软件工程

一、软件过程

软件过程的概念

 件 工 程 过 程 (Software Engineering 
Process) 是 为 获 得 软 件 产 品 ， 在 软 件 工 具 支 持 
下 由 软 件 工 程 师 完 成 的 一 系 列 软 件 工 程 活 动 
软 件 工 程 的 过 程 则 是 将 软 件 工 程 的 方 法 和 工 
具 综 合 起 来 以 达 到 合 理 、 及 时 地 进 行 计 算 机 软 
件 开 发 的 目 的 。 
过 程 定 义 了 方 法 使 用 的 顺 序 、 要 求 交 付 的 文 
档 资 料 、 为 保 证 质 量 和 协 调 变 化 所 需 要 的 管 理 
、 及 软 件 开 发 各 个 阶 段 完 成 的 里 程 碑 

经典软件过程模型的特点(瀑布模型、增量模型、演化模型、统一过程模型)

**瀑布模型**

简单，分阶段，阶段间存在因果关系，

各个阶段完成后都有评审，允许反馈，不支持

用户参与，要求预先确定需求

需求易于完善定义且不易变更的软件系统

[快速原型模型](http://wiki.mbalib.com/wiki/%E5%BF%AB%E9%80%9F%E5%8E%9F%E5%9E%8B%E6%A8%A1%E5%9E%8B)

不要求需求预先完备定义，支持用户参与，

支持需求的渐进式完善和确认，能够适应用户需求的变化

需求复杂、难以确定、动态变化的软件系统

[增量模型](http://wiki.mbalib.com/wiki/%E5%A2%9E%E9%87%8F%E6%A8%A1%E5%9E%8B)

软件产品是被增量式地一块块开发的，

允许开发活动并行和重叠

[技术风险](http://wiki.mbalib.com/wiki/%E6%8A%80%E6%9C%AF%E9%A3%8E%E9%99%A9)较大、用户需求较为稳定的软件系统

[迭代模型](http://wiki.mbalib.com/wiki/%E8%BF%AD%E4%BB%A3%E6%A8%A1%E5%9E%8B)

不要求一次性地开发出完整的软件系统，将软件

开发视为一个逐步获取用广需求、完善软件产品的过程

需求难以确定、不断变更的软件系统

[螺旋模型](http://wiki.mbalib.com/wiki/%E8%9E%BA%E6%97%8B%E6%A8%A1%E5%9E%8B)

结合瀑布模型、[快速原型模型](http://wiki.mbalib.com/wiki/%E5%BF%AB%E9%80%9F%E5%8E%9F%E5%9E%8B%E6%A8%A1%E5%9E%8B)和迭代模

型的思想，并引进了[风险分析](http://wiki.mbalib.com/wiki/%E9%A3%8E%E9%99%A9%E5%88%86%E6%9E%90)活动

需求难以获取和确定、软件开发风险较大的软件系统

[RUP](http://wiki.mbalib.com/wiki/RUP)

