软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(4):831−845 [doi: 10.3724/SP.J.1001.2012.04043]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 MOF 的软件体系结构分析结果集成框架*

陈湘萍[1,2], 黄 罡[1,2+], 宋 晖[1,2], 孙艳春[1,2], 梅 宏[1,2]

[1](北京大学 信息科学技术学院 软件研究所,北京 100871)

[2](高可信软件技术教育部重点实验室(北京大学),北京 100871)

MOF Based Framework for Integration of Software Architecture Analysis Results

CHEN Xiang-Ping[1,2], HUANG Gang[1,2+], SONG Hui[1,2], SUN Yan-Chun[1,2], MEI Hong[1,2]

[1](Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[2](Key Laboratory of High Confidence Software Technologies of Ministry of Education (Peking University), Beijing 100871, China)

+ Corresponding author: E-mail: huanggang@sei.pku.edu.cn

Chen XP, Huang G, Song H, Sun YC, Mei H. MOF based framework for integration of software architecture analysis results. *Journal of Software*, 2012,23(4):831−845. http://www.jos.org.cn/1000-9825/4043.htm

Abstract: In software adaptation, multiple analysis methods are used for analyzing system, planning adaptation strategy, and decision-making. Because the analysis results are interpreted and understood with SA (software architecture) model as context, synthesizing analysis results and the SA model become important. However, current researches on integration of analysis methods do not pay enough attention to the integrating analysis result with the SA model. Considering integration challenges existing in meta-model, model, and view levels, this study proposes an MOF (meta-object facility)-based framework for integration of analysis results. The framework provides the following: An ADL (architectural description language) extension mechanism with support for upward compatibility; automatic generation of model transformation for model synthesis, and code generation to extend modeling tool for view of synthesized model. In the case study, this study uses the framework to integrate three analysis methods for reliability evaluations and fault tolerant reconfiguration, and applies these analysis methods to analyze SA model of J2EE system Ecperf.

Key words: software architecture; model driven; MOF (meta-object facility)

摘 要: 运行在网络环境下的软件在自适应过程中,需要集成多种分析方法来进行分析、规划和决策.由于自适应的决策程序或者设计人员以 SA(software architecture)模型作为解析和理解分析结果的上下文,这使得分析结果与 SA 模型的集成尤为重要.但是,现有的分析方法集成框架多关注于提供输入、执行分析、从而得到分析结果的过程,对分析结果的集成关注不够.针对分析结果与 SA 模型集成中元模型、模型和视图 3 个层次的挑战,提出一种软件体系结构分析结果集成框架.框架使用 MOF(meta-object facility)元建模技术提供 ADL(architectural description language)的扩展机制;使用自动生成模型转换实现 SA 模型与分析结果的合成;使用代码生成技术扩展建模工具为扩展后的 ADL 提供模型视图.最后以 3 种分析方法——两种可靠性评估方法和容错风格的规划方法为例,使用集成框架将其加以集成并应用于 Ecperf 系统的 SA 模型的分析中,从而展示集成框架的可行性和有效性.

* 基金项目: 国家重点基础研究发展计划(973)(2009CB320703); 国家创新研究群体科学基金(60821003)
收稿时间: 2010-02-18; 修改时间: 2010-06-09; 定稿时间: 2010-07-20

关键词: 软件体系结构;模型驱动;MOF(meta-object facility)
中图法分类号: TP311 文献标识码: A

作为网构软件的基本特性之一,自适应性是指软件系统在预设策略的指导下自动地监测系统状态信息,并在必要时对自身进行调整,以提供更好的服务[1].软件在自适应过程中,需要考虑适应目标、运行系统的状态和行为以及如何保证系统的正确性和质量属性等因素,以得到自适应调整的策略.这个决策的过程需要使用分析方法,通过分析将一个复杂的论题或者本体分解为较小的部分和侧面,从而更好地理解问题.

软件在自适应中,使用多种方法来获得分析、规划和决策所需的分析结果,其对分析结果的使用也有着新的特点:一方面,分析结果不再仅供设计维护人员参考,而是作为其他分析、规划方法的输入,或者直接作为自适应调整的决策.例如,关注可靠性的系统会将规划方法的分析结果作为可靠性分析方法的输入,从而评估规划方案的可靠性;另一方面,自适应的调整需要综合考虑多个分析结果,而这些分析结果描述不同的软件特性,是软件体系结构(software architecture,简称 SA)模型中不同元素的属性.SA 模型成为解析和理解分析结果的上下文,如果 SA 模型与分析结果分布在各自独立的模型中,那么后续的分析或者规划程序将无法解析和使用这些结果来进行分析,也不利于理解.

自适应软件对分析方法集成的需求,除了要求为分析方法提供输入、执行分析、从而得到分析结果以外,更注重分析结果与 SA 模型的集成.分析结果与 SA 模型的集成指的是将分析模型作为 SA 模型或者 SA 模型中元素的属性,按照分析结果与 SA 模型的关系将其组织为一个包含分析结果的 SA 模型.例如,将可靠性分析和规划结果与 SA 模型集成,在获得的 SA 模型中除了包含原软件体系结构建模语言(architectural description language,简称 ADL)中定义的属性之外,还包含新增的每个构件的可靠性以及候选的自适应调整方案.

目前,在对集成框架的研究中,大多关注于对分析方法提供输入、执行分析、从而获得分析结果的过程.已有的工作按照对集成进行支持的方式,主要分为两类:一类以集成框架为中心,为分析方法的设计提供约束和指南,使得按照规则设计的分析方法可以很容易地加入到分析方法中,代表性工作包括 ADD 方法[2]和 FDAF 方法[3]等;另一类工作以分析方法为中心,通过提供框架或者基础设施,为不同实现方式的分析方法提供输入适配或者执行上的辅助,代表性工作包括 XTEAM 框架[4]、KAMI 框架[5]、DUALLY 项目[6]等.这些工作对于分析结果的使用关注不够.

分析结果与 SA 模型的集成,需要面对的挑战包括:原有的 ADL 不能描述包含了分析结果的 SA 模型;SA 模型中没有包括分析结果;建模工具无法提供合成后模型的视图.针对元模型、模型和视图这 3 个层次中存在的挑战,本文提出一种软件体系结构分析结果集成框架.框架基于集成的分析方法,为用户提供扩展后的 ADL、包含分析结果的 SA 模型以及其辅助的建模工具.框架的解决方案使用元对象机制(meta-object facility,简称 MOF)提供 ADL 的扩展机制;使用自动生成模型转换实现 SA 模型与分析结果的合成;使用代码生成技术扩展建模工具来提供扩展后 ADL 的视图.其实现基于通用的建模语言 ABC ADL 及其建模工具,为分析方法集成人员和分析方法使用者两类用户提供支持.

最后,本文在实例研究部分以 3 种分析方法——两种可靠性分析方法和一种基于容错风格的规划方法为例,展示如何使用框架集成和使用分析方法.集成了分析方法后的框架对 Ecperf 系统的运行时刻 SA 模型进行分析,验证加入容错风格对系统可靠性的提高,以展示集成框架的可行性和有效性.

本文第 1 节介绍集成框架结构与解决方案.第 2 节从元模型、模型和视图 3 个层次介绍解决方案,并介绍工具的实现.第 3 节介绍如何使用框架集成可靠性分析和基于容错风格的规划的 3 种分析方法,并应用于 Ecperf 系统的运行时刻 SA 模型,通过分析验证加入容错风格对于系统可靠性的提高,从而展示集成框架的可行性和有效性.第 4 节介绍相关工作.第 5 节讨论集成框架的适用范围和有效性.第 6 节总结全文并对未来工作进行展望.

1 框架结构与解决方案

合成 SA 分析结果与 SA 模型,目标是获得包含分析结果的 SA 模型.这个模型按照分析结果与 SA 模型元素的关系,将分析结果作为 SA 模型中元素的属性,与 SA 模型中原有元素组成新的模型.例如,合成 SA 模型以及构件可靠性分析结果得到的目标 SA 模型中,构件实例增加了可靠性的属性,其属性值是可靠性分析结果.由于 SA 模型与分析结果使用不同的元模型进行描述,是各自独立的模型,在不同的工具中进行查看和编辑.相对应地,支持分析方法结果集成的框架需要面对来自元模型、模型和工具这 3 个层次的挑战.

