ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.8, August 2006, pp.1707−1716
DOI: 10.1360/jos171707
© 2006 by *Journal of Software*. All rights reserved.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

多维关注分离的模型驱动过程框架设计方法*

段玉聪[1,2+], 顾毓清[1]

[1](中国科学院 软件研究所,北京 100080)

[2](中国科学院 研究生院,北京 100049)

A Multi-Dimensional Separation of Concerns Approach for Model Driven Process Framework Modeling

DUAN Yu-Cong[1,2+], GU Yu-Qing[1]

[1](Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: duanyucong@263.net, http://www.iscas.ac.cn

Duan YC, Gu YQ. A multi-dimensional separation of concerns approach for model driven process framework modeling. *Journal of Software*, 2006,17(8):1707−1716. http://www.jos.org.cn/1000-9825/17/1707.htm

Abstract: Currently, model driven software development processes largely take the form of the integration of several developing approaches. However comparing, selecting, composing of developing approaches usually rely on experiences lacking systemic guidelines. In this paper, a multi-dimensional separation of concerns approach for process frameworks constructing is proposed. Taking abstraction, generality, behaviorism as meta concerns, developing approaches are compared. Incorporated with the expected evolving curves of these meta concerns, implementing frameworks of developing processes are constructed. This work will be beneficial to meet nonfunctional requirements on model driven development processes like improving efficiency, traceability and ensuring consistency.

Key words: multi-dimensional separation of concerns; process framework; model driven development

摘 要: 当前,许多模型驱动软件项目过程采用多种开发方式相结合的形式.但开发方式的比较、选择、组合操作却缺少系统化的方法指导.提出一种多维关注分离的开发过程框架设计方法.采用一般化、行为化和抽象化作为元关注维,对开发方式进行比较.结合这三维的期望演化曲线,给出过程实现模型框架.对于模型驱动开发过程的提高开发效率、增强可跟踪性和保证一致性等非功能性需求有一定的参考意义.

关键词: 多维关注分离;过程框架;模型驱动开发

中图法分类号: TP311 文献标识码: A

随着 OMG MDA(model driven architecture)[1]和 UML(unified modeling language)[2]规范的发展,模型驱动开发 MDD(model driven development)[1]正在被越来越多的用户所接受.不同的组织从不同的角度和背景出发,提

* Supported by the Innovation Foundation of the Institute of Software, The Chinese Academy of Sciences under Grant No.K5CX045415 (中国科学院软件研究所创新基金)

Received 2004-07-22; Accepted 2005-12-13

© 中国科学院软件研究所 http://www.jos.org.cn

出了很多开发方式,如面向对象方式 OOD(object oriented development)、基于组件方式 CBSD(component-based software development)[3]、面向模式(pattern)方式 POD(pattern oriented design)[4]、面向服务(service)方式 SOA(service oriented architecture)[5]、面向特征(feature)方式 FOP(feature oriented programming)[3,6]、面向关注(concern)方式 COSA(concern-oriented software architecture)[7]、面向用户方式 USE(user-oriented software engineering)[8]、面向刻面(aspect)方式 AOSD(aspect-oriented software development)[9]、面向 Agent 方式 AOSE(Agent-oriented software engineering)[10]、基于角色(role)方式[11]、针对需求的用例(usecase)驱动方式[12]以及经典的面向功能(function)方式 FOD(function oriented design)等.

上述的许多开发方式能够对不同类型 MDD 生命期的一个或多个阶段提供支持.当前,许多 MDD 过程框架[13]采用多种开发方式相结合的形式,以期满足表达要求,获得综合效益.然而,利用不同开发方式构建 MDD 开发过程[14]框架存在两个主要问题:

(1) 不同开发方式在开发过程的不同阶段以及不同项目领域背景下,存在表达能力和效率上的差别.从整个开发过程效率提高的角度考虑,要求每一阶段的实现都选择效率最高的开发方式.这一问题的解决,客观上要求有一种对不同开发方式进行选择和比较的方法.

(2) 不同开发方式之间存在表达和作用范围的重叠(overlap).表达范围重叠包括概念共享(concepts sharing)[15],如不同的开发方式包含表达相同领域核心概念的子设计元素;作用范围重叠包括动态行为交错,类似于经典的面向刻面的交叉(crosscutting).这些重叠在实现过程中转化为表达信息冗余[16]后,可能导致随后模型演化过程中的不一致[16−18].除了对开发方式的比较和选择以外,在过程框架构建中恰当地组合各种开发方式,也有助于减少表达冗余.

针对以上问题,我们采用了多维关注分离(multi-dimensional separation of concerns,简称 MDSOC)[19,20]的方法进行分析和解决.不同开发方式的区别是多方面的,因而问题(1)的解决要求从不同利益相关者(stakeholder)的关注角度出发进行考虑.在开发过程中贯穿利用从问题(1)的解决中得到的分析结果来回避表达和作用范围的重叠,能够减少信息冗余.这个问题本质上是一个在开发过程中维持关注分离的问题.