可改造、扩展和剪裁：可以对它进行设计、

开发、维护和发布；强调迭代开发

复杂和需求难以获取和确定的软件系统；

软件开发项目组拥有丰富的软件开发和管理经验

过程评估与CMM/CMMI的基本概念

CMM等级

 力 等 级 
第 一 级 初 始 级 
（ 最 低 级 ） 
第 二 级 可 重 复 级 
第 三 级 已 定 义 级 
第 四 级 已 管 理 级 
第 五 级 优 化 级 
（ 最 高 级 ） 
特 点 
软 件 工 程 管 理 制 度 缺 乏 ， 过 程 缺 乏 定 义 、 混 乱 无 
序 。 成 功 依 靠 的 是 个 人 的 才 能 和 经 验 ， 经 常 由 于 缺 
乏 管 理 和 计 划 导 致 时 间 、 费 用 超 支 。 管 理 方 式 属 于 
反 应 式 ， 主 要 用 来 应 付 危 机 。 过 程 不 可 预 测 ， 难 以 
重 复 。 
基 于 类 似 项 目 中 的 经 验 ， 建 立 了 基 本 的 项 目 管 理 制 
度 ， 采 取 了 一 定 的 措 施 控 制 费 用 和 时 间 。 管 理 人 员 
可 及 时 发 现 问 题 ， 采 取 措 施 。 一 定 程 度 上 可 重 复 类 
似 项 目 的 软 件 开 发 。 
已 将 软 件 过 程 文 档 化 、 标 准 化 ， 可 按 需 要 改 进 开 发 
过 程 ， 采 用 评 审 方 法 保 证 软 件 质 量 。 可 借 助 CASE 
工 具 提 高 质 量 和 效 率 。 
针 对 制 定 质 量 、 效 率 目 标 ， 并 收 集 、 测 量 相 应 指 
标 。 利 用 统 计 工 具 分 析 并 采 取 改 进 措 施 。 对 软 件 过 
程 和 产 品 质 量 有 定 量 的 理 解 和 控 制 。 
基 于 统 计 质 量 和 过 程 控 制 工 具 ， 持 续 改 进 软 件 过 
程 。 质 量 和 效 率 稳 步 改 进 。 
关 键 过 程 
需 求 管 理 ， 项 目 计 划 ， 项 目 跟 踪 和 监 控 ， 软 件 子 
合 同 管 理 ， 软 件 配 置 管 理 ， 软 件 质 量 保 障 
组 织 过 程 定 义 ， 组 织 过 程 焦 点 ， 培 训 大 纲 ， 软 件 
集 成 管 理 ， 软 件 产 品 工 程 ， 组 织 协 调 ， 专 家 审 评 
定 量 的 软 件 过 程 管 理 和 产 品 质 量 管 理 
缺 陷 预 防 ， 过 程 变 更 管 理 和 技 术 变 更 管 理 

基本概念

CMMI（Capability Maturity Model Integration，[能力成熟度模型集成](https://baike.baidu.com/item/%E8%83%BD%E5%8A%9B%E6%88%90%E7%86%9F%E5%BA%A6%E6%A8%A1%E5%9E%8B%E9%9B%86%E6%88%90)）

将各种能力成熟度模型（即：Software CMM、Systems Eng-CMM、People CMM和Acquisition CMM）整合到同一架构中去，由此建立起包括软件工程、系统工程和软件采购等在内的诸模型的集成，以解决除软件开发以外的软件系统工程和软件采购工作中的迫切需求。

CMMI框架包括[软件能力成熟度模型](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E8%83%BD%E5%8A%9B%E6%88%90%E7%86%9F%E5%BA%A6%E6%A8%A1%E5%9E%8B)CMM 2.0草案，系统工程能力成熟度模型，软件采购能力成熟度模型，继承产品和过程开发等。

CMMI的:“[关键过程域](https://baike.baidu.com/item/%E5%85%B3%E9%94%AE%E8%BF%87%E7%A8%8B%E5%9F%9F)”25个，“目标”105个， “关键实践”485条。

CMMI的评估方式：

自我评估：用于本企业领导层评价公司自身的软件能力。

主任评估：使本企业领导层评价公司自身的软件能力，向外宣布自己企业的软件能力。

CMMI的评估类型：

软件组织的关于具体的软件过程能力的评估。

软件组织整体软件能力的评估（软件能力成熟度等级评估）。

CMMI的基本思想

1、解决软件项目过程改进难度增大问题

2、实现软件工程的并行与多学科组合

3、实现过程改进的最佳效益

equirement management 
software project planning 
software project tracking oversight 
software subcontract manage

