计算机科学1990№.5

面向对象并发程序设计导引

Akinori Yonezawa和Mario Tokoro

面向对象的并发程序设计是一种程序设计方法学，也是一种设计方法学。在这种程序设计中，待建立的系统被模拟为一个称之为对象的可并发执行程序模块的集合，它们之间的相互作用靠发送消息来进行。本文系《面向对象的并发程序设计》文集的卷头文章，虽然重点是介绍有关这一方法学的当前工作，但有一定的普遍意义。

背景

面向对象的并发程序设计这一技术的研究出于我们需要设计和建造强有力的，而且是灵活的计算机软件系统，以满足社会需求的不断增长。随着信息化社会的发展，计算机系统应能解决更复杂的问题并提供更高级的服务。尽管以适度开销，生产更快更大的机器的技术可能继续向前发展，但直接使用这种机器不会产生能满足我们所要求的计算机系统。我们需要的是并行工作，使得我们能充分利用由多台计算机组成的大量计算结点并使它们合作运行。

并行性

从概念上来说，并行工作是吸引人的，但有效地利用并行性并非易事。在计算机科学传统分支，如操作系统设计和分布式数据库中，已经发展了控制并发性的许多有用的技术。然而，这些技术对建立所要求的系统是不够的，因为待构造的这些系统的构件之间需要更广泛的各种相互作用和更高的并发程度。

实现并行性的中心问题是什么活动和谁的活动应当并行执行，以及这些并发活动应当怎样相互作用。在设计一个体现并行性的软件系统中，这些问题归结为应如何将该系统分解为可并行活动的构件的问题，以及如何划分构件的功能的问题。为了保持系统的透明性，分解应该是自然的和模块化的。那么，每个构件应当以什么样的风格或形式表示呢？正如本卷研究报告中提出并作简要讨论的一样，“对象”的概念为我们提供了一种非常有希望的形式。

对象

“对象”这一术语几乎独立地在计算机科学的各个领域中出现过。早在70年代，差不多同时涉及到了一些表面上看来不同，但相互

9. 朱海滨等，“Smalltalk-80虚拟机在VAX/VMS上的实现”，小型微型计算机系统(10)，1989。
10. 朱海滨，“基于Smalltalk语言的面向对象程序设计”，微小型计算机开发与应用（6），1989。
11. 朱海滨、陈火旺，“从Smalltalk语言的结构看‘软插件’的形成”，计算机科学（3），1990。
12. 朱海滨，“Smalltalk语言虚拟象的设计与实现”，第二届系统软件研讨会，哈尔滨，7，1989。
13. 朱海滨、徐锡山，“面向对象的程序设计——方法、语言及实现”，计算机工程与科学(3)，1990。
14. 朱海滨、胡守仁，“面向对象计算机研究中的几个问题”，计算机工程与科学(3)，1990。

有联系的概念。所有这些概念旨在用对象来表示按模块化方式分解的系统构件或知识表示的模块单元以便管理复杂的软件系统。已出现在各个领域内的对象的典型概念有：

- 在并行计算模型和AI程序设计中以响应消息的方式执行动作的计算结点（agents），即Hewitt的角色（actors）；
- 带有类/实例和超类/子类分层的信息包，诸如Simula和Smalltalk中的对象，模拟中的对象，AI和更一般的程序设计中的对象；
- 程序设计语言中的抽象数据类型；
- 在知识表示中知识和专家知识的模块或单元，如Minsky的框架；以及
- 操作系统中受保护的资源。

这些概念的共同特性为对象是一个逻辑或物理实体，它们是自封闭的，并配有一个统一的通信协议。在面向对象的框架中，计算和信息处理表示为对象之间传递的消息序列。由于对象是具有统一的通信协议的自封闭实体，因此，将一个系统分解成一个对象的集合是非常灵活的，最后产生的系统结构变得非常自然。此外，统一的通信协议和"自封闭"又保护对象防止非法存取。这样就支持了对象之间有规则的相互作用。这些优点向我们提供了一种非常完备的并发程序设计基础。所以，我们相信，面向对象的并发程序设计是实现并行的一种强有力的程序设计/设计方法学，在这种程序设计中，待建立的系统表示为一个并发可执行对象的集合，而系统各部分之间的相互作用由消息传递来表示。

用对象进行模拟

我们周围有各种系统，这些系统各部分的并发和合作活动是实现系统目标所必需的。人类社会，团体组织，工厂系统，人类认知系统，计算机，以及通信网络等等，是这类系统的典型实例。借助对象和对象之间的消息传递模拟这些系统常常是相当容易和自然的。因此，我们设计和实现软件系统的方法学可以直接借鉴这些系统中所出现的并行解题模式。而且这种借鉴给我们发明更复杂的并行解题模式奠定了基础。

