计算机科学 1997 Vol. 24 No. 5

支持 DBPM 的数据库模型*)

A Database Model for DBPM

唐培和 张伟刚 殷作勤 李春贵
(广西工学院计算中心 柳州 545005)

刘连芳
(广西计算中心)

摘 要 The research of software process arouses great interest. This paper discusses a database model in details, which is used to supporting the Decision-Based Process Model——DBPM, and defines main data objects and operations. This model can serve as a means for the development of tools and environments to the DBPM.

关键词 Decision, Software process, DesignBase, Model

近年来，人们普遍认识到，软件产品和其它产品一样，其生产率和质量的高低与生产产品的过程有关，改进软件过程的质量，可最终获得高质量的软件产品。因此，从八十年代开始，学术界和工业界逐渐把注意力集中到软件过程本身的研究上来。1987 年，L. Osterweil 指出："软件过程也是软件"[1]，这种观点在学术界得到了广泛的赞同。进入九十年代以来，国际软件工程界在软件过程方面加快了研究步伐，目前的研究重点是描述过程模型的建模语言及其性能以及过程驱动的软件工程环境等方面。

1 支持软件过程的数据库——设计库

软件过程是建立、维护和演化软件产品的整个过程中所有技术活动和管理活动的集合，也可以认为，一个软件过程是一组受约束的协同活动，其中一些活动是自动执行的[2]。

对软件过程的研究，大致可划分为三个不同的方面。首先是过程建模；其次是过程实施(Enact)；再次是研究以过程为中心的环境与工具。

软件过程模型是对软件过程的抽象描述，这种抽象描述可以是形式化的、半形式化的，也可以是非形式化的。进行这种抽象描述的工程活动就称为软件过程建模。通常，以过程建模语言来构造软件过程模型，这样的语言已多达数十种，典型地，如 ALF, HFSP, APPL/A, Marrel, SLANG, 以及 XYZ/PME 等[2]。

本文所涉及的过程模型是基于决定(Decisions)的过程模型，模型的描述是以半形式化和非形式化相结合的形式进行的。有关此模型的详细内容另文讨论。该模型的基本思想是：软件开发与维护的过程实际上就是不断地认识问题、解决问题的过程，针对不同的问题，需要做出不同的决定，以便确定合理的解决问题的方案，并由此获得解决每一个问题的代码。因此在整个软件过程中，决定是至关重要的，关系到软件的质量和产品的性能。基于决定的过程模型就是以决定为中心的过程模型。

在基于决定的软件开发与维护过程中，有大量的过程信息需要存贮、管理与维护，不可避免地需要建立相应的数据库及其管理机制。我们把支持基于决定的过程模型的数据库称之为设计库(DesignBase)。

2 设计库的核心模型

支持软件过程的设计库涉及过程信息的存贮模型以及数据对象的操作。过程信息在设计库中以不同对象类型来表示，设计库的操作主要有增加、删除、替换、索引和访问等。

2.1 核心对象类型

设计库中有四种主要的、不同类型的对象，用以描述设计过程的不同方面，即问题元(Problem Elements)、设计决定(Design Decisions)、断言(Assertions)和代码视集(Code View)。核心对象以属性

*)广西区科委青年科学基金资助项目

(Attributes)集来表示。这些属性大致上可分为三类:①语义属性,描述特定对象的语义,这种属性的描述可以是形式的、半形式的(结构化语言或图形)或者非形式的,典型地,如描述问题语义的属性;②关系属性,用于描述该对象与其它对象的关系,既有正向关系,也有反向关系;③状态属性,表示对象的当前状态。

问题元　描述待解决的问题及其解的有关信息。这些信息由设计者记录,用于解释和陈述"做什么"这一需要设计者解决的问题。设计过程就是从问题开始的。

在设计库的核心模型中,我们用 Description、Reference、Result_from、Subject_of 以及 Status 五种属性来描述问题元对象。属性 Description 确定问题元的语义。在该模型中,假定语义描述是文本形式的。如果支持环境许可,也可以是形式的或半形式的。通过属性 Reference,可存取更多的信息,如参考文献、会议备忘录或其它相关信息等。

