人工智能用于软件原型化方法的探讨

黄 玲 （武汉大学计算机科学系）

马 江（武汉大学软件工程国家实验室）

摘 要

Applying the methods and techniques of artificial intelligence to software engineering is becoming a general trend. Can we use artificial intelligence to software prototyping? In this paper, we try to answer these questions and make some discussions for exchanging views.

一、引 言

传统生命周期方法的缺陷，促进了软件工程中原型化方法的发展。其宗旨是用较小的代价，较快的速度生成可供人们对需求和目标系统性能进行审定的系统模型或“例子”。但是由于软件原型化方法本身并不完善，加上涉及原型开发本身的开销，目前，利用原型化方法成功地开发实际产品系统的例子并不多见。与之相反，人工智能却在许多实际领域取得了成功，如：各种实用专家系统和专家系统工具的建造，并已逐步形成了较完整的方法和技术。那么，能否将人工智能用于软件原型化方法，加快软件原型的开发呢？本文在分析原型化方法的基础上，从人工智能程序设计方法和知识表示方法的角度，阐明了人工智能用于软件原型化方法的可能性，并作了一些探讨。

二、软件原型化方法

软件原型化方法是针对传统生命周期方法的缺陷而产生的，它采用演示原型的方式来启发，判断，揭示系统的需求。一般地，软件原型化方法可分为两类：抛弃式原型方法和进化式原型方法。

抛弃式原型方法，把目标放在原理的证明上，因此着眼点是快速开发一个演示系统，用于明确需求及检验系统设计的可行性。一旦用户认可，原型即被搁置一边，“真实”的系统按生命周期方法重新开发。在该方法中，代码的编写不考虑其可维护性或效率。

进化式原型方法，是通过增量式开发方法逐步达到产品的要求。这种方法比抛弃式原型方法用得少些，但对建立信息系统极为有用。由于信息系统的逻辑设计与物理实现之间存在差别，使得进化式原型方法成为可能。进化式原型方法强调系统逻辑设计的合

[5] Ramamorthy, C.V. "the Problems and Future of Software Engineering" IEEE Centennial The State of Comhuter, Oct.1984, Vol 17, No.10

[6] Shooman, M.L. "Software Engineering: Design, Reliability and Management" McGraw-Hill Inc.1983

[7] 孙振飞等，“软件工程概论”湖南科技出版社 1987.8

[8] 胥正辉，“软件开发原型化方法”《软件产业》1988.8

法性，而不是系统的物理实现。评估是基于系统的有效性而不是效率。（效率在原型变为真实系统或产品过程中被精化）。

目前，软件原型化方法主要用于以下几个方面：

①用于精化对系统要求的理解，帮助定义用户需求。

②用于评估多种设计的可行性。

③用于动态模拟目标系统的全部或部分功能，帮助识别复杂的可行子集。

三、人工智能用于软件原型化方法

下面我们着重从人工智能程序设计方法及其知识表示方法两个方面来阐明人工智能用于原型化方法的可能性。

3.1 人工智能程序设计方法

人工智能通常按照它模拟人类智能的目标来定义，理想地，人工智能程序应具有与人类推理能力相类似的功能。而与之相比，传统的计算机程序，则仅仅满足于加速一个定义好的计算的进行。

通常人工智能程序设计方法具有如下两个显著的特点：

①能够针对不断提出的要求容易加以修改，扩充。在传统的过程式程序设计方法中，许多构成解答的事实和结论赖以得到的处理过程都是包含在代码之中的。因而不能很好地适应变化的要求。

②能够应付"粗略说明"或"不完全知识"而产生的不完全结果。理想地，人工智能程序能够解决它前所未遇到的问题，甚至在不知道解决这一问题的确切处理步骤情况下。

上述的两个特点实际上同原型的探查实质是一致的，是人工智能程序设计方法中推理机与知识库相分离的结果。

目前，人工智能程序设计方法通常包括两部分：知识库和推理机。知识库包括推理规则，应用领域知识或特定应用过程；推理机则包含用于发现解答的查询方法及推理逻辑，它利用知识库中的信息来找出解答。理想地，推理机应独立于知识库，以使得程序的能力只需通过扩充或修改知识库的内容而增强。