以上问题解决的基础是恰当地关注体系的选取.受 Egyed[16]进行视图宏观比较的视图空间工作的启发,本文采用了各种利益相关者有共同自然语言理解基础的 3 个概念:行为化(behaviorism)、一般化(generality)、抽象化(abstraction)作为元关注维.基于这 3 个概念,从开发方式的静止比较和动态演化集成方面出发,构建了一个 MDD 过程框架设计原型方法,称为 BAG/SD(behaviorism, abstraction, generality/static, dynamic)系统.

本文第 1 节介绍行为化、抽象化和一般化概念.第 2 节给出对不同开发方式在比较空间的比较和定位.第 3 节基于 3 个元关注维的演化曲线模型给出过程框架设计方法.第 4 节给出一个过程框架示例.第 5 节进行与相关工作的比较.最后是对全文的总结和对下一步工作的展望.

1 一般化、抽象化、行为化概念

1.1 3个概念的解释

针对 MDD 过程特点,本文结合文献[16]和 UML[2]中的相关概念定义给出相关概念解释.

一般化表示更普通的元素和更详细的元素之间的分类关系.一般化在元模型中对应组合层次体系中可一般化元素和更普通元素的有向继承(inheritance)关系或定型(subtyping)关系.更详细的元素与更普通的元素充分兼容,并且可能拥有其他附加信息.一般化水平表示被讨论概念内部交流信息的普遍适用程度.

抽象化是一种连接代表不同抽象层次或不同视角的两个元素或元素集合的依赖关系.抽象在元模型中对应描述提供者(provider)和客户(customer)之间映射的依赖关系.根据具体的抽象的定型,这种映射可以是形式化或非形式化的,单向或双向的.UML 中抽象的标准定型类有导出(derive)、实现(realize)、求精(refine)和跟踪(trace),分别由定型〈〈derive〉〉,〈〈realize〉〉,〈〈refine〉〉和〈〈trace〉〉表示.抽象化水平对应视图中描述的信息被分解的程度(粒度水平).

行为化表达系统对象之间的动态行为,包括它们的方法、合作(collaboration)、活动(activity)和状态历史.

© 中国科学院软件研究所 http://www.jos.org.cn

系统的动态行为被描述为系统随时间的一系列变化.行为化水平对应视图所交流的交互信息的量.

1.2 概念的区分

(1) 从时间概念角度的一种区分

行为化与时间概念显式直接相关.一般化不涉及时间概念,只用于对象间的静态结构,描述规范中那些与时间无关的元素.抽象化主要用于对象间的静态结构,但有可能隐式涉及时间概念.静态结构中的元素表示对应用的有意义的概念,可能包括抽象、真实世界和实现概念.

(2) 在元模型中基于数据类型的操作意义区分

一般化操作基于显式维护操作对象和操作结果共同的类型标识的前提,其变化部分对应 gf 类型体系下的属性层次,具有一定的量变意义.一般化的作用结果形成了元素之间的类型继承体系.抽象化操作对应一般化以外的类型概念层次内部的组合、分解、裁减、生成等情形,具有一定的质变意义.

注意,在实际使用情形中,很多表达概念是混合使用的,只能从宏观和相对的意义上进行界定.例如,UML 中混合行为图和结构图元素展示嵌套在内部结构中的状态机的情形[2].还应当注意,这 3 个概念的区分仅仅从静止角度是不够的,实践中应当在过程环境中运动地加以考察.

2 开发方式比较与选择

2.1 开发方式比较

参照文献[16]视图空间的结构,对 3 个元关注构成的定位空间进行如下分区:抽象维被分为抽象和具体;一般化维被分为普通和明确;行为维被分为结构和行为.针对相关开发方式的核心概念的含义,对与 MDD 关系较密切的开发方式在分解空间 8 个区域进行归类,如图 1 和表 1 所示.

