SEC1 软件开发概述

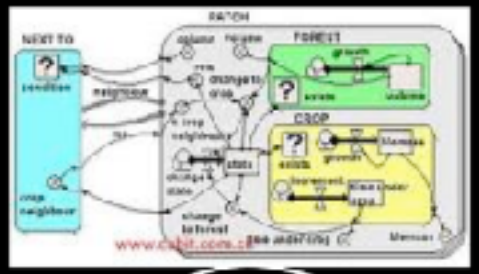
没有银弹

- 软件的开发是一种高度复杂的智力活动。
- 1960’s年代，人们提出了“软件危机”的说法，来描述软件开发的困难，并提出了“软件工程”的概念。
- Brooks在1986年“没有银弹”一文中提到软件开发具有复杂性、不一致性、可变性和不可见性，说明了软件开发是一项困难的任务，不存在特效药。

软件开发

- 使用软件解决我们现实世界中的问题。
- 现代计算机只能执行机器世界中的二进制代码。
- 通常，我们会先用自然语言来描述现实世界的问题；然后通过智力活动对该问题进行思考，为其设计计算机可以完成的解决方案；最后用具体的程序设计语言来实现该解决方案，并通过翻译程序翻译成二进制代码，在计算机上执行以解决该问题。

概念结构



高级语言

中级语言

低级语言

人

编译器

问题

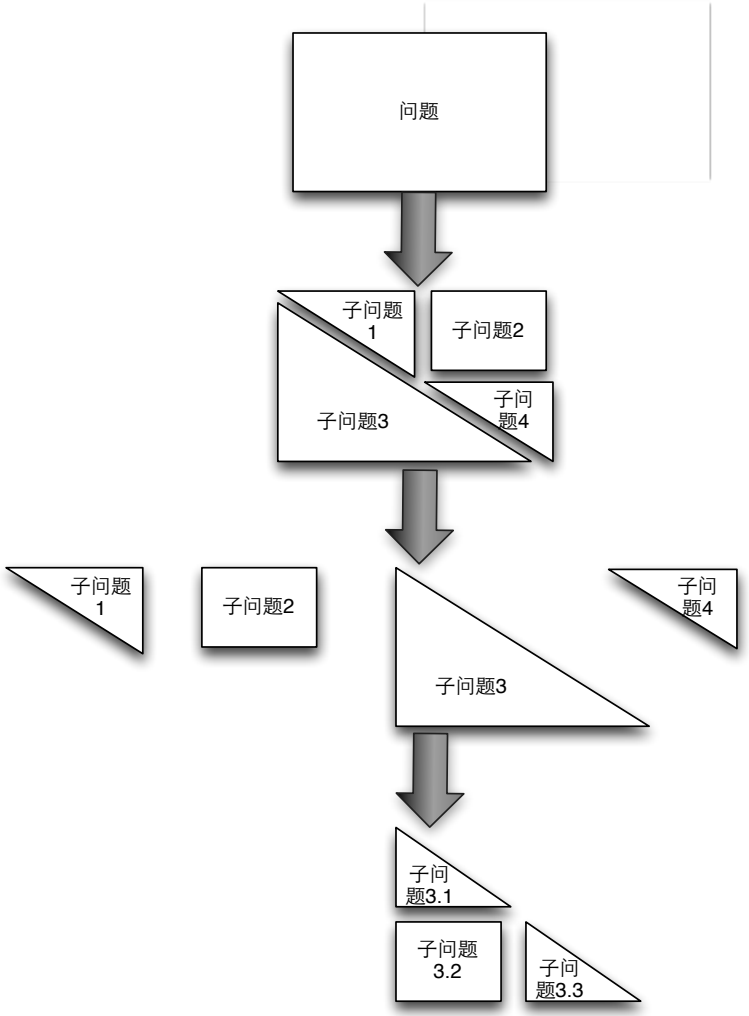
- 人们通过“自然语言”来描述遇到的问题。
- 计算机无法理解自然语言。
- 自然语言有二义性。

