国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第13卷第2期 1991年6月 Vol. 13 No. 2

软件开发自动化

博士论文摘要

殷建平 陈火旺 张克均
(电子计算机系)

摘 要 软件开发到底是一个什么过程？这个过程中的哪些工作能自动化？这是软件开发自动化的两个核心问题。本文以这两个问题为中心来展开讨论，试图提出一些见解以改进过去的研究工作。值得指出的是，基于这里的思想，我们已经实现了一个用于开发 Ada 软件的支持系统。实验表明：在该系统的支持下，开发者能以最少的干预半自动地实现从问题的自然语言描述到程序包规范的转换。

关键词 软件工程，软件开发自动化，人工智能，自然语言理解

分类号 TP311

过去，在研究软件开发自动化时，往往隐含地假定事先已有一个一致的、完全的初始规范，然后考虑从这个规范开始怎样开发出结果软件。但是，对这个规范是如何得来的以及这个规范是否真正反映了用户的需求却研究甚少。这样做的结果实际上是把软件开发的难点转移到了初始规范的书写上。然而，对一个大型工程，要一次得到满足这种要求的初始规范，将是相当困难的。另外即使得到了满足这种要求的初始规范，现有的研究成果也并不足以保证可从它自动生成结果软件。这一系列的困难和不足严重地阻碍了软件开发自动化的研究和应用。面对这样的困境，我们不得不反思过去所走过的路，不得不重新认识软件开发自动化中的两个核心问题：1. 软件开发到底是一个什么过程？2. 这个过程中的哪些工作能自动化？

1 软件开发过程

软件开发的起点无疑应该是用户给出的问题的自然语言描述 P0，其中用户列出他们期望的系统功能和系统性质。这样的描述通常是不简洁的、冗余的、多词同义的、一词多义的、不精确的、上下文相关的、不完全的，甚至是不一致的。另一方面，软件开发的终点一般是用某种具有良定义的文法和语义的编程语言来书写的可执行程序 Prog，用户希望这个程序应该‘真实地实现’或‘准确地翻译’描述 P0 所要求的系统功能和系统性质，并且，我们常常要求这个程序是简洁的、无冗余的、精确的、完全的、一致的、优

1991年3月15日收稿

化了的，甚至是可重用的。这样，一个最简单的软件开发模型可图示成图 1 所示。

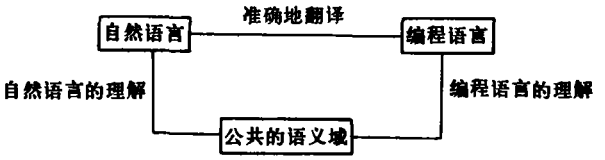

图 1　一个最简单的软件开发模型

也就是说，所谓一个自然语言句子被准确地翻译成一个编程语言句子是指：在公共的语义域上，对自然语言句子的理解与对编程语言句子的理解相同。过去，程序员凭经验和直觉为用户直接编写所需的程序就是这个模型的一个实例。那时，公共的语义域建立在程序员的头脑中，程序员凭着自己对自然语言的理解和对编程语言的理解来确认他所书写的程序是否准确地翻译了问题的自然语言描述。由于编程语言本身的特点决定了它必须涉及过多的实现细节，所以，它与自然语言之间的距离是遥远的，结果，由于负担过重，程序员的确认往往是靠不住的。这一点已被大量的事实所证明。

另一方面，由于‘自然语言的理解’一般说来难以机械化，所以，让机器来自动完成这个确认工作是不现实的。

既然这样，为了提高软件开发的可靠性，我们不得不把软件开发分解成一个由若干步组成的过程，每一步的输出构成下一步的输入。这时，开发者只需确认每一步的输出是否准确地翻译了这一步的输入。由于每一步仅仅执行一些简单明确的翻译工作，因此，这样的确认往往是十分明显的，结果有效地提高了软件开发的可靠性。至于开发步数，一般由具体问题的性质来确定。