敏捷宣言与敏捷过程的特点。

敏捷宣言，也叫做敏捷软件开发宣言，正式宣布了对四种核心价值和十二条原则，可以指导迭代的以人为中心的软件开发方法。

敏捷宣言强调的敏捷软件开发的四个核心价值是：

个体和互动高于流程和工具

工作的软件高于详尽的文档

客户合作高于合同谈判

响应变化高于遵循计划

敏捷选择提出的12条原则已经应用于管理大量的业务以及与IT相关项目中，包括商业智能（BI）。12原则包括：

1.通过早期和连续型的高价值工作交付满足“客户”。

2.大工作分成可以迅速完成的较小组成部门。

3.识别最好的工作是从自我组织的团队中出现的；

4.为积极员工提供他们需要的环境和支持，并相信他们可以完成工作；

5.创建可以改善可持续工作的流程；

6.维持完整工作的不变的步调。

7.欢迎改变的需求，即使是在项目后期。

8.在项目期间每天与项目团队和业务所有者开会。

9.在定期修正期，让团队反映如何能高效，然后进行相应地行为调整。

10.通过完成的工作量计量工作进度。

11.不断地追求完善。

12.利用调整获得竞争优势。

二、软件需求

软件需求的概念

软件[需求分析](https://baike.baidu.com/item/%E9%9C%80%E6%B1%82%E5%88%86%E6%9E%90)就是把软件计划期间建立的

[软件可行性分析](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%8F%AF%E8%A1%8C%E6%80%A7%E5%88%86%E6%9E%90)求精和细化，分析各种可能的解法，并且分配给各个软件元素。需求分析是软件定义阶段中的最后一步，是确定系统必须完成哪些工作，也就是对目标系统提出完整、准确、清晰、具体的要求。

需求工程的基本过程

需求获取：通过与用户的交流，对现有系统的观察及对任务进行分析，从而开发、捕获和修订用户的需求；

需求建模：为最终用户所看到的系统建立一个概念模型，作为对需求的抽象描述，并尽可能多的捕获现实世界的语义；

形成需求规格：生成需求模型构件的精确的形式化的描述，作为用户和开发者之间的一个协约；

需求验证：以需求规格说明为输入，通过符号执行、模拟或快速原型等途径，分析需求规格的正确性和可行性，包含有效性检查，一致性检查，可行性检查和确认可验证性；

需求管理：支持系统的需求演进，如需求变化和可跟踪性问题。

分层数据流模型

用例和场景建模及其UML表达(用例图、活动图、泳道图、顺序图)

 围 与 类 用 域 类 
执 巧 运 行 或 者 模 拟 动 薈 族 的 雪 义 
整 性 飞 'ML 中 各 个 要 完 整 一 致 的 表 
事 的 关 条 
司 性 ． UML 中 闷 卩 № 
{ b 雇 ， protected, private, 
package 〕 
弁 范 ！ 为 事 ． 夭 糸 命 芒 ] 
一 些 要 求 
的 饰 ： 各 种 号 的 含 义 
旷 犋 制 ' 对 示 符 号 古 行 自 廴 展 
閹 汾 ： 类 、 对 二 翁 祛 ， 口 、 实 
约 束 褻 曙 加 新 的 义 或 者 改 变 已 存 
征 的 一 中 剛 [ 制 
规 则 
机 制 
构 造 块 
UML 
关 以 { 鳃 聚 含 俄 ） ' 亘 存 在 的 关 
实 ' 类 元 之 《 《 也 义 关 系 ， 一 实 现 了 另 一 个 呆 证 的 契 
约 《 囗 实 刷 》 
泛 化 ： 一 般 皇 的 关 糸 ， 子 类 到 父 类 承 的 泛 化 
注 记 事 物 ： UM [ 中 部 分 〔 E 噁 ， s ． 对 圍 及 笑 羔 等 阱 行 的 
说 明 冫 
分 组 事 物 ： 描 述 UML 團 中 的 螟 分 〔 E Packal»e ． 元 〕 
行 为 事 物 ： 借 L 牖 L 囝 中 空 0 及 时 百 上 荇 大 {E.g 《 n 地 on 
． 坝 黑 以 功 一 动 作 舍 ） 
梅 引 ： L 牖 L 酗 的 部 分 ， 備 概 念 壶 者 理 酗 篷 、 g 
Class ． 貝 石 稻 同 属 性 和 作 的 对 象 匣 合 ） 
帛 例 的 ' 站 用 户 角 度 。 体 现 用 户 提 0 功 
《 系 统 峥 黠 《 表 现 。 与 类 与 葵 之 司 的 关 
包 ： 将 糸 中 繁 歃 的 类 。 关 糸 銎 辑 关 糸 成 0 
活 动 到 ； 《 类 或 者 对 篆 完 成 糸 tfi 一 糸 动 
时 序 图 ！ 从 消 息 发 生 顶 字 方 面 糸 所 中 对 象 之 闫 的 交 
到 象 绢 识 笔 构 万 面 而 糸 中 对 交 与 美 糸 
构 僻 ． 親 年 的 湖 睡 组 部 分 
獲 中 实 灯 环 境 中 # 行 部 况 
状 到 
涔 类 或 者 对 篆 江 率 蚜 中 的 所 有 状 

