软件生存周期与速成原型方法

张维业　杨芙清　唐世谓

（北京大学计算机科学技术系）

摘　要

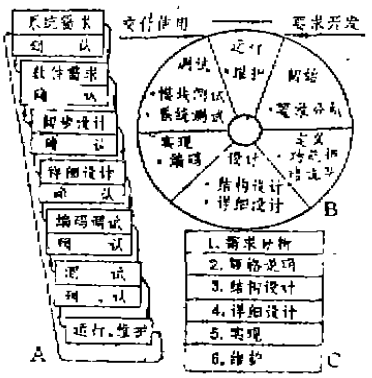
在采取软件工程方法进行大型软件开发过程中，传统思想正面临着一些难以解决的问题。本文通过分析这些问题，介绍了速成原型方法，并着重探讨了原型开发过程和速成原型方法对软件生存周期的影响，提出了采用速成原型方法的软件生存周期模型。

一、引　言

在计算机发展的短暂历史进程中，软件生存周期的提出已有相当一段时间。这一概念的建立是软件工程学的一次重要突破，它使得人们可以根据软件开发不同阶段的特点，研究和应用相应的技术、方法和工具，同时，软件开发技术与工具的迅速发展又促使软件生存周期模型作相应的修改。

传统的软件生存周期模型有过多种定义描述，其典型代表是所谓"瀑布模型"(图1A)。此外，美国国家标准局给出的模型（图1B）与Freeman等人的定义描述（图1C）都表达了类似的内容，它们的核心都是将软件开发划分为分析、设计、编码、测试及维护这种阶段性开发过程。随着软件开发技术与方法的进一步发展，目前又出现了一些新模型，如螺旋式模型（Spiral model）、花形模型（flower model）等等。

本文首先分析传统软件生存周期模型所面临的问题，结论是：规格说明的难以完善、需求变动和通讯中的误解与模糊性是进行严格线性开发难以克服的重大障碍。然后，针对这些问题，介绍速成原型方法的基本思想、运用方式、构造原型的要求、原型方法的优点与不足，着重讨论原型开发过程以及原型在软件生存周期中的作用，最后提出了采用原型方法的软件生存周期模型。



注：A——瀑布模型，B——美国国家标准局的模型　C——Freeman定义

图1　软件生存周期模型

二、传统思想及其面临的困难

在采用传统软件生存周期模型进行大型软件开发过程中，开发人员努力施行严格定义的方法，试图在每一活动过程结束后，通过严格的阶段性复审与确认，得到该阶段的一致、完整、正确和无二义性的良好文档资料，以"冻结"这些文档资料为该阶段结束标志，保持其不变，作为下一阶段活动的唯一

基础，从而形成一个理想的线性开发序列，以每一步的正确性和完整性来保证最终系统的质量。

传统思想之所以强调每一阶段的严格性。尤其是开发前期的良好的规格说明，主要是源于过去软件开发的经验：在开发阶段后期修正不完善的规格说明将花费巨大的代价。于是人们付出极大的努力来加强各阶段活动的严格性，特别是前期的系统分析阶段，希望得到完善的规格说明以减少后期不易估量的浪费。在传统思想中，人们一般认为：只要认真努力，总可以通过详尽分析，确定完整准确的规格说明，从而完全明确系统的各种需求；只要采取一套严格规定的术语及表达方式，就一定可以准确清楚地表达和通讯，这样便可以在严格、规范的开发管理下得到完美的结果。