计算模型

在讨论面向对象的并发程序设计时，必然要涉及到若干问题。下面我们简略地提出以下一些主要问题的四个重点方面：计算模型、描述语言/系统、体系结构和应用。

这种计算模型实质上与各种对象概念有关。由于每种对象概念具有自封闭和统一的通信协议这样的共同特性，因而经常是从几个侧面来刻画每一种对象的概念：一个对象通过一条消息的到达被激活，而在一连串的动作之后又变成不活动的对象，或者一个对象可以总是在活动并通过执行某些命令接受消息。在命令、函数或逻辑方式中，什么样的内部对象动作是允许的？而又怎样刻划它们呢？每一个动作都应当借助消息传递来表示吗？计算所涉及到的所有实体都应当是对象吗？在一个对象内部一个以上的动作串可以同时发生吗？

消息发送可以是点-点通信，也允许是广播通信。对象之间的消息传递可以是同步的，也可以是异步的。消息发送次序既可以按消息到达的次序进行，也可以不按消息到达的次序进行。什么样的信息可以封闭在消息中？应当提供何种同步机制？消息是公平处理的吗？或者某些消息可以不处理吗？是否要作有关时间或状态的相对性的假设？

在一个计算模型中，如何选择这些特性取决于所讨论的何种信息处理模式，以及用这种计算模型框架设计和实现什么样的应用系统。为了严格地说明这样一种计算模型，我们需要进行数学处理。建立这种数学基础不是无足轻重的，因为面向对象的计算的动态特性（即对象的动态生成和内部对象连接拓扑结构的动态变化）要求我们发展新的数学方法。

语言和系统

面向对象的并发程序设计语言应当提供一些语言结构，这些语言结构应尽可能直接

地针对系统构件的行为表达程序员/设计者的直觉，并且还应当能表示更为精细的算法细节。此外，它们应能有助于构造健壮而又具有良好结构的软件系统。为此，必须考虑能表示诸如类-子类分层，继承、类型/类说明和推理，建立在对象之上的模块概念、内部模块/对象名作用域的控制、错误处理等语言结构。与领域有关的特性，象实时控制和表处理功能需要结合到各种不同的语言中，以便写出各种应用程序。

在高质量的软件产品中，语言结构不是唯一的因素。采用我们的方法学进行实际的程序设计应当在良好的程序环境中进行。这种环境为程序设计和调试提供了与各个对象的当前状态（和整个系统）有关的相应信息。由于我们要处理一种复杂的现象，即"并行"，因此，我们得为各个并行计算模型设计完整的调试方案，并为这些方案设计良好的接口，因为在开发程序时有效地显示调试信息是必需的。

体系结构

实现一个真正好的程序设计和运行环境只有靠良好的计算机体系结构的支持。面向对象的计算实质上是综合了动态对象的生成，对象的约束，消息缓冲，方法搜索和单元回收等，其中每一个都必须实时执行。传统的体系结构能够有效地执行常规的面向过程语言的编译代码。虽然编译技术已被用来加速执行传统体系结构上的面向对象程序，但要求有一种新的计算机体系结构能支持面向对象计算的动态特性仍然十分强烈。

在面向对象的并发计算中，每一个对象和其它对象一起并发运行。因此，对单一处理器的执行来说，频繁的上下文切换是不可避免的。减少并发执行上下文次数的编译程序和运行时的技术是重要的研究课题。而且，结构上对有效的上下文切换的支持也是不可缺少的。

然而，最根本的方法看来是在一个并行/分布的体系结构上并行地执行一个并发程序。由于对象是一个具有统一通信协议的自封闭的实体，所以面向对象的模型自然适应于由通信子系统互连的一组处理器-存储器对所组成的分布计算系统。这种解决方法是相当吸引人的，但仍然存在一些研究问题：我们怎样将一组合作对象分解成一些子集，并让每一个子集常驻在各个不同的处理器中？一个对象应当在哪一个处理器上生成以及一个对象如何从一个处理器搬迁到另一个处理器？分布的单元怎样才能有效地回收？系统的性能和处理器-存储器对的数量相称吗？此外，当一个应用系统变得很大，而基础的处理系统也相应变得很大时，会带来系统可靠性和弹性问题。

