一个集成化的软件工程支撑环境

杨芙清 方裕 唐世渭 杨冬青

(北京大学)

AN INTEGRATED SOFTWARE ENGINEERING SUPPORT ENVIRONMENT

Yang Fuqing, Fang Yu, Tang Shiwei and Yang Dongqing

(*Beijing University*)

ABSTRACT

Integration is one of the essential problems in software engineering support environment construction. It involves concepts, methodologies, techniques and tools. The integration has its logical meaning and physical attributes. In this paper, the architecture of an integrated software engineering support environment hosted by UNIX system was introduced. In this environment, the information base and the user-interface were designed as its main integration mechanisms. It can support object-oriented programming. This environment offers a series of base languages based on C and can assist different users in software development and project management. The whole environment was divided into the environment layer, the object layer and the data layer so that it was organized as a layered virtual machine. Therefore, this implementation makes the environment robust, open-ended and tailorable.

摘 要

软件工程支撑环境的集成化问题是构造环境的中心环节，它涉及到概念、方法、技术和工具等方面，有其内在的逻辑含义和物理表征。本文介绍了一个在UNIX 系统上构造的集成化的软件工程支撑环境。该环境以环境库和用户界面为其主要的集成机制，支持面向对象的程序设计方法。它提供了一个基本语言序列，可以支持不同层次的用户从事软件开发活动和项目管理活动。在实现上，采用分层虚机器的构造方法，从而保证了系统的坚固性、开放性和可剪裁性。

1989 年6 月15 日收到，1989 年12 月2 日定稿。

软件工程支撑环境是指在基本硬件和宿主软件的基础上，为支持系统软件和应用软件的开发和维护而设置的一个软件系统。大型软件系统的开发是一个相当复杂的过程，现代软件工程的理论研究和实践为软件开发和维护提出了一套颇有指导意义的原则、方法和技术。但是，由于这些技术和方法本身的复杂性以及实施时可能引起的管理方面的实际困难，使得它们迟迟不能在实践中发挥应有的作用。在这种情况下，软件工程支撑环境的出现是必然的归宿。

过去十余年间，有关软件工具、软件工程支撑环境的研究是一个相当活跃的领域。大量的软件工具以及不少软件工程支撑环境的出现对软件工程实践活动起到了很大的推动作用。尽管现有的软件工具在功能和性能上还存在种种缺陷，但它们基本上覆盖了软件生命周期的各个阶段，可以支持或部分支持软件开发和维护的各项活动，并逐渐开始为广大软件技术人员接受和采用。

但是，大量的软件工具都是独立研制的，彼此之间没有联系，使用方式差异很大，甚至互不兼容。现有的软件工程支撑环境大多不能连续地、系统地支持整个软件开发和维护的过程。因此，如何构造一个能够支持软件生命全周期的、适合不同层次软件技术人员使用的、支持包括技术活动和管理活动等软件生产各方面活动的集成化的软件工程支撑环境，成为一个重要的研究课题。本文提出的集成化软件工程支撑环境，正是试图将环境中的各种工具和辅助设施有机地结合起来，向软件技术人员提供协调的、连续的和使用方便的各种支持，从"物质上"更好地帮助软件技术人员使用复杂的软件工程方法和技术，从而更好地完成软件开发和维护任务。另一方面，也可以"强制地"使某些尚不习惯于使用软件工程方法和技术的人员使用这些技术和方法来从事开发和维护工作。

§1. 支撑环境的集成化问题

任何大型软件系统的研制都存在着集成问题。支撑环境的集成化问题涉及概念、方法、技术和工具等方面。这里，概念集成是环境集成化思想的基础。提供一个统一的概念框架，以此为基础集中支持一个软件开发方法学，研究所涉及到的软件技术和设施，开发相应的工具和辅助设施并将它们有效地组织起来，就构成了集成的主要内容。为了使构筑的环境更有效，也需要对宿主软件做适当的改造。此外，构造一个统一的用户界面，使环境所提供的各类服务在用户面前更加友善。所有这一切，就是集成化软件工程支撑环境研制的全部内容。