解决方案

- “一个相互牵制关联的概念结构，是软件实体必不可少的部分，它包括：数据集合、数据条目之间的关系、算法、功能调用等等。”[Brooks, 1986]。软件开发者通过智力活动，根据计算机的运行原理，所具有的计算能力，来设计针对某个特定问题的解决方案。

解决复杂性

- 复杂性是软件开发面临的主要困难之一。
- 问题分解。



算法

- 在确定的解决方案中，我们会形成一个相互牵制关联的概念结构，其中的算法是指完成一个任务所需要的具体步骤和方法。
- 以下是Donald Knuth在他的著作《The Art of Computer Programming》里对算法下的定义：
  - 输入：一个算法必须有至少零个输入量。
  - 输出：一个算法应有至少一个输出量，输出量是算法计算的结果。
  - 明确性：算法的描述必须无歧义，以保证算法的实际执行结果是精确地符合要求或期望，通常要求实际运行结果是确定的。
  - 有限性：依据图灵的定义，一个算法是能够被任何图灵完备系统模拟的一串运算，而图灵机只有有限个状态、有限个输入符号和有限个转移函数（指令）。而一些定义更规定算法必须在有限个步骤内完成任务。
  - 有效性：又称可行性。能够实现，算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现。

编程

- 确定了解决方案以后，接下来的工作就是使用一种程序设计语言，将解决方案转换成程序。
- 程序设计语言是根据计算机运行原理设计的语言，所以它与自然语言的差异较大，是计算机指令表达的一种方式，不存在歧义性。

机器语言与汇编语言

- 机器语言（Machine Language）是机器的自然语言，它是唯一一种计算机能直接理解并执行的语言。
- 对机器语言进行上层抽象，形成了汇编语言，提高了可读性与可移植性，使得程序变得更加容易修改和调试。

面向问题的语言

- 高级语言使用的是易于理解的符号和英文单词组成的语句，接近自然语言，独立于计算机硬件系统，每条语句都有相当于若干条低级语言语句的功能。
- 如目前流行的Java，C++，Python，PHP，C#，（Visual）Basic，Objective-C，JavaScript等

TIOBE Programming Community Index for February 2017

Feb 2017	Feb 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.676%	-4.47%
2	2		C	8.445%	-7.15%
3	3		C++	5.429%	-1.46%
4	4		C#	4.902%	+0.50%
5	5		Python	4.043%	-0.14%
6	6		PHP	3.072%	+0.30%
7	9	⌃	JavaScript	2.872%	+0.67%
8	7	⌄	Visual Basic .NET	2.824%	+0.37%
9	10	⌃	Delphi/Object Pascal	2.479%	+0.32%
10	8	⌄	Perl	2.171%	-0.06%
11	11		Ruby	2.153%	+0.10%
12	16	⌃⌃	Swift	2.125%	+0.75%
13	13		Assembly language	2.107%	+0.28%
14	38	⌃⌃	Go	2.105%	+1.81%
15	17	⌃	R	1.922%	+0.73%
16	12	⌄⌄	Visual Basic	1.875%	+0.02%
17	18	⌃	MATLAB	1.723%	+0.63%
18	19	⌃	PL/SQL	1.549%	+0.49%
19	14	⌄⌄	Objective-C	1.536%	+0.13%
20	23	⌃	Scratch	1.500%	+0.71%

选择Java作为本课程学习语言（技术）-来自知乎

- 1、至今为止，java是大量实际得到应用的语言中，可读性最强，最利于阅读和理解，语法最严谨和规范的语言（当然，这也可以理解为语法繁琐的另一种表述）之一.
  - java是目前最佳的算法及数据结构教学语言
  - java是传统的软件开发过程（生命周期管理）的最佳实践语言，即在传统的“需求-设计-代码-测试”这样的一个过程中，java是最能够最大程度贯彻和实践软件工程学的理论的。
  - java的白盒测试方面的表现非常出色和易行
  - java在代码评审，缺陷管理，开发规范约束，大团队的协同开发方面，有着无可争辩的突出优势

