文章编号:1001 - 9081(2008)01 - 0224 - 02

C^4ISR 体系结构产品实体关系一致性分析方法

罗爱民,姜 军,罗雪山
(国防科学技术大学 信息系统与管理学院,长沙 410073)
(amluo@ nudt. edu. cn)

摘 要:在多视图体系结构描述方法中,实体数据存在一定的重复性,保证分布在不同产品中相同数据的一致性是体系结构设计的基本要求。以美国国防部体系结构框架 1.0 为基础,提出一种实体关系数据一致性分析方法。该方法利用不同产品数据建立实体关系矩阵,并通过实体关系矩阵判断不同产品中相同数据一致性。最后以作战节点—作战活动关系的一致性分析为例,证明该方法的可行性。

关键词:体系结构;产品;一致性
中图分类号: TP302.1;E96 **文献标志码**:A

Consistency analysis method of entity relationship of C^4ISR architecture products

LUO Ai-min, JIANG Jun, LUO Xue-shan
(*College of Information Systems and Management, National University of Defense Technology, Changsha Hunan* 410073, *China*)

Abstract: According to multi-view architecture description methodology, the entities are overlapped in different products; it is the basic requirement to keep consistency of identical entities in different products. Based on DoD Architecture Framework 1.0, the consistency analysis methods was provided in the paper, in which entities relationships matrix was developed according to the architecture product, and consistency was judged by relationships matrix. Finally, the case of consistency analysis of relationship between operational node and operational activity was given, and the method has been proved feasible.

Key words: architecture; product; consistency

0 引言

美国国防部体系结构框架 1.0(DoD AF1.0)[1] 采用多视图的体系结构描述方法,从多个不同的角度描述 C^4ISR 系统,较方便地反映各类风险承担者的需求和愿望,易于形成对体系结构整体的描述,同时便于各类风险承担者从不同的角度理解体系结构,也便于他们之间的交流,促进各类人员对体系结构描述达成共识。

不同视图是各类风险承担者从不同视角描述体系结构的产物。在体系结构设计中,一些多类风险承担者共同关心的内容,如体系结构完成的作战活动、具有的系统功能等,通常在多个视图都会描述,因此,各视图和产品的描述内容存在一定的重叠,出现不同视图或产品对同一内容重复描述的现象。如在 DoD AF1.0 中,作战活动模型(OV-5)、作战信息交换矩阵(OV-3)都描述了信息交换内容,作战节点连接关系描述(OV-2)和 OV-5 都包含作战活动的相关信息。由于是对同一对象相同内容的重复描述,因此分布在不同产品中的这些数据必须保持一致。实体关系一致性是体系结构数据一致性中的一个重要部分。实体关系一致性分析是判断在多个产品中定义的相同实体关系是否一致。如 OV-2 以作战节点描述为主线,定义了作战节点与作战活动之间的关系。而 OV-5 以作战活动描述为主线,也定义了作战节点与作战活动之间的关系。对于同一体系结构来说,其中的作战节点与作战活动关系是一定的。因此,从逻辑上讲,这两个产品设计的作战节点与作战活动之间的关系必须一致。实体关系一致性分析就是分析与判断类似于这种实体关系是否一致。本文主要研究利用实体关系矩阵分析实体关系一致性的方法。

1 基于实体关系矩阵的实体关系一致性分析方法

根据 DoD AF1.0 定义的框架和产品,以及产品实体关系设计的特点,产品间的实体一致性分析可采用以下方法[2]:

第 1 步:判断实体数据是否一致。

判断实体关系是否一致,首先应该判断构建实体关系的实体数据是否一致,这是保证实体关系一致性的基本前提。实体数据一致性可以通过对两个产品中的实体数据集合比较来进行判断。