应用

由于对象能够直接了当地表示概念/物理实体，我们的方法学足以支持在各种不同的抽象层次上模拟各种并行系统：从抽象设计描述到生成可执行程序。具有实时特性的基于对象的并发程序设计语言适用于编写各种应用领域的程序，如音乐合成，动画片的合成，办公信息系统，工厂自动化系统，等等。甚至在执行速度是关键的应用领域中，我们的方法学对于开发结构良好的原型系统也是特别重要的。

基于我们的方法学的设计说明为构造带有并发性和实时性的可靠软件系统提供了一个坚实的基础。这种设计说明还能用于可执行的规格说明，此种方法在设计诸如操作系统，办公信息系统和实时控制系统等这类系统中是特别有前途的。

我们的方法学的主要应用领域之一是分布式人工智能（DAI）。DAI所考虑的是通过一组非集中和松散耦合的知识源进行合作解题。事实上，对象是具有计算功能和通信能力的知识源。正如前面提到的，借鉴我们周围的组织和系统中看到的各种解题模式对于构造智能系统是有用的，而这种借鉴可直接模拟为对象的并发活动。所以，我们的方法学对AI的并行工作做出了实质性的贡献。

关于当前的研究

面向对象的并发程序设计相对说来是一个新的而且正在迅速发展的方向。应当应用，试验并把这种方法学推广到各个领域，如人工智能、软件工程、系统程序设计、办公信息系统、过程控制系统、模拟等方面。我们相信，目前技术的推广和进一步的研究成果将为未来计算机系统奠定重要的软件基础。

下面我们简要地介绍一下面向对象的并发程序设计方面的研究工作，其中包括与方法学有关的基本问题的讨论，几个程序设计语言和应用。

在"Act1 并发的面向对象的程序设计"中，Henry Lieberman解释了角色（Actor）模型的原理，该模型将程序组织成活动对象的集合，对象之间通过消息传递以并行方式通信，并且还讨论了程序设计语言Act1。该语言把二个特殊的角色结合在一起，它们是提高并发性的定单（角色）和限制并行性的串行化（角色）。

Actor模型的创造者 Carl Hewitt 和 Gul Agha在"利用 Actor 模型进行并发程序设计"一文中指出，"具有局部状态变化，动态可重构性和固有并发性的共享对象模型的能力是任何并发模型所希望的性质。他们还介绍了将面向对象的程序设计和函数式程序设计两者优点结合在一起的角色模型的最新发展情况。

Akinori Yonezawa 和他的小组在"用面向对象的并发语言 ABCL/1 建模和编程"一文中介绍了一个面向对象的计算模型，设计这一模型是为了对大量各种并发系统进行建模和描述，此外，他们还详细说明了一个基于他们的计算模型，采用 ABCL/1语言的分布式解题模式。

在论文"ABCL/1分布式计算"中，Etsuya Shibayama和Akinori Yonezawa 介绍了分布计算中典型问题的算法，如 ABCL/1中的离散事件模拟和并发存取控制。它们的技术是以重算（roll-back），封锁和流水线为基础的。

Yasuhiko Yokote 和 Mario Tokoro在"Concurrent Smalltalk 的并发程序设计"一文中将并发结构，同步机制和原子对象运用到Smalltalk-80 中，他们给出各种实例程序来说明一种用内部对象同步来进行并发程序设计的新风格。

Yutaka Ishikawa和Mario Tokoro在论文"Orient84/k：一个用于知识表示的面向对象的并发程序设计语言"中提出了一种表示并发对象知识的模型。他们还介绍了一种并发程序设计语言/系统，在对象框架中提供面向对象的，基于逻辑的，以及面向精灵（常驻内存的系统进程，译者注）的程序设计范例，并用溢出一临界（spill-crisis）专家系统加以说明。

在文章"*POOL-T*：一个并行面向对象的语言"中，Pierre America设计了一种描述基于并行体系结构的更大系统面向对象的并行语言。根据他对面向对象的程序设计，模块化程序设计和抽象数据类型的含义及其相互之间差异的研究，他引入了并发机制和类型，但没有继承机制。

Pierre Cointe, Jeam-Pierre Briot 和 Bernard Serpette在"表格系统：面向对象的程序设计在音乐方面的应用"一文中指出，面向对象这一比喻与应用计算机进行音乐作曲和合成这一需求的一个子集合相吻合。他们介绍了在 IRCAM 上进行实际音乐作曲和合成的面向对象的语言系统的运行情况。

在论文"Omega 的并发策略执行"中，Giuseppe Attardi简要地说明了基于描述的知识表示系统 Omega，并讨论了如何使用 Omega进行描述的并发推导过程。

〔赵敦琢译自 Object-Oriented Concurrent Programming, MIT Press, 1987, pp.1～7。曹 华、诸 忠校〕