1.1 集成化的逻辑表征

(1) 概念集成

在我们的环境中，以一个统一的概念框架来支配环境的各项设施和活动，这就是面向对象。它要求环境的用户(包括工具研制者)在考虑问题求解时，面向客观世界的问题领域进行自然分割，以更接近于人类思维的方式建立问题模型，把问题抽象和计算能力抽象更有机地结合起来，设计尽可能直接体现问题求解的软件。

面向对象是结构化思想的更高一级体现。如果说结构化程序设计是面向机器求解模型的话，面向对象的设计方法则是面向问题认识模型的。

(2) 软件工程开发方法的集成

软件工程开发方法的集成有两个方面的含义。首先，环境应支持软件生命周期的各个阶段，这自然涉及到与项目有关的各类人员的活动，不仅要支持程序员和主程序员，

也应该支持系统分析员和项目负责人；不仅要支持技术活动，也应该支持管理活动。其次，环境应在统一框架的基础上，支持各种软件开发方法和维护方法。作为面向对象的自然结果，我们的环境以结构化方法为基本前提，支持各种与此相关的开发方法。

(3) 软件技术的集成

集成化软件工程支撑环境的研制涉及到几乎所有的软件技术，例如程序设计语言技术、操作系统技术、数据库技术、图形技术、人机界面技术、网络技术、人工智能技术等。只有这样才能支持环境辅助设施和和工具的研制以及环境在不同硬件系统上的配置。由于我们的环境旨在支持传统软件的开发和维护，并在集中式计算机系统上配置，因此，除了网络技术以外，集成了几乎所有的现代软件技术。

(4) 工具的集成

作为方法和技术的有形体现，工具的集成是逻辑集成的最低层。这里，它涉及到建立工具之间的联系、构造工具之间的通讯机制、解决使用工具时的并发控制和将不同工具组织起来连续地支持软件项目开发等。在可能的情况下，还要考虑组成工具的更小单位—工具片段，以解决工具的重组问题。在我们的环境里，工具和工具片段都作为对象，数据也是对象，问题就归结为对象之间抽象关系的处理和共享。

1.2 集成化的物理表征

(1) 统一的用户观点

环境中的所有实体都是对象，用户本身也是对象。不同层次的用户只是对象的类型不同，概念上并无实质性差别，它们由环境统一组织起来，赋以不同的权限和不同的活动范围，协调一致地进行工作。另一方面，用户又遵循一种一致的方式来存取系统中的成分，按照统一的方式来开发软件，精心组织的活动管理使得用户对环境的认识变得一致、简单，从而大大简化了项目管理的复杂程度。

(2) 统一的数据存放方式

环境中的所有对象都以一种共同的内部表示存放在库中，环境对库中存放的数据有一致的解释方法和管理手段。数据和解释机制的结合满足了信息的使用要求，使数据库上升到了信息库。不仅如此，在不断增添解释机制的基础上又赋以特定的使用规则，这些使用规则体现了软件工程实践的实际含义，使信息库与软件工程支撑环境的实质内容成为一体，称之为环境库。

(3) 统一的操作模式

统一的操作模式指的是友善的用户界面和一致的操作方法。环境配有图形辅助的用户界面和合理的求助机制。环境提供了一组风格类似的、以C 语言为基础的基本语言，供工具开发者、项目开发者和环境管理人员使用，达到了内容和形式上的统一。

§2 环境设计

2.1 基本模型

环境的基本模型如图1 所示。

<table>
<tr><td colspan="2">开发方法和活动管理</td><td rowspan="5">用户界面</td></tr>
<tr><td colspan="2">用户组织和视图组织</td></tr>
<tr><td>存储管理</td><td>类型管理系统</td></tr>
<tr><td>系统</td><td>配置管理系统</td></tr>
<tr><td colspan="2">UNIX 系 统</td></tr>
</table>