为了记录软件开发的中间结果，规范语言这个概念便因此而产生了。规范语言的使命是在自然语言和编程语言之间架起一座自然的桥梁以便于客观地记录被开发系统的本质性质，如果设计的规范语言没有很好地完成这个使命，那么软件开发过程就在走弯路。过去，人们在研究软件开发过程时，往往偏向于首先提出一种极抽象的规范语言，然后考虑从这种语言到编程语言的变换。这样做实际上犯了‘本末倒置’的错误。因为软件开发的最终目的是为了得到一个满足用户要求的可执行程序，规范语言的引入只是为了达到这个目的的一种手段。事实上，正是由于这个错误导致了前面所指出的困难和不足。

基于上面的分析，一个改进了的软件开发模型可图示成图 2 所示。

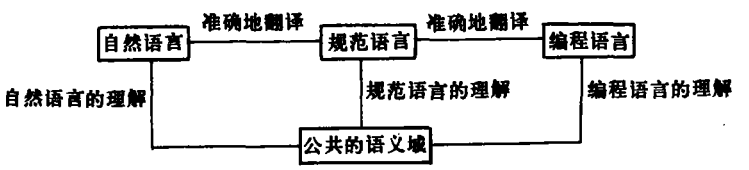

图 2　一个改进了的软件开发模型

这个模型实际上隐含地假定了问题的自然语言描述 P0 确实准确地反映了用户的需求。然而，正如前面所指出的：这样的描述通常是不简洁的、冗余的、多词同义的、一词多义的、不精确的、上下文相关的、不完全的，甚至是不一致的。这样，随着软件开发过程的进展，随着开发者对问题的认识的不断深入，问题描述本身也必然经历一个进化过程以便逐步消除描述 P0 中的上述不良性质。这样，软件开发的结果一方面是消除不良性质的进化了的问题描述 P_n，另一方面是它的实现 Prog，因此，一个真实的软件开发过程可图示成图 3 所示：

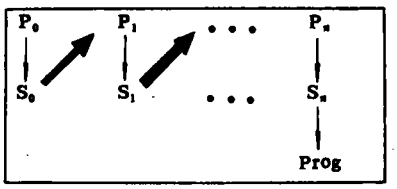

图 3　一个真实的软件开发过程

这里，P_i（0<=i<=n）是问题的自然语言描述，P0 为初始描述，P_{i+1}由 P_i 进化而来，P_n 是最后的描述；S_i（0<=i<=n）是由 Pi 导出的规范，在由 P_i 推导规范的过程中，若在 S_i 处发现由于 P_i 的某种不良性质而不能继续往后推导时，则进化 P_i 成 P_{i+1}以消除这种不良性质。Prog 是 S_n 的实现。

图 3 所示的软件开发过程可粗略地分成两个大的开发阶段。第一个阶段（P_0——→S_n）的主要任务是：1. 实现文法形式的翻译；2. 消除由于自然语言描述的不良性质所造成的影响。第二个阶段（S_n——→Prog）的主要任务是：1. 优化规范；2. 生成可执行的程序。对此，我们将在下面进行深入的讨论。

2　软件开发的主要任务

2.1　实现文法形式的翻译

语言之间的共通性建立了翻译的理论基础。这种共通性决定了语义相同的两个句子，在语法结构和词汇方面必定存在某种对应关系。具体来说，如果一个自然语言句子被准确地翻译成一个规范语言‘句子’，那末在这两个句子的文法形式之间必然存在某种对应关系。换句话说，如果一个规范语言描述相对于一个自然语言描述是有效的，那么这个自然语言描述中必定隐含着那个规范语言描述的‘影子’。一个好的规范语言描述应能尽量完整地保留这些‘影子’，这样做的目的是为了在没有给出规范语言描述的精确语义的情况下，仍然能够借助于对应的自然语言描述来理解它。