(1) 原有的 ADL 不能描述包含了分析结果的 SA 模型.分析结果可能是系统的正确性、一致性、质量属性、验证结果或者自适应的规划结果等.这些分析结果关注于不同的领域,不可能在一种 ADL 中考虑到所有的领域需求;而另一方面,描述分析结果的元模型也可能与 ADL 中描述同类信息的元模型不同.因此,框架需要对 ADL 中进行扩展以支持分析结果的描述;

(2) SA 模型和分析结果存储在各自独立的模型.由于 SA 分析方法是由研究人员根据不同的领域需求提出来的,通常并不特定于 SA 建模方法.这使得最终是以 SA 模型作为理解和解析上下文的分析结果,但却不是 SA 模型中的部分.因此,框架需要合成模型以得到包括分析结果的 SA 模型;

(3) 建模工具无法提供合成后模型的视图.建模工具是人与模型交互的主要方式之一,对自动化程度不高的软件自适应过程非常重要.由于扩展的 ADL 相比于原有的 ADL,增加了新的建模元素,而建模工具并不能支持这些新增的元素,也不能支持使用扩展后的 ADL 描述的合成后模型,因此,框架需要扩展建模工具来提供合成后模型的视图.

针对这 3 个挑战,本文提出一种分析结果的集成框架,总览图如图 1 所示.框架从元模型、模型和视图这 3 个方面,为集成的分析方法提供扩展后的 ADL、包含分析结果的 SA 模型及其视图.

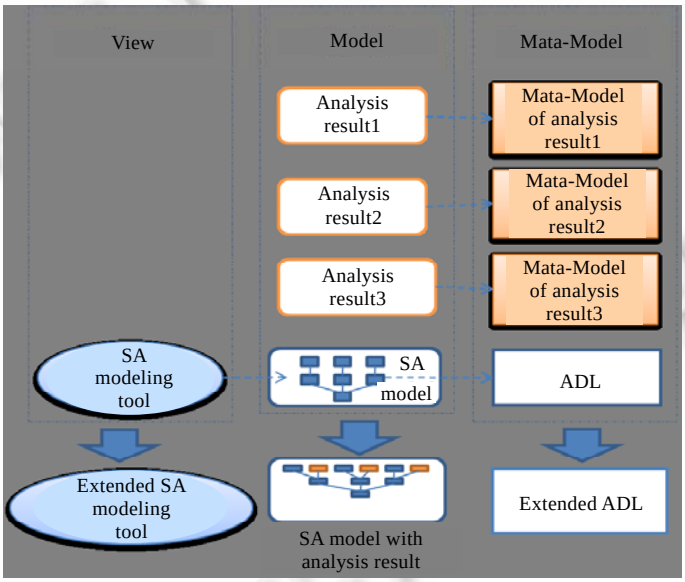


Fig.1 Overview of the integration framework

图 1 框架总览

在 ADL 的扩展方面,框架考虑元模型描述方式的约束和 ADL 的向上兼容性要求,基于 MOF 提出了一种 ADL 的扩展机制.分析方法的集成人员使用扩展机制描述分析结果与 SA 模型之间的关系,从而定义分析结果的集成方式.

在模型的合成方面,通过框架考虑分析方法的使用者不需要了解分析结果与 SA 之间的合成关系,关注为

模型合成提供自动化支持.框架选择模型转换来实现分析结果与 SA 模型的合成,使用每个模型转换实现一类分析结果的合成.在此基础上,框架根据分析结果集成方式的定义,通过解析元模型和元模型之间的关系生成模型转换的实现,并针对多种分析方法的情况进行优化.

在视图的扩展方面,框架考虑分析方法的集成和使用者都不了解建模工具的设计和构造,关注对建模工具扩展的自动化支持.框架使用自动代码生成技术,为没有建模工具的 ADL 构造基本的模型编辑器来提供模型视图;对已有建模工具的 ADL,在其建模工具中增加分析结果的显示和编辑功能,以提供模型视图.

2 框架实现

本节首先按照元模型、模型和视图这 3 部分介绍集成框架的解决方案,最后介绍框架的工具实现.

2.1 元模型扩展机制

为了使 ADL 能够支持对包含了分析结果的 SA 模型的描述,集成框架需要提供扩展的机制,并根据分析结果与 ADL 之间的关系动态地扩展 ADL,得到所需的 ADL.ADL 的扩展机制,必须满足如下两个方面的要求:

(1) 满足元模型描述方式所带来的约束.由于 ADL 和扩展后的 ADL 都是使用同一元建模技术来描述的,因此它们的描述方式以及扩展方式受到元建模技术所提供的建模概念的约束;

(2) 向上兼容性,即扩展后的 ADL 能够描述使用原 ADL 描述的软件体系结构.

集成框架考虑上述的两类要求,为 ABC ADL 提供扩展机制.ABC ADL 是一种支持构件组装的通用软件体系结构建模语言[7],为体系结构建模提供了核心元素的定义,并将其组织为资产(asset)、配置(configuration)和行为(behavior)这 3 类,用于描述可复用的软件体系结构资产、软件结构信息和行为信息.

在元模型的描述中,ADL 使用 MOF[8]进行描述.MOF 是由 OMG 组织定义的一种标准化的元模型建模方式,主要包括 4 类建模概念:(1) 类(classe):用于建模 MOF 的元对象(metaobject);(2) 关联(association):用于建模元对象之间的关系(binary relationship);(3) 数据类型(DataType):用于建模其他数据;(4) 包(package):用于对模型进行模块化.

因此,使用 MOF 描述的 ABC ADL 的扩展机制允许增加的建模概念只能为以上 4 种中的一种或者多种.分析结果在 SA 模型中,可能是模型的属性、元素或者 SA 模型中元素的属性.因此,扩展机制支持在 ADL 中增加类、数据类型和属性来定义新增的元素或属性,增加关联来建立新增元素与已有元素的关系.而由于原 ADL 已经包括了 SA 建模所需的主要元素,扩展的部分只是对建模能力的增强,并不需要修改或者增加元模型定义中的包结构.因此,扩展机制不允许在 ADL 中增加新的包,或者对包结构进行更改,新增的元素添加进原 ADL 根元素所在的包.

为了保证 ADL 的向上兼容性,使得扩展后的 ADL 能够描述使用原 ADL 描述的 SA 模型,扩展机制需要保证对 ADL 的扩展不会改变原 ADL 中的元素及元素之间关系的定义和语义.由于 ADL 的扩展是在已有的建模能力基础上增加建模能力,改变 ADL 不会删除或者修改已有元素以及它们之间的关系.向上兼容性的要求主要表现在扩展机制中对新增元素与已有元素之间关系的约束上:在扩展机制中,要求增加的元素都必须是可选而不是必须的;在扩展时增加的元素之间的关系不能是在两个已有元素之间的关系,而必须是在两个新增元素之间或者一个新增元素与原有元素之间的关系.

基于以上两个方面的要求,集成框架提供的 ABC ADL 扩展机制支持:

(1) 增加元素定义,也就是类的定义;

(2) 增加已有元素的类属性定义;

(3) 增加元素之间关系的定义,其建立的关系包括:组合(composite)和聚合(aggregate);

(4) 增加数据类型的定义.

扩展的定义包括名字(name)、类型(type)、扩展点(extension point)、多重性(multiplicity).对于元素,名字用于在建立元素与原有元素之间关系时标识元素;对于属性,名字用于属性命名.定义中的名字在包含这个元素或者属性的元素定义中必须是唯一的.例如,当为 ADL 中的 ComponentInstance 元素增加属性 reliability 来记录其

对外提供服务的可靠性时,这个名字 reliability 必须与 ComponentInstance 中包含的其他属性和元素名不同.扩展点指定扩展的对象,而多重性是扩展点与分析结果之间关系的属性,用于描述一个扩展点对应的分析结果的数量.