图1 基本模型

环境以UNIX 为宿主软件，在Micro-VAX / II 和SUN-3 / 160 上运行。环境以用户界面和环境库为主要的集成机制，软件工具和开发项目本身作为集成的对象。在这个意义下，整个环境就由宿主软件以及配有良好的用户界面、装入各种工具的环境库系统所组成。其中环境库包括存储管理系统、配置管理系统、类型管理系统、用户组织和视图组织设施、开发方法和活动管理设施组成，体现了概念、方法、技术和工具的集成。用户界面则是内在的集成机制的外部体现。

2.2 开发方法和活动管理设施

开发方法和活动管理设施的目标在于：提供一种组织和控制的手段，支持软件技术人员组织软件开发活动以及接纳各种工具。软件开发的特点之一是分阶段按步骤进行，每一阶段或步骤完成一个特定的任务，这样一个完成特定任务的过程称为一个活动。显然，任何活动都是遵循一定的开发方法的，可以用一个规程来加以描述，例如，通常的编码活动可以按照编辑、编译、连接、排错这样一个模式来进行，所以规程可以看作活动的抽象描述，并为活动提供了行为规范，而活动则是某个规程在特定目标上的一次实施。这一部分设置了规程定义语言MDL 和活动管理系统。MDL 提供了定义规程的手段，活动管理系统则将某个规程赋予一个活动并控制其执行。图2 表示了一个规程从定义到赋予一个活动并实施的过程。

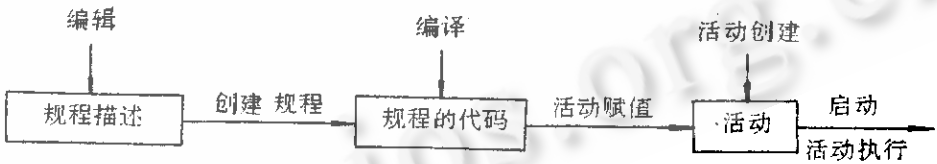

图2 规程描述到活动执行的过程

活动管理系统设置一个活动库，用以标识、记录、分配每一个活动并控制其运行。

这一部分的另一个主要成分是工具定义语言OECL，供工具研制者使用。工具研制者可以用OECL 编写工具并将其存入环境库中。OECL 以C 语言为基础加以扩充，并提供了在已定义的类型和工具上组合新工具的手段。OECL 语言定义的工具作为分程序的扩充，直接将工具和对象联系起来，不再引入类型的概念。

2.3 用户组织和视图组织设施

用户组织和视图组织的目标在于：组织用户，提供工作台的概念，以使合理地管理软件开发项目，有效地控制任务分配、权限赋予和资源共享，加强环境的安全性。此外，

为了反映和控制数据层模型的某些局部特征，提供了视图的概念，并为此向用户提供对象视图定义语言OVDL。

所有利用环境进行工作的人员(环境控制员、项目负责人、开发组长、程序员) 都是环境的用户，他们按树型结构组织起来，如图3 所示。图上每一个结点都是一个工作台，上级用户可以创建和注销下级用户，并给他们分配权限和传递消息。

