

**软件工程专业导论报告**



题目 我眼中的软件工程

学 号 201806061108

姓 名 胡皓睿

班 级 计实1801

提交日期 　 2019/4/27

**题目**

**摘要：**本文是关于软件工程的一个概述，展现了软件工程从出现到发展的过程。软件开发是一个把用户需要转化为软件需求，把软件需求转化为软件设计，用软件代码来实现软件设计，对软件代码进行测试，并签署确认它可以投入运行使用的过程。在这个过程中的每一阶段，都包含有相应的文档编制工作。

**关键词：**软件工程，计算机，软件危机

一**、**引言

当第一台数字计算机出现在20世纪40年代早期时，[11]使它们运行的​​指令被连接到机器中。从业者很快意识到这种设计不灵活，并提出了“存储程序架构”或冯·诺依曼架构。因此，“硬件”和“软件”之间的划分始于抽象被用来处理计算的复杂性。编程语言开始出现在20世纪50年代早期[1]，这也是抽象的另一个重要步骤。 Fortran，ALGOL和COBOL等主要语言于20世纪50年代末发布，分别用于解决科学，算法和业务问题。 David Parnas于1972年[2]介绍了模块化和信息隐藏的关键概念，以帮助程序员处理日益复杂的软件系统。“软件工程”一词的起源归因于各种来源。 “软件工程”一词出现在1965年6月出版的计算机和自动化公司提供的服务清单中，并在1966年8月的ACM通讯（第9卷，第8期）中更正式地使用了“ ACM会员“由ACM主席Anthony A. Oettinger执掌; [3]，它也与1968年由第一届软件工程会议Friedrich L. Bauer教授召开的北约会议的标题联系在一起[4]。当时被认为是“软件危机”。[5][6][7]第40届软件工程国际会议（ICSE 2018）以弗雷德里克·布鲁克斯[[8]和玛格丽特·汉密尔顿的全体会议主题演讲庆祝50年的“软件工程”。[9]1984年，软件工程研究所（SEI）成立，是一家联邦政府资助的研发中心，总部位于美国宾夕法尼亚州匹兹堡的卡内基梅隆大学校园内。 Watts Humphrey创建了SEI软件过程计划，旨在了解和管理软件工程过程。引入的流程成熟度级别将成为开发的能力成熟度模型集成（CMMI-DEV），其定义了美国政府如何评估软件开发团队的能力。ISO / IEC JTC 1 / SC 7小组委员会收集了现代的，普遍接受的软件工程最佳实践，并作为软件工程知识体系（SWEBOK）发布。[10]

**二、**我眼中的软件工程

**2.1 软件开发的流程**

软件开发是一个把用户需要转化为软件需求，把软件需求转化为软件设计，用软件代码来实现软件设计，对软件代码进行测试，并签署确认它可以投入运行使用的过程。在这个过程中的每一阶段，都包含有相应的文档编制工作。

软件开发过程当中，遵循一定的流程，主要包括系统分析、系统设计、系统编码、系统测试以及系统的维护等几个阶段。依次概述如下：

1、系统分析

系统分析包括软件需求分析和系统可行性分析。软件需求分析就是回答做什么的问题。它是一个对用户的需求进行去粗取精、去伪存真、正确理解，然后把它用软件工程开发语言（形式功能规约，即需求规格说明书）表达出来的过程。系统可行性分析就是通过需求调查来确定此系统是否具有可行性。

2、系统设计

系统设计可以分为概要设计和详细设计两个阶段。实际上软件设计的主要任务就是将软件分解成模块。概要设计就是结构设计，其主要目标就是给出软件的模块结构，用软件结构图表示。详细设计的首要任务就是设计模块的程序流程、算法和数据结构，次要任务就是设计数据库，常用方法还是结构化程序设计方法。

3、系统编码

系统编码是指把软件设计转换成计算机可以接受的程序，即写成以某一程序设计语言表示的"源程序清单"。

4、系统测试

系统测试的目的不是验证软件的正确性，而是以较小的代价发现尽可能多的错误。测试从需求阶段开始，此后与整个开发过程并行，换句话说，伴随着开发过程的每一个阶段，都有一个重要的测试活动，它是预期内按时交付高质量的软件的保证。

5、系统维护

系统维护是指在已完成对软件的研制（分析、设计、编码和测试）工作并交付使用以后，对软件产品所进行的一些软件工程的活动。即根据软件运行的情况，对软件进行适当修改，以适应新的要求，以及纠正运行中发现的错误。编写软件问题报告、软件修改报告。在实际开发过程中，软件开发并不是从第一步进行到最后一步，而是在任何阶段，在进入下一阶段前一般都有一步或几步的回溯。在测试过程中的问题可能要求修改设计，用户可能会提出一些需要来修改需求说明书等。