第 2 步:建立关系矩阵。

根据不同产品对同一实体关系的描述,分别建立实体关系矩阵,这些实体关系矩阵体现了实体之间的关系,是分析判断实体关系一致性的基础。

第 3 步:分析、修改实体关系矩阵,构建能够相互比较的修正关系矩阵。

第 2 步得到的实体关系矩阵是利用不同产品设计的关系数据建立的,在矩阵大小、矩阵行列的排列顺序上可能存在差别,这一步就是在第 2 步建立的关系矩阵的基础上,对这些矩阵进行修改,使得实体关系矩阵在大小、行列的排列顺序和含义上一致,将多个关系矩阵转换到一个公共的空间,形成修正实体关系矩阵,保证关系矩阵能够比较。

收稿日期:2007 - 07 - 26;**修回日期**:2007 - 10 - 19。 **基金项目**:国家"十一五"预研项目(513060103)。

作者简介:罗爱民(1970 -),女,湖南邵阳人,副教授,主要研究方向:C^4ISR 体系结构; 姜军(1979 -),男,湖北武汉人,博士研究生,主要研究方向:C^4ISR 体系结构; 罗雪山(1965 -),男,湖南人,教授,博士生导师,主要研究方向:C^4ISR 系统基础理论、系统分析与设计。

第 4 步:利用修正关系矩阵判断实体关系是否一致。

利用修正实体关系矩阵进行比较运算,对所得的结果进行分析,进而判断实体关系是否一致。

上述方法可用于各种实体关系的一致性分析判断。下面以作战节点—作战活动关系一致性为例说明上述方法的使用。

2　作战节点—作战活动关系一致性分析方法

按照 DoD AF1.0,OV-2 和 OV-5 分别描述了作战节点与作战活动之间的关系,在满足实体数据一致性的前提下,根据 OV-2 中定义的作战节点与作战活动之间关系,建立作战节点与作战活动关系矩阵 $\boldsymbol{M}_{\mathrm{ON\_OA}}$,其中 $\boldsymbol{M}_{\mathrm{ON\_OA}}$ 的行表示作战节点集,$\boldsymbol{M}_{\mathrm{ON\_OA}}$ 的列表示作战活动集,$\boldsymbol{M}_{\mathrm{ON\_OA}}$ 的元素记为 $(\boldsymbol{M}_{\mathrm{ON\_OA}})_{ij}$:

$$(\boldsymbol{M}_{\mathrm{ON\_OA}})_{ij}=\begin{cases}1, & \text{位于第 } i \text{ 行的作战节点完成第 } j \text{ 列的作战活动}\\ 0, & \text{位于第 } i \text{ 行的作战节点不完成第 } j \text{ 列的作战活动}\end{cases}$$

同样方法,根据 OV-5 的数据构建作战节点与作战活动关系矩阵 $\boldsymbol{M}_{\mathrm{OA\_ON}}$,其中 $\boldsymbol{M}_{\mathrm{OA\_ON}}$ 的行表示完成作战活动的作战节点集,$\boldsymbol{M}_{\mathrm{OA\_ON}}$ 的列表示作战活动集,$\boldsymbol{M}_{\mathrm{OA\_ON}}$ 的元素记为 $(\boldsymbol{M}_{\mathrm{OA\_ON}})_{ij}$:

$$(\boldsymbol{M}_{\mathrm{OA\_ON}})_{ij}=\begin{cases}1, & \text{位于第 } i \text{ 行的作战节点完成第 } j \text{ 列的作战活动}\\ 0, & \text{位于第 } i \text{ 行的作战节点不完成第 } j \text{ 列的作战活动}\end{cases}$$