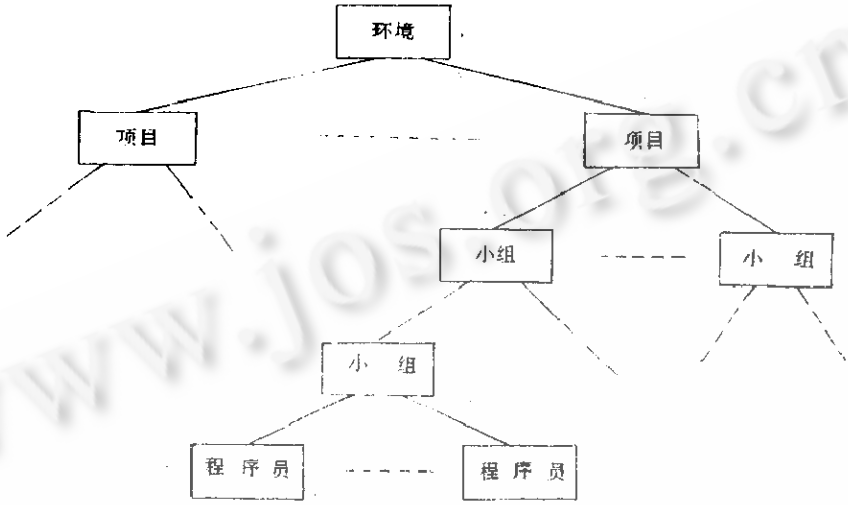

图3 用户组织

视图是对象实体的全部(或部分) 特性在授权方式下的表现和再塑造。视图是在类型上定义的。OVDL 规定了视图的定义规则，视图可以嵌套定义，即在视图上可以定义视图。显然，OVDL 为OECL 提供了对软件实体的语用约束。

2.4 类型管理系统

对象是集成化软件工程支撑环境中的基本成分和独立实体，它刻划了问题领域内实体的基本特征，同时又体现了该实体在计算机环境中的处理能力。因此，对象作为面向问题的抽象和面向机器的抽象的结合体，以一种自然的和语义保真的方式在软件系统中描述了问题域的抽象。

类型是对一组具有相似特征的对象的描述。其中，类刻划了一组对象的共有结构和共同的行为性质，在概念层上对对象进行了区分。型则是类的某种实现方式，即在实现层上以"模板"的形式来规定了对象在特定实现环境下的物理构成和处理能力。类和型的统一，构成了面向对象的程序设计方法的基础，也是我们集成化软件工程支撑环境的理论基础。

在我们的环境中，任何对象都有自己的类型。另一方面，类型作为一个实体，本身也是一个对象。对象之间存在着一定的抽象关系。从实现的角度来看，所有的对象都可以看作由一个私有存储以及一组在它上面施行的操作所组成。我们提供了对象定义语言OTDL 来描述对象的类型和处理对象的实例化，OTDL 支持刻划对象之间抽象关系的四种语义抽象机制:

分类(classification): 描述了类型和实例之间的关系，从而建立了高层对象(类型) 和较低层对象(实例) 之间的关系，即as-a 关系。

类属(generalization): 描述了对象类之间的关系，即一组具有类似特征的对象和刻划它们共性的对象之间的关系，例如类型的继承关系。子类(导出类型) 既有其父类(基类型) 的特性，又可以定义自己固有的特性，描述了is-a 关系。

聚合(aggregation): 描述了对象之间的组成关系，即一组相互关联的对象可以组成一个新对象，刻划了对象及其组成成分之间的关系，即component-of 关系。

集合(association): 描述了一个对象集合和单个对象之间的关系，把一组对象组成的集合视作一个对象，强调作为集合的特征，即element-of 关系。

环境中设置了若干基本类型，即这些类型的实例对象中不再含有其它对象，这种对象称为基本对象，这种类型又称预定义类型，如整型、实型、UNIX 文件型等，用户不必自己定义这些类型。非基本对象即为复合对象。环境中的所有对象组成一个对象家族，图4 表示了它们的继承关系。