总的说来，软件开发是一个环环相扣的设计和实施过程，整个系统开发的过程当中，系统分析和设计是重中之重。只有把握好系统分析，才能使后续改动尽可能多的减少；只有把握好系统设计，才能保证软件的根基比较稳固。也即是它们很大程度上决定着软件开发的周期以及寿命。另外，完美的开发团队和开发过程的合理控制是软件成功开发关键要素之一。

**2.2 目的**

软件工程的目标是：在给定成本、进度的前提下，开发出具有适用性、有效性、可修改性、可靠性、可理解性、可维护性、可重用性、可移植性、可追踪性、可互操作性和满足用户需求的软件产品。追求这些目标有助于提高软件产品的质量和开发效率，减少维护的困难。

（1）适用性：软件在不同的系统约束条件下，使用户需求得到满足的难易程度。

（2）有效性：软件系统能最有效的利用计算机的时间和空间资源。各种软件无不把系统的时/空开销作为衡量软件质量的一项重要技术指标。很多场合，在追求时间有效性和空间有效性时会发生矛盾，这时不得不牺牲时间有效性换取空间有效性或牺牲空间有效性换取时间有效性。时/空折衷是经常采用的技巧。

（3）可修改性：允许对系统进行修改而不增加原系统的复杂性。它支持软件的调试和维护，是一个难以达到的目标。

（4）可靠性：能防止因概念、设计和结构等方面的不完善造成的软件系统失效，具有挽回因操作不当造成软件系统失效的能力。

（5）可理解性：系统具有清晰的结构，能直接反映问题的需求。可理解性有助于控制系统软件复杂性，并支持软件的维护、移植或重用。

（6）可维护性：软件交付使用后，能够对它进行修改，以改正潜伏的错误，改进性能和其它属性，使软件产品适应环境的变化等。软件维护费用在软件开发费用中占有很大的比重。可维护性是软件工程中一项十分重要的目标。

（7）可重用性：把概念或功能相对独立的一个或一组相关模块定义为一个软部件。可组装在系统的任何位置，降低工作量。

（8）可移植性：软件从一个计算机系统或环境搬到另一个计算机系统或环境的难易程度。

（9）可追踪性：根据软件需求对软件设计、程序进行正向追踪，或根据软件设计、程序对软件需求的逆向追踪的能力。

（10）可互操作性：多个软件元素相互通信并协同完成任务的能力。 [2]

**2.3 存在的问题**

软件工程将其从业者视为遵循明确定义的解决问题的工程方法的个体。这些方法在各种软件工程书籍和研究论文中有详细说明，总是具有可预测性，精确性，降低风险和专业性的内涵。这种观点导致[由谁？]要求许可，认证和编纂知识体系作为传播工程知识和成熟领域的机制。软件开发人员已经提出软件工艺作为替代方案，强调软件开发人员自身的编码技能和责任，而没有专业性或任何规定的课程导致临时问题解决（工艺）没有工程（缺乏可预测性，精确，缺少风险缓解，方法是非正式的，定义不明确）。软件工艺宣言扩展了敏捷软件宣言[12]，并在现代软件开发和中世纪欧洲学徒模式之间进行了比喻。软件工程扩展了工程并利用工程模型，即工程过程，工程项目管理，工程要求，工程设计，工程构建和工程验证。这个概念是如此新颖以至于很少被人理解，并且被广泛误解，包括软件工程教科书，论文以及程序员和工匠的社区。软件工程的核心问题之一是其方法不够经验，因为对方法的现实验证通常不存在或非常有限，因此软件工程经常被误解为仅在“理论环境”中可行。今天软件开发中使用的许多概念的创始人Edsger Dijkstra在2002年去世之前拒绝了“软件工程”这一概念，认为这些术语与他称之为计算机科学的“激进新颖性”的术语相差甚远：许多这些现象都以“软件工程”的名义捆绑在一起。由于经济学被称为“悲惨的科学”，软件工程应该被称为“命中注定的纪律”，注定因为它甚至无法实现其目标，因为它的目标是自相矛盾的。当然，软件工程本身就是另一个有价值的原因，但这就是洗眼：如果你仔细阅读它的文献并分析它的奉献者实际做了什么，你会发现软件工程已经接受了它的章程“如果不能你如何编程” 。[13]

参考文献

[1] Leondes, Cornelius T. (2002). Intelligent Systems: Technology and Applications. CRC Press. p. I-6. ISBN 978-0-8493-1121-5. 1.4 Computers and a First Glimpse at Al (1940s)

[2] Campbell-Kelly, Martin (April 1982). "The Development of Computer Programming in Britain (1945 to 1955)". IEEE Annals of the History of Computing. 4 (2): 121–139. doi:10.1109/MAHC.1982.10016.