- 2. java有着目前为止，最丰富，最强大的IDE开发环境，这是历史原因形成的，包括商业因素在内
- 3. java有着现有所有语言中，拥有最长的产品线，适用性是最广的语言之一。从前端的app应用，到嵌入式，到web，到服务器应用
- 4. java拥有所有语言中最丰富的类库和代码资源
- 5. 至今为止，在被广泛应用的编程语言中，java依然是开发效率最高的语言。之所以使用编程语言这个词，是为了和脚本语言，以及4GL开发工具区隔开来。这些语言和编程语言相比，其适用性都有很大局限，而且后者（4GL）和OS平台紧耦合关联。事实上，近10多年来，脚本语言领域突飞猛进（有赖于web应用提供了广阔的舞台），而编程语言基本上自java之后就没有太多大的新鲜事了。

商业层面

- 1. java是目前为止唯一的，在商业和开源领域都得到大力推广，推荐和使用的语言，其背后的推力是任何语言所不能比拟的。
- 2. java的诞生和发展，赶上了千载难逢的好机遇，历史造就了java。

高级语言

- 优点
  - ①高级语言具有易学性
  - ②高级语言具有易读性。
  - ③高级语言程序易移植。
  - ④高级语言为程序员带来了便捷。
- 缺点
  - ①高级语言“翻译”后，目标程序冗长。
  - ②高级语言“翻译”后，目标程序运行速率不高。
  - ③高级语言程序无法直接访问、控制硬件。

编译和解释

- 编译器（Compiler）和解释器（Interpreter）都是将某种高级编程语言写成的源代码转换成低级编程语言目标代码的电脑程序。
- 编译：源代码（source code）→预处理器（preprocessor）→编译器（compiler）→汇编程序（assembler）→目标代码（object code）→链接器（linker）→可执行程序（executables）。
- 解释器是一个执行程序的虚拟机，一次能够翻译高级语言程序的一段、一行、一条命令或一个子程序。
- 解释技术更容易进行开发和调试，但开销很大；编译技术则可以产生更高效的代码，能够更加有效使用内存，对代码进行优化，程序执行更快。

软件开发

- 1950’s年代，计算机主要用于科学计算，程序员主要是硬件工程师和部分科学家。
- 1960’s年代以后，计算机逐步应用到企业和各种商业环境中，数据处理和事务计算成为了重要的工作内容，而这时的程序员还没有多少经验和原则可以遵循，这时的软件开发是“工艺式”的。“个人英雄主义编程”。
- 1960’s年代后期，“软件危机”。

“软件工程”

- “软件工程”一词1968年首先在NATO（北约）组织的一次会议上作为正式的术语出现，标志着这一新的学科的开始。
- 会议组织者Brian Randell：“我们特意选择‘软件工程’这个颇具争议性的词，是为了暗示这样一种意见：软件的生产有必要建立在某些理论基础和实践指导之上——在工程学的某些成效卓著的分支中，这些理论基础和实践指导早已成为了一种传统。”

“工艺式”的软件开发的问题?

- 这是因为“工艺式”的软件开发难以解决软件开发的高度复杂性和对软件质量的要求。
- “狗窝”：“摩天大楼”
- 摩天大楼的复杂度和质量要求远比狗窝要高的多。如果要应对更复杂、更高质量要求的软件开发，我们也必须采用工程化的方法来开发，这样才有可能取得成功。

定义

- 软件工程作为一个新兴学科，一直以来都缺乏一个得到大家公认的定义。
- IEEE在软件工程术语汇编中的定义
- 软件工程是：
  - ①将系统、规范、可度量的方法应用于软件的开发、运行和维护，即将工程应用于软件。
  - ②对①中所述方法的研究。

- 计算机科学技术百科全书的定义如下:
- 软件工程是应用计算机科学、数学及管理科学等原理，开发软件的工程。软件工程借鉴传统工程的原则、方法，以提高质量、降低成本。其中，计算机科学、数学用于构建模型与算法，工程科学用于制定规范、设计范型（Paradigm）、评估成本及确定权衡，管理科学用于计划、资源、质量、成本等管理。