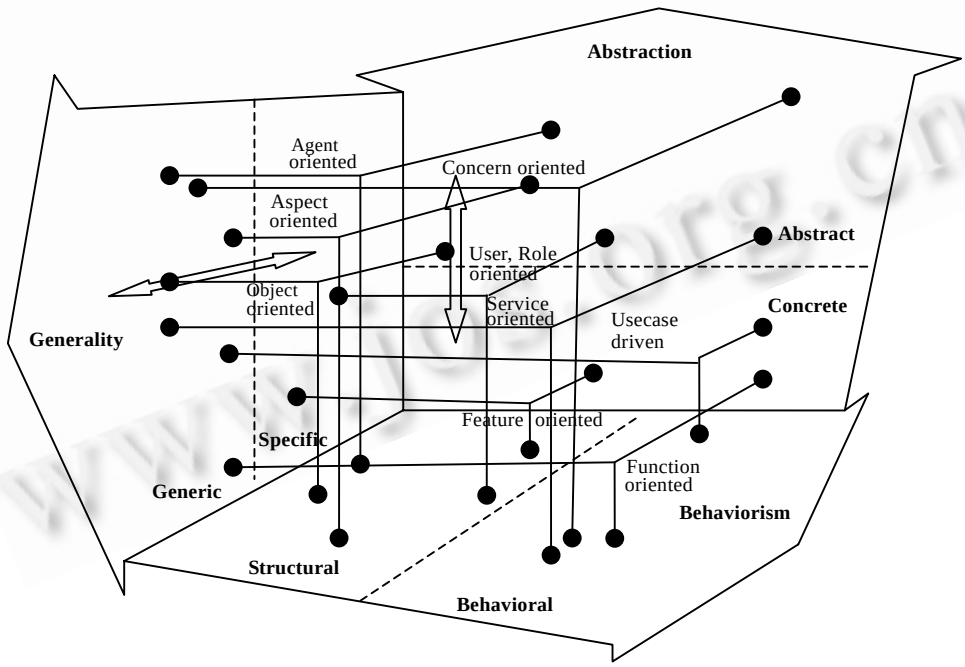

Fig.1 Locating space of developing approaches

图 1 开发方式定位空间

这些开发方式包括:面向对象方式 OOD、基于组件方式 CBSD、面向 Agent 方式 AOSE、面向服务(service)方式 SOA、面向功能(function)方式 FOD、面向接口方式 IDD、面向特征(feature)方式 FOP、面向关注(concern)

© 中国科学院软件研究所 http://www.jos.org.cn

方式 COSA、面向用户方式 USE、面向刻面(aspect)方式 AOSD、基于角色(role)方式、针对需求的用例(usecase)驱动方式等.面向模式(pattern)方式 POD 等由于其覆盖领域的相对广泛,故没有在这 8 个区域标出.

Table 1 Developing approaches in Fig.1

表 **1** 在图 1 中的开发方式

Area number	Abstraction level	Generality level	Behaviorism level	Candidate development techniques and approaches
(a)	Abstract	Generic	Structural	Object oriented, Agent oriented, Aspect oriented
(b)	Abstract	Generic	Behavioral	Service oriented, Concern oriented
(c)	Abstract	Specific	Behavioral	Component oriented (not labeled out)
(d)	Abstract	Specific	Structural	Role based, User oriented
(e)	Concrete	Generic	Behavioral	Function oriented, Usecase driven
(f)	Concrete	Generic	Structural	Data structure oriented (not labeled out)
(g)	Concrete	Specific	Structural	Feature oriented
(h)	Concrete	Specific	Behavioral	

这里的比较结果有助于在 MDD 过程不同阶段选择每一阶段表达能力和实现效率较高的开发方式.利用这些选择得到的开发方式构成的 MDD 过程框架能够获得较高的综合效益.同时,对不同开发方式的比较和选择也有助于对它们之间存在的表达和作用范围的重叠进行回避,减少静态信息冗余.当然,这里对表达范围重叠的解决主要集中在其静态形式即概念共享(concepts sharing)[15]部分的重叠上.

注意,各种开发方式在比较空间中的位置具有阶段相对性,即针对 MDD 的不同阶段开发方式在比较空间中的相对位置有一定的变化性.图 1 和表 1 只是示意性地展现了静态的简化比较结果.在实践中,要取得更准确的结果要求从运动的角度考察.像图 1 中面向对象方式的双向箭头所展示的那样,一些开发方式的定位可能在某些维度有一定的自由度范围.不同开发方式对应的项目性质(如大量使用现成类和模式的项目)也会对开发方式在分类空间中的定位产生影响.

2.2 关注分离演化

一般地,MDD 过程框架组成成分在第 2.1 节的 3 个元关注维的宏观变化方向为[16]:

(a) 一般化维,普通→明确;

(b) 抽象化维,抽象→具体;

(c) 行为化维,结构→行为.

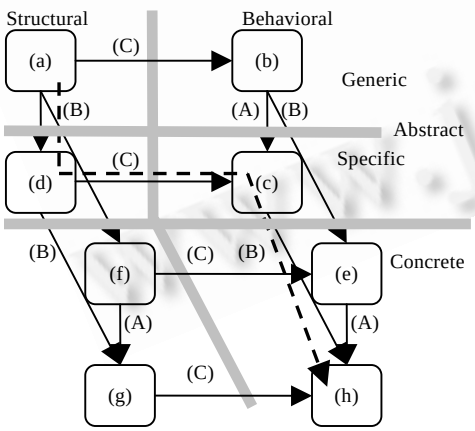

Fig.2 Transformation methods and paths

图 2 转换方法和路径

其中:一般化维与抽象化维的变化可以主要是信息的形式发生转换;而行为化维并不存在结构到行为的转换.在 MDD 过程中,信息从非形式化形态向形式化形态转变.在此过程中,结构化信息和行为化信息的绝对数量都是增长的.这里的变化表示被行为化和结构化信息所占比重的相对变化.从静止角度考察定位,表 1 中面向对象方式和面向刻面方式的宏观定位为抽象、普通、结构.它们对应图 2 中 3 种变化方向的起点区域(a),这意味着它们对 MDD 生命期过程简化模型其余 7 个区域存在潜在的表达替代能力.这也是后面过程框架示例中采用刻面方式辅助整个软件开发生命期关注分离的一个原因.

