基于活动图的 C⁴ISR 能力需求过程建模及验证

刘大伟 王智学 禹明刚
(解放军理工大学指挥信息系统学院 南京 210007)

摘 要 当前对 C⁴ISR 系统能力需求的描述大多基于图形、文字等静态模型,对信息和数据的具体操作没有进行定义,以至于对象之间的行为过程没有详细说明。缺乏可执行动态语义的能力模型是不可执行的,因此提出了一种基于活动图的能力需求过程建模方法,为可执行体系结构的建模仿真提供支撑。首先给出了系统过程模型的定义,在 C⁴ISR 系统能力元概念模型的指导下,通过扩展 UML 活动图得到系统的能力需求过程元模型。然后用本体表示能力需求过程元模型语义,通过对本体的逻辑推理实现对 C⁴ISR 系统能力需求过程元模型的验证。

关键词 能力需求,可执行,活动图,过程建模,模型验证
中图法分类号 TP302.1 **文献标识码** A

Modeling and Validation of Capability Requirement Process Based on Activity Diagram

LIU Da-wei WANG Zhi-xue YU Ming-gang
(Institute of Command Information System, PLA University of Science and Technology, Nanjing 210007, China)

Abstract Current descriptions of C⁴ISR system capability requirement are mostly based on static model such as literature and static diagram, and lack of definition of specific operation to information and data, which leads to lack of detailed explanation of behavior between objects. Since the capability model which lacks executable dynamic semantics is unexecutable, a modeling approach to capability requirement process based on activity diagram was proposed to support the modeling and simulation of executable architecture. Firstly, definition of system process model was presented, and with the guidance of C⁴ISR capability metamodel, capability requirement process metamodel was built by extending UML activity diagram. Then ontology was used to describe the semantics of capability requirement process metamodel, and validation of C⁴ISR capability requirement process metamodel was done through reasoning of ontology.

Keywords Capability requirement, Executable, Activity diagram, Process modeling, Model validation

C⁴ISR 系统因系统层层嵌套、环环相扣,对其研究需要涉及各种复杂因素而难以确定和管理而被称为复杂系统(SoS)。进入 21 世纪以来,西方发达国家的武器装备需求管理由传统的基于威胁的规划策略转为基于能力的需求规划,由此也产生了基于能力的军事需求方法[1]。目前,基于能力的需求方法中比较有代表性的分别是美军的"联合能力集成与开发系统(joint capability integration and development system, JCIDS)"[2]和英军在其"国防部体系结构框架 MODAF v1.1"中用于描述能力产品的战略视图(StV2~StV6)[3]。文献[4]提出了一种基于能力的系统需求建模分析过程和研究方法。该研究明确了基于能力需求获取的分析方法,特别在 UML 建模方法的基础上,统一了基于能力需求分析的建模过程。文献[5]提出了一套基于 UML 的 C⁴ISR 系统能力需求建模过程和方法,从作战使命和任务分析入手,分析作战任务目标,围绕目标构建作战用例和活动模型,由此构建能力需求模型。文献[6]首先采用描述逻辑语言形式化描述能力需求模型,确定模型的形式化语义,然后提出能力需求模型的一致性检验规则,最终在一致性检验规则的约束下,借助自动推理引擎,实现模型的一致性检验。

尽管前期在基于能力的 C⁴ISR 系统需求工程方法和技术的研究上取得了一些成果,但当前对 C⁴ISR 系统能力需求的描述大多基于图形、文字等静态模型,对信息和数据的具体操作没有进行定义,以至于对象之间的行为过程没有详细说明。缺乏可执行动态语义的能力模型是不可执行的,这样不易于前期分析及设计中发现问题,增加了后期开发的风险。近年来在系统工程领域许多学者致力于可执行体系结构(Executable Architecture, EA)的研究[7,8],主要是针对动态模型,包括活动图、时序图等,通过动作语义使活动图能"动"起来,达到动态观察和验证作战相关概念的目的。本文的主要工作及创新点为:1)扩展事件驱动建模,实现了活动图外部事件处理建模;2)扩展数据传递机制,在保持对系统数据传递描述效果的前提下降低了数据传递建模的复杂度;3)扩展基于能力的需求过程元模型并生成适用过程建模的 UML Profile;4)把过程模型转换为 OWL 本体进行推理验证。

