需要具体做的事情



uploaded file写格式的东西，conditions of satisfaction是desired behavior

conditions of satisfaction可以让设计team更好的理解要执行什么，让客户更好的理解他的产品是否往他所预想的方向去了。

agile user story的好处是你可以以不同的level of detail来描述user story,最宽泛的user story叫做epic，描述了整个项目的主要目标

由于epic太过庞大，对于agile team来说不可能在一个iteration之内完成，所以需要把他分成detail，

方式一共有两种，一，把一个user story分成多个user story，二，给user story加上conditions of satisfaction

用于sprint level的小user story太小也太多，为了满足更高层次的计划，我们需要更少、更少细节、更抽象的项目。否则，我们就会陷入一堆几乎无关紧要的细节之中。

如果只有一个(小的)事例，我们将有义务在非常细粒度fine-grained上定义所有的需求

分类方法一：

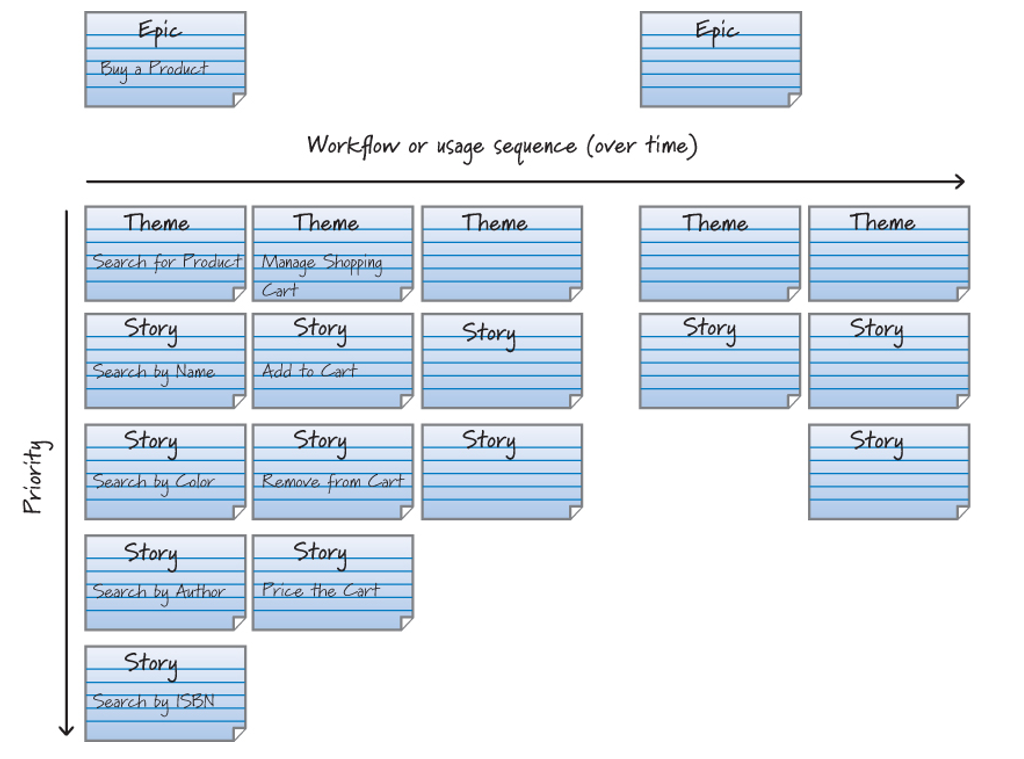
Epic：最宽泛的,几个月完成

Feature：次宽泛的，几个星期完成