比较矩阵 $\boldsymbol{M}_{\mathrm{ON\_OA}}$ 和 $\boldsymbol{M}_{\mathrm{OA\_ON}}$,建立修正关系矩阵。在体系结构设计中,设计作战节点的产品是 OV-2,设计作战活动的产品是 OV-5,因此,建立修正关系矩阵时,必须以 OV-2 中的作战节点集和 OV-5 中的作战活动集为基础。若 $\boldsymbol{M}_{\mathrm{ON\_OA}}$ 和 $\boldsymbol{M}_{\mathrm{OA\_ON}}$ 的行与列存在差别,则按照矩阵 $\boldsymbol{M}_{\mathrm{ON\_OA}}$ 行的组成和顺序,$\boldsymbol{M}_{\mathrm{OA\_ON}}$ 列的组成与顺序,对 $\boldsymbol{M}_{\mathrm{ON\_OA}}$ 和 $\boldsymbol{M}_{\mathrm{OA\_ON}}$ 进行修改,建立 OV-2 的修正实体关系矩阵 $\boldsymbol{M}'_{\mathrm{ON\_OA}}$ 和 OV-5 的修正实体关系矩阵 $\boldsymbol{M}'_{\mathrm{OA\_ON}}$。修正关系矩阵中的元素值和原来矩阵对应元素的值一致。修正关系矩阵 $\boldsymbol{M}'_{\mathrm{ON\_OA}}$ 和 $\boldsymbol{M}'_{\mathrm{OA\_ON}}$ 均为 $N\times M$ 的矩阵,其中 N 表示 OV-2 中作战节点集中元素的个数,M 表示 OV-5 中作战活动集中元素的个数。

对修正关系矩阵 $\boldsymbol{M}'_{\mathrm{ON\_OA}}$ 和 $\boldsymbol{M}'_{\mathrm{OA\_ON}}$ 进行减法运算,令:

$$\boldsymbol{A}=\boldsymbol{M}'_{\mathrm{ON\_OA}}-\boldsymbol{M}'_{\mathrm{OA\_ON}}$$

通过分析矩阵 $\boldsymbol{A}$ 中元素 a_{ij} 的值,判断作战节点—作战活动关系是否一致。判别规则如下:

1) 如果 $a_{ij}=1$,表示 OV-2 定义了第 i 个作战节点与第 j 个作战活动之间的关系,但该关系在 OV-5 中没有定义;

2) 如果 $a_{ij}=-1$,表示 OV-5 定义了第 i 个作战节点与第 j 个作战活动之间的关系,但该关系在 OV-2 中没有定义;

3) 如果 $a_{ij}=0$,表示 OV-2 定义的第 i 个作战节点与第 j 个作战活动之间的关系在 OV-5 中有相同定义,两者一致。

3　实例

某系统体系结构产品 OV-2 的设计如图 1 所示。图中:N_i 表示作战节点,A_i 表示节点完成的作战活动。

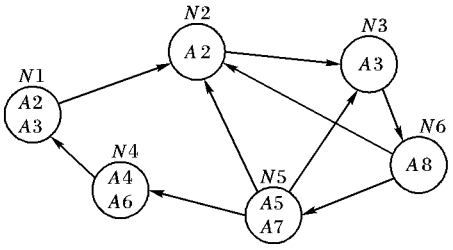

图 1　作战节点连接关系描述 OV-2

根据图 1 定义的作战节点关系,建立矩阵 $\boldsymbol{M}_{\mathrm{ON\_OA}}$:

$$\boldsymbol{M}_{\mathrm{ON\_OA}}=\begin{array}{c}\begin{matrix}A1&A2&A3&A4&A5&A6&A7&A8\end{matrix}\\ \begin{bmatrix}0&1&1&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&1&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}\end{array}\begin{matrix}N1\\N2\\N3\\N4\\N5\\N6\end{matrix}$$

根据图 2 定义的作战活动模型,建立矩阵 $\boldsymbol{M}_{\mathrm{OA\_ON}}$:

$$\boldsymbol{M}_{\mathrm{OA\_ON}}=\begin{array}{c}\begin{matrix}A1&A2&A3&A4&A5&A6&A7&A8\end{matrix}\\ \begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&1&1&0&0\\0&0&0&0&0&0&1&1\end{bmatrix}\end{array}\begin{matrix}N1\\N2\\N3\\N4\\N6\end{matrix}$$