元素的类型可以使用 MOF 中已定义的原子类型、原 ADL 中除了 SA 类型以外的所有类型或者自定义类型.前两种类型是预定义类型,用户在使用时只需指定类型名,无需重新进行定义;自定义类型需要由用户提供类型名,并使用 MOF 描述类型定义.自定义类型可以是 MOF 支持的 4 种类型中的任何一种:枚举类型(enumeration type)、结构类型(structure type)、集合类型(collection type)和别名类型(alias type).在定义时,用户可以通过引用类型名使用预定义类型.用户自定义类型的类型名在集成框架必须是唯一的,其名字不能与 MOF, ADL 中已有类型或者其他用户自定义类型同名.

例如,在 ADL 元素 Topology 中增加子元素 tobeConfig 来描述对运行系统进行自适应规划之后的目标配置图,由于规划的配置图也是描述构件连接子之间的拓扑结构,用户可以直接使用 ABC ADL 中已有的 Config 类型作为扩展的 tobeConfig 元素的类型.如果 MOF 和 ADL 中已有的类型不适用于扩展内容的描述,用户可以使用 MOF 重新进行定义.

在 MOF 中支持的元素之间的关系包括如下 3 种:组成、聚合以及继承.由于增加原 ADL 中元素之间的关系会影响向上兼容性,因此扩展机制对新增的元素之间关系的支持只包括两种:新增元素之间的关系和新增元素与 ADL 中已有元素之间的关系.当用户自定义类型时,扩展机制允许在定义中的元素之间建立组成、聚合、继承这 3 种关系中的一种或者多种.由于在自定义类型定义中,无论使用新的定义或者复用预定义类型所创建的都是新的元素,并不会影响 ADL 中已有的元素.因此,扩展机制不限制关系的增加.当用户建立新增元素和 ADL 中已有元素之间的关系时,扩展机制只允许建立组成和聚合关系.因为新增元素已有对应的类型定义,它不能与 ADL 元素之间增加继承关系.在此基础上,扩展机制要求每一个扩展的元素需要建立与 ADL 中至少一个元素之间的组成关系,这个原 ADL 中的元素称为扩展点.扩展点可以是 ADL 中各个层次中的任意一个类元素,包括 ADL 的根元素 SA.如果扩展的元素与已有的 ADL 中其他元素不存在直接的关系,可将其作为新的组成部分加入根元素 SA 中.另一方面,扩展机制不允许建立从原有元素到新增元素之间的组成关系.例如,不能增加一个新的元素,其中包含 ADL 的根元素 SA.

2.2 自动模型合成

SA 模型和分析结果分布在各自独立的模型中,需要将其合成以获得扩展后的模型.框架关注于为模型的合成提供自动化支持,需要考虑的主要问题包括:选择合适的方式实现模型合成、自动生成模型合成所需实现、在运行中执行自动的模型合成.

模型合成需要复制模型中的信息,并按照其元模型之间的关系将其组织成目标模型.对于同一种分析方法,虽然每一次分析后需要合成的模型实例是不同的,但是 SA 模型和分析结果之间的关系是确定的.这种关系由分析结果和 ADL 与目标模型之间的关系来决定,也是由分析结果元模型与 ADL 之间的关系来决定.模型转换提供了一种描述如何从一类元模型获得目标模型的方法,适用于实现由元模型关系决定的模型合成.使用模型转换描述一类分析结果与 SA 模型之间的合成,可以一次定义之后应用于对不同 SA 模型的分析结果集成,有利于复用和减少开发的代价.因此,本文中的方案使用模型转换来实现模型合成.

实现合成 SA 模型和分析结果的模型转换,需要复制 SA 模型中的元素,然后将分析结果按照对应关系加入到其扩展点中,形成符合扩展后 ADL 的模型.模型转换需要维持分析结果与体系结构元素之间的对应关系,只涉及元素的复制,并不需要额外的计算过程.例如,一个构件可靠性分析方法,分析结果名字为 reliability,类型为 Double,表示每个构件实例的可靠性属性.使用这种分析方法对包含两个构件实例的 SA 模型进行分析,得到两个 reliability 的分析结果,假设分别为 0.9 和 0.5.模型转换在复制每个构件实例(ComponentInstance)的信息时,需要从分析结果中复制 reliability 的数值作为构件实例的属性.合成的目标模型中包含两个构件实例 *A* 和 *B*,其中,*A* 的 reliability 属性为 6,*B* 的为 3,代表其各自可靠性.

在实践中,实现模型合成的模型转换的逻辑并不复杂.但是由于 ADL 中定义的元素较多,例如在 ABC ADL

中的元素定义就有 145 个,用户需要编写模型转换语句来复制每个元素的属性和关系,不仅耗时且容易出错.因此,框架自动生成模型转换来减少集成的工作量.

由于模型转换描述的合成模型时所需要保持的对应关系是由扩展 ADL 时所描述的 ADL 与分析结果元模型之间的对应关系决定的,因此这里,本文借鉴模型转换研究中将模型转换当作一个模型的思想[9],如图 2 所示,将扩展定义中的 ADL 与分析结果的关系作为模型,通过实例化来获得模型的合成所需要的模型转换描述.

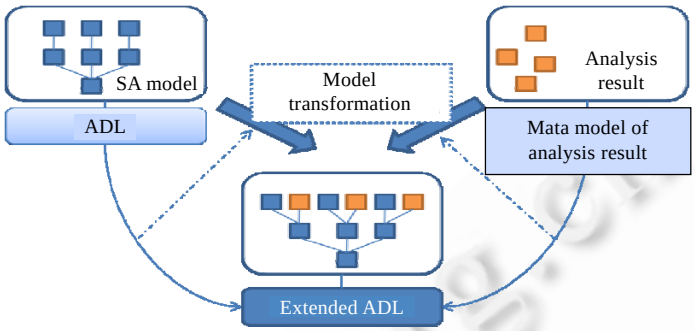


Fig.2 Relationship between SA model and analysis results

图 2 SA 模型和分析结果的关系

算法的思想是:首先解析 ADL,对每个元素定义生成对应的复制语句;之后,解析分析结果的元模型 AR_MM,对每个元素定义生成对应的复制语句,并将其插入到扩展点所对应的复制语句后.由于 SA 模型和分析结果分属于独立的模型,在复制过程中需要为来源于不同模型的复制语句生成相应模型的标识.转换后的目标模型使用扩展后的 ADL 作为元模型.

算法的输入包括:ADL、分析结果的元模型 AR_MM、集成定义中的分析结果名字 name 和扩展点 extensionPoint.GenerateCode(function,element,model)用于生成模型转换代码,从特定模型 model 中通过特定方式 function 获得元素 element,或者生成特定用途的代码.具体的算法如下:

```
Initialization: GenerateCode(transformationFileHeader)
For e in ADL
    GenerateCode("copy",SubElements(e),SAmodel)
    GenerateCode("copy",Attributes(e),SAmodel)
    If e is extensionPoint
        GenerateCode("copy",name,analysisResultFile)
        endif
endfor
For e in AR_MM
    GenerateCode("copy",e,analysisResultFile)
 endfor
```

实现模型合成的模型转换逻辑较为简单,目前主流的模型转换语言都能够满足其实现要求.在本文的集成框架中,选择了应用较广并且有成熟的编译执行工具的 ATL[10]模型转换语言.在运行时刻,框架通过使用 ATL 的执行引擎来进行模型转换.框架生成和执行模型转换的过程对于用户是透明的.

对于 SA 模型上的多种分析方法,框架可以通过生成并顺序执行每种分析方法的模型转换,从而得到集成多种分析结果的模型.由于增加一种方法需要多执行一次模型合成,集成分析结果的时间随分析方法的个数呈线性增长,而且多次的模型转换中存在着对模型中的某些元素的重复复制.考虑到分析结果与 SA 模型之间以及分析结果之间存在着相对的独立性,可以对集成多种分析结果的模型转换进行优化.

