计算机科学2002Vol.29№.10

UML与SDL结合使用的研究

Research on Using UML with SDL

魏定国[1,2] 吴时霖[1]

(复旦大学计算机系 上海200433)[1] (广东商学院 广州510320)[2]

Abstract In this paper we investigate how Unified Modeling Language(UML)can be used together with the Specification and Description Language(SDL) standardized by International Telecommunication Union(ITU). In our research,it is discovered that a SDL user can benefit from UML,similarly,a UML user is able to take advantage of the SDL. Especially for a real-time system development,UML should be utilized in analysis of the requirement,and SDL should be used in system design.

Keywords UML,SDL,Software architecture, Real-time system

1 引言

在介绍如何将SDL与UML结合在一起使用之前,让我们先来回顾一下UML和SDL各自的适应范围。它们性能上的差异在于它们在工程应用中的不同的阶段。UML的非形式化的自然特性使其成为一种性能优越的建模语言,它应用完全的面向对象的分析(OOA)方法。然而对于实施,UML却不能胜任,通常要用传统的程序设计语言如C++/Java来完成,而SDL的形式化定义使得它取代传统的程序设计语言有了可能。OOA与SDL相结合已有相当长的时间了,先用OOA(主要是类图和用例/概要)进行需求分析和识别对象,然后用SDL/MSC来精确定义系统的架构和系统的需求与系统的行为。很自然,我们可以用UML来进行需求分析,用SDL进行系统设计。

然而,我们认为UML和SDL之间的差异并不是一件坏事,因为它们对开发的各阶段提供不同支持是非常有用的。就一个应用系统开发的各个方面来说,这两种语言适应的范围是相当大的。UML所适应系统开发的相关方面主要包括:

(1)逻辑架构,如使用类图和状态图;(2)需求架构,使用用例图;(3)物理架构,用组件和展开图等;(4)系统开发的工作流,使用活动图。

UML宽广的应用范围是它的优势之一,但很难有对每一个方面都支持的工具,这也就使之成了一个较大的问题。而SDL中却采用不同的方法,它只着眼于应用软件的逻辑架构,也就是如何将应用系统分解成逻辑组件的层次结构,进而对每一个组件的行为、与其它组件或其应用环境之间的交互进行精确的定义。这些特性集中到这种语言上,给SDL用户带来真正的好处:

(1)能够适合分布式应用、实时处理和并发应用的设计。

(2)它有精确的语义、完整的行为定义。

(3)它易于对层次结构、封装和接口进行精确定义。

我们都知道,在应用SDL进行应用开发时,很大程度是用并发状态机来描述的,它主要是用异步通信机制来进行通信。这使得SDL成为一种特殊用途的语言,非常适合于并发、分布、实时应用系统的开发,因为这些系统用并发状态机来描述是特别有效的。

SDL系统是一个应用系统的完全描述。由于SDL形式化定义了完整的动态语义,SDL系统分析和验证的支持工具都是非常有效的。大多数SDL工具对语法与语义分析、SDL模型的仿真、形式化验证、测试都提供了广泛的支持。正是由于精确、完整SDL定义使得我们可以直接从SDL生成应用程序。大多数SDL工具提供了代码生成器,不需要作任何修改和扩充就可以工作在各种目标环境中,如不需要操作系统支持的裸机或集成到实时操作系统中使其成为一个任务。SDL的代码生成与高级的自然特性相结合提供了基于通信和扩展有限自动机模式的应用系统开发的一个极为有效的方法。

在SDL中的架构与人机接口等关键工作是由专门的结构图来完成,结构图用来定义SDL应用的层次、通信路径和人机界面。这些都使得SDL非常适合大规模的开发,它可以将复杂的应用系统分解成易于处理的组件,使其能够由不同的团队来进行开发。

2 将UML与SDL相结合

我们从系统设计的三个不同方面来说明怎样将UML与SDL结合在一起使用:(1)展开与物理分布;(2)详细描述系统的行为;(3)架构和通信交互。

最后我们再讨论在分析阶段如何使用UML的非形式化的特性,然后在SDL设计模型中将UML所分析的结果形式化。

2.1 展开与物理分布

2.1.1 SDL交互图 在SDL发展成一种设计语言之后,由于非常贴切实施,这种语言的精确性非常接近于常规的程序设计语言。这意味当进行架构建模时,就要采用自顶向下的设计方法。但由于SDL缺乏对可选描述或视点的灵活支持,要对可视化系统的不同视点进行不同的SDL描述仍然不切实际。