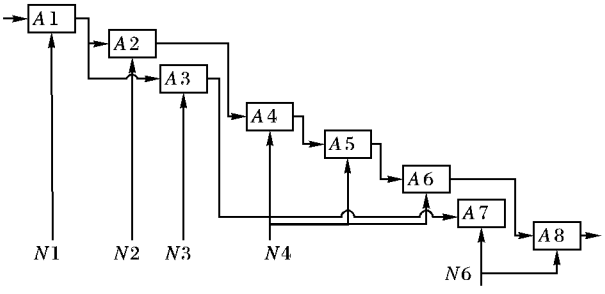

图 2　作战活动模型 OV-5

比较矩阵 $\boldsymbol{M}_{\mathrm{OA\_ON}}$ 和 $\boldsymbol{M}_{\mathrm{ON\_OA}}$,建立修正矩阵:

$$\boldsymbol{M}'_{\mathrm{OA\_ON}}=\begin{array}{c}\begin{matrix}A1&A2&A3&A4&A5&A6&A7&A8\end{matrix}\\ \begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&1&1&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&1\end{bmatrix}\end{array}\begin{matrix}N1\\N2\\N3\\N4\\N5\\N6\end{matrix}$$

$\boldsymbol{M}'_{\mathrm{ON\_OA}}=\boldsymbol{M}_{\mathrm{ON\_OA}}$,则:

$$\boldsymbol{M}'_{\mathrm{ON\_OA}}-\boldsymbol{M}'_{\mathrm{OA\_ON}}=\begin{bmatrix}0&1&1&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&1&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}-\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&1&1&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&1\end{bmatrix}=\begin{bmatrix}-1&1&1&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&-1&0&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&0&0&-1&0\end{bmatrix}$$

从上面矩阵运算的结果可以看出,在 OV-2 和 OV-5 中,作战节点 $N2$,$N3$ 与作战活动的关系,作战活动 $A4$,$A6$,$A8$ 与作战节点的关系是一致的,其他作战节点与作战活动的关系设计不一致。

(下转第 228 页)

陷数据时需填写数量繁多的信息项,并在众多的属性值类别中做出选择,这必然会影响缺陷数据收集的效率和精确性。

准则5 软件缺陷数据的定义应能够根据软件组织的实际需要而改变。

缺陷数据的定义不是一成不变的,软件组织在不同的成熟度阶段,以及在开发不同领域的软件产品时,对缺陷信息的要求会有所差别,这就要求缺陷管理工具具有缺陷属性及其值的自定义功能。

3 软件缺陷数据定义举例

综合以上各准则,我们在开发"软件缺陷管理和度量系统(BMMS)"时定义了表1所示的默认的缺陷属性。表中的"填写时机"是指在缺陷跟踪流程中的哪一个环节将缺陷属性值输入到系统中。

表1 BMMS的默认缺陷数据定义

缺陷属性	含义	填写时机
编号(defectCode)	缺陷的唯一编号(由系统自动生成)	报告缺陷
标题(title)	缺陷的标题	报告缺陷
报告人(reporter)	缺陷的报告人,通常是发现缺陷的人员	报告缺陷
报告时间(reportTime)	缺陷被报告时的日期	报告缺陷
项目(projectID)	缺陷所属的产品项目	报告缺陷
模块(moduleID)	缺陷所属的软件产品模块	报告缺陷
环境(Environment)	缺陷被发现时的软、硬件环境	报告缺陷
描述(Description)	重现缺陷的操作步骤及缺陷所引发的失效现象	报告缺陷
活动(Activity)	缺陷是在执行什么活动时被检测到的,如需求审查、设计审查、代码走查、各种测试、用户的使用等	报告缺陷
严重程度(Level)	缺陷对软件产品及其运行环境所造成的影响的严重程度,如关键的、严重的、一般的、轻微的,等等	报告缺陷
附件(Accessory)	附加的说明文档或屏幕截图等	报告缺陷
修复人(dealer)	缺陷的修复人	将缺陷分配给某人员进行修复
状态(State)	缺陷当前的状态,如打开、已分配、已解决、已关闭等	执行了各种缺陷处理任务后
解决方式(resolution)	解决缺陷的方式,如修复、挂起、延迟、不能重现等	解决缺陷后
类型(Type)	对于某一来源,导致缺陷被引入的具体的区域,例如对于设计来源,可能的类型包括功能、接口定义、数据定义、算法设计等	解决缺陷后
来源(Origin)	产生此缺陷的原由,最早可追溯到哪个开发活动中的问题,如需求分析、设计、编码、维护等	解决缺陷后
关闭时间(closeTime)	缺陷被关闭时的日期,缺陷被关闭代表其生命周期的结束	关闭缺陷
原因(Cause)	导致缺陷被引入的根本因素,例如开发者的疏忽、通信障碍、技术或领域知识的缺乏、文字表达的错误等	缺陷数据分析后