分析结果与 SA 模型之间以及分析结果之间存在着相对独立性,分别表现为:

(1) 分析结果与原有的 SA 模型之间是相对独立的.分析的结果是对 SA 信息进行分析推理后得到的,但分析的过程不会影响 SA 模型的信息;

(2) 应用于同一个 ADL 的不同分析方法的结果之间是独立的,不会互相影响.因为分析方法由不同的设计人员定义,如果分析方法 *A* 依赖于分析方法 *B*,那么分析方法 *A* 应该是在扩展了分析方法 *B* 的结果的 ADL 上所定义的分析方法.

分析结果与 SA 模型之间的独立性保证了 SA 模型作为分析方法的输入不会在分析过程中被修改.因此,无论集成单个还是多个分析结果,模型转换中均包含相同的复制 SA 模型中原有的元素的赋值语句.分析结果之间的独立性,使得对不同分析结果的赋值不会互相影响.因此,模型转换中复制不同分析结果的语句的执行顺序不会影响转换结果.

框架在集成多种分析结果时,使用优化生成算法生成模型转换来实现合成.算法与生成集成单个分析结果的模型转换算法类似,所不同的是,算法需要解析多个分析结果的集成定义,在对应扩展点插入分析结果的复制语句后.对应于上述生成集成单个分析结果的模型转换的算法,算法在第 4 行判别扩展点的部分,需要判别多个扩展点组成的列表,生成从对应分析结果文件中获取元素的模型转换语句.生成的模型转换以 SA 和多个分析结果模型作为输入,生成集成多种分析结果的模型.与多次模型转换相比,优化后的模型转换减少了对原 SA 模型中元素的多次复制,执行时间随分析结果的数量和复杂程度有所增加.

2.3 建模工具扩展的代码生成

建模工具是用户与模型之间进行交互的主要方式之一.由于无论是分析方法的集成人员还是使用者都不了解建模工具的实现,因此框架需要提供视图来支持扩展后的 ADL,而不需要用户去理解或者实现建模工具本身.集成框架根据用户对视图扩展的需求,使用代码自动生成技术,分别为不存在建模工具的 ADL 和已有建模工具的 ADL 提供所需的视图.

建模工具一般包括两个部分:模型编辑器和图形化视图.SA 建模工具通常提供一个模型编辑器,辅助用户查看和编辑模型中的元素和属性.在此基础上,工具提供 SA 模型中的一阶实体,也就是构件和连接子的可视化的线框图,一般称为配置图.由于扩展的 SA 分析结果是对 SA 模型的补充,并不是模型中的一阶实体,因此,其视图由建模编辑器提供,要求对建模工具的模型编辑器进行扩展.

模型驱动的软件工程为特定领域的工具和应用开发提供了很多帮助.对于没有建模工具支持的 ADL 而言,集成框架使用 Eclipse 的开源项目 Eclipse 建模框架(Eclipse modeling framework,简称 EMF)[11]来自动生成 ADL 的视图.EMF 是一个支持基于结构化数据开发建模工具的建模框架,它为工具开发提供代码生成的基础设施.EMF 以一个使用 XMI 描述的模型为输入,生成一系列模型所需的 Java 类及其所需的辅助类.EMF 提供工具生成代码,并且对代码提供运行时刻的支持,从而得到能够查看和编辑模型的建模工具.

对于已有建模工具支持的 ADL,例如 ABC ADL,其建模工具除了提供基本的编辑功能外,还提供模型分析、检查、运行时刻管理等功能.由于这些建模工具为建模语言的使用提供了方法学上的支持,因此框架通过对已有的建模工具进行扩展来提供扩展后 ADL 的视图.工具的扩展关注两方面的要求:一方面是在模型编辑器中提供扩展后建模元素的视图,另一方面是保持已有的建模能力,不影响原有功能.

框架在 EMF 框架及其提供的代码生成功能的基础上,根据 ADL 的扩展生成 ABC TOOL 扩展的代码.ABC TOOL 是一个基于 EMF 和 GMF 的 Eclipse 插件,其模型编辑器使用 EMF 生成基础设施的代码,代码组织为模型、适配器和编辑器这 3 个层次.因此,框架针对 ADL 的每个扩展增加对应的模型和适配器代码,并在相应扩展点中增加子元素定义.由于 EMF 框架支持根据元模型的变化增量生成代码,也就说,对于增加了方法或者变量的元模型,EMF 可以生成支持新增元素的建模工具,并保证在这个过程中不影响原有代码和原有的功能.集成框架利用这个机制,将扩展后的 ADL 作为修改后的元模型,之后使用模型和适配器代码生成的相应接口,生成 ABC TOOL 视图扩展的代码.

2.4 集成框架的工具实现

在本文的解决方案中,基于 ABC ADL,使用了 MOF 元建模技术、模型转换、自动代码生成等技术.在框架工具的实现中,也相应集成 ABC TOOL、Eclipse Ecore、ATL 引擎和 Eclipse EMF 等支持工具,为框架的使用者提供工具支持.

框架的使用者包括两类:集成人员和用户.集成人员根据分析方法及其结果的表示方式,定义分析结果的元模型及其与 ADL 中其他元素的关系;框架的最终用户是软件自适应设计人员或者维护人员,他们根据自适应需求选用所需的分析方法,并且使用扩展后的 ABC TOOL 编辑使用扩展后 ADL 描述的、包含了分析结果的软件体系结构模型.

框架的实现结构如图 3 所示.左部是框架所需的输入,包括由集成人员提供的分析方法定义、由分析方法使用者给出的 SA 模型和分析结果实例.SA 模型可以是设计阶段的制品或者由监测运行系统得到.分析结果是分析方法的输出,分析方法的实现可以使用框架支持的方式定义,也可以是外部工具.ABC ADL 和 ABC TOOL 是框架中预置的 SA 建模语言和工具.模型转换文件是框架生成的内部文件,它对于用户是透明的.图 4 右部是框架的输出,包括包含分析结果的 SA 模型、扩展后的 ADL 和 ABC TOOL.

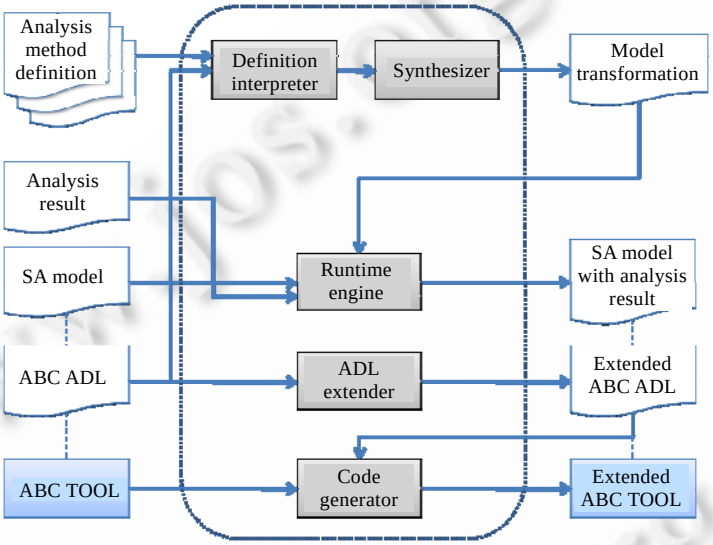


Fig.3 Architecture of framework implementation

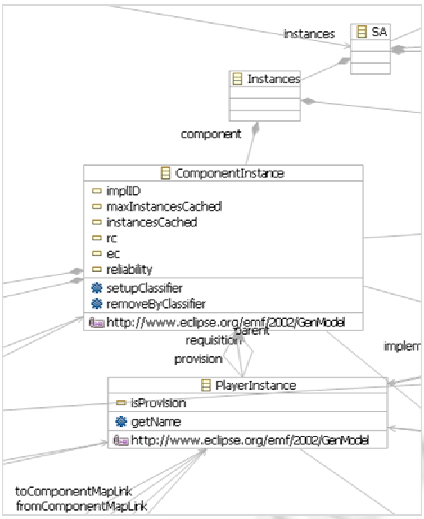
图 3 框架实现的体系结构

框架的实现包括 5 个构件:

- 定义解析器(definition interpreter):框架中基础设施模块,负责解析分析方法定义和 ADL.其中,分析方法定义组织为 XML 文件,由集成人员通过用户界面输入并存储;MOF 定义的 ABC ADL 使用 Ecore 存储.解析器集成 XML 解析器以及 EclipseEcore 插件,为 ADL 的扩展和模型合成提供所需信息;
- 合成器(synthesizer):用于生成模型合成所需的模型转换文件的模型.其输入包括分析方法的集成相关定义和 ADL,生成合成 SA 模型和分析结果的模型转换.其具体算法如第 2.2 节中所述,可以生成单个或多个分析结果集成所需的模型转换.生成的模型转换使用 ATL 语言进行描述,存储为.atl 文件;
- 运行引擎(runtime engine):运行时刻执行模型转换的模块,负责在输入的 SA 模型或分析结果发生改变时,执行由合成器生成的模型转换.引擎中集成了 ATL 引擎以实现解析和执行 ATL 描述的模型转换;
- ADL 的扩展器(ADL extender):实现 ABC ADL 扩展机制的模块,其输入包括 ADL、分析方法定义中的分析结果元模型以及扩展需求,输出扩展后的 ADL.由于分析结果是 Ecore 所支持的原子类型,或者是使用 Ecore 描述的复杂类型,而 ADL 也是使用 Ecore 描述的,因此,对 ADL 的扩展是在其定义文件中增

加相应的元素定义,并建立两者之间的关系.扩展器集成 Eclipse 中已有的工具包,使用其编辑相关的接口来实现;

- 代码生成器(code generator):负责管理和扩展 ABC TOOL 代码的模块.生成器基于扩展后的 ADL,在 ABC TOOL 的代码中增加扩展部分的模型编辑器代码.生成器的实现集成了 EMF 代码生成的 codegen 部分,解析 ADL 扩展中增加的分析结果定义,并调用 codegen 中相应的接口来增加代码.




```
rule generateConfig{
   erom Config: adllConfig()
to tobeConfig: adllConfig{
   name←Config.name,
   component←Config.component,
   connector←Config.connector,
   link←Config.link,
   componentMapLink←Config.componentMapLink,
   connectorMapLink←Config.connectorMapLink,
   componentinstanceFlag←Config.componentinstanceFlag,
   connectorinstanceFlag←Config.connectorinstanceFlag,
   designPath←Config.designPath
   }
   do {
   for {c in tobeConfig.component}{
   c.reliability←Config.component→at{tobeConfig.component→
   indexOf(c)→reliability(1.0,10.0,Config.link);}
   }
}
```

Fig.4 Extended ADL and model transformation file generated by the integration framework

图 4 框架生成的扩展后 ADL 和模型转换文件

3 实例研究

关注可靠性的软件,需要分析系统运行时刻的可靠性,在可靠性不满足要求时加以调整.调整过程中,软件还需要在已有的 SA 上进行规划,并评估调整方案的可靠性,确保调整后的系统能够满足可靠性要求.本文针对这一场景,选择两种可靠性分析方法[12,13]和基于容错风格的软件体系结构规划方法[14].其中,前两种方法用于对软件的配置进行可靠性分析和预测,第 3 种方法基于软件体系结构风格生成具有容错能力的体系结构配置方案.本节分别介绍集成人员如何使用框架集成这 3 种方法,并介绍用户如何使用单个方法、并行或者串行使用多种方法分析 Ecperf 系统的运行时刻 SA 模型.ECPerf 系统是 J2EE 的一种参考实现.

3.1 集成和使用SBRA可靠性分析方法

SBRA 可靠性分析方法以软件体系结构中的构件和连接子配置图作为输入,根据每个构件的可靠性(rc)和执行时间(ec)、每个连接子的可靠性(rk)和连接可能性(pk)以及配置情况来计算系统的可靠性.为了更好地定位各个构件对可靠性的影响,本文使用这种方法计算每个构件的可靠性,用以表示以这个构件为开始点的系统可靠性,方法细节可参考文献[12].

3.1.1 集成阶段

分析方法集成人员通过工具界面定义分析结果的名字(name)、类型(type)、扩展点(extension point)、多重性(multiplicity)等,集成人员也可以使用模型转换定义分析方法得到工具对于分析方法定义以及依赖元素提示等辅助性的支持.对于可靠性分析方法,集成人员定义分析结果 reliability 是一个 Double 类型的实数,作为 SA 中构件实例(component instance)元素的属性.该分析方法具有 ATL 描述的实现,集成人员将其加入,并确认存储.

框架在用户确定之后,首先存储集成相关的定义,并对 ADL 的扩展和模型的合并进行预处理.框架生成的制品包括一个用于记录集成相关定义的 xml 文件、扩展后的 ADL 定义、使用 ATL 描述模型合并的文件.这些

文件按照一定的方式组织存储在项目工作区中.扩展后能够描述构件实例可靠性的 ADL,如图 4 左部所示,元素 ComponentInstance 中增加了 Double 类型的属性 reliability.生成的合成模型所需的模型转换文件,如图 4 右部所示,受篇幅所限,图中只截取生成配置相关的模型转换.模型转换中的赋值语句完成对元素 Config 中所有属性和子元素的复制,“do”部分代码则用于生成配置图中的每个构件实例的 reliability 属性.代码中调用的 reliability 函数即为使用模型转换实现可靠性分析的方法.

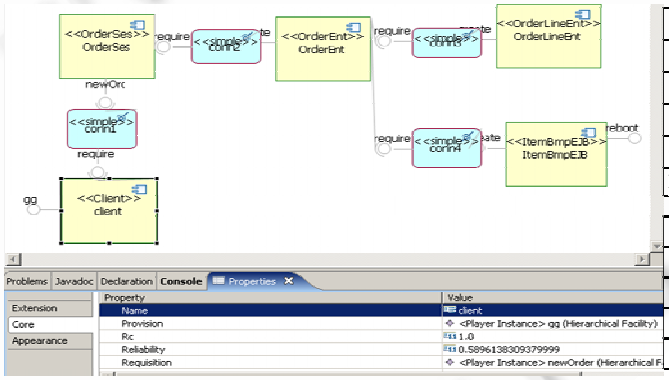
3.1.2 用户选择和使用分析方法阶段

框架扩展 Ecore 编辑器,在编辑器中增加一页作为用户选择分析方法的界面.在这个页面中,框架将集成阶段定义的分析方法以列表的形式组织起来,供用户选择.用户可以使用编辑器打开 ADL,进入分析方法页面选择所需的分析方法.此时,用户看到列表中的可靠性分析方法,并选择使用这种方法.

用户选择了分析方法之后,框架在元模型、模型和工具这 3 个层次对分析方法的集成进行处理.由于用户只选择了可靠性分析方法,因此框架可以使用预生成的扩展 ADL 作为元模型,使用模型转换文件来进行模型合成.在工具层,框架基于扩展后的 ADL,对 ABC TOOL 源代码进行增加.框架遵循 EMF 的代码框架,增加的代码集中于两个插件:处理模型操作的插件 cn.pku.edu.abcadl 和处理模型适配的插件 cn.pku.edu.abcadl.edit.由于框架在 ADL 中增加元素 ComponentInstance 的属性来支持可靠性描述,相对应地,框架在这两个插件中的元素 ComponentInstance 的模型类和适配类中,分别增加变量和函数来实现显示和编辑 reliability 属性的功能.