然而，对于当前的大型软件项目，在开发前期用户常常对系统仅有一个模糊的想法，很难明确确定和表达对系统的全面要求，开发者对所要解决的问题更是模糊不清。经过详细的系统分析和定义，尽管可以得到一份较好的规格说明，但却很难期望该规格说明能将系统的一切都描述得完整、准确、一致并与实际环境相符，很难通过它在逻辑上推断出（而不是在实际运行中判断评价）系统的运行效果，并以此达到各类人员对系统的共同理解，其原因主要是缺乏知识：用户缺乏了解计算机系统可为其提供何种服务，解决何种问题的知识，开发人员缺乏用户专业领域的知识。因此难以预见系统的实现效果和软件系统将对用户工作环境的影响等等，从而难以保证规格说明能完整准确反映用户的真正意图。由于知识背景的不同、工作中的疏漏、和通讯媒介的局限性，通讯中的误解更无法避免。同时，随着项目向前推进，用户会产生新的要求或因环境变化希望系统也能随之变化。开发者也可能在设计开发过程中遇到某些未曾预料的实际困难，而希望在需求中有所权衡。总之，规格说明的难以完善、需求变动以及通讯中的误解与模糊性成为进行严格线性开发的重大障碍。尽管在传统软件生存周期中通过加强复审与确认、全面测试并设立维护阶段以减缓上述困难，但均未在根本上解决这些问题。

三、速成原型方法

在传统思想的支配下，人们花费极大努力进行严格开发，但终究难以接近理想目标，一切活动都掺杂着若干未能预料的疏漏。于是人们不再仅仅追求开发活动的极度严格性和准确性，而开始考虑传统思想中的一些基本观念是否应当改变。通过传统的系统分析方法，能够得到完整准确的规格说明吗？光凭细心和认真能避免通讯中的各种误解与疏漏吗？不完善的规格说明必然导致巨大浪费吗？

由于软件技术与工具的迅速发展，软件开发，特别是设计与实现的工作越来越方便，自动化程度越来越高，对系统的局部修改或局部重开发往往不再耗费极大代价。于是人们考虑由快速分析而迅速构造出一个小型软件系统，满足用户基本要求，使用户在试用过程中受其启发，逐步确定各种需求，从而产生了所谓速成原型方法。

1. 基本思想 通常，原型是指模拟某种产品的原始模型。在软件开发当中，原型是软件的一个早期可运行版本，它反映最终系统的部分重要特性。速成原型方法是利用原型辅助软件开发的一种新思想。它要求经过简单分析，快速实现软件系统的一个原型，用户同开发者等有关人员在试用原型的过程中加强通讯与反馈，通过反复评价和改进原型系统，逐步减少分析与交互中的误解，弥补遗漏，进一步确定各种需求细节，适应需求的变化，从而提高最终系统的质量。

2. 运用方式 虽然速成原型方法是在研究分析阶段的技术和方法中产生的，但它

更针对于传统思想所面临的困难，因而也面向软件开发的其它阶段及全过程。由于软件项目的特点和运用原型的目的不同，原型表现为不同的作用形式。在文献[1]中作者提出如下三种类型：（1）探索型（Exploratory Prototyping），目的是要弄清对目标系统的要求，确定所期望的特性，并探讨多种方案的可行性。它主要针对开发目标模糊，用户与开发者对项目都缺乏经验的情况。（2）实验型（Experimental prototyping），用于大规模开发和实现之前考核方案是否合适，规格说明是否可靠。（3）演化型（Evolutionary prototyping），其目的不在于改进规格说明，而是将系统建造得易于变化，在改进原型的过程中将原型演化成最终系统。它将原型方法的思想扩展到软件开发的全过程，适于满足需求的变动。

由于运用原型的目的和方式不同，在使用原型时亦应采取不同的策略。一种称为抛弃策略（throw-away），将原型用于传统软件生存周期的一个阶段，促使该阶段的开发结果更加完整、准确、一致、可靠。该阶段结束后，原型随之作废。另一种称为附加策略（Add-on），对应于演化型的开发形式，原型由最基本的核心开始，逐步附加新要求，反复修正，发展为最终系统[3]。

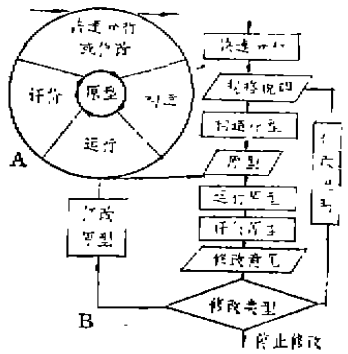
采用何种形式、何种策略运用速成原型方法主要取决于软件项目的特点、人员素质及可供支持的原型开发工具和技术等，要根据实际情况的特点加以决断。