[3] Parnas, David (December 1972). "On the Criteria To Be Used in Decomposing Systems into Modules". Communications of the ACM. 15 (12): 1053–1058. doi:10.1145/361598.361623. Retrieved 2008-12-26.

[4] "The origin of "software engineering"". Retrieved 17 Nov 2017.

[5] Randall, Brian. "The 1968/69 NATO Software Engineering Reports". Retrieved 17 Nov 2017.

[6] Sommerville 2008, p. 26

[7] Peter, Naur; Randell, Brian (7–11 October 1968). Software Engineering: Report of a conference sponsored by the NATO Science Committee (PDF). Garmisch, Germany: Scientific Affairs Division, NATO. Retrieved 2008-12-26.

[8] Randell, Brian (10 August 2001). "The 1968/69 NATO Software Engineering Reports". Brian Randell's University Homepage. The School of the Computer Sciences, Newcastle University. Retrieved 2008-10-11. The idea for the first NATO Software Engineering Conference, and in particular that of adopting the then practically unknown term "software engineering" as its (deliberately provocative) title, I believe came originally from Professor Fritz Bauer.

[9] 2018 International Conference on Software Engineering celebrating its 40th anniversary, and 50 years of Software engineering. "ICSE 2018 - Plenary Sessions - Fred Brooks". Retrieved 9 Aug 2018.

[10] 2018 International Conference on Software Engineering celebrating its 40th anniversary, and 50 years of Software engineering. "ICSE 2018 - Plenary Sessions - Margaret Hamilton". Retrieved 9 Aug 2018.洪庆根,陈铁柱.本科教学质量工程的构建与实践[J].高等工程教育研究, 2006, 36(4):117-119.

[11] 陶华亭．《软件工程概论》：高等教育出版社，2007

[12] Beck, Kent; et al. (2001). "Manifesto for Agile Software Development". Agile Alliance. Retrieved 14 June 2010.

^13] Dijkstra, E. W. (1988). "On the cruelty of really teaching computing science". Retrieved 2014-01-10.

**附录1**

**一、文献综述**

文献综述是学生在开提前阅读过某一主题的文献后，经过理解、整理、融会贯通，综合分析和评价而组成的一种不同于研究论文的文体。文献综述的目的是反映某一课题的新水平、新动态、新技术和新发现。从其历史到现状，存在问题以及发展趋势等，都要进行全面的介绍和评论。在此基础上提出自己的见解，预测技术的发展趋势，为开题奠定良好的基础。

**二、文献综述的格式**

文献综述介绍与主题有关的详细资料、动态、进展、展望以及对以上方面的评述。因此文献综述的格式相对多样，但总的来说，一般都包含以下四部分：即前言、主题、总结和参考文献。撰写文献综述时可按这四部分拟写提纲，再根据提纲进行撰写工作。

**前言部分**，主要是说明写作的目的，介绍有关的概念及定义以及综述的范围，扼要说明有关主题的现状或争论焦点，使读者对全文要叙述的问题有一个初步的轮廓。

**主题部分**，是综述的主体，其写法多样，没有固定的格式。可按年代顺序综述，也可按不同的问题进行综述，还可按不同的观点进行比较综述。不管用那一种格式综述，都要将所搜集到的文献资料归纳、整理及分析比较，阐明有关主题的历史背景、现状和发展方向，以及对这些问题的评述。主题部分应特别注意代表性强、具有科学性和创造性的文献引用和评述。

**总结部分**，将全文主题进行扼要总结，提出自己的见解并对进一步的发展方向做出预测。

**参考文献，**因为它不仅表示对被引用文献作者的尊重及引用文献的依据，而且也为评审者审查提供查找线索。参考文献的编排应条目清楚，查找方便，内容准确无误。关于参考文献的使用方法，可以网上查询。

附录2

**2-3 能够运用网络等现代技术获取信息和文献资料，并通过文献研究来分析和总结解决软件领域复杂工程问题的可能途径**

**3-5 具备创新意识，能够在软件设计中发现创新点，并掌握基本的创新方法**

**7-1 理解软件领域复杂工程实践对外部环境以及社会可持续发展影响**

**12-1 能够自主进行文献检索和资料查询，及时获取和跟踪计算机领域的前沿技术和最新进展**