表1中的缺陷数据定义充分地考虑到使属性之间没有重叠,每个属性的取值之间也有明显的界限,从而最大限度地实现了正交性。除"来源"和"原因"属性值的判断带有一定主观性外,其余属性均是根据开发活动中的客观事实直接选择其取值,降低了输入错误数据的可能性。"原因"属性是在进行缺陷数据分析后输入的,之所以设置这个属性,是因为对于某些严重缺陷,需要分析其产生的根本原因(Root Cause),将分析结果记录下来,预防类似缺陷的再次发生。

4 结语

缺陷数据的定义对于软件缺陷管理具有极为重要的意义,按照本文提出的方法,软件组织首先要根据自身的成熟度确定缺陷管理的目标,然后定义缺陷管理过程,使缺陷管理过程与组织的整体软件过程无缝地集成在一起,在此基础上,根据目标和过程的信息需要以及有效性、简明性等一系列准则来定义缺陷数据。

这一方法的实现需要工具的支持,一个优秀的缺陷管理工具应具有以下特征:

1)集成工作流引擎,可灵活配置缺陷跟踪流程;

2)具有灵活的用户管理和权限分配功能;

3)支持缺陷属性及属性值的自定义。

目前大多数缺陷管理工具在这些特性上都不够完善,在BMMS项目的后续研究工作中,我们将继续深入研究软件缺陷数据的定义方法和分析方法,并进一步完善相应的缺陷管理工具。

参考文献:

[1] GRADY R. Software failure analysis for high-return process improvement decisions [J]. Hewlett-Packard Journal, 1996, 47(4):15-24.

[2] FREDERICKS M, BASILI V. Using defect tracking and analysis to improve software quality [EB/OL]. [2007-06-20]. http://www.dacs.dtic.mil/techs/defect/.

[3] 李明树, 王青. 基于过程控制的软件质量管理[J]. 电子学报, 2002, 30(12A):2032-2035.

[4] FENTON N E, PFLEEGER S L. 软件度量——严格而实用的方法:影印版[M]. 2版. 北京:机械工业出版社, 2004.

[5] LANUBILE F, SHULL F, BASILI V. Experimenting with error abstraction in requirements documents [C]// Proceedings of 5th International symposium on software metrics. Bethesda: IEEE Computer Society Press, 1998:114-121.

[6] CHILLAREGE R, BHANDARI I, CHAAR J, *et al*. Orthogonal defect classification - A concept for in-process measurements [J]. IEEE Transactions on Software Engineering, 1992, 18(11):943-956.

(上接第225页)

4 结语

采用本文提出的方法可以对实体关系一致性进行有效的分析。由于作战节点和作战活动都具有层次分解的关系,因此,对于a_{ij}为1或-1的元素要深入分析,要结合父子节点或父子活动的关系,进一步判断在逻辑上关系数据是否一致。

参考文献:

[1] DoD Architecture Framework Working Group. DoD architecture framework Version 1.0 Volume I: Definitions and guidelines [S], 2003.

[2] 罗爱民. 基于信息模型的C⁴ISR体系结构设计与分析方法研究[D]. 长沙:国防科学技术大学, 2006.