在简化模型中,当把表 1 中的区域(a)看作出发区域,把区域(h)看作目的区域时,从区域(a)到区域(h)在各个元关注维上至少需要一次转换/转化.因而,整个转换过程至少需要 3 个单独的简单变化步骤.图 2 是开发方式之间的简单转换路径示意,实线箭头表示简单转换.MDD 过程框架需要组合简单转换来保证连续性和可扩展性.这里称简单转换的集成和序列化执行为复杂转换.容易看出,共存在 6 条从区域(a)到区域(h)的不同的复杂转换,带箭头虚线所示

© 中国科学院软件研究所 http://www.jos.org.cn

“(a)→(d)→(c)→(h)”就是其中一条.

2.3 区域转换

开发方式的选择要满足区域转换的两个主要方面:

1) 区域集成应当转换所有冗余信息;

2) 中间区域需要捕获所有冗余信息.

对转换冗余有 3 种弥补方法[16]:使用附加的中间区域;增加新的转换;扩展存在的模型.图 3 显示了过程框架不支持从表 1 中区域(a)到表 1 中区域(c)的直接转换.代替地,用区域(d)作为中间区域,把该转换分为两个简单转换:“(a)→(d)”和“(d)→(c)”.

作用范围重叠包括动态行为交错,类似于经典的面向刻面的交叉(crosscutting).这些重叠在实现过程中转化为表达信息冗余后,会导致随后模型演化过程中的不一致.在过程框架构建中,恰当地组合各种开发方式有助于减少动态发生的行为交错引起的表达冗余.

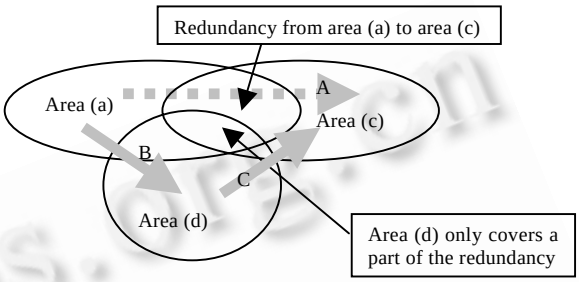

Fig.3 Areas transformation

图 3 区域转换

3 开发过程框架模型设计

3.1 元关注维演化曲线模型

组成 MDD 过程框架模型的开发方式在 3 个元关注维的演化过程既是灵活变化的,又在一定程度上符合一定的规律.下面展示一般情况下从运动角度考察 MDD 生命期的需求、架构设计[7,9,19,21]、设计、实现 4 个阶段 3 个元关注维的演化曲线模型.每一维的演化示意图中有 3 条演化曲线,分别对应相对于不同的利益相关者群体的演化.它们包括:a) 相对用户、客户、需求分析人员等;b) 相对设计、架构、代码实现、维护人员等;c) 前两种情况的系统综合.

抽象化维演化曲线如图 4 所示.曲线 a)表示在演化初期,用户、客户、需求分析人员等拥有的关于最终系统模型的信息多为非形式化的,能够达成一致的信息往往尚未细节化,相对最终系统实现抽象度高的信息所占比重较大.在架构设计阶段,增加的信息大多拥有较高抽象度,故抽象度曲线上升.在设计阶段尤其是详细设计阶段,抽象度水平开始下降.这既是因为这一阶段的工作开始以对先前高抽象度表达信息的细节化为主,也由于一些先前积累的较高抽象度的信息开始退出其有效作用范围.曲线 b)表示在演化初期,设计、架构、代码实现、维护人员等对关于最终系统模型的信息所知甚少或者大部分是高抽象度信息.在演化过程中,随着信息总量逐渐增加和信息形式化水平的提高,具体细节信息所占比重越来越大.曲线 c)表示曲线 a)和曲线 b)的加权综合曲线,即系统整体在抽象维的演化.

在 MDD 过程中,a)与 b)两类利益相关者都同时追求两个主要目标:可理解性与可实现性.但他们对在不同阶段对这些目标的追求的侧重不同.其中,a)类利益相关者在过程开始的需求和架构设计阶段,更侧重于保证模型对各种利益相关者的可理解性.这样是为了保证对形式化方法通常不熟悉的利益相关者能够和架构设计者流畅的交流,从而持续引入正确的需求.另外,曲线 a)在这些阶段对整体演化曲线的影响起着主要作用.图 4 中阴影部分对应这些阶段.向下的箭头表示在可理解性为第一位,可实现性为第二位的情况下,曲线 a)和曲线 b)对应利益相关者群体所操纵建模信息集合之间抽象度差距的减少以 b)向 a)趋近为主的原则.在阴影部分以后的阶段,向上的箭头表示在可实现性为第一位,可理解性为第二位的情况下,曲线 a)和曲线 b)对应利益相关者群体所操纵建模信息集合之间抽象度差距的减少以 a)向 b)趋近为主的原则.