分类方法2：

Epic

Theme:用来概括某一功能的user story，它代表着一系列有共同detail的集合//也是user story



user story 的几个特点：

independent: 用户描述应该是独立的，或者至少彼此之间是松散耦合的。呈现高度相互依赖关系的故事使评估、优先级和计划变得复杂

Negotiable,valuable: story不是预先需求文档形式的书面契约。story是可以讨论negotiable细节detail的交流

故事必须对客户、用户或两者都有价值。,技术故事可能很有价值

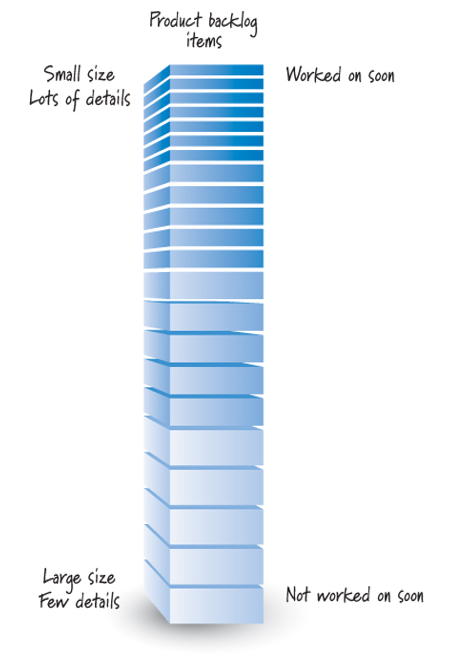
***Estimatable, Small, Testable:*** 故事应该由设计、构建和测试它们的团队进行评估

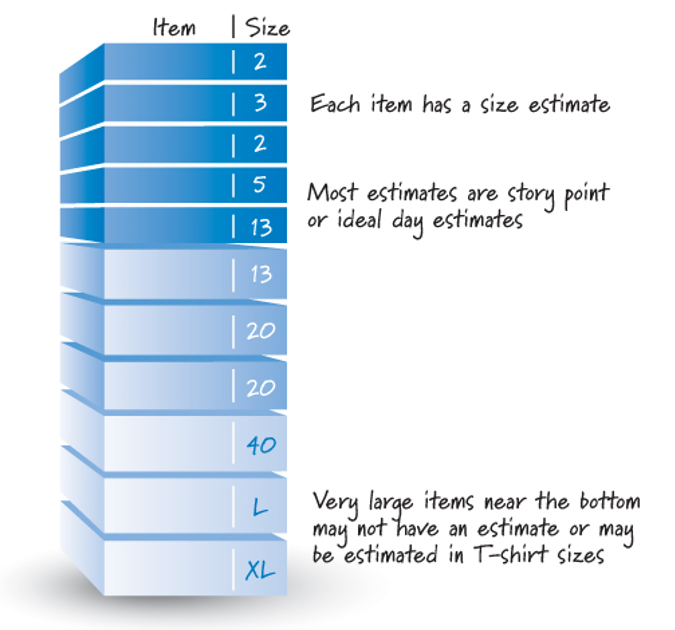
当我们计划处理故事时，故事的大小应该适当。sprint中编写的故事应该很小。

故事应该以二进制的方式进行测试——它们要么通过相关的测试，要么失败。可测试意味着有良好的接受标准

knoledge acquistion stories：知识学习story，通过这个story，我可以学习到什么，比如说，as a developer，我想通过对比两种Implement方法来看看那个快