源于最初设计问题(原始设计问题)的问题元,其关系属性 Result_from 的值是空的,否则问题元至少是某个设计决定的结果。反过来,问题元至少是某个设计决定的主题,这可通过关系属性 Subject_of 来描述。

状态属性 Status 指明问题元是否已被处理(Processed),或悬而未决(Pending),或是否将被放弃(如果它来自设计库中一个被放弃的决定)。问题元的状态和决定的状态是相关的。

Problem :: Description × Reference × Result_from × Subject_of × Status
Description::STRING
Reference::STRING
Status=PROCESSED |IGNORED|PENDING
Result_from=Decision_id|NULL
Subject_of=Decision_id
Decision_id::TOKEN

设计决定　是设计库中的核心概念,描述设计与维护过程中可能做出的决定(动作或选择),涉及"怎么做"的问题。对设计决定的描述,采用了六种属性,即属性 Description, Justification, Alternatives, Justification_of_choice, Subject 以及状态属性 Status。

属性 Description 描述设计者针对特定问题作出决定(动作)的语义,在这里仍以非形式的方式(文本方式)来描述;属性 Justification 描述设计者为什么做出这种决定,或者"为何对此感兴趣?"这一类的问题。而属性 Subject 则描述决定的作用对象(问题元)。

属性 Alternatives 记录设计者解决同一问题时可能采用的解决方案。解决一个问题的方案很可能不止一个,设计者必然会在多个方案中选择其中一个方案。属性 Justification_of_choice 则用来描述设计者为什么选择这个方案而不是别的。

设计决定是针对问题的,每个问题必须有且只能有一个设计决定。每个设计决定将导致若干新的问题或断言。例如,针对问题 P,设计者可做出这样一种决定,将其分解为问题 P1 和 P2,然后分别求解。

如果设计者为求解一个问题而做出了某种决定,则其状态为活的(Active);否则,如果一个决定已被考虑用来求解一个问题,但未被选中,则其状态为睡眠状态(Sleeping)。

Decision :: Subject × Description × Justification × Alternatives × Justification_of_choice × Status
Subject=Problem_id
Description::STRING
Justification::Assertion_id
Justification_of_choice::Assertion_id
Alternatives=Alternative_id→Alternative
Alternative=Solution
Solution::STRING
Status=ACTIVE|SLEEPING
Problem_id, Assertion_id, Alternative_id::TOKEN

断言　这类数据对象记录与描述特定问题及其设计信息,以证明做出这样的设计决定是合理的。它们解释设计过程或最终产品"为什么这样做?"

设计过程中,设计者使用他们的设计策略和问题域方面的知识,这些知识几乎不包含在形式化的文档之中。我们提出在断言中记录特定问题与设计知识,便于设计者论证他们的决定为什么是合理的。

Assertions=Assertion_id→Assertion
Assertion::Description×Reference×Justification
Description::STRING
Reference::STRING
Justification::STRING
Assertion_id::TOKEN

代码视集　就是基于特定软硬件环境、解决指定问题的源程序代码。和传统意义下的代码片段(程度段)不同,它是一种代码片集。

代码视集是根据问题来划分的,既与功能性需求有关,也与非功能性需求有关;而传统的程序段(程序块)是根据功能来划分的。因此前者的代码行可以是连续的,也可能是离散的;而后者的代码行通常是连续的。并且,两个不同的代码视集的交集可能是非空的。

CodeViews=Cid→CodeView
CodeView::Subject×Codes×Status
Subject=Sid
Codes=Lid→Line

Line ::STRING
Status=ACTIVE|SLEEPING
Cid,Sid,Lid::TOKEN

综合以上分析，我们可以得出设计库的核心模型，如图1所示。

2.2 设计库的基本操作

设计库管理、维护与设计过程有关的信息(记录)。除被管理的数据对象外，设计库应提供一系列有关管理、生成、查询、访问(浏览)等方面的操作及其支持工具。利用这些工具，人们可以产生(增加)新的数据对象；删除无意义的数据对象；按多种方法查询设计库中已有的数据对象；以及访问(浏览)特定数据对象的每个属性。限于篇幅，本文只讨论查询与浏览操作，其它操作就不做详细描述了。