© 中国科学院软件研究所 http://www.jos.org.cn

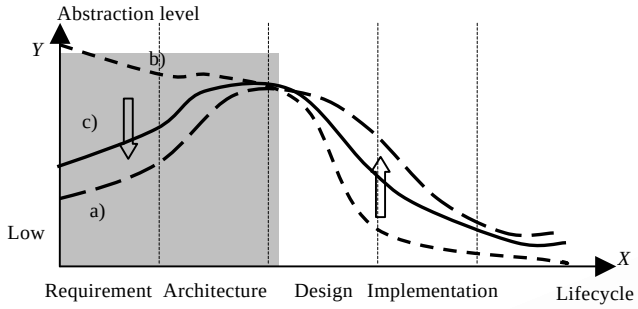

Fig.4 Instance of abstraction dimension evolution curves

图 4 抽象维演化曲线示意

一般化维演化曲线如图 5 所示.表面上看,它和抽象化维的演化曲线有相似之处,但实际上两者是独立变化的两类曲线.由于一般化维对应的表达元素的变化对具体项目应用逻辑的细节内容的依赖度比抽象化维高,故其曲线变化任意性比抽象化维要大.在没有具体项目背景信息的情况下,整体上曲线 a)、曲线 b)、曲线 c)三条曲线的变化较为接近.在开始阶段,一般化维的活跃依赖于可一般化元素集合元素数目的增加和成熟(非抽象化维的变化).因此,模型的整体作用活跃区域相对于抽象化维在开发生命期模型中靠后.在可一般化元素集合变化活动相对减弱、集合相对稳定时,可一般化元素的一般化维变化开始增强.在实现阶段后部,曲线 a)保持较高的位置表示程序系统内部功能的编码阶段工作对用户、客户、需求分析人员对系统认识的一般化水平无影响.

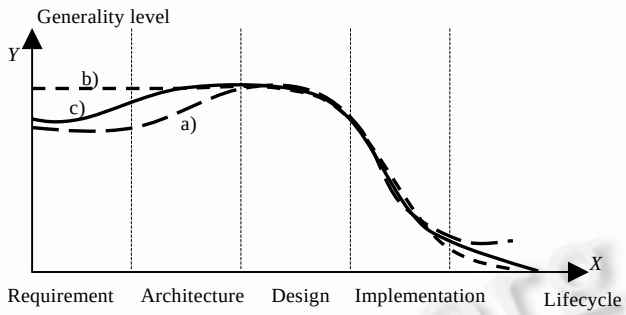

Fig.5 Instance of generality dimension evolution curves

图 5 一般化维演化曲线示意

行为化维演化曲线如图 6 所示.它比一般化维演化曲线具有更强的项目依赖性.不同项目需求和实践下的行为化曲线变化很少有规律可循.

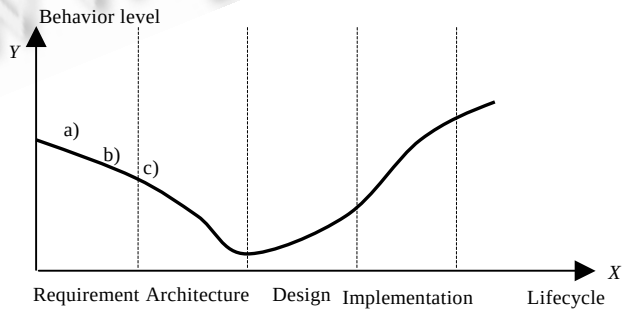

Fig.6 Instance of behaviorism dimension evolution curves

图 6 行为化维演化曲线示意

© 中国科学院软件研究所 http://www.jos.org.cn

由于系统的行为化信息并没有显著地受利益相关者分类的影响,故图 6 中曲线 a)、曲线 b)、曲线 c)用同一曲线表示.示意曲线表示在某 MDD 生命期过程中的架构设计阶段,结构化信息量的增加速度比行为化信息量的增加速度要快,在信息总量中,行为化信息的比重下降;在设计阶段,尤其是详细设计阶段,行为化信息量的增加速度则超过结构化信息的增加速度,行为化信息在信息总量中所占的比重上升.在实际迭代开发过程中,不同的开发细节将引起曲线的波折变化,如需求阶段采用场景(scenarios)到用例的抽取方法,就会引起类似变化.

3.2 BAG/SD建模过程

利用开发方式定位结果,结合演化曲线提供的演化过程中某时刻 3 个元关注维的坐标值的大小和变化方向,可以进行多层次(如不同抽象化水平)和多角度的 MDD 过程模型设计.设计过程是灵活的,可以根据参与的利益相关者群体的实际特点、项目类型、目标以及实施细节约束等进行定制.这里给出主要实践步骤(为了方便描述,这里用 DA 表示开发方式集合,ABT 表示抽象化,GL 表示一般化,BHV 表示行为化):

1) 对业界已存在的 n 种 MDD 相关开发方式集合 $DA=\{da_i\}(1\le i\le n)$,在如图 1 所示三维元关注空间内进行相对定位.其中每一开发方式 da_i 在定位空间中的坐标记作 $SPACECOD_{da_i}=(ABT_{da_i},GL_{da_i},BHV_{da_i})$.

2) 根据利益相关者群体的经验状况、过程实践以及产品期望需求等方面,在考虑可理解性、可实现性优先级等的情况下,分别设计 MDD 生命期系统模型在 3 个元关注维的期望演化曲线.

3) 在 MDD 生命期上选定 m 个时刻$\{t_j\}(1\le j\le m)$,每一时刻 t_j 对应步骤 2)得到的每一演化曲线,可以得到一个该演化曲线上的 Y 轴坐标值.t_j 在 3 个元关注维的坐标值共同决定一个三维坐标,记作 $CURVECOD_{t_j}=(ABT_{t_j},GL_{t_j},BHV_{t_j})$.