本文使用运行时刻 SA 管理框架 SM@RT[15]获得 Ecperf 系统的运行时刻 SA 模型,并将其作为输入.在 Ecperf 系统运行时刻,每个构件的可靠性(rc)、执行时间(ec)、每个连接子的可靠性(rk)和连接可能性(pk)以及分析之后得到的每个构件的可靠性(reliability)如图 4 右部所示.分析结果中,client 构件对外提供服务的可靠性为 0.589 6,不能满足用户的需求.主要问题在于:构件 ItermBmpEJB 本身的可靠性较低,只有 0.5;而构件 client 的 70%的请求依赖于这个构件提供的服务,导致构件 client 本身的可靠性也随之降低.

在运行时刻,集成框架首先对用户输入的 SA 模型进行模型转换,得到包含 reliability 属性的 SA 模型,并使用扩展后的 ABC TOOL 将其打开.用户界面如图 5 所示,可视化图形界面与原 ABC TOOL 相同,但在模型编辑器和图元的属性列表中增加了分析结果部分.可以看到,在构件实例的属性列表中,增加了属性 reliability.当 SA 模型发生变化时,框架会重新执行分析和模型的合并,更新模型.



构件名	Rc	Ec	Reliability
Client	1.0	1.1	0.589 6
OrderSes	0.99	2.3	0.595 6
OrderEn	0.99	1.3	0.607 7
OrderLineEnt	0.99	1.2	0.99
ItemBmpEJB	0.5	1.3	0.5

连接子名	Rk	Pk
Conn1	0.99	1.0
Conn2	0.99	1.0
Conn3	0.99	0.3
Conn4	0.9	0.7

Fig.5 SA model of system Ecperf including reliability analysis result

图 5 包含可靠性分析结果的 Ecperf 系统体系结构模型

3.2 集成和使用ABSRM可靠性分析方法

ABSRM 可靠性分析方法以软件体系结构中的构件和连接子配置图作为输入,根据 SA 风格建立 SA 对应的状态图,然后根据每个构件的可靠性(rc)和连接可能性(pk)以及配置情况来计算系统的可靠性.方法细节可参考文献[13].为了更好地定位各个构件对可靠性的影响,本文使用这种方法计算每个构件的可靠性,用以表示以这个构件的状态作为开始点的子系统的可靠性.

3.2.1 集成阶段

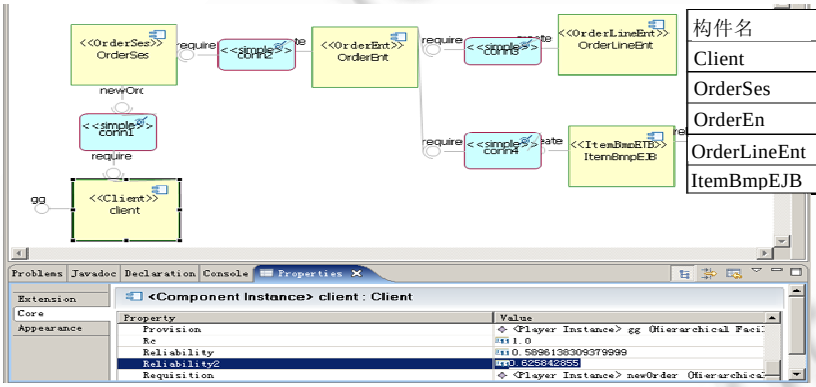
对于 ABSRM 可靠性方法,集成人员定义分析结果 reliability2 是一个 Double 类型的实数,作为 SA 中构件实例(component instance)元素的属性.

框架在用户确定之后,存储集成相关的定义,并对 ADL 的扩展和模型的合并进行预处理.在扩展后的 ADL 中,元素 ComponentInstance 中增加了 Double 类型的属性 reliability2.框架同时生成了合成模型所需的模型转换文件.

3.2.2 用户选择和使用两种可靠性分析方法阶段

在第 3.1 节和第 3.2 节中,集成人员已经集成了两种可靠性分析方法.用户可以在分析方法选择页面看到包含这两种分析方法的列表,并且可以同时选择这两种可靠性分析方法.在使用多种分析方法时,框架不复用预生成的扩展 ADL 和模型转换文件.框架在元模型、模型和工具这 3 个层次对分析方法的集成进行处理.在生成的扩展后的 ADL 中,元素 ComponentInstance 中增加了属性 reliability 和 reliability2.在生成的模型转换文件中增加了计算可靠性 reliability 和 reliability2 的 ATL 代码.在扩展后的 ABC TOOL 中增加了对 ADL 中扩展的两类分析结果的显示和编辑功能.

本文仍使用 Ecperf 系统的运行时刻 SA 模型作为输入.经过分析后,加入了两种具有可靠性分析结果的 SA 模型,如图 6 所示.可以看到,在模型编辑器中所选择的 client 构件的属性列表中,增加了 reliablity1 和 reliability2 两个属性.图 6 右部的表格显示了使用不同分析方法得到的可靠性评估结果,两种分析结果比较相近.这是因为,这个子系统中构件是顺序执行的,基于运行序列和基于状态的评估方式对系统的评估影响不大.其中差异产生的原因主要是由于 ABSRM 方法只考虑构件的可靠性(rc)和连接可能性(pk),而没有考虑连接子的可靠性(rk)所致.



构件名	Reliability	Reliability2
Client	0.589 6	0.589 6
OrderSes	0.595 6	0.595 6
OrderEn	0.607 7	0.607 7
OrderLineEnt	0.99	0.99
ItemBmpEJB	0.5	0.5

Fig.6 SA model of system Ecperf including two reliability analysis results

图 6 包含两种可靠性分析结果的 Ecperf 系统体系结构模型

3.3 集成和使用基于容错风格的规划方法

基于容错风格的软件体系结构规划方法以用户描述的体系结构风格和软件的配置图作为输入,输出加入容错能力后的目标配置图.其中,风格描述作为 SA 模型中资产(asset)中的元素,包括风格要实施的实例以及风格的目标模型的描述.方法细节可参考文献[14].

3.3.1 集成阶段

对于基于容错风格的规划方法,分析方法集成人员定义分析结果 tobeConfig,用于描述规划结果,也就是目标的 SA 配置图.集成人员使用原 ADL 中定义的类型 Config 作为元素 tobeConfig 的类型,把元素作为 SA 中拓扑结构元素 topology 的子元素.添加分析结果 tobeConfig 定义的方式与添加可靠性分析方法时类似.

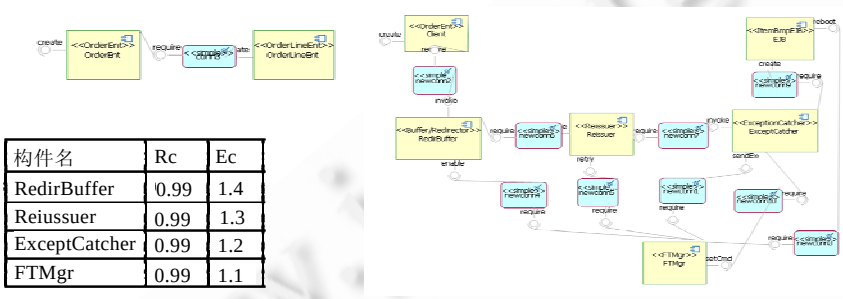
框架在用户确定之后存储集成相关的定义,并对 ADL 的扩展和模型的合并进行预处理.在扩展后的 ADL

中,元素 topology 中增加了 Config 类型的子元素 tobeConfig.同时,框架生成了合成模型所需的模型转换文件.

3.3.2 用户选择和使用分析方法阶段

当用户在分析方法选择界面中选择了基于容错风格的规划方法之后,框架在元模型、模型和工具这 3 个层次上进行处理.框架使用预生成的扩展 ADL 作为元模型,使用模型转换文件来进行模型合成.之后,框架基于扩展后的 ADL,在处理模型操作插件 cn.pku.edu.abcadl 和模型适配的插件 cn.pku.edu.abcadl.edit 中生成扩展相关的代码.框架在 ADL 中增加元素 topology 的子元素,相对应地,需要增加显示和编辑这个子元素的代码.由于新增子元素的类型 Config 在原有的 ADL 中已有定义,也就是说,ABC TOOL 中已有 Config 类型的模型类和适配类,因此,框架只在元素 topology 的模型类和适配类中增加子元素定义的代码,然后复用 Config 类型的模型和适配处理的代码.