1 系统过程元概念模型及 C⁴ISR 能力元概念模型

1.1 系统过程元概念模型

一个系统任务在执行的过程中需要完成各种动作,例如目标识别、态势分析、导弹射击等。动作的执行是由事件触发的,在动作执行结束时又可能产生新的事件,从而触发下一个

刘大伟(1989-),男,硕士生,主要研究方向为需求工程与体系结构技术,E-mail:liudw295@163.com;王智学(1961-),男,教授,博士生导师,主要研究方向为军事通信学、系统工程、软件工程;禹明刚(1986-),男,博士生,主要研究方向为系统工程、需求工程。

动作。并且，动作的执行需要零或多个参数，事件中又可能携带相关的数据，动作的参数需要引用系统数据，事件中携带的数据需要保存到系统数据中。因此，动作、事件和系统数据是组成过程模型的基本元素[10]。

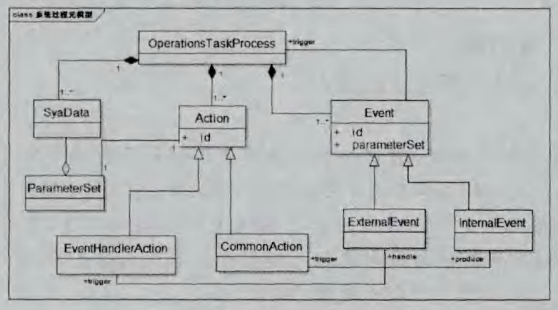

图 1　系统过程元概念模型

在图 1 所示的系统过程元概念模型中，一个系统任务需要执行多个动作和处理多个事件，动作分为通用操作动作、事件处理动作。作战过程包含了多个系统数据，动作的执行和事件的处理都涉及到系统数据的传递。

定义 1（动作，Action）　动作是一个五元组〈id，ps，te，fe，fes〉[10]。

id 是动作的标识，ps 是执行该动作所需要的参数集（Parameter Set），te 是触发动作执行的入口事件（Trigger Event），fe 是该活动执行完成时产生的事件（Fire Event），fes 是 fe 的集合。

动作完成一个具体的功能；动作的执行所需的一个或多个参数，由其动作参数集 ps 规定；动作的执行需要由一个驱动事件 te 进行驱动；动作在执行完成之后会产生 fes 中的一个 fe 来表示动作执行的结束，同时 fe 也指出了动作执行的结果，fes 则描述了动作所有可能出现的执行结果。

根据动作执行功能的不同，动作又可以分为事件处理动作和常规动作。

定义 2（事件处理动作，Event Handler Action）　事件处理动作是指对某一外部事件进行响应，根据同一入口事件的不同情况派生出动作分支。

定义 3（常规动作，Common Action）　常规动作指除事件处理动作外任务所有的逻辑执行动作。

定义 4（事件，Event）　事件是一个二元组〈id，ds〉[10]。

id 是事件的标识，ds 是事件中携带的数据集合（Data Set）。

定义 5（内部事件，Internal Event）　是指业务执行过程中由动作执行完成后同步产生的事件。

定义 6（外部事件，External Event）　是指除内部事件以外，需要在系统实体间通过网络传递的异步事件。

定义 7（系统数据，Sys Data）[10]　系统数据指系统执行过程中所使用的所有数据。事件中携带的数据需要保存到相应的系统数据中；事件处理动作和常规动作执行所需的参数需要引用系统数据。

此外，系统过程的执行需要由一个事件触发，当该触发事件到来之后任务开始执行，并对该事件进行处理，根据处理的结果产生相应的事件驱动下一个动作，直到过程结束。

1.2　C⁴ISR 能力元概念模型

元概念模型是基于能力的系统需求分析方法的核心，是应用概念模型的抽象。元概念模型总结了美军 DoDAF[11] 和英军 MODAF[3] 的主要思想，它抽取了 C⁴ISR 系统体系结构中最为基本的概念以及概念间的关系（由于文章篇幅有限，这里已经对元概念模型作出适当裁减，见图 2）。