软件工程知识域（SWEBOK 3.0，2014）

- 软件需求
- 软件设计
- 软件构造
- 软件测试
- 软件维护
- 软件配置管理
- 软件工程管理
- 软件工程过程

软件工程模型与方法

软件质量

软件工程职业实践

软件工程经济学

计算基础

数学基础

工程基础

“软件工程道德规范和专业实践”

- 第一条原则中就提出软件工程师的一切行为原则的总纲：不能违背公众利益。
- “计算机及其相关技术正逐渐成为推动政府、教育、工业、商业、医疗、娱乐和整个社会发展的核心技术，软件工程师正是通过亲身参加或者教授软件系统的分析、说明、设计、开发、授证、维护和测试等实践工作，为社会做出了巨大贡献。也因为他们在开发软件系统中所起的重要作用，软件工程师有很大机会去为社会做好事或者给社会带来危害，有能力让他人以及影响他人为社会做好事或者给社会带来危害。为了尽可能确保他们的努力应用于好的方面，软件工程师必须做出自己的承诺，使软件工程师成为有益的和受人尊敬的职业，为了符合这一承诺，软件工程师应当遵循下列职业道德规范和实践要求。”

软件开发生命周期（Software Development Life Cycle，SDLC）

- 软件开发生命周期指软件产品从开发到报废的生命周期，通常周期中包括了需求分析、软件设计、实现与调试、测试与验收、部署、维护等活动。
- 软件报废：不需要其功能；很难在合理的成本范围内继续演进开发原有软件。

软件开发过程

- 简称软件过程。
- 软件开发过程是指一个软件产品开发的方法，它描述了软件开发中的活动和任务。简单的说，过程就是软件开发中的一系列活动，如果能够按照这些活动进行工作，我们就可以获得预想的结果。
- “过程决定质量”

- 常见的软件开发过程模型有瀑布、迭代、螺旋模型和敏捷软件开发等，有很多分类的方法。

- 当前有关于软件开发过程的讨论与争议很多，还没有一种大家共同认可的开发过程，但多数人都同意软件过程对软件开发具有重要的指导意义，软件开发不应该是无序的、混乱的。

- Why?

    - 软件项目通常都是很复杂的系统，开发人员需要按照一定的开发模型来分解工作任务，按照一定的步骤进行工作以完成项目的开发。如果没有开发模型的存在，开发人员很容易迷失在复杂系统中，缺乏指引，陷入混乱。

创建-修补
(Build-Fix)

- 编码-修补式软件开发被分成编码和修补两个阶段，它没有合理的开发计划，系统的开发由一系列仅能满足当前需要的短期决策构成，开发者往往很随意的做出决定，也不会给出设计的原因。
- 适用于软件规模小、质量要求低。学生作业。
- 问题：大规模、高质量。
    - 当增加新的功能到系统中时，因为系统中充斥了各种临时的局部设计决定，新模块的添加异常艰难。同时，系统缺陷也无处不在，并且难以修补，可能在修复一个缺陷时又带来新的缺陷。

瀑布模型

- 描述了软件开发的基本框架。
- 多数人已经认识到瀑布模型的缺陷，但瀑布模型仍然可以帮助我们认识软件开发活动，并对其提供指导。
- 瀑布模型（Waterfall Model）是由W.W.Royce在1970年最初提出的，它按时间顺序描述了一个软件项目的开发。

- 瀑布模型核心思想是根据开发活动来分解项目。瀑布模型将软件生命周期划分为制定计划、需求分析、软件设计、程序编写、软件测试和运行维护等六个基本活动。

- 瀑布模型设计了一系列软件开发阶段，按时间顺序展开开发活动，从需求分析开始直到产品发布和维护，每个阶段都会产生循环反馈。如果在某个阶段出现了问题，那么最好“返回”上一个阶段并进行修改。项目开发过程像是从一个阶梯“流动”到下一个阶梯，这也是瀑布模型名称的由来。

- 优点：帮助开发人员理解软件开发中的活动和任务，界定清晰的软件开发检查点。

- 问题I：在实践中很少有项目能够以纯线性的方式进行，通过回到前面的阶段或修改前面某个阶段的结果是非常常见的现象，而这时带来的成本增加和系统开发的混乱是很难避免的。