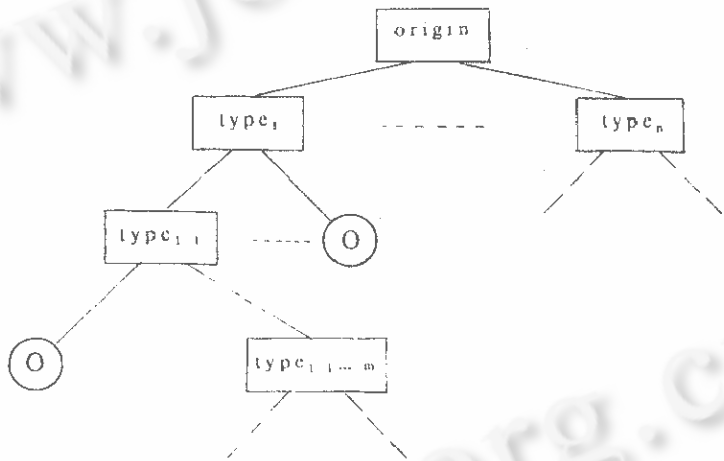

图4 类型系统组织模型

图中origin 是所有类型的基类型，代表了最基本的程序设计处理手段。系统中只有origin 不能有实例。其它类型都是导出类型。

类型管理系统还提供了一套辅助设施，用以处理对象(实例) 的产生、消亡以及组织管理。OTDL 则提供了类型的描述手段，所有这一切都支持了系统的开放性。

2.5 配置管理系统

配置管理系统记录环境中所有软件实体演变的历史信息，进行版本、发布、基线的标识和控制，维护各软件实体之间的关系，解决诸如who、what、when、where、why 和How 的问题并实现系统自动生成。环境中每个软件开发项目的所有版本采用树型结构加以组织和管理。在环境中，对象在某一时刻的状态记录称为一个版本。这样，任一对象的演变历史都可以从该对象的版本树中反映出来。从而保证了软件开发项目的可见性和可追溯性。

2.6 存储管理系统

存储管理系统负责环境中所有物理信息的组织和管理，提供对信息的存取和检索机制。软件开发和维护活动产生的大量数据如文档、程序、程序的内部形式等，在形式上有其特有的复杂性和相关性，因此不宜用传统的数据库来管理。在我们的环境中，存储管理系统除了提供常见的数据类型支特以外，还提供了支持复杂数据结构(例如文档、form、关系、UNIX 文件等) 的组织管理以及数据类型的复合构造，以满足软件工程活动的需要。在这里，所有数据都组织成嵌套表的形式，所有嵌套表构成一棵树，显然，这棵树也代表了嵌套表之间的构成关系。图5 表示了存储组织模型。

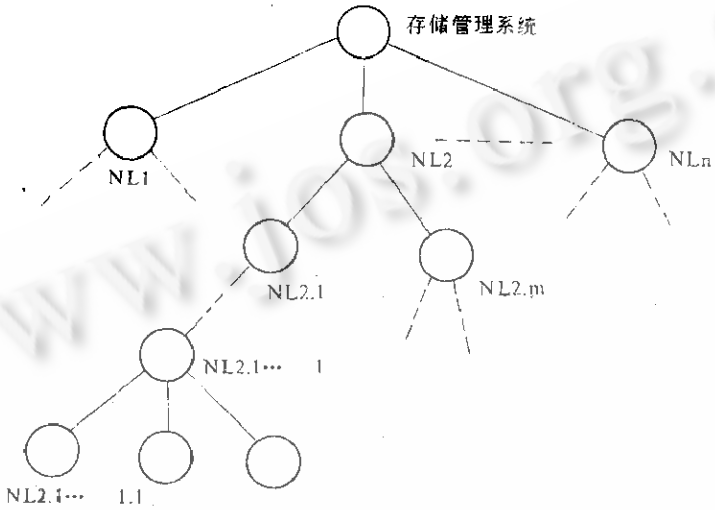

图5　存储组织模型

图中叶结点表示基本的嵌套表，非叶结点表示构造型的嵌套表。这种统一的存储模式支持了统一的对象概念。围绕着嵌套表树，提供了一套辅助设施和操作。

§3. 环境实现

我们采用分层虚机器的方法来构造整个环境。整个环境分为三层：环境层、对象层和数据层。各层之间相互联系，在基本数据的基础上不断地添加必要的辅助信息和处理、解释机制，最终支持环境的所有功能。图6 表示了环境的层次。

数据层解决数据的存贮和管理，事务的组织。它除了提供常规的数据类型以外，还提供了一些软件工程活动需要的复杂数据类型。在这一层上存放了环境中所有的数据，且都以嵌套表的形式组织起来。