软件工程的实践表明：软件维护在整个软件生存期中占有极其重要的地位，而软件的理解是进行软件维护的前提和基础。遗憾的是，以往的各种方法并没把软件的开发同软件的理解有机地结合起来。应该说，这种状况是导致软件危机的一个主要的原因。难怪计算机大师 D. B. Knuth 教授热心于研究‘有文化的编程’(literate programming) 并声称“编程不再是指示计算机做什么，而是告诉人们要计算机做什么，程序的可理解性和表述风格的优美构成有文化的编程的最高准则。”其实，我们也遵循了相同的“最高准则”，文

法形式的翻译把易为人们所理解的自然语言和不易被人们理解的规范说明语言有机地联系起来，使形成的规范说明具有双重语义，结果，使软件的开发与软件的理解溶为一体，从根本上提高了规范说明的可读性和可理解性。

传统的翻译方法包括语法分析法，句型匹配法和语义分析法等，一般认为：翻译最好以语法、语义和环境相统一的最小翻译单元的‘最小语言模式’的对应为主，分析为辅，对应是机械的，可由系统自动完成。为此，必须建立相当数量的概括性很强的最小语言模式及其对应翻译；分析需要理解，往往必须依靠人工干预，甚至要求分析员按理解后的意思重新加以叙述。

2.2 消除由于自然语言描述的不良性质所造成的影响

(1) 不简洁

自然语言描述中往往包含一些反复出现的合成概念，由于不得不反复地定义它们，所以这时的自然语言描述通常是不简洁的。在由这样的自然语言描述推导规范的过程中，一般可采用‘折叠’的方法来解决问题。也就是说，先把合成概念定义成一个新概念，然后在所有出现该合成概念的地方，用这个新概念代之。

(2) 冗余

开发者常常会遇到这样情况：稍作分析后就能发现，用户列出的系统功能并不是相互独立的，也许有些功能实际上是另一些功能的特例，甚至有些功能并不是系统所必需的。另一种情况是：用户在描述一种系统功能时，很可能提及一些并不影响系统行为的概念。这时冗余现象便出现了。对此，一般可采用‘删除’的方法来解决问题。

(3) 多词同义

多词同义现象在自然语言中是相当普遍的。在这里，多词同义是指：从软件开发的角度来看，不同的词（概念）具有相同的指称。为了有效地控制这种现象在规范中的出现，一般可采用‘换名’的方法来消除不必要的概念。

(4) 一词多义

所谓一词多义，就是指：从软件开发的角度来看，相同的词的不同出现具有不同的指称。为了消除这种多义性，一般可采用‘代换’的方法将不同的指称区分开来。

但是，也应该看到：二义性的消除，多少都带有猜测或推测的色彩。更何况，由于自然语言本身的毛病，现实生活中大量存在连人也可能解决不了的二义性，即所谓真二义性。这时，除非上下文中具有对同一事实的非二义的描述，否则，人也无法理解其准确含义。总而言之，根据合乎上下文情理的理解，可得出合乎情理的翻译，然而，最后的确认者应该是客户而不是分析员。

(5) 不精确

用词不当很可能造成不精确的描述。这时，简单的‘换名’往往就能解决问题。更常见的情况与概念的丰富内涵有关。在由问题描述推导规范的过程中，开发者必须选取一个特定的角度来观察每个概念，必须经历一个分析、抽象、取省的过程以确定对软件的开发有意义的有关方面，然后用适当的形式精确地表示这些方面。这就是概念的定义。值得指出的是：软件开发作为一种以个人为中心的活动，不同的人，由于具有不同的知识背景或不同的观点，对同一概念的抽象和取省常常是不同的，结果可能得出完全不同

的模型。这是正常现象。一般说来，问题的自然语言描述往往定义了一个解空间，由于每个开发者对其中出现的不精确概念采取了一种特定的理解。所以他生成的总是上述解空间的一个特解，这是一种自然的思维过程。