3. 构造要求 原型不同于最终系统。它需要迅速实现，投入运行。因此必须注意在功能和性能上的取舍。既要忽略一切暂时不太关心的部分以加速原型的实现，又要充分体现原型的作用，满足评价原型的要求。根据构造原型的目的，要明确确定考核，评价原型的内容，如针对界面形式、系统结构、功能或模拟性能等。所构造的原型也有可能是一个忽略细节的整体系统，也可能仅是一个局部，如用户界面，部分程序系统，或数据库模式等等。总之，在运用原型方法进行开发之前，必须明确运用原型的目的，从而决定分析与构造内容的取舍。

4. 优点与不足 速成原型方法通过实际运行系统进行反馈，加强了通讯，为用户提供了一种直观方法来有效地评价系统，促使用户更加主动地参与项目开发活动，促进各类人员的密切合作，更加适应需求的变化并减少了误解，能有效地提高最终系统的质量，特别是用户接受性，为全系统良好地提供用户使用奠定了基础。但是这一方法目前还尚未成熟，缺乏丰富而且强有力的软件工具和开发环境，并缺少对这种开发方法进行有效管理的研究。

四、原型与软件生存周期

1. 原型开发过程 由于原型开发的形式和策略直接影响软件生存周期的模型，为此我们首先分析原型的开发和使用过程，作为讨论软件生存周期的基础。图2A给出了原型开发过程模型，图2B进一步描述了这一过程。



注：A——模型，B——使用

图2 原型开发过程

〈1〉快速分析 在分析人员与用户紧密配合下，迅速确定系统的基本需求。根据原型所要体现的特性（界面形式、处理功能、总体结构或模拟性能等）描述基本规格说明，以满足开发原型的需要，这里关键要注

意分析与描述内容的选取，围绕运用原型的目标，集中力量，确定局部的需求说明，从而尽快开始构造原型。

〈2〉构造原型　在快速分析的基础上，根据基本规格说明尽快实现一个可运行系统。这里要求具备强有力的软件工具支持，并忽略最终系统在某些细节上的要求，如安全性、坚固性，例外处理等等。主要考虑原型系统能够充分反映所要评价的特性，而暂时删略一切次要内容。例如：如果构造原型的目的在于确定系统的输入界面形式，可以借助输入界面的自动生成工具(如Form Generator)，由界面形式的描述和数据域的定义立即生成简单的输入模块，而暂时忽略有关后处理工作及参照检查、值域检查等内容，从而迅速提供用户使用。如果要利用原型确定系统的总体结构，可借助菜单生成器迅速实现系统的控制结构，忽略转贮、恢复等维护功能，用户通过运行菜单了解系统的总体结构。总之，在此阶段要求快速实现，投入运行。

〈3〉运行原型　这是进行高度通讯、发现问题、消除误解的重要阶段。由于原型忽略了很多内容，集中反映要评价的特性，外观看来不免残缺不全。用户要在开发人员的指导下运行原型，在使用过程中努力发现各种不合要求的部分。各类人员在共同运用原型的过程中进一步加深对系统的了解及相互之间的理解。

〈4〉评价原型　在运行试用原型的基础上，考核评价原型的特性，分析运行效果是否满足规格说明的要求以及规格说明的描述是否满足用户的愿望。纠正过去交互中的误解与分析中的错误，增添新的要求，并满足因环境变化或用户的新想法而引起的系统要求的变动，提出全面的修改意见。

〈5〉修改　根据评价原型的活动结果(修改意见)进行修改。若原型运行效果未满足规格说明的要求，说明对规格说明存在不一致的理解或实现方案不够合理，则根据明确要求迅速修改原型（相当于新的构造过程)。若规格说明不准确(有模糊性或未反映用户意图)，不完整（有疏漏)，不一致，或需求有所变动和增加，则修改并确定新的规格说明，重新构造或修改原型。