对象层在数据层的基础上为各类数据添加辅助信息并重新加以整理，提供了对象、类型、视图的描述手段和处理设施，提供了对象的安全设施和权限控制、错误恢复，记录对象的演变历史和对象之间的关系。

环境层提供了软件开发规程、软件开发活动、软件工具、软件开发工作台等概念，完成对软件开发、维护活动的全部支持。

目前，我们已在Micro-VAX / II 和SUN-3 / 160 上完成了环境原型的实现，正在用环境提供的语言序列来描述工具，充实集成化软件工程支撑环境。

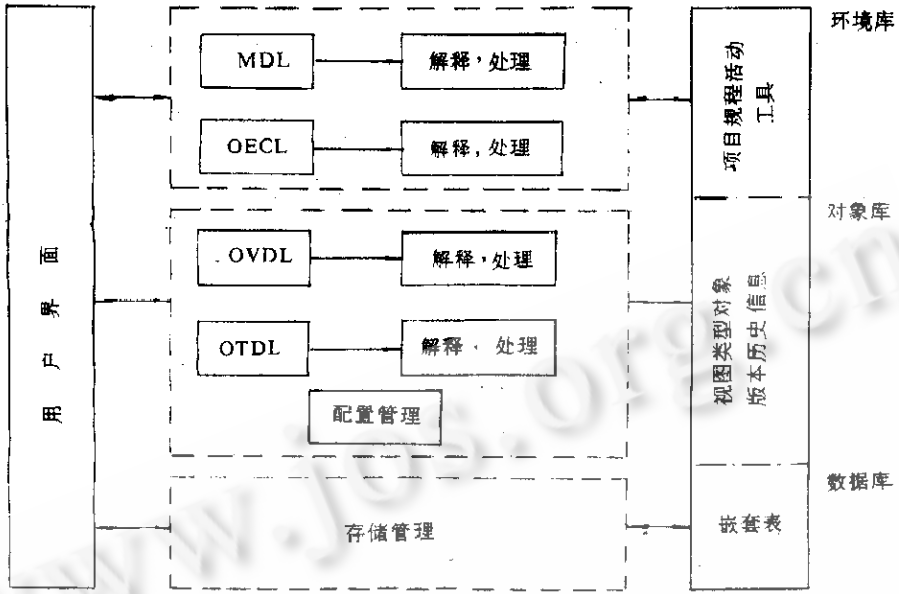

图6 环境的层次

§4. 结论

1. 集成化的软件工程支撑环境全面地集成地支持软件开发和维护任务。它可以支持软件开发的各个阶段，支持开发和维护的各项任务。同时，又支持各阶段之间的衔接和各任务之间的协调和共享。

2. 环境库和用户界面是我们环境的主要集成机制。

3. 环境支持面向对象的方法，实现了问题领域抽象和计算机处理能力抽象的统一。

4. 环境提供了一套基于c 语言的基本语言序列，支持了不同层次的软件开发人员，扩大了应用范围。

5. 环境库的设计支持了系统的开放性和可剪裁性。

6. 分层虚机器的实现方法保证了系统结构的清晰性，提高了系统的可靠性。

7. 环境库的设计为专用开发环境的研制提供了基础。不同性质的工具加入环境库就可以使环境具有不同的支持能力。

参考文献

[1] 杨芙清等，“软件工程支撑环境中集成化问题研究”，《计算机科学》，1987.4。

[2] Goldberg. A, "Smalltalk 80: The Interactive Programming Environment", Addison-Wesleg, 1983.

[3] Peter Wegner, "Classification in Object-Oriented Systems", SIGPLAN Notices, 1986. 10.

[4] M. L. Brodie, "Association: Abstraction for Semantic Modelling", CACM, 1983.

[5] Smith J. M etc, "Database Abstraction: Aggregation and Generalization", CACM, 1977. 6.

[6] S. Cook, "Languages and Object-Oriented Programming" S. E Jornal, 1986.3.

[7] T. Rentsh. "Object-Oriented Programming", SIGPLAN Notices, 1981.7.