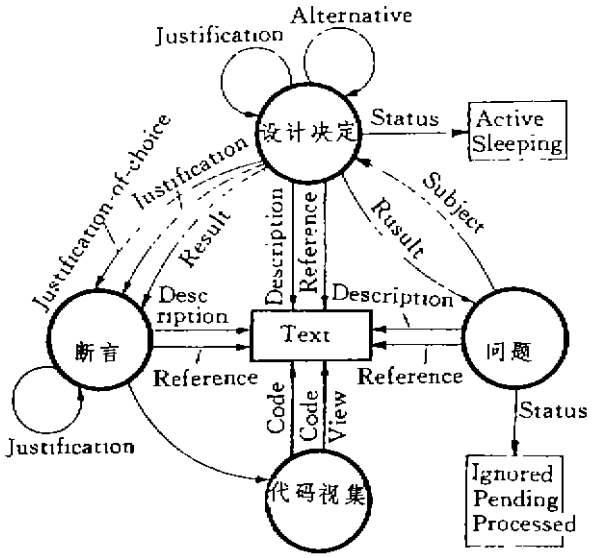

图1 设计库模型

查询操作 设计库由数据对象集以及施加于数据对象集上的操作所组成，设计库中的每一个数据对象都有其特定的标识符。核心模型至少应提供两种方式来定位(Addressing)设计库中的数据对象，即标识符和关键字。为描述方便，用 DesignBase 表示设计库，用 Object 表示设计库中的数据对象，Operations 为施加于数据对象上的操作，不难得出：

DesignBase::Objects×Operations
Objects=Object_id→Object
Object_id::TOKEN
Operations::……

核心数据对象的类型前面已做过讨论。根据对象的标识符，定位其对应的对象，相对来说比较容易，这里，我们定义操作 AddressingProblem 和 AddressingDecision 来完成。

AddressingProblem:Problem_id→DesignBase→Problem
AddressingDecision:Decision_id→DesignBase→Decision
AddressingCodeView:CodeView_id→DesignBase→CodeView

另一种定位数据对象的方法是通过关键字的匹配来实现。给定关键字(Keyword)后，扫描设计库中所有的对象，能与关键字匹配的数据对象可能有多个，也可能没有。在这里，我们定义了 SearchingProblem 和 SearchingDecision 这两种最基本的操作。

SearchingProblem:Keyword→DesignBase→Problem_set
SearchingDecision:Keyword→DesignBase→Decision_set

访问操作 设计库中对某一个数据对象的访问涉及数据对象的属性，典型地，如增加一个属性；删除一个属性；更改指定属性的值；观察指定属性的值；以及浏览数据对象所有的属性等。因此，在这里我们定义了下面五种基本操作。如果需要，还可以定义更多的操作，在此不做深入讨论。

ChangingAttributeValue:(Attribute×NewValue)→Object→Object
DisplayingAttributeValue:Attribute→Object→Value
BrowsingAllValue:()→Object→Value_set
DeleteAttribute:Attribute→Object→Object
AddAttribute:Attribute→Object→Object

结束语 本文只讨论设计库中的一些基本操作，仅有这些操作是不够的。须指出的是，本文未做详细讨论或提及的操作并非是次要的或可有可无的；相反，可能是必须的，不可缺少的。此外，因篇幅所限，没有给出每一种操作详细的语义描述及其实现细节。

软件过程建模的目的在于：使人们易于理解软件过程并在此基础上进行交流；支持软件过程中各参与人员之间的通讯与协调；支持对软件过程的分析与改进；支持对软件过程的实施与管理；支持过程重用；提供对过程的指导与自动执行支持等。

研究过程模型与过程实施对于软件生产工程化、自动化和过程重用有重要意义；可以使软件项目具有更好的可预见性；使软件产品具有更高的质量；从根本上改善软件生产过程和软件生产率；使软件组织过程更成熟些，以适应各行各业日益加剧的对软件产品的需要。

参 考 文 献

[1]L. Osterweil, Software processes are software too. Proc. of the 9th Intl. Conf. on Software Engineering, Los Alamitos, CA: IEEE Computer Soc. Press, 1987

[2]柳军飞、唐稚松，软件过程建模语言研究，软件学报，7(8)1996