人工智能中推理机与知识库分离的另一重要结果，是有机会用与嵌入过程式程序中过程分支的不同方法来表示"知识"。

3.2 人工智能知识表示方法

以往人工智能活动一直集中在对查询机制和推理机的研究上，前提是优良的查找机制可适用于各种知识库。然而近年来，研究者开始意识到，关键在于知识表达。现在人们已经普遍认为推理机的查找机制应根据用于组织知识的知识库的方法来进行设计、修改，即将重点放在知识表达问题上。

目前，人工智能中已发展了许多知识表达方法，包括谓词逻辑、规则、语义网、框架、过程式知识库等。逻辑程序是一些子句集合，它们真假逻辑地推出一个结论。规则对应于IF-THEN形式。过程是知识库包含一些处理说明和激活机构以及控制结构。语义网是一种面向语义的结构。其中节点代表对象、概念或情况，弧代表概念间的关系。框架则是关于数据和对应的处理信息模块，是一种更面向语义的结构。

虽然基于逻辑的表达方法严密且可验证，但它们使用起来比特定应用领域的语义网或框架更困难一些。例如对象间的多级关系需一组逻辑规则表示，精确定义这些规则并保证它们的内部相容性是困难的。相反语义网却可以简化对这同一关系的描述，用图表示，方便了设计者和用户，并且这种构造性方案更容易得到高度统一的推理机制。

3.3 人工智能用于软件原型化方法的探讨

人工智能程序设计方法的灵活性、易变性，以及能够针对不完全知识开发的特点和丰富的强有力的知识表示方法，使得它在以下几个方面能够用于软件原型化方法开发。

（1）利用人工智能程序设计方法中的推理机快速产生抛弃式原型，避免了重新开发原型所需的时间。实际上，推理机可看成

一个能产生许多原型的可重用工具。每个原型都有其自己的识识库，适应性由系统的知识库的表达提供。一个成熟的验证过的推理机无需针对原型的每次修改重新检验——原型的能力能容易地随知识库的改变而改变和增强。这样用户能很容易地通过修改知识库而得到满足自己需求的演示原型。对进化式原型而言，还必须具有在一段时间内能便于性能变化的结构。

原型应揭示出需进一步说明的问题和系统需求，它们使得大型系统在项目开始对问题了解很少或不够时，无需构造一个详细的程序组织结构，而是通过构造演示原型逐步获得。原型的这种探查实质使得它可针对一个不完全的知识库而建立。推理机利用现有知识库来确定一个解答的可能与否，这使得设计者将注意力集中在所希望的知识子集上而不是花时间去解释还未得到的知识。如果原型加速而容易，开发者／用户能通过修改和扩充知识库逐步完善原型，使得可能对难以理解的应用问题求出解答或获得用户的真正需求。

（2）利用人工智能中非过程式知识表达方法，如：规则、语义网、逻辑等组织知识，使说明和编码都更容易。此时知识库本身可成为系统的"说明"，从某种意义上讲，可理解为提供了应用系统的形式化文档(documentation)——抛弃式原型的结果形式。

人工智能中允许"跟踪"知识库而找出特定要求的解答，"跟踪"实际上是用于推理机推理过程的"自动文档化"的一种形式，可将它作为系统的形式化规格说明。这是因为：

(a) 它记录了结论是如何得到的。推理若是由人做或掩藏在代码之中，则通常被丢失。该记录对设计者、用户和系统维护者理解结论是非常必要的。

(b) 它可用来识别知识库中构成不正确结论的不完全或不正确信息。

(c) 当原型系统准备转化为产品时，若要将程序翻译成效率更高的过程式代码，它有助于定义过程逻辑。

（3）将人工智能知识表达方法与语义数据模型化技术相结合，增强数据模型的表达能力。语义数据模型化技术是从实体-关系模型[1]及Smith和Smith的工作发展来的。它类似于人工智能中语义网表示。目前这是一种较实用的支持信息系统原型开发的手段。它在以下几个方面支持软件原型化开发。