backlog item,未交付订单项

细节的，小的先做，优先级从上到下慢慢降低，

回估计具体size，而大的就不会

========================================================================

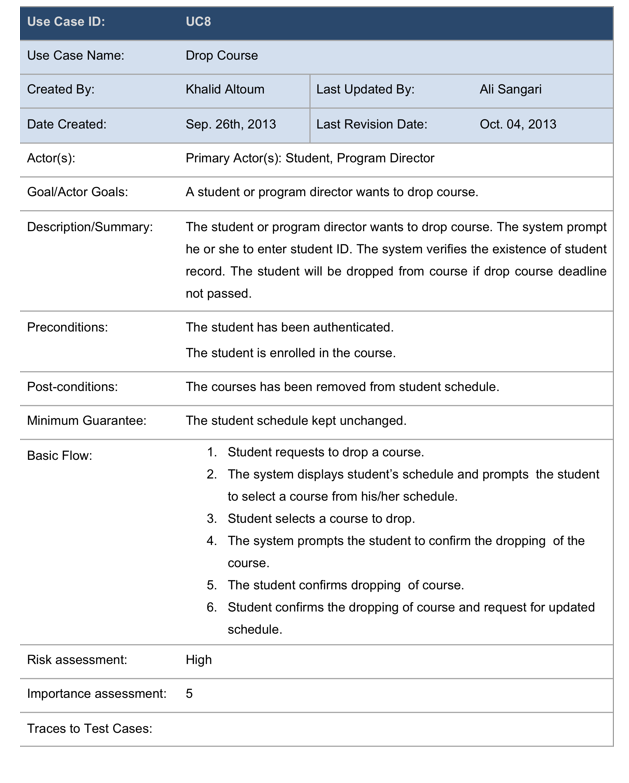
Use cases 用途案例

描述一种用途的简短场景scenario

描述系统的使用的一个集合

文字和图

不包括非功能性需求



========================================================================

Project Backlog Management

Rational Unified Process

Agile Unified Process

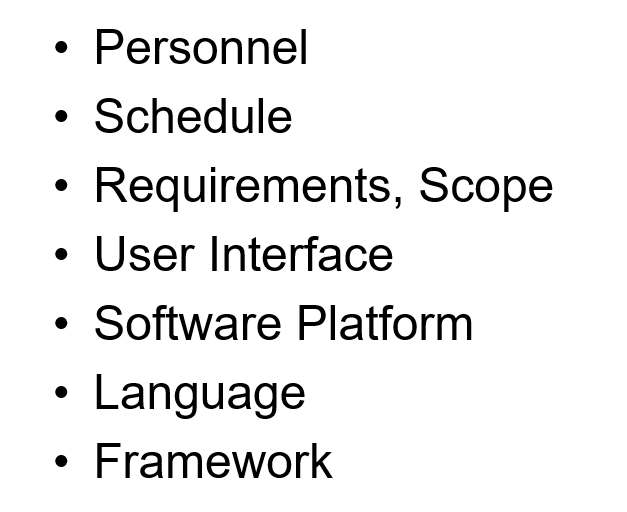
Enterprise Unified Process

========================================================================

Risk

风险是指所有不确定的领域，包括要执行的任务和团队成员实现这些任务的能力

例子：



spiral model的risk

根据客户的选择，如果risk太大，整个项目可以中止。

风险因素可能包括

开发成本超支,

运营成本误判,

根据客户的判断，任何其他可能导致最终产品不令人满意的因素。

========================================================================

Estimation types

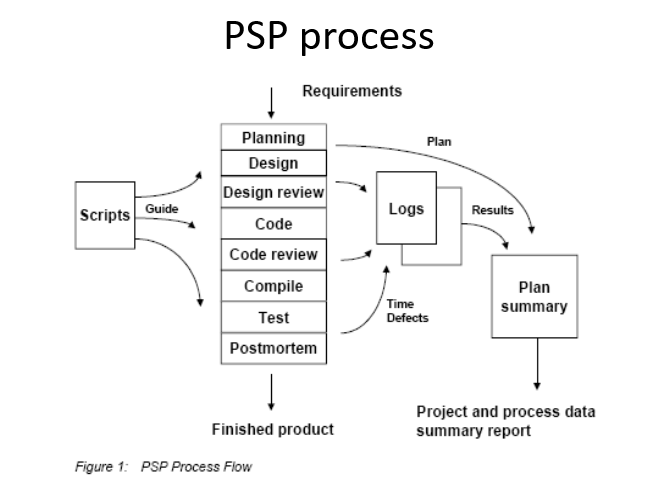
Some more process types

=================================================================

Personal software process

psp

* Help software engineers to:
  + Improve their estimating and planning skills.
  + Make commitments they can keep.
  + Manage the quality of their projects.
  + Reduce the number of defects in their work.
* •帮助软件工程师:
* -提高他们的评估和计划技能。
* -做出他们能够遵守的承诺。
* -管理项目质量。
* -减少他们工作中的缺陷数量。