[用例](https://baike.baidu.com/item/%E7%94%A8%E4%BE%8B)图由参与者（Actor）、用例（[Use Case](https://baike.baidu.com/item/Use%20Case)）、[系统边界](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F%E8%BE%B9%E7%95%8C)、箭头组成，用画图的方法来

 例 图 中 涉 及 的 关 系 有 ： 关 联 、 泛 化 、 包 含 、 扩 展 。 
如 下 表 所 示 ： 
关 类 型 
说 明 
表 示 符 号 
关 泛 包 扩 
联 化 含 展 
参 与 者 与 用 例 间 的 关 
参 与 者 之 间 或 用 例 之 间 的 关 
包 括 
用 例 之 间 的 关 
用 例 之 间 的 关 

. 关 联 (Association) 
表 示 参 与 者 与 用 例 之 间 的 通 信 ， 任 何 一 方 都 可 发 送 或 接 受 消 息 。 
〖 箭 头 指 向 〗 ： 指 向 消

. 包 含 (Include) 
包 含 关 系 用 来 把 一 个 较 复 杂 用 例 所 表 示 的 功 能 分 解 成 较 小 的 步 骤 。 
〖 箭 头 指 向 〗 ： 指 向 分 解 出

. 依 R(Dependency) 
以 上 4 种 关 系 ， 是 UML 定 义 的 标 准 关 系 。 但 VS2010 的 用 例 模 型 图 中 ， 添 加 了 依 赖 关 系 ， 用 带 箭

 含 (include) 、 扩 A(extend) 、 泛 化 (Inheritance) 的 区 别 ： 
条 件 性 ： 泛 化 中 的 子 用 例 和 include 中 的 被 包 含 的 用 例 会 无 条 件 发 生 ， 而 extend 中 的 延 伸 用 例 的 发 生 是 有 
条 件 的 ； 
直 接 性 ： 泛 化 中 的 子 用 例 和 extend 中 的 延 伸 用 例 为 参 与 者 提 供 直 接 服 务 ， 而 include 中 被 包 含 的 用 例 为 参 
与 者 提 供 间 接 服 务 。 
对 extend 而 言 ， 延 伸 用 例 并 不 包 含 基 础 用 例 的 内 容 ， 基 础 用 例 也 不 包 含 延 伸 用 例 的 内 容 。 
对 Inheritance 而 画 
一 个 用 例 图 示 例 ： 
RP 
用 户 資 u 彆 改 
， 子 用 例 包 含 基 础 用 例 的 所 有 内 容 及 其 和 其 他 用 例 或 参 与 者 之 间 的 关 系 ； 
角 色 信 息 鲦 护 
用 尸 贷 护 
笤 理 鲦 护 
用 户 资 注 锕 
导 出 资 
包 括 ， 
用 P 懷 息 彆 改 
用 户 贷 u 查 询 