(a) 产生无二义性的模型及相应地能被开发者／用户理解的图表。

(b) 能描述广阔范围的详情，适合于支持全信息系统开发过程。

(c) 独立于任何特定的文件管理系统或DBMS，能用于推进任何环境下的实现。

(d) 它们对于开发模型及评价模型质量有详细说明和方法学。

语义数据模型化技术方法学的存在使得不是数据模型化专家的人们，也可能开发出好的原型。原型的内容相容性可得到验证。产生的原型可用来支持信息系统计划和开发。然而这种方法不足的地方在于语义表达能力方面，因此发展的方向应是更多地借用人工智能知识表达方法来扩充，增强数据模型的表达能力。

目前，一种面向对象的模型化技术正在迅速发展，它将人工智能中的框架、过程式知识库方法与语义网原理融合在一起。对象模型的一个优势在于它的外部说明与内部说明分开，从对象外面可见的属性以及通过向对象发一消息就可唤醒方法来描述一个概念模型。该模型不依赖于特定计算机环境中对象是如何实现的。这种概念模型与实际实现方法的独立性，使得对象模型技术对进化式原型开发很理想。

一个具有上述特点的面向对象的语义数据模型可具有如下特点：

(a) 面向对象的；

(b) 借用人工智能中的框架与对象描述说明性知识，对象、消息描述过程性知识；

(c) 具有较强的数据抽象与继承的能

力（基于AKO，APO，AMO，AIO等）；

(d) 支持复杂的约束及版本控制；

(e) 模型独立于具体开发方法。当结合具体软件开发方法的概念、规程、规则时，可描述软件开发过程性知识。

尽管在上述方面，人工智能用于软件原型化方法存在优点，但实际上将它用于原型化方法的确存在一些问题，这主要表现在性能和可预测方面。最为明显的不足是大多数人工智能程序比过程式代码慢，它要求特殊的机器且可能耗费太多的内存。利用推理机产生的原型可能依赖于硬件、软件工具及作图工具，而这些通常在一般程序设计环境下是难以得到的。所以人工智能机器一般常适合于建立抛弃式原型。即使产品系统可以从人工智能原型发展得到，但人工智能方法也可能不如适合原型那样适合产品系统，因为人工智能方法一个潜在困难在于它们的解答难以检验，这次得到的正确解答不能保证下次稍加修改后仍能得到一个正确解。况且，所得到的解答可能不是最优的结果。因此，将会发现产品系统的性能很不稳定，对同样的问题可能得到不同的解答。

另外，人工智能知识表达方法和程序设计方法并不普遍适用，它们表达可由过程式方法方便说明的系统会很低劣。用知识库定义详细情况和过程比常规说明定义更困难一些，对于那些存在一已知处理步骤的可以精确表达和形式化的问题，过程式程序比基于知识的人工智能程序效率高得多。

四、结 论

人工智能程序设计方法满足了开发一个好的原型的许多要求，主要表现为：

待定应用信息的知识库与查找解答的推理机的分离，它们使得修改变得灵活而容易，工作原型可快速开发出来，能处理不完全知识且可用于定义需求。

人工智能知识表达方法能较好地描述广泛的语义信息及方便而严密地组织知识，因此可将它与支持信息系统原型化方法的数据模型化技术相结合，扩充增强其对原型的描述、表达能力，更好地支持信息系统原型化的开发。

软件工程中吸收人工智能的成果，已经成为当前软件工程中的主要发展方向之一。本文就人工智能用于软件工程中原型化方法，做了一些探讨。我们希望这些探讨能够有益于软件原型化方法的完善和发展。

参 考 文 献

[1] Smith J.M.and Smith DCP, "Database abstractions: aggregation and generalisation" ACM transactions on Databae Systems vol.2 No.2 pp 105-133 (June 1977)

[2] D.C.INCE, "Software Prototyping progress and prospects" Information and Software Technology Vol 29, No.1, 1987

[3] MES Loomis and TPL Loomis,Prototyping and artificial inteligence" 1986

[4] 王俊彦等"论数据模型在软件开发环境中的作用和实现"，武汉大学学报 软件工程专刊，1988.6

[5] 何克清，"计算机软件工程学"，武汉大学出版社 1983年