4) 对应坐标序列$\{CURVECOD_{t_j}\}(1\le j\le m)$中的每一个单独三维坐标,在步骤 1)得到的开发方式坐标集合$\{SPACECOD_{da_i}\}$中查找与其取值最接近的坐标 $SPACECOD_{da_i}$.$\{SPACECOD_{da_i}\}$所对应的开发方式构成序列$\{da_i\}$.

5) 对开发方式序列$\{da_i\}$进行评估(执行间隙检查,例如,规定相邻开发方式之间的转化必须满足简单转化原则,即规定连续执行的相邻开发方式的作用区域之间最多只能在一个元关注维上存在差异).如果不能通过评估,则考虑执行:

a) 在发现间隙的相邻开发方式之间引入中间开发方式;

b) 重复步骤 3)调整生命期时刻划分,重新确定$\{CURVECOD_{t_j}\}$;

c) 重复步骤 2)考察演化曲线的合理性,并进行修改.

6) 通过评估的开发方式序列$\{da_i\}$就构成 MDD 过程框架执行模型.

4 示 例

本节展示使用 BAG/SD 方法以及前几节的分析结果和曲线对一个消息中间件系统进行的工作.这里给出主要实现过程:

1) 为描述简单起见,采用第 2.1 节中图 1 和表 1 的开发方式定位结果,作为步骤 1)的开发方式坐标集合$\{SPACECOD_{da_i}\}$.规定:在区域集合{抽象,普通,行为化}内,坐标值取 1;在区域集合{具体,明确,结构}内,坐标值取 0.

2) 采用第 3.1 节的利益相关者群体划分和图 4、图 5、图 6 的演化曲线 c)作为期望演化曲线.

3) 在演化曲线的需求、架构设计、设计、实现阶段中各取一个时刻 t_j,并在实现阶段末尾增选一个时刻.构成由 5 个时刻构成的序列$\{t_j\}(1\le j\le 5)$.对应得到的坐标序列$\{CURVECOD_{t_j}\}$={(0,1,1),(1,1,1),(1,1,0),(0,0,0),(0,0,1)}.

4) 经查找,坐标序列$\{CURVECOD_{t_j}\}$对应表 1 中的区域序列{(e),(b),(a),(g),(h)}.从项目的特点考虑,选择对应开发方式序列{基于用例,面向关注,面向对象组件,面向特征,((h)对应编码过程)}.

5) 评估发现,从设计到实现阶段的开发方式转换“(1,1,0)→(0,0,0)”中有多于 1 个维度的差异,即不符合简单

© 中国科学院软件研究所 http://www.jos.org.cn

转换原则.

重复步骤 3),在设计到实现阶段转换中增加一个取值时刻点,得到坐标值(0,1,0).在转换“(1,1,0)→(0,0,0)”中插入该中间点后,转换“(1,1,0)→(0,1,0)→(0,0,0)”可以满足简单转换原则.$SPACECOD_{da}$=(0,1,0)对应表 1 中的区域(f),暂时没有开发方式可选.经分析,面向对象组件方式可以扩展支持这一区域,故不必增加中间开发方式.

6) 最终确定的 MDD 过程框架执行模型为开发方式序列{基于用例,面向关注,面向对象组件,面向特征,((h)对应编码过程)}.

实现过程中的过程框架为一系列子空间序列连续转换的形式,其概念框架如图 7 所示.支持相应开发方式的工作主要包括:关注空间[22,23]、用例空间和用例完整行为视图[24]、组件空间[3,16]、特征空间[3]等.增加面向角色方式是可以更好地维护利益相关者分离的关注[25].面向刻面方式的引入,可以支持必要的关注分解和集成.经考察,它们的引入满足简单转换原则的要求.当前,角色、关注、特征、刻面等概念的定义并不十分统一,这里使用的概念含义主要参考文献[7,19,22,26,27]和 IEEE-Std-1471[21]的定义.

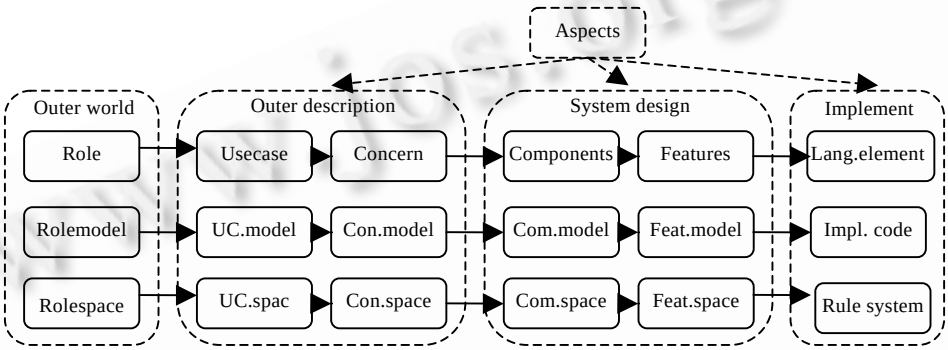

Fig.7 Concept framework

图 7 概念框架

5 相关工作比较

在开发过程建模方面,BAG/SD 方法不是像 AAG(activity artifact graph)[28]、AHEAD(adaptable and human-centered environment for the management of design processes)[14]等方法那样完全独自开发一套新方法,而是充分继承现有开发方式的成熟经验,并在保证兼容 UML 建模系统等的积累知识的情况下,针对 MDD 过程建模的特点进行补充.