在实际工作中,在设计活动开始时,可能需要研究几种架构,但却只有一个被选,然后将所有设计结果在这个架构中描述。

SDL所提供的语言结构是:

魏定国 副教授,博士生,主要从事计算机通信与网络技术、数据库的研究。**吴时霖** 教授,博士生导师,主要从事计算机通信与网络技术、Petri网理论及应用的研究。

(1)交互式图的定义

· 系统、块、块嵌套、进程;

· 用于活动对象的面向对象描述:系统类型、块的类型、进程类型。

(2)通信结构的定义

· 信号、信号表、信号数据的定义;

· 在结构元素之间通信路径的信道。

所选择的SDL架构与最终选择的物理架构通常是有出入的,SDL在这种情况只是一个逻辑架构,它应尽可能使得设计和维护能够有效地进行。

2.1.2 UML实施图 当考虑到与物理架构相关的说明,UML提供两个不同的实施图:组件图和展开图。

组件图描述软件组件与组件之间的关系。UML就是通过组件、类、依赖关系、和接口等概念来描述的。

展开图是关于可执行系统的物理架构和组件运行时的特性描述,它包含与组件图相同的信息,但它通过节点的概念作了很多的扩充,节点是系统环境中运行软件或其它物理设备物理机器或处理器。在节点之间有一些通信关联,附带描述通信介质、通信协议和数据编码等。

该图的目的就是要给出系统的各部分在实现时是怎样在不同的计算机中分布。可以用来描述在设备中的活动对象,如任务、进程和线程。UML也可以提供几种可选择的展开图,并给出几种可能不同的系统实现。

2.1.3 将SDL与展开图相结合 可以看到SDL虽然具有方便、明确的概念能够很好地用于描述逻辑架构的设计,但它却缺乏对可选架构视图的支持,还缺乏实施细节概念如描述物理分布、仿真视图、编译和链接信息、下载与执行建立等。即使是最新版的SDL还不尽人意。

UML组件和展开图可以基本上解决这些问题。我们建议在设计活动中使用SDL建立逻辑架构,并用UML组件图和展开图附上那些不能用SDL形式描述的信息。UML组件图给出逻辑设计制品是怎样与物理元素如文件与库相联系的,而UML展开图是描述逻辑组件是怎样与物理组件相联系的。

2.2 行为的描述

虽然UML和SDL都提供了描述状态机行为的方法,但各有其长处和不足,通常按下述原则来应用它们。UML应在分析阶段使用,因为UML的状态图工具对于行为分析极为方便实用,具有对SDL设计阶段所描述的SDL进程行为进行概括综述的能力;SDL最好用于设计阶段,因为提供的概念适合于描述行为的细节,这正是UML所缺乏的。很显然UML状态图和SDL进程行为的描述是在同一个信息模型中的两种相互补充的视图。

2.2.1 UML状态图 用状态图进行描述有两个主要的好处:可扩展性和紧凑性,而且状态图提供几个非常好用的概念用于行为建模,这是传统的SDL所不能提供的。可扩展和紧凑的特性使得UML状态图特别适合分析阶段,因为在此阶段概述比细节更重要。

表达复合状态的能力是非常有用的,这正是状态图所提供的。描述状态层次结构可以使用自顶向下的方法对行为建模,当在分析阶段采用以行为为中心的方法进行系统开发时,这就显得特别有用。另外一个好用的概念是并发子状态,在分析阶段应用这个概念对并发的建模确实是一个好的方法,但在设计阶段,注意力是从概述到细节,并发子状态的描述不但不会简化状态机,反而会增加其复杂性。几个概念可能同时存在于状态图和SDL进程行为中,如动作。在UML状态图中对这些定义经常有些含糊不清,这正是需要SDL的原因所在。然而在设计阶段状态图对SDL进程图的补充是非常有用的,使用状态图可以隐藏其细节,突出其状态与事件。状态图在设计阶段不能很好地适应各个方面,因为缺乏模型化细节行为的概念,它不能提供数据模型。

2.2.2 SDL进程流程图 SDL进程行为的描述在系统分析中是非常有用的,然而,如果要描述层次和并发状态机就会变得非常复杂,因为SDL进程描述缺乏状态图所支持的那种概念。SDL仅支持非层次状态机,只能将平行行为建模成分离的进程实例。如果为了抓住和提供大量的信息并对问题进行形式化,这些限制或许是有意义的。在细节行行的建模变得非常重要的设计阶段,SDL具有很大的优越性。SDL进程的行为包括的概念有任务、决策、算法、过程和其它用于应用程序开发所必需的程序设计概念。SDL还提供一种独立于数据模型的完全目标语言。所有这些概念相结合使得SDL成为完全的设计语言。与UML状态图不同的是,SDL有明确定义的语义,它定义了信号的处理、排队规则、定时器的操作等等,使SDL适应于编译,也能在SDL语言水平上进行调试,能在SDL级别上进行应用程序的开发。