在问题的自然语言描述中，一个概念的前后常常带上它的修饰语。这样，在精确定义概念之后，还必须考虑如何由修饰语推导出精确的概念限制（constraint）。这就是限制的定义。

另外，规范语言常常要求精确地说明参量的方式（mode），常量的类型和值，如此等等，这样的信息一般隐含在自然语言描述中，只能间接地分析出来。

（6）上下文相关

因为自然语言是上下文相关的，所以每个句子都可从其环境中获得大量隐含信息。另一方面，在实现文法形式的翻译时，显然只能获得文法上明确指出的信息。这样，在翻译之后还必须设法使环境信息明确化。一般来说，可通过‘参量的增加’和‘操作的换名’来获得这种效果。这时，新增参量名多半来自原问题描述或其它操作的现有参量名。

（7）不完全

第一种不完全性是由于用户在描述一种系统功能时遗漏了一些影响系统行为的重要内容所致。被遗漏的内容一般可通过‘参量的增加’和‘操作的换名’来获得，结果使不完全的规范趋于完全化。这时，新增参量名多半靠分析员输入。第二种不完全性是由于用户列出的系统功能还不足以使目标系统正常运行所致。这种情况往往只能在开发的后期经过深入分析之后才能发现。这时必然导致自然语言描述的进化和对应规范的扩充。第三种不完全性是由于用户常常没有考虑系统的异常情况所致。‘异常的建立’能使系统的异常情况明确地表示出来，结果使得异常情况的控制和处理成为可能。

（8）不一致

所谓不一致，就是指用户的要求前后矛盾，结果根本不可能开发一个满足用户要求的软件。这种不一致性有时很明确，有时却是隐含的，必须经过深入分析之后才能发现。不管怎样，一经发现，就要通过修改自然语言描述的有关部分来加以解决。

2.3 优化规范

规范的优化主要考虑两件工作：一是信息的隐蔽；二是规范的重用。这里就不详细展开了。

2.4 生成可执行的程序

这里的主要工作是：用具体的数据类型实现抽象的数据类型。如果事先已为每种抽象的数据类型提供了多种预选的实现，那么，这时开发者或系统便可根据实际情况灵活选取数据结构和算法以得到高效的实现。

3 软件开发自动化

为了回答软件开发的哪些工作能自动化这个问题，就必须深入分析机器和人的能力到底有多大，长短如何？在进行这种分析时，不可避免地将涉及到智能问题。关于什么是智能，这里把它区分成两类：一类是可机械化的智能，这种智能可借助机器来实现。这样它必定是基于逻辑的（当然，即使基于逻辑的也并不一定能机械化）。另一类是不可能

机械化的智能，这种智能是指一种判断能力，这种判断是合理的却是非逻辑的，其合理性以经验（一种非逻辑的知识）为基础，而从这种判断所产生的结果得到证实（而不是证明）。当然，这两类的划分是相对的，是可能随时间而变化的。也就是说，随着认识的不断深入，合理的东西可能上升为逻辑的东西，最后成为可机械化的东西。但是，隐含在这种相对性的背后的一种绝对的界确实存在，这一点在可计算性理论中早就作出了明确的回答。

按照图 3 指出的软件开发过程，软件开发自动化包含两个方面的内容：

1. 自动确定开发步；2. 自动执行开发步。

所谓自动确定开发步是指：通过对当前状态的分析，自动确定下一步该干什么。为此，就必须获取软件开发者的知识，研究软件开发专家们隐含采用的问题求解过程的理论模式。例如，有的开发者常采用以下的模式：