当前,软件开发相关领域的概念含义模糊问题阻碍了利益相关者的交流,使软件模型的形式化和演化的自动化水平难以提高,概念澄清工作已经变得十分迫切.文献[29]甚至认为“当前软件工程中的根本问题与其说是技术性的,倒不如说是概念性的”.与 UML 建模系统基础概念相关的工作最为引人瞩目,如 William Frank 等人的 3C(clear,clean,concise)[30]建议用 15 个核心概念来替代 UML 当前标准的 144 个概念.基于本体的方法[31]在这一领域的应用能够有效削减相关概念自然语义含义引起的理解上的模糊.一些研究[32]直接面向业界,针对开发过程管理给出了策略和工具解决方案.BAG/SD 方法通过对相关开发方式的作用范围的约束,在一定程度上也有助于相关概念的澄清工作.

6 结论及进一步的工作

随着各种开发方式的进一步成熟,它们的由作用域扩展而引起的表达相互覆盖的问题将进一步凸现.同时,它们各自的特色也将进一步明显.前者将使引入 BAG/SD 方法进行开发方式区分的重要性得到加强,后者将进一步保证 BAG/SD 方法进行区分的可行性.BAG/SD 方法可以用来对 RUP(rational unified process)等方法提供辅助支持,有助于满足对 MDD 过程的一些非功能性需求,如提高开发效率、增强可跟踪性和保证一致性等.

© 中国科学院软件研究所 http://www.jos.org.cn

进一步的工作将包括:在研究和应用中对 BAG/SD 方法不断完善,与领域工程[6]结合建立符合 MDA 领域分类体系的演化曲线模型库,以期降低早期投资比重,获得更强的实践价值;我们还将研究 BAG/SD 方法的扩展对基于 UML 的 MDD 过程建模细节的支持[33].

致谢 我们感谢 SoftMetaWare 公司的 J. Bettin,Charles University 的 M. Vladimir,University of Paderborn 的 H. Giese,University of Auckland 的 J. Grundy 等老师和朋友,在与他们的早期讨论中获得很多建议和鼓励.我们也向对本文提出宝贵意见的评审老师表示衷心感谢.

References:

[1] Miller J, Mukerji J. MDA Guide version 1.0.1. omg/2003-06-01, 2003. http://www.omg.org/docs/omg/03-06-01.pdf

[2] Unified modeling language: Superstructure, version 2.0. Revised Final Adopted Specification, ptc/04-05-02, 2004. http://www.omg.org/uml/

[3] Jia Y. The evolutionary component-based software reuse approach [Ph.D. Thesis]. Beijing: Institute of Software, The Chinese Academy of Sciences, 2002 (in Chinese with English abstract).

[4] Forbrig P, Laemmel R, Mannhaupt D. Pattern-Oriented development with rational rose. RationalEdge, 2001. http://www-128.ibm.com/developerworks/rational/library/content/RationalEdge/jan01/PatternOrientedDevelopmentwithRationalRoseJan01.pdf

[5] Pallos MS. Service-Oriented architecture (SOA): A primer. eAI Journal, 2001. http://www.eaijournal.com/PDF/SOAPallos.pdf

[6] Zhang W, Mei H. A feature-oriented domain model and its modeling process. Journal of Software, 2003,14(8):1345−1356 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1345.htm

[7] Kande M. A concern-oriented approach to software architecture [Ph.D. Thesis]. Lausanne: Swiss Federal Institute of Technology, 2003.

[8] Wasserman AI. User software engineering and the design of interactive systems. In: Proc. of the 5th Int'l Conf. on Software Engineering. Washington: IEEE Computer Society Press, 1981. 387−393.

[9] Rashid A, Moreira A, Araujo J. Modularisation and composition of aspectual requirements. In: Proc. of the 2nd Int'l Conf. on Aspect-Oriented Software Development. New York: ACM, 2003. 11−20.

[10] Bresciani P, Perini A, Giorgini P, Giunchiglia F, Mylopoulos J. Tropos: An Agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems, 2004,8(3):203−236.

[11] VanHilst M, Notkin D. Decoupling change from design. In: Proc. of the ACM SIGSOFT'96 Symp. on the Foundations of Software Engineering. New York: ACM, 1996. 58−69.

[12] Regnell B, Kimbler K, Wesslen A. Improving the use case driven approach to requirements engineering. In: Proc. of the 2nd IEEE Int'l Symp. Requirements Engineering. Washington: IEEE Computer Society Press, 1995. 40−47.

[13] Tan WA, Zhou BS, Zhang L. Research on the flexible simulation technology for enterprise process model. Journal of Software, 2001,12(7):1080−1087.