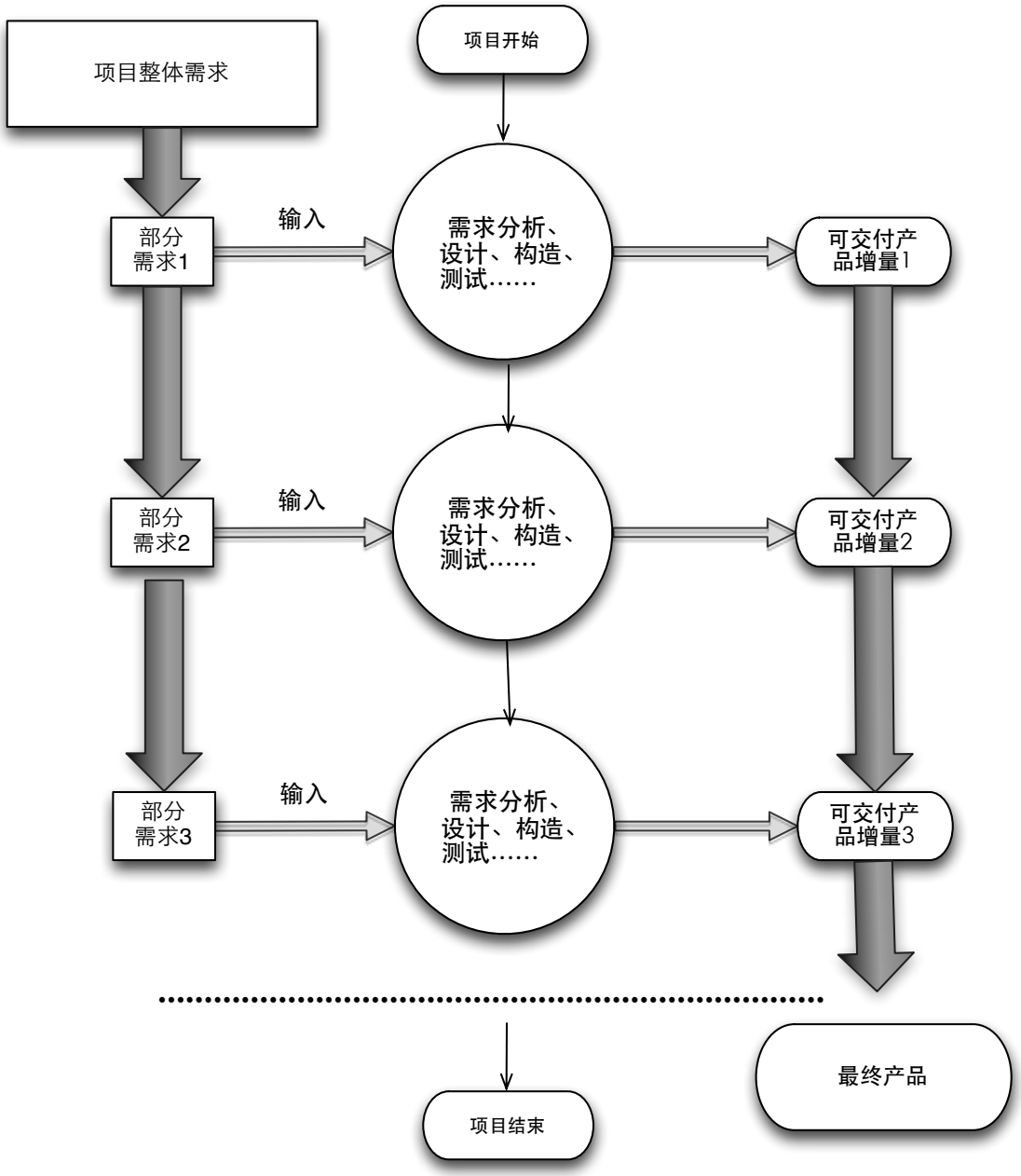
- 问题2：在瀑布模型中，开发人员很难判断其前期的工作是否正确。计划、需求分析和设计中的错误往往要到产生了测试、集成时才能够发现，而在前期，我们没有足够的信息来进行这种判断。