修改过程代替了初始的快速分析，从而形成原型开发的循环过程。用户与开发者在这种循环过程中不断接近系统的最终要求。

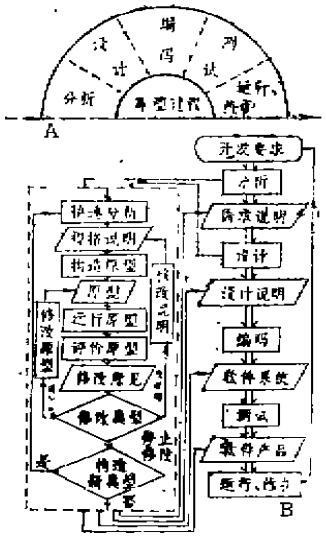
速成原型方法在基本思想上已突破了传统思想中的阶段划分，以上仅是为了说明表述的方便，将原型运用过程分段解释。在强有力的软件工具支持下，上述各种活动往往交融在一起，或合而为一或交叉进行。例如快速分析与构造原型有可能通过在大型软件库中选取功能部件及确定组合关系而同时进行。运行、评价和修改有可能在各类人员共同使用和随时交互过程中交织在一起，而不再象传统软件生存周期中那种严格的阶段划分，线性推进了。

2．采用速成原型方法的软件生存周期

速成原型方法的提出使得传统的软件生存周期在思想方法上受到了影响，但如果仅在局部运用原型方法，将原型过程用于传统周期的一个阶段内，那么传统软件生存周期模型依旧不变，只是阶段内部的软件开发活动采用了新方法。若原型过程代替了传统周期的多个阶段，软件开发过程则成为一种新的形式。

图3A表示运用原型方法的软件生存周期模型，图中原型过程处于核心，表示可在周围各阶段中引入原型方法，也可合并若干阶段，由原型过程代替。图3B详细描述了各种运用原型方法的软件开发过程。其中在原型过程末尾附加了"是否构造新类型"的判断，这是针对于在全系统开发当中有可能为不同目的而运用多个原型的情况。

〈1〉辅助或代替分析阶段　在分析阶段利用速成原型方法旨在获取良好的需求说明。在整体上依旧采取传统模式，努力达到严格的软件开发，把原型作为需求说明的补充形式或加细形式，通过



注：A——模型，B——开发过程

图3 采用速成原型方法的软件生存周期

运用原型尽可能使得需求说明完整、一致、准确和无二义性，从而代替了仅由复审和确认提高需求说明质量的传统方法。尽管在总体上依旧采用传统思想，但在阶段内却体现原型方法的思想，不再依赖纸面上的分析说明来确定完好的需求说明，而是给用户提供一个可运行系统，通过试用发现问题，从而确定用户需求，既高效又准确。

〈2〉辅助设计阶段 在设计阶段引入原型，对需求说明快速分析实现方案，构造原型。通过运行，考察设计方案的可行性与合理性。设计说明可因此得到完善。原型有可能成为设计的总体框架，或设计结果的一部分及设计文档的补充形式。

〈3〉代替分析与设计阶段 这种方式将速成原型方法运用到软件开发的整体过程上，突破了传统思想的严格阶段性开发要求，不再考虑完善的需求说明.分析、定义与设计交织在一起，通过原型的构造、评价与改进的循环过程向最终系统的全面要求接近。它在分析的同时也考虑到了设计实现要求，因而更有效地确定系统要求与设计说明。用户、分析员与设计者紧密配合，不再象以前那样仅以纸上的说明相联系，而是通过共同使用原型进行高效通讯与开发。在原型过程结束后，同时得到良好的需求说明与设计说明，原型系统有可能成为系统的总体结构，或作为系统雏形以供进一步开发实现。

〈4〉代替分析、设计与编码阶段 这种方式是在强有力的软件工具与环境支持下，通过原型过程的反复循环直接得到软件系统，交付系统测试。这已经属于演化型开发方法。由开始的基本原型出发，一直演化为软件的整体系统，从满足开始的基本需求一直扩展到满足系统的一切要求。