[14] Heller M, Schleicher A, Westfechtel B. A management system for evolving development processes. In: Integrated Design and Process Technology, IDPT 2003. Austin: Society for Design and Process Science 2003. http://www-i3.informatik.rwth-aachen.de/private/bernhard/idpt03b.pdf

[15] Clarke S, Walker RJ. Composition patterns: An approach to designing reusable aspects. In: Proc. of the 23rd Int'l Conf. on Software Engineering (ICSE). Washington: IEEE Computer Society Press, 2001. 5−14. http://www.cs.ubc.ca/labs/spl/papers/2001/icse01-cp.pdf

[16] Egyed A. Heterogeneous views integration and its automation [Ph.D. Thesis]. University of Southern California, 2000.

[17] Sage AP, Lynch CL. Systems integration and architecting: An overview of principles, practices, and perspectives, systems engineering. The Journal of the Int'l Council on Systems Engineering, 1998,1(3):176−226.

[18] Hausmann JH, Heckel R, Taentzer G. Detection of conflicting functional requirements in a use case-driven approach. In: Proc. of the ICSE 2002. Washington: IEEE Computer Society Press, 2002. 105−115.

© 中国科学院软件研究所 http://www.jos.org.cn

[19] Kandé MM, Strohmeier A. On the role of multi-dimensional separation of concerns in software architecture. In: Proc. of the OOPSLA 2000 Workshop on Advanced Separation of Concerns in Object-Oriented Systems. 2000. 1–6. http://trese.cs.utwente.nl/workshops/OOPSLA2000/papers/kande.pdf

[20] Tarr P, Ossher H, Harrison W, Sutton SM. *N* degrees of separation: Multi-Dimensional separation of concerns. In: Proc. of the '99 Int'l Conf. on Software Engineering. Washington: IEEE Computer Society Press, 1999. 107–119.

[21] IEEE Architecture Working Group. Recommended practice for architectural description. IEEE P1471/D5.2 Information Technology Draft, 1999.

[22] Hilliard R. Aspects, concerns, subjects, views. In: Proc. of the 1st Workshop on Multi-Dimensional Separation of Concerns in Object-oriented Systems (at OOPSLA'99). 1999. 1–4. http://www.cs.ubc.ca/~murphy/multid-workshop-oopsla99/position-papers/ws08-hilliard.pdf

[23] Jr Sutton SM, Rouvellou I. Issues in the design and implementation of a concern-space modeling schema. 2001. http://www.research.ibm.com/hyperspace/workshops/icse2001/Papers/sutton.pdf

[24] Plasil F, Mencl V. Getting "Whole Picture" behavior in a use case model. Trans. of the SDPS: Journal of Integrated Design and Process Science, 2003,7(4):63–79.

[25] Giese H. Separation of concerns and roles in the object coordination nets approach. 2001. 1–6. http://www.info.uni-karlsruhe.de/~pulvermu/workshops/aop2001/beitraege/giese.pdf.

[26] Shi ZZ, Shi J. Perspectives on cognitive informatics. In: Patel D, Patel S, Wang YX, eds. Proc. of the 2nd IEEE Int'l Conf. on Cognitive Informatics (ICCI 2003). London: IEEE Computer Society, 2003. 129–136.

[27] Herrmann S. Views and concerns and interrelationships—Lessons learned from developing the multi-view software engineering environment PIROL [Ph.D. Thesis]. Berlin: Technical University, 2002.

[28] Yoon IC, Min SY, Bae DH. Tailoring and verifying software process. In: Proc. of APSEC 2001. Washington: IEEE Computer Society Press, 2001. 202–210.

[29] Tekinerdogan B. Synthesis based software architecture design [Ph.D. Thesis]. Enschede: University of Twente, 2000.

[30] Frank W, Tyson KP. Be clear, clean, concise. Communications of the ACM, 2002,45(11):79–81.

[31] Jin Z, Lu R, Bell DA. Automatically multi-paradigm requirements modeling and analyzing: An ontology-based approach. Science in China (Series F), 2003,46(4):279–297.

[32] Wang Q, Li MS. Software process management: Practices in China. Lecture Notes in Computer Science, 2005,3840:317–331.

[33] Duan YC, Cheung SC, Fu XL, Gu YQ. A metamodel based model transformation approach. In: Proc. of 3rd ACIS Int'l Conf. on Software Engineering Research, Management & Applications (SERA 2005). Washington: IEEE Computer Society Press, 2005. 184–191.

附中文参考文献:

[3] 贾育.基于演化构件的软件复用方法[博士学位论文].北京:中国科学院软件研究所,2002.

[6] 张伟,梅宏.一种面向特征的领域模型及其建模过程.软件学报,2003,14(8):1345–1356. http://www.jos.org.cn/1000-9825/14/1345.htm

[13] 谭文安,周伯生,张莉.企业过程模型的柔性模拟技术研究.软件学报,2001,12(7):1080–1087.

段玉聪(1977－),男,河南南阳人,博士,助理研究员,主要研究领域为软件工程,过程建模与质量保证.

顾毓清(1940－),男,研究员,博士生导师,主要研究领域为软件开发方法学,计算机辅助软件工程,CSCW.

© 中国科学院软件研究所 http://www.jos.org.cn