  - 按照软件工程中的贝姆定律（Boehm's Law），在开发过程中越晚修正缺陷，代价就会越高。

- 问题3:只有到项目开发的后期才能看到能够运行的软件，这使得开发团队很难在早期和用户就需求进行验证和讨论，增加了需求误解的可能性，同时也对团队的开发士气造成了不好的影响。

迭代式软件开发

- 迭代式软件开发根据软件项目的不同功能子集来分解项目。
- 在迭代式软件开发中，整个开发工作被组织为一系列小的项目，被称为一系列的迭代。
- 每个迭代周期结束时都应该得到一个经过测试的，集成起来的基本可用的软件产品，某些少量程序缺陷的修复或用户培训可以放在最后一个迭代周期后进行。


项目整体需求
项目开始
部分
需求1
输入
需求分析、
设计、构造、
测试……
可交付产
品增量1
部分
需求2
输入
需求分析、
设计、构造、
测试……
可交付产
品增量2
部分
需求3
输入
需求分析、
设计、构造、
测试……
可交付产
品增量3
项目结束
最终产品

- 特点1.易于应对需求变更
  - 迭代式软件开发可以方便的在每一个迭代结束时修改原有的需求，以应对相应的变更。
  - 用户在看到一个迭代的结果后，会重新审视原来提出的需求，修正原有的需求或增加新的需求。

- 特点2. 提高团队士气
  - 开发人员的士气和对项目的热情部分决定了软件项目的成败。
  - 开发人员通过每次迭代都可以在短期内看到自己的工作成果，有助于他们增强信心，更好地完成开发任务。

真的是迭代式开发?

- “我们正在进行一次分析迭代，接下来会有两次设计迭代。”
- “这次迭代的代码中有很多缺陷，我们在下次迭代时修复它们。”

- 有一个判定标准就是，每一次迭代结束时，系统中的代码需要经过测试，正确的集成起来，并且达到基本可交付的产品级品质。

Why

- 测试与集成都是非常难于估算的开发活动，不能把这样的活动放到迭代开发的最后进行，否则就无法得到迭代式开发的益处。
- 每次迭代的结果是开发团队在本迭代内工作真实有效的反馈，它有助于开发团队及时调整开发计划和软件开发实践。
- 测试和集成工作应该做到：即使本次迭代的软件不进行发布，那么如果要进行发布的话，也不需要大量的工作。