2.3 架构与交互

2.3.1 SDL交互图 正如上面所说的,SDL系统和框图是用来描述系统的逻辑架构,最适合作设计活动。SDL严格的语法规则和形式化的语义使得CASE工具支持扩展的静态分析和高级的逻辑仿真、自动运行错误检测。SDL系统和框图既能捕获系统的结构又表达了高级设计的通信。

SDL语言的缺陷之一是缺乏对同样功能的可选视图支持。现在的新版SDL仍然缺乏对继承关系的图示说明。

2.3.2 UML类图 在UML中从图的类型来说可以称其为静态结构图。类图和只包含实例的对象图之间有时还是有些不同。要区分类图和对象图主要是历史的原因,在必要时可以用相同的图来描述类和对象。

UML静态图对概括综述、任选视图、继承关系(通信与层次)、灵活的视图(没有严格语法与语义)和包关系都提供了良好的支持。

UML的灵活性是其很大的一个优点,如果系统不太大,那么所有的类和关系可以表示成一单个的图。对于大系统或为了更好地理解可以用几个图来描述,每一个用来描述系统架构的一部分,或者同一类视图。例如

(1)架构视图用来描述组件的最高层次;

(2)细节架构视图用于系统的各个部分;

(3)通信视图给出通信类和它们通信路径;

(4)继承视图给出各个类之间的继承关系。

2.3.3 将SDL与类图相结合 SDL与UML静态的结构图相结合能够改进系统的设计说明。所定义的SDL架构能够保证设计期间不但使组件易于处理,而且使得系统便于维护、组件可以重用。

UML静态图应当作为一种附加的视图用于改善对系统组件和它们之间关系的理解。在SDL缺乏对可视说明支持的地方显得尤为重要,如在结构元素或数据上的继承关系、包依赖或抽象数据类型(ADT)/数据类的定义等。同样,SDL以一个集成的方式与UML类图概念相结合,允许这些概念直接在SDL图中使用。

将 UML 和 SDL 结合在一起的另外一个目的是将 UML 更多的非形式化的方法用于分析。在这种情形下 UML 静态图和状态图在 SDL 设计中是可重用的。这种方法下面作了进一步详细的阐述。

3 系统分析与设计

3.1 动态需求的分析

在九十年代中期，将用例的概念引入了面向对象的方法学中开始向一个新的模式转移，极大地影响了需求工程研究的领域。到那时动态需求的处理与描述还没有得到与软件的静态需求同样的注意。可能是由于实时软件业(包括不同的电信领域)引进面向对象的概念比整个软件业要慢一些的缘故。

用例模式已被证明是非常有用的，这是由于它具有以下三个方面的优良特性：

(1)把注意力集中在系统的用户视图，使用系统作为用户的目标。

(2)集中注意概要，以最简单的方式描述用法例子，确保外行也容易读懂和理解按这些规则建立起来的需求文档。

(3)进一步将用例求精成的序列图，集中注意系统各部件之间、外部系统与用户之间传送的信息。

然而传统的序列图缺乏形式化的定义，缺乏使用例的部件可重用的细节说明，在分布式、实时系统中还缺少描述通信行为所必需的概念。但幸运的是，最近的消息序列图(MSC)使得 UML 的概念得到了进一步的扩展：

(1)能够以类似流程图符号给出行为概要的层次说明，应用的操作符号有选择、序列、任选、异常和并行，因此在描述中具有高度的灵活性。

(2)在层次说明中对每一个项目的通信行为都有一个详细的描述。

(3)以严格的方式重用行为概要的各部分，通过消息、定时器和实例的替代使得 MSC 基准的前后能准确使用。

(4)在事件、事件组或完全的概要中能够描述绝对或相对时间约束。

(5)既能描述同步消息传递又能描述异步消息传递。

(6)能够提供细节数据和表达式说明和约束，不但增强了语言对细节的描述能力，而且具有很高的灵活性。

(7)能够描述实际的实时概要如消息超越，消息丢失和意外的消息。

用例对系统组件早期的确立是至关重要的，也是关系到系统的边界。当使用 MSC 进行详细的用例描述时就更是如此了。由于组件的群聚或分组通常是由组件之间的通信活动级别来决定的，设计者不但要精确定义系统的组件和对象，而且要选择系统的架构。