 如 何 阅 读 活 动 图 
阅 读 简 单 活 动 图 
用 户 下 订 单 
用 户 选 择 支 
付 方 式 
川 户 取 消 或 订 
单 超 过 时 限 
订 单 取 消 
生 成 送 货 单 
NO 
收 款 
供 应 商 送 货 
[Yes) 
修 改 订 单 项 
所 有 订 单 项 己 
状 态 
送 货 完 毕 
[Yes) 
（ 0 〗 订 单 完 成 

 动 图 的 主 要 元 素 
· 初 始 节 点 和 活 动 终 点 ： 用 一 个 实 心 圆 表 示 初 始 节 点 ， 用 一 个 圆 圈 内 加 一 个 实 心 圆 来 表 示 活 动 终 点 
· 活 动 节 点 ： 是 活 动 图 中 最 主 要 的 元 素 之 一 ， 它 用 来 表 示 一 个 活 动 
用 户 下 订 单 
len=a.length+l 
dispatch(aOrder) 
消 息 
表 达 式 
文 字 描 述 
· 转 换 ： 当 一 个 活 动 结 束 时 ， 控 制 流 就 会 马 上 传 递 给 下 一 个 活 动 节 占 在 活 动 图 中 称 之 为 “ 转 换 " ， 用 一 条 带 箭 头 的 直 线 来 表 示 
活 动 图 的 主 要 元 素 
· 分 支 与 监 护 条 件 ： 分 支 是 用 菱 形 表 示 的 ， 它 有 一 个 进 入 转 换 （ 箭 头 从 外 指 向 分 支 符 号 ） ， 一 个 或 多 个 离 开 转 换 （ 箭 头 从 分 支 符 号 指 向 外 ） 。 而 每 个 离 
开 转 换 上 都 会 有 一 个 监 护 条 件 ， 用 来 表 示 满 足 什 么 条 件 的 时 候 执 行 该 转 换 。 
a>Ol 
(a<0 
· 分 岔 与 汇 合 

 泳 道 的 活 动 图 
客 户 
用 户 下 订 单 
用 户 选 择 支 
付 方 式 
[No 
[Yes 
户 取 消 或 i 
单 超 过 时 限 
订 单 取 消 
系 统 
生 成 送 货 单 
收 款 
[No) 
丨 有 订 单 
项 己 送 货 完 
[Yes) 
Q) 订 单 完 成 
供 应 商 
供 应 商 送 货 
修 改 订 单 项 
状 态 

数据模型建模及其UML表达(类图)

 、 什 么 是 类 图 
类 图 (Class diagram ） 主 要 用 于 描 述 系 

<https://www.cnblogs.com/silent2012/archive/2011/09/07/2169946.html>

行为模型建模及其UML表达(状态机图)。

<https://www.cnblogs.com/sura/archive/2012/07/01/2572083.html>

三、软件设计与构造

软件体系结构及体系结构风格的概念

设计模式的概念

模块化设计的基本思想及概念(抽象、分解、模块化、封装、信息隐藏、功能独立)

软件重构的概念

软件体系结构的UML建模(包图、类图、构件图、顺序图、部署图)

接口的概念

面向对象设计原则(开闭原则、Liskov替换原则、依赖转置原则、接口隔离原则)

内聚与耦合的概念、常见的内聚和耦合类型。

<https://www.cnblogs.com/xiaym896/p/5400677.html>

四、软件测试

软件测试及测试用例的概念

单元测试、集成测试、确认测试、系统测试、回归测试的概念

调试的概念、调试与测试的关系

测试覆盖度的概念

白盒测试、黑盒测试的概念

代码圈复杂度的计算方法

白盒测试中的基本路径测试方法

<http://www.51testing.com/html/44/n-3713444.html>

黑盒测试中的等价类划分方法

<https://www.cnblogs.com/lonelywolf/archive/2012/03/20/2407468.html>