开发长度

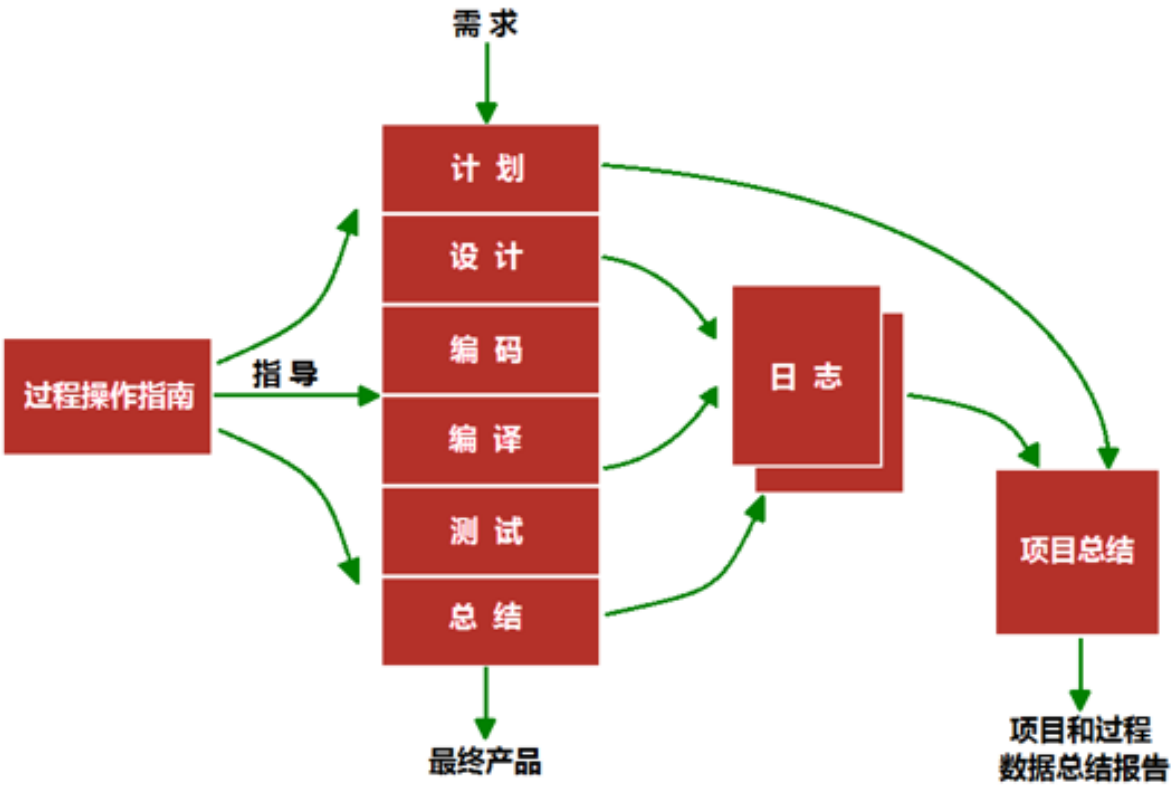
- 增加迭代时间来完成既定的项目功能。
- 固定时间长度（time boxing）的迭代周期：工作节奏、功能优先级。
- 敏捷软件开发都是迭代式开发过程。

个人软件过程

- 个人软件过程（Personal Software Process，PSP）由美国卡内基梅隆大学软件工程研究所（Software Engineering Institute，SEI）的Humphrey等开发，于1995年推出，着重于软件开发人员的个人培训，品质改善和工期估算。

- PSP能够提供：
  - 1、个体软件过程原则；
  - 2、软件开发工程师如何制定准确的计划；
  - 3、软件工程师为改善产品质量需要采取的措施；
  - 4、度量个体软件过程的改进；
  - 5、流程的改变对软件工程师个人能力的影响。

- PSP是包括了数据记录表格、过程操作指南和规程在内的结构化框架。



PSP原则

- “过程质量决定最终产品质量”
  - 软件系统的整体质量由该系统中质量最差的某些组件所决定。
  - 软件组件的质量取决于开发这些组件的软件工程师，更加确切的说，是由这些工程师所使用的开发过程所决定。
  - 作为合格的软件工程师，应当自己度量、跟踪自己的工作，自己管理软件组件的质量。
  - 作为合格的软件工程师，应当从自己开发过程的偏差中学习、总结，并将这些经验教训整合到自己的开发实践中，也就是说，应当建立持续地自我改进机制。

时间度量

- 一条完整的时间日志记录应该包含以下基本信息：编号、所属阶段、开始时间、结束时间、中断时间、净时间以及备注。

- “理想工作时间”，flow

缺陷度量

- 一个典型的缺陷日志包含以下几项内容：编号、发现日期、缺陷类型、注入阶段、消除阶段、消除时间、关联缺陷以及缺陷描述。

PSP缺陷类型描述

编号	缺陷类型	描述
1	Documentation	注释、提示消息错误等；
2	Syntax	拼写、标点、打印、指令格式错误等；
3	Build，Package	变更管理、库以及版本控制方面的错误等；
4	Assignment	变量的申明、重命名、域以及限制方面的错误；
5	Interface	过程调用接口、输入输出、用户接口方面的错误；
6	Checking	错误信息、不充分检验等方面的错误；
7	Data	数据结构与内容方面的错误；
8	Function	逻辑、指针、循环、递归、计算以及功能性错误；
9	System	配置、计时、内存方面的错误；
10	Environment	设计、编译、测试或者其它支撑环境的问题引发的错误。