The first PSP quality principle is that engineers are personally responsible for the quality of the programs they produce. Because the software engineer who writes a program is most familiar with it, that engineer can most efficiently and effectively find, fix, and prevent its defects.

The PSP provides a series of practices and measures to help engineers assess the quality of the programs they produce and to guide them in finding and fixing all program defects as quickly as possible.

In addition to quality measurement and tracking, the PSP quality management methods are early defect removal and defect prevention.

第一个PSP质量原则是工程师个人对他们生产的程序的质量负责。因为编写程序的软件工程师最熟悉它，所以该工程师能够最有效地发现、修复和预防其缺陷。

PSP提供了一系列的实践和措施来帮助工程师评估他们生成的程序的质量，并指导他们尽快发现和修复所有的程序缺陷。

PSP的质量管理方法除了进行质量测量和跟踪外，还包括早期的缺陷消除和缺陷预防。

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

software process

基本四要素 Software Specification, Software Development, Software Validation .Software evolution

waterfall model的基本交流方式：documentation

Growing software: incremental；agile

Agile重要的：individuals and interactions个人与交流;working software 可以运行的软件;Customer Collaboration 客户合作，responding to change 对应变化

Scrum正确的，每次需要三到四周 的planning; 产品特点在开发之前就被选好了;要做的工作在最后被评估；软件实现后交付给系统stakeholder

四种agile techniques 在extreme programming里

1.Requirements expressed as scenarios or user stories需求表示为场景或者用户故事

**Requirements are based on simple customer stories or scenarios that are used as a basis for deciding what functionality should be included in a system increment**

需求基于简单的客户故事或场景，这些客户故事或场景被用作决定系统增量中应该包含哪些功能的基础

2.Refactoring:重构

Programming team look for possible software improvements and make these improvements even where there is no immediate need for them. This improves the understandability of the software and so reduces the need for documentation.

编程团队寻找可能的软件改进，即使在不需要他们也要进行这些改进。这提高了软件的可理解性，从而减少了对文档的需求。

3.Pair programming

pair programming involves programmers working in pairs, developing code together

4.Test-first development

Writing tests before code clarifies the requirements to be implemented.

在代码澄清要实现的需求之前编写测试。

test-first development存在的问题

programmers may take short-cuts when developing tests so that the system tests are incomplete

Some tests can be difficult to write incrementally

It is difficult to estimate the completeness of a test set

程序员在开发测试时可能会走捷径，这样系统测试就不完整了

有些测试很难增量编写

很难估计测试集的完整性

non functional requirements from the list of requirements

the system should support for keystone as an authentication backend. This allows full integration with nova,swift,and other OpenStack services that share this common authentication method

The system should support at least 100 nodes registered and the system should be able to select a number of nodes allocated to a given project

Enterprises need to be compliant with data security laws and regulations that require sensitive data to be encrypted at rest as well in motion

系统应该支持keystone作为身份验证后端。这允许与nova、swift和其他共享这种公共身份验证方法的OpenStack服务进行全面集成

系统应该支持至少100个注册节点，并且系统应该能够选择分配给给定项目的节点数量

企业需要遵守数据安全法律和法规，这些法律和法规要求在静态和动态中对敏感数据进行加密

动词要identifier，形容词vague，软件要version，user 就是requirement on users，because就是complex sentence，error less

DevOps approach

devops is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality

devops是一组实践，目的是在确保高质量的情况下，减少向系统提交更改和将更改放入正常生产之间的时间