本文仍使用 Ecperf 系统的运行时刻 SA 模型作为输入.SA 资产中预定义的容错风格是简单重试风格,如图 7 所示,左部是需要进行容错的配置实例,中间是容错风格的描述.简单重试风格监测构件的请求,在失效的情况下重新向构件发送请求.风格的描述中还包括了对构件的可靠性(rc)和执行时间(ec)以及连接子的可靠性(rk)和连接可能性(pk)的预测值.



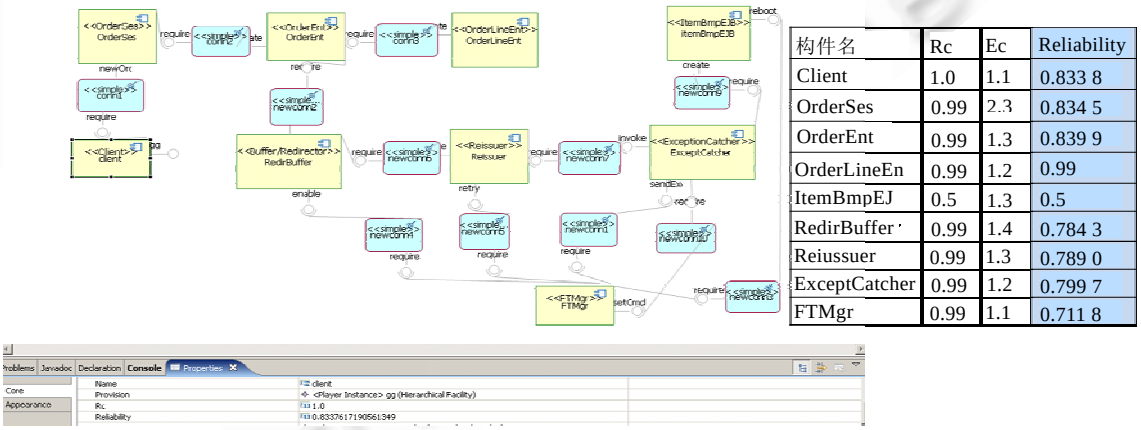
构件名	Rc	Ec
RedirBuffer	0.99	1.4
Reiussuer	0.99	1.3
ExceptCatcher	0.99	1.2
FTMgr	0.99	1.1

连接子名	Rk	Pk
newconn1	0.99	0.25
newconn2	0.99	0.7
newconn3	0.99	0.25
newconn4	0.99	0.25
newconn5	0.99	0.25
newconn6	0.99	1.0
newconn7	0.99	1.0
newconn9	0.9	1.0
newconn10	0.99	0.5

Fig.7 Fault-Tolerant architecture style

图 7 容错风格

在运行时刻,集成框架首先对用户输入的 SA 模型进行模型转换,得到包含了 tobeConfig 元素的 SA 模型,并使用扩展后的 ABC TOOL 将其打开.由于 Config 类型的元素在 ABC TOOL 中有对应的可视化编辑器,因此 Ecperf 系统 SA 模型的分析结果 tobeConfig 可以使用图形化的编辑器打开,如图 8 所示.



构件名	Rc	Ec	Reliability
Client	1.0	1.1	0.833 8
OrderSes	0.99	2.3	0.834 5
OrderEnt	0.99	1.3	0.839 9
OrderLineEn	0.99	1.2	0.99
ItemBmpEJ	0.5	1.3	0.5
RedirBuffer	0.99	1.4	0.784 3
Reiussuer	0.99	1.3	0.789 0
ExceptCatcher	0.99	1.2	0.799 7
FTMgr	0.99	1.1	0.711 8

Fig.8 Software architecture model including two kinds of analysis results

图 8 使用了两种分析方法之后得到的软件体系结构模型

3.4 串行使用两种分析方法进行分析

在第 3.1 节和第 3.3 节中,集成人员使用框架集成了可靠性分析方法和基于容错风格的规划方法.由于框架支持用户串行使用多种不同的分析方法,因此用户对于不能满足可靠性要求的 Ecperf 系统,可以先使用容错风格规划得到目标的配置图,然后再分析此规划结果可能的可靠性.

用户首先使用基于容错风格的规划方法分析 Ecperf 的 SA 模型,其步骤与第 3.2 节相同.之后,用户继续选择可靠性分析方法,对所有配置图中的构件实例进行分析.由于使用两种分析方法,框架不能复用预生成的扩展 ADL 和模型转换文件.框架在元模型、模型和工具这 3 个层次对分析方法的集成进行处理.在生成的扩展后的 ADL 中,元素 Config 中增加了子元素 tobeConfig,元素 ComponentInstance 中增加了属性 reliability.生成的模型转换文件实现对扩展了规划结果 tobeConfig 描述后的 ADL 中所有元素的复制,并且增加计算可靠性 reliability 的 ATL 代码.扩展后的 ABC TOOL 中增加了对 ADL 中扩展的两类分析结果的显示和编辑功能.

经过分析后,加入容错风格的配置结果如图 8 所示,表中是规划结果中每个构件的可靠性预测结果.其中,构件 OrdeLinerEnt 和 ItemBmpEJB 的可靠性与原 SA 模型相同,这是由于这两个构件不依赖其他构件提供的服务,其可靠性由构件本身的可靠性决定.Client 构件对外服务的可靠性从 0.589 6 增加到 0.833 8,这是由于加入的捕抓异常的构件 ExceptionCatcher 通过重试失效请求,提高了构件 OrderEnt 的可靠性.

4 相关工作

目前,对基于模型的分析方法的集成提供支持的研究工作已有很多,根据其目标和对分析方法集成的支持方式,可以分为如下两类:

第 1 类工作以框架为中心,为分析方法的设计和实现提出准则,使得按照这种方式定义和实现的分析方法能够比较方便地集成到框架中,代表性的工作包括属性驱动的设计方法(attribute-driven design,简称 ADD)[2]以及形式化设计分析框架方法(formal design analysis framework,简称 FDAF)[3]等.这类工作为分析方法的集成提供了有用的指南,也注重集成分析结果与 SA 中的信息来进行分析和决策.但是由于这些工作不能支持已有的分析方法,也缺乏工具支持,在实践中没有得到广泛接受;

第 2 类工作以现有的软件体系结构分析方法和工具为中心,关注于减少集成分析方法和工具的代价和工作量,代表性的工作包括 XTEAM[4],KAMI[5],DUALLY[6]等.其中,XTEAM 和 KAMI 都提出集成软件体系结构分析方法的框架.这些框架中提供了基础设施或者代码框架等,使得用户通过增加输入模型的定义、工具的调用方式的描述或者实现等,就可以在框架中集成和执行分析方法.DUALLY 则关注于集成不同 ADL 所提供的分析能力.DUALLY 支持用户描述不同 ADL 的映射关系,并使用这些映射关系自动生成模型转换.生成的模型转换将 SA 模型转换为使用其他 ADL 描述的 SA 模型,从而实现 ADL 之间语言和工具的互操作能力.这些工作为 SA 分析方法的集成提供了有力的支持,但是对于分析结果的集成关注不足.

由于软件自适应技术的发展和应用,分析方法集成中对于分析结果的合成需求更显重要.比照上述工作,本文的创新点在于关注对分析结果和 SA 模型的集成,关注合成中元模型、模型和工具这 3 个层次存在的问题.框架使用元建模技术、模型转换技术以及自动代码生成技术,为集成的分析方法提供扩展后的 ADL、包含分析结果的 SA 模型以及其辅助的建模工具.