3.2 静态需求的分析

软件开发的早期活动(即分析)的目标是要弄明白要做什么，定义工程的目标，包括识别和理解系统的需求。其主要任务之一就是要识别系统与环境之间的边界。另外一个任务就是要概括系统的信息模型。这些活动要求用易于理解的语言来表达，而不是形式化。所描述的范围模型和信息模型是系统的面向对象建模的基础，这些早期的活动最好用 UML 来完成。UML 类图具有强大的表达能力，只需使用很少的几个概念就可以完成分析建模，这当然是以损失精度为代价。然而，由于 UML 类图的优点和缺陷使得它用于分析阶段非常理想，因为分析阶段主要强调的是表达和良好的建模能力，而不必关心系统细节的精度。

由于类图的建模概念比较少，因此它比较容易创建和理解。这在分析阶段是非常重要的，因为它涉及到与各个部分之间的通信，包括外部与工程组之间的通信和内部的各种通信。类图还有其它强大、高效的表达能力。例如 UML 允许同一模型有不同的视图，这便于强调模型中我们特别感兴趣的方面。另外，在类图中很少使用抽象，这使得它容易升级，容易用合适的详细程度呈现每一个视图。UML 还能提供一个紧凑的表示，这使得它在 SDL 系统中描述全局概要是非常方便有用的。

UML 类图的特性是与范围无关，这使得 UML 描述问题从信息系统延伸到嵌入式系统之中。UML 描述的模型可以适应多个领域，其广阔的应用范围可以使它能够应用到很多特殊领域，使用很多的特殊符号，特别适合于处理一些特殊问题。

尽管如此，我们还是主张使用 SDL 进行应用开发(特别是对于实时系统)，只是要在开发中充分运用 UML 概念与方法，其原因如下：

(1)SDL 描述的需求规格说明准确无二义性，表达接口的规格说明特别方便。

(2)SDL 模型是一个可执行的模型，方便仿真，能确保模型的准确完整。

(3)UML 模型可以作为 SDL 可执行模型的文档，增强它的可理解性，方便其取舍。

正是由于可执行的规格说明使错误能够被尽早检测出来，也就能够增强 SDL 模型的正确性，大大减少了将错误带到设计阶段可能性，有利于减少开销(到了设计阶段才发现错误，纠正错误的开销就会大得多)。

3.3 系统设计

当完成了分析，进入到设计阶段，注意力就集中到怎样做了。设计阶段的目标就是要对系统分析阶段所定义的问题提供解决方案，也就是定义系统的架构，包括认可其结构，详细说明系统各组件之间的接口。在架构说明中任何含糊或不明确将很可能导致实施活动的费时重复的工作。虽然 UML 的强大描述能力使得它也可以用于后来的活动，但只能适于一个较小的范围，因为设计应当更注重精度和细节，应当选择能够精确说明问题的语言。设计语言应当能够管理控制详细的程度、支持可扩展性，提供到实施阶段的一个平滑的转换。

在实时系统应用领域用 SDL 来表达设计是一个明智的选择。在分析阶段选择 UML 作为工具对 SDL 设计不会产生什么负面的影响，因为分析阶段的 UML 表达能够很方便平滑地转换成设计阶段的 SDL 表达。SDL 有几个概念，使得它成为一个性能优越的设计语言。其一是 SDL 能够直接编译成应用程序，这大大地减少了实施工作。其二是 SDL 的图示化的符号使其容易理解，是一个性能优越的文档处理语言。传统的 UML 设计方法要用 UML 图作为源代码的文档，需要维护，以保证源代码与 UML 图之间的同步。在 SDL 中，文档本身就是源代码。SDL 包含很多细节的概念支持可扩展性和复杂的行为，这正是 UML 建模语言所忽略的。另外 SDL 支持高级的调试，能在正式实施之前执行并验证所做的设计，将设计错误减少到最小程度。在实际工作中如果采用 UML 进行设计，为了确保其正确性，必须进行手工检查，这是一个费时、费力、开销大的工作。由于 UML 有这样的缺陷，有时只要大

(下转第159页)