分析系统的功能→建立对应的操作→建立操作的参量→

定义参量的类型→定义类型的限制→定义限制的常量→

考虑规范的重用→考虑信息的隐蔽→生成可执行程序

确实，随着越来越多的软件开发经验上升为标准的软件开发模式，要建立一个专家系统来帮助软件开发并不是完全不可能的。但是，也应该看到：软件开发是一个多维的问题，而每个模式都有局限性，不可能在所有维上都获得成功。所以，对一个特定的问题，到底应该采用哪一种模式，这是一个高度智能化的问题，恐怕机器难以对此作出准确的判断。对所有问题都采用一种‘公共汽车式’的模式，这种宗教式的信仰是不现实的。举个例来说，一般认为：相似的问题可以采用相同的模式来开发。但是，现在的困难是：连什么叫做‘相似的问题’也难以给出一个定量的标准。这样看来似乎可得到如下的结论：根据已有的模式，机器能够展示可供选用的开发步，却不能准确地断言应该采用什么开发步。另一方面，开发者往往能凭自己的直觉来判断下一步该干什么，甚至发现新的开发模式。因此，现实的做法是：把开发者作为机器的扩充，在机器展示了可供选用的开发步后，开发者凭着自己的直觉来确定到底选取哪种开发步。

在确定开发步后，特定开发步的具体执行是发挥机器作用的有利时机。一般说来，在具体执行特定的开发步时，往往只对该步的输入的一小部分进行重写，而剩下的部分却丝毫不受影响。这种重写和复制工作可以而且应该由机器来完成，因为对这些机械活动，机器比人更可靠。这时，机器就象一个‘奴隶’，它的机械性工作减轻了开发者的负担，避免了由于粗心而引进的错误。结果，开发者便可把他的主要精力集中于对问题的实质性困难的研究以便形成正确的设计思想和开发路线。另外，正是由于在这方面能得到机器的帮助，‘分步开发’才能达到既可靠又不损失生产率的效果。

下面简要地讨论一下以下任务的自动化：1. 方法形式的翻译；2. 概念的定义；3. 程序的生成。

前面已经指出：语言之间的共通性决定了语义相同的两个句子，在语法结构和词汇方面必定存在某种对应关系。但是，语言之间的差异性决定了这种对应关系并非简单的一一对应。正是这种复杂性，决定了全自动地实现从自然语言到规范语言的翻译是相当困难的，甚至是不可能的。解决问题的一条途径是仅考虑自然语言的一个特定的子集，构

造对这个子集的翻译程序，但是，这样做时，由于所考虑的子集往往加了过多的限制，所以它实际上是不自然的。另一条途径是：人和机器共同来完成翻译工作。人主要负责作出一些高度智能化的决策来引导翻译过程，而机器主要完成复制和简单的文字处理。这样，人和机器取长补短，构成一个高效的翻译系统，有效地实现从自然语言到规范语言的翻译。

为了定义一个概念，常常必须获取用户的问题域中的知识，必须分析知识的深层结构。理想的做法是：通过事先建立一个特定域的知识库来提高自动化程度，但是，一个现实的困难是：要具体开发软件之前，一般并不知道开发过程中将需要哪些知识，结果难以给出一个准则来判断：哪些知识该保存在知识库中，哪些知识不必保存在知识库中。另一方面，即使对一个较小的领域，其领域知识也可能大得惊人，甚至是无限的。这样，要建立一个穷举其所有知识的知识库也是不可能的。知识库太大，不但浪费空间，而且还延长查找时间，也许还不一定有用；知识库太小，也许解决不了什么问题。既然难于预先确定知识库，那么，一种较为保险的做法是：当某个开发步确实必需某种领域知识时，开发者便将他所拥有的领域知识联机交互地注入到规范之中。这样便要求开发者必须拥有问题域的知识，因为很难设想：一位对问题域一无所知的人能开发出一个满足用户需求的程序，所以，对开发者提出这样的要求也是可以接受的。当然，开发者也并不一定是问题域的专家，这样，当需要某种他不具备的领域知识时，他也必然要向领域专家请教，经历一个学习过程。这似乎是现实可行的做法。