5 讨 论

本文中分析结果的集成框架并不假设 ADL 和分析结果的描述形式或者存储方式,但在框架实现过程中特定 MOF 元模型描述标准来支持 ADL 和扩展与模型的合并.其中,ADL 和分析结果的元模型都使用 MOF 描述,分析结果存储为 XML 或者 XMI 文件.从原理上讲,本文中扩展机制的两个要求也同样适用于设计使用其他描述方式的 ADL 的扩展机制.对于使用其他方式定义元模型的分析结果,集成框架尝试着将其元模型转换为 MOF 描述,从而集成分析结果.例如,框架通过使用 Eclipse EMF 中的转换工具,将 Schema 定义转换为 MOF 描述的元模型,从而支持集成使用 Schame 定义模型组织方式的分析结果.

集成框架目前已集成的分析方法,除了本文实例研究中介绍的 3 种方法之外,还包括 COMPAS[16]和 JMT[17]等分析方法和工具的分析结果.在集成过程中,集成人员需要给出集成的定义,包括分析结果的名字、类型、扩展点以及多重性,并提供分析结果的元模型,也就是分析结果的类型定义.在此基础上,集成人员可以在框架的工具中使用模型转换实现分析方法.在实例研究中,我们使用大约 8 人时的方式来集成 SBRA 分析方法.这是由于 SBRA 分析方法并没有得到分析工具的支持,大部分时间被用于实现分析方法本身.分析结果集成的定义只包括 5 项,对于熟悉分析方法和 SA 模型的集成人员而言,给出定义并不困难.

集成定义是可复用的.分析方法使用者根据分析任务,可以在框架中选择串行或者并行地集成一个或者多个分析结果.由于 ADL 扩展满足向上兼容性,ADL 中已有的扩展和新的扩展不会互相影响.例如,两个分析结果集成定义分别对 ADL 中同一个元素进行扩展,扩展元素不会影响 ADL 已有元素或者已有元素之间的关系,两个扩展后的元素之间也不存在关系.也就是说,处于分析方法串行执行的不同阶段的分析方法,其集成定义只描述分析结果与 ADL 之间的关系,集成定义可以复用.集成框架通过将 ADL 更新为扩展了其他分析结果的 ADL,重新生成模型转换,使得分析结果的集成可以适应不同的分析方法的执行顺序.

6 总结与展望

软件在自适应过程中,使用多种分析方法来获得分析、规划和决策所需的分析结果,并以 SA 模型作为理解和解析分析结果的上下文.但是,分析结果和 SA 模型是各自独立的模型,这为分析结果的集成带来了挑战.本文关注于分析结果和 SA 模型的集成,提出一种基于 MOF 的分析结果集成框架.解决方案涉及元模型、模型和视图这 3 个层次,分别使用元建模、模型转换和代码生成等技术.框架实现以通用的软件体系结构建模语言 ABC ADL 及其建模工具为基础,通过集成分析方法,为用户提供扩展后的 ADL、包含分析结果的 SA 模型以及其辅助的建模工具.

本文提出集成框架目前只关注从分析结果的角度集成分析方法,对于分析方法和工具的集成支持仍然不够全面.下一步,我们将在已有工作的基础上,关注对分析方法提供输入、执行分析、从而获得分析结果的过程,重点解决集成过程中输入适配和多分析方法执行规划等问题.

References:

[1] Mei H, Huang G, Zhao HY, Jiao WP. A software architecture centric engineering approach for Internetware. Science in China (Series F: Information Sciences), 2006,49(6):702−730. [doi: 10.1007/s11432-006-2027-1]

[2] Bass L, Clements P, Kazman R. Software Architecture in Practice. 2nd ed., Westford: Addison-Wesley, 2003.

[3] Dai LR, Cooper K. Modeling and analysis of non-functional requirements as aspects in a UML based architecture design. In: Proc. of the 6th Int'l Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing and 1st ACIS Int'l Workshop on Self-Assembling Wireless Network (SNPD/SAWN 2005). Maryland: IEEE, 2005. 178−183. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1434886 [doi: 10.1109/SNPD-SAWN.2005.54]

[4] Edwards G, Medvidovic N. A methodology and framework for creating domain-specific development infrastructures. In: Proc. of the 23rd IEEE ACM Int'l Conf. on Automated Software Engineering. L'Aquila: IEEE, 2008. 168−177. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4639320 [doi: 10.1109/ASE.2008.27]

[5] Epifani I, Ghezzi C, Mirandola R, Tamburrelli G. Model evolution by run-time parameter adaptation. In: Proc. of the 2009 IEEE 31st Int'l Conf. on Software Engineering. Washington: IEEE Computer Society, 2009. 111−121. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5070513&tag=1 [doi: 10.1109/ICSE.2009.5070513]

[6] Malavolta I, Muccini H, Pelliccione P, Tamburri DA. Providing architectural languages and tools interoperability through model transformation technologies. IEEE Trans. on Software Engineering, 2010,36(1):119−140. [doi: 10.1109/TSE.2009.51]

[7] Mei H, Chen F, Feng YD, Yang J. ABC: An architecture based, component oriented approach to software development. Journal of Software, 2003,14(4):721−732 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/721.htm

[8] Object Management Group. Meta object facility (MOF) specification. Technical Report, Object Management Group (OMG), 2002. http://www.omg.org/spec/MOF/2.0

[9] Bézivin J, Büttner F, Gogolla M, Jouault F, Kurtev I, Lindow A. Model transformations? transformation models! In: Proc. of the MoDELS 2006. LNCS 4199, Berlin: Springer-Verlag, 2006. 440−453. http://www.springerlink.com/content/31pt5242j7745410/ [doi: 10.1007/11880240_31]

[10] Jouault J, Kurtev I. Transforming models with ATL. In: Proc. of the Model Transformations in Practice Workshop (MTIP), MoDELS Conf. Montego Bay: Springer-Verlag, 2005. 128−138. http://www.springerlink.com/content/7143g735r4j59463/

[11] EMF Project. 2010. http://www.eclipse.org/modeling/emf/

[12] Liu TC. Research on self-recovery of middleware services [Ph.D. Thesis]. Beijing: Peking University, 2006 (in Chinese with English abstract).

[13] Wang WL, Wu Y, Chen MH. An architecture-based software reliability model. In: Proc. of the '99 Pacific Rim Int'l Symp. on Dependable Computing. Hong Kong: IEEE, 1999. 143−150. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=816223 [doi: 10.1109/PRDC.1999.816223]

[14] Li JG, Chen XP, Huang G, Mei H, Chauvel F. Selecting fault tolerant styles for third-party components with model checking support. In: Proc. of the 12th Int'l Symp. on Component-Based Software Engineering. East Stroudsburg: Springer-Verlag, 2009. 69−86. http://www.springerlink.com/content/h65027670813351r/ [doi: 10.1007/978-3-642-02414-6_5]

[15] Huang G, Song H, Mei H. SM@RT: Applying architecture-based runtime management into Internetware systems. Int'l Journal of Software and Informatics, 2009,3(4):439−464.

[16] Parsons T. Automatic detection of performance design and deployment anti-patterns in component based enterprise systems [Ph.D. Thesis]. Dublin: University College Dublin, 2007.

[17] Bertoli M, Casale G, Serazzi G. The JMT simulator for performance evaluation of non-product-form queueing networks. In: Proc. of the Annual Simulation Symp. Norfolk: IEEE Computer Society, 2007. 3−10. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4127197 [doi: 10.1109/ANSS.2007.41]

附中文参考文献:

[7] 梅宏,陈锋,冯耀东,杨杰.ABC:基于体系结构、面向构件的软件开发方法.软件学报,2003,14(4):721−732. http://www.jos.org.cn/1000-9825/14/721.htm

[12] 刘天成.中间件服务自修复技术研究[博士学位论文].北京:北京大学,2006.

陈湘萍(1981－),女,广东潮州人,博士生,主要研究领域为软件体系结构.

孙艳春(1970－),女,博士,副教授,主要研究领域为软件工程,软件复用及构件技术,计算机支持的协同工作.

黄罡(1975－),男,博士,副教授,CCF 会员,主要研究领域为软件工程,分布式计算,中间件.

梅宏(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件复用和软件构件技术,分布对象技术.

宋晖(1983－),男,博士生,主要研究领域为软件体系结构.