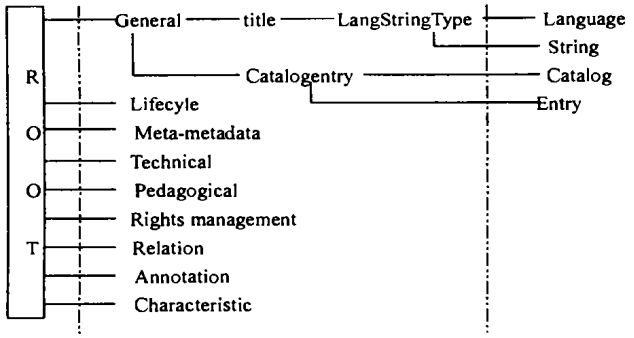

图2 媒体素材库元数据详细集元素示意图

小结 本文在分析了已有的教育资源的元数据基础之上，仔细分析了现代远程教育系统下媒体素材的特点，针对远程教育系统中用户的不同，提出了媒体素材库的层次型元数据模型。现代远程教育媒体素材库元数据标准规范和统一了远程教育系统中媒体素材的描述，保证了使用媒体素材的一致性和简洁性，为远程教育环境下由媒体素材组建个性化的学习内容奠定了基础。本文提出的媒体素材库元数据模型仅是一种新的尝试，希望对远程教育资源建设的规范化有一定的指导和促进作用。

参 考 文 献

1 The Dublin Core element set. http://WWW. purl. org/DC/about/element_set. htm

2 The IEEE Learning Object Meta-data (LOM) base document. http://ltsc. ieee. org/doc/wg12/scheme. html

3 The Instructional Management System Project (IMS) website. http://www. imsproject. org

4 http://www. fgdc. gov/metadata/metadata. html

5 Gateway of Education Material (GEM) website. http://www. gem. org

6 英国 UKOLN 网站. http://www. ukoln. ac. uk/

7 澳大利亚 EDNA 元数据网站. http://standards. edna. edu. au

8 教育部高等教育教育资源规划可行性报告，1999

9 教育部现代远程教育工程教育资源建设规范，1999

(上接第152页)

体上表达某项问题，就使人相信问题已经得到了解决，但实际上将问题传给了下一个阶段。

尽管 UML 太概括了以至它不能成为表达设计的自然选择，但在 SDL 设计中仍然可以从 UML 中获得益处。利用 UML 紧凑的符号提供 SDL 架构的概括综述。UML 中的图非常适合描述 SDL 包之间的关系，而 SDL 却没有这种图示，因此可以借助 UML 来表示。SDL 类型之间的继承层次关系也可以图示化，在继承层次中 SDL 类型表示成 UML 中的类。

结束语 UML 和 SDL 结合起来使用非常有效，比单独使用能够提供更好的开发支持。现在的 UML/SDL 工具通常分解成两个独立的部分，一个是基于 UML 用于面向对象的分析，另外一个是基于 SDL 用于设计与实施，然后提供两模型间的链接与转换。这是当今 SDL 工具最通常的结构，能够给用户提供一个良好的解决方案，充分利用 UML 和 SDL 两种语言的优势。

然而总是用两种以上的工具或语言，必然增加开销，因为在两种不同的工具或语言之间需要切换。因此将 UML 和 SDL 合并在一起这是当今的趋势。合并 SDL 和 UML 工具是可能的，能够使得 UML 和 SDL 结合使用更好方便平滑。从 UML 开发者来看，他能够得到由 SDL 带来的便利，如在 UML 工具中他也可以仿真、形式化验证，完全的应用生成和大规模分布系统的直接定义；从 SDL 开发者来说他可以在他的 SDL 工具中直接使用强有力的面向对象的分析工具，他还能从其它可以利用的 UML 模型中获得益处；从研发单位来看 UML 和 SDL 合并也是非常重要的，因为这可以减少用户使用 SDL 和 UML 的培训费用，可以相互重用，工程师可以方便地从基于 SDL 的开发转到基于 UML 的开发，可以大大地减少因此而引起的投资。而且当不同的符号统一规范了之后，由分析团队建立的分析模型就能很方便地由设计团队来使用。

我们可以得出这样的结论，将非形式化的 UML 世界和 SDL 的形式化的语义相结合在各类系统的开发中带来很多便利。我们认为有必要将它们集成在一起，建立性能更为优越的建模设计环境，以便更加适应复杂分布式实时系统的大规模的工业化分析与设计。

参 考 文 献

1 OMG，UML1. 3 June1999

2 ITU-T Recommendation Z. 100. Specification and Description Language(SDL)Nov. 1999

3 bourhfir C，Aboulhamid E. Test cases selection from SDL specifications，Computer Network，2001. 693～708

4 ITU-T Recommendation Z. 120：Message Sequence Chart(MSC)，Nov. 1999

5 魏定国，吴时霖. 基于 UML 和 Petri 网的用户界面原型的研究. 计算机科学，2001，12

6 Engels G，Heckel R，Saucer S. UML -A universal Modeling Language? ICATPN2000，LNCS，2000. 24～38