〈5〉代替全部开发阶段 这是最典型的演化型开发方法，它完全摆脱了传统的软件工程开发方式，放弃了传统软件生存周期模型，通过反复的原型过程，直接得到软件产品，交付运行。系统测试作为评价原型的一部分工作融入原型开发过程。它不再强调开发的严格阶段性与高质量的阶段性文档资料，不再追求理想的开发模式，而是在反复循环过程中，提高通讯效率，更便利地发现问题与解决问题。将系统构造得适于变化，满足需求变动和其它修改的需要，在循环前进中达到最终系统。

五、 小 结

传统软件开发方法中的严格线性开发思想面临着许多难以解决的问题，速成原型方法的提出对软件开发的很多方面都带来了重要影响。它突破了传统思想，引入了新的概念、新的观念，新的思想方法。利用原型作桥梁，在实际使用过程中加强通讯与反馈，从而减少误解与疏漏，适应需求的变动。同时，它对软件生存周期模型带来新的影响，把速成原型方法从引入分析阶段扩展到软件开发的其它阶段，直至全部软件开发过程，完全改变了线性开发方式，通过放宽阶段性严格要求而加强反馈来提高最终系统的质量，为提高软件开发效率开辟了新的途径，同时促进了有关工具与环境的深入研究和迅速发展。

参考文献

〔1〕 Christiane Floyed"A Systematic Look at Prototyping" Approaches to Prototyping 1984

〔2〕 Tannra Taylor, Thomas A. Standish "Initial Thoughts on Rapid Prototyping Technigues" ACM SIGSOFT SEN Vol. 7 No. 5 1982

〔3〕 Hans. E. Keus "Prototyping, a more

SRA软件开发环境与工具的研究

郝克刚　　乔广俊

（西北大学计算机科学系）

SRA (Software Research Associates,Inc. Company) 是日本一家有影响的软件公司，建立较早，研究范围广泛。目前开发的领域也多种多样。本文只介绍该公司在软件开发环境方面的研究工作。

为了提高软件生产效率和质量，SRA采用了集成化软件开发环境的概念，其特点如下：

应用广泛性——应是内部结构灵活的复杂的环境；

用户友好性——整个系统应使用户好学易用；

工具重用性——采用工具箱办法，包括一些模块化元素，以便需要时重用；

功能整体性——用合理方便的机制把单一功能的工具组成复杂结构处理各用户的特殊要求；

集中数据库——信息由公共数据库管理。

完全具有上述特点的理想环境尚不存在，SRA采用UNIX作为环境原型的基础，在VAX/UNIX上建立了适应于各类应用的软件工程环境。其中开发的一些公用软件工具，代表了在UNIX上多年积累的经验，可以提高软件生产率和可靠性。这些工具包括：

·交互式软件开发/维护工具　从1968年开始，SRA集中了从事各种应用软件开发和熟悉UNIX系统本身的专家，建立了一个内容丰富的程序设计环境，其中包括不被UNIX本身支持的软件需求定义、程序设计、测试和维护等软件生存周期的各个方面。

·用户界面软件　初级用户界面软件可以使没有多少经验的人迅速从UNIX提供的程序设计辅助设施中获益；普通用户界面软件，能使用户方便地利用那些被看作UNIX专业人员独有的特点。

·项目资料支持工具　文件管理工具用一个数据库对整个项目所产生的资料进行控制；更改控制工具用于管理源程序和测试数据的不同版本，也支持软件配置的控制。

·项目计划管理工具　如PEGASUS可以用图示手段描述项目计划和实际开发过程。

下面介绍SRA研究的环境中的主要工具。

reasonable approach to system development" ACM SIGSOFT SEN Vol 7 No.5 1982

[4] John. L. Connell "The Impact of Implementing a Rapid Prototyping on System Maintenance" AFIPS 1985

[5] Bernard H. Boar, Application Prototyping, John Wiley & Sons, Inc. 1984