因为编程语言肯定具有操作语义，所以，如果设计的规范语言也具有操作语义，那么从规范语言到编程语言之间的翻译便可完全自动化。换句话说，程序的生成可完全自动地进行。这样，规范语言中选用的数据类型既要具有良好的数学性质以便开发者能借助常用的数学概念来构造问题的模型，又要具有操作语义以便自动生成结果程序。

4 结 语

本文仅仅概述了文[1]中的一些思想和方法。而完整的工作还包括规范说明语言的设计、支持环境的设计、实现与应用等。限于篇幅，这里没作具体介绍，有兴趣的读者请参考[1]。

参 考 文 献

[1] 殷建平. 基于变换的软件开发方法及其支持环境. 工学博士学位论文，国防科技大学研究生院，1990

Software Development Automation

Yin Jianping Chen Huowang Zhang Kejun

(Department of computer Science)

Abstract

What on earth is the software development process? Which tasks in this process can be automated? These are two key problems to software development automation. Taking them as the heart

of matter this paper sttempts to advance some ideas so as to improve the past researches . It is worth pointing out that on the basis of these ideas a support system to develop Ada programs has been implemented. Experiments have shown that under the support of the systen developers can semiautomatically implement the transformation from description of problems in natural language to package specification in Ada with the least interference.

key words software engineering, artificial intelligence, software development automation, natural language understanding

---

中国电子学会 1991 年学术活动预报（一）

	学术活动名称	时 间	地点	承办单位和联系人
1	第四届全国核医学电子学学术会议	1991 年下半年	安徽	北京核仪器厂 刘伟
2	第五届全国毫米波亚毫米波学术会议	1991 年 5 月	青岛	机电部 22 所 李开春、胡大璋
3	全国计算机辅助生产管理学术交流会	1991 年 8 月	北京	机电部自动化所 章以钧
4	先进制造系统的信息技术会议（ITAMS'90）	1991 年 9 月	南京	华东工学院 邓子琼
5	SMT、SMD 学术研讨会	1991 年 6 月	南京	生产技术专业学会 白经天
6	第二届电子技术应用学术会议	1992 年 5 月	待定	中国电子学会总部
7	第二届中国神经网络学术大会	1991 年 12 月	南京	中国电子学会总部
8	电子信息科学领域归国学者学术讨论会	1991 年 6～7 月	北京	中国电子学学术部 章笑南
9	第二届专用集成电路学术讨论会	1992 年	北京	中国电子学会总部
10	智能信息系统学术交流会	1992 年	北京	中国电子学会总部
11	全国第五届光纤通信学术会议	1991 年 2 季度	天津市	天津电子仪表局张朝仪 天津大学杨恩泽 天津光纤联合公司 伍继祥
12	第五届中国医药信息学大会 CMLA'91	1991 年 9 月	大连	辽宁卫生厅 王良骥
13	91 年全国薄膜学术讨论会	1991 年 9 月	北京	电子材料学学会 张龙光
14	第五届语言通讯、图象学术讨论会	1991 年 11 月	北京	中科院声学所李昌立、诸维明
15	91 年全国微波会议	1991 年 10 月	陕西三原县	空军导弹学院 侯勉
16	第五届全国微波能应用学术讨论会	1991 年 5/6 月	北京	机电部 12 所真空电子学会微波电子学学会任志详、黄伟嘉
17	第四届全国生物磁学会议	1991 年 9—10 月	上海	上海磁疗协会周顺义
18	第七届全国集成电路和硅材料学术年会	1991 年 10 月	成都	电子科技大学张波、陈星弼
19	91'国际雷达学术会议	1991 年 10 月	北京	中国电子学会总部 方敏
20	国际广播电视技术研讨会	1991 年 8 月	广东珠海市	广播电视学会 毛至攸
21	中国 1991 年国际电路与系统学术会议	1991 年 6 月	深圳	中科院电子所邹谋炎 深大电子系陈德源