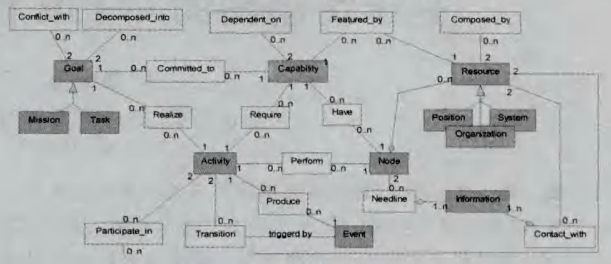

图 2　C⁴ISR 能力元概念模型

图 2 中阴影框为概念类，空白框为关系类，概念类和关系类之间的基数约束表示在元概念模型约束下的系统模型中相应的关系和概念实例之间的基数约束。例如：关系类 Dependent_on 和概念类 Capability 之间的基数约束表示在构建的系统模型中，一个关系 Dependent_on 的对象要连接两个类 Capability 的对象，一个类 Capability 的对象可以连接 0～n 个 Dependent_on 关系对象。在能力元概念模型中，系统需求开发的核心元素是能力（Capability），系统的活动（Activity）、事件（Event）、资源（Resource）都与能力有关联。

C⁴ISR 能力元概念模型是应用概念模型的抽象，能够引导和规范领域能力需求过程元模型的建立：

（1）元概念给出了领域概念的确切含义，在建立领域模型过程中避免了二义性；

（2）元模型为领域建模提供了模型语义约束规则，可以作为领域模型完整性检查的知识，为规范建模和一致性检查奠定了基础。

2　能力需求过程元概念模型

活动分析是分析为了实现某个任务目标所开展的各种活动、活动的先后关系、参与活动的实体以及开展活动所需要的相关能力。UML 活动图被定义为一个包含了 Action Node、Control Flow、Acitivity Parameter Node 等要素的多元组，活动图可以很好地描述以动作、事件、系统数据为核心的系统过程，通过在活动图中扩展相关概念和机制来实现对系统基于能力的过程描述。

扩展 UML 活动图的方法是使用 UML 的扩展机制 UML Profile 中的构造型（Stereo type）对系统执行过程中涉及到的动作、事件、系统参数、能力等概念进行描述。扩展过程应遵循两条基本原则：

（1）扩展要保证模型的完整性和无二义性。完整性和无二义性也是基于逻辑推理的模型验证和可执行仿真模型生成的基础；

（2）建模过程简洁，模型易于维护。

2.1　扩展的事件驱动建模

在基于能力的过程模型中，动作的执行是以事件的传递为基础的，依靠各种事件的传递引发动作的执行。同一个实体（泳道）内动作间事件的交互是同步模式，动作执行完成后同步产生事件，该类事件为内部事件；而不同的实体（泳道）间事件的交互具有异步性的特点，实体不能够准确知道事件到来的时刻，该类事件为外部事件。以区域防空系统为例，在实体“区域指挥中心”内部，动作“威胁评估”向“行动部署”传递

的事件为内部事件，实体“情报中心”的动作“目标探测”向实体“区域指挥中心”的动作“威胁评估”传递的“探测结果”事件为外部事件。

活动图中关于过程的描述主要依靠的是内部动作的结果，即控制流来描述，对于内部事件和外部事件没有提供不同的处理方式，而系统过程的执行需要对内部事件和外部事件作不同的处理。因此，基于活动图搭建过程模型的关键点之一就在于如何在模型上描述对内部事件和外部事件的不同处理。

内部事件是在动作执行完成后同步产生的，因而内部事件的处理比较容易进行描述。对于表示某个内部事件 E 的控制流来说，其源端节点是产生内部事件的动作，其目标节点是源端动作在执行完成后产生 E 时需要执行的下一个节点。

对于外部事件来说，其反映的是连接本实体与其他实体或外部环境的网络所发生的变化，当且仅当变化发生时才会到来，具有异步性的特点，实体不能够准确知道外部事件到来的时刻。然而，外部事件的到来总是由某些实体动作的执行所引起的或者在某种情况下会出现，所以实体可以在有可能出现外部事件之前对外部事件注册其处理方式，从而在外部事件到来时能够正确对其进行处理。例如在区域防空系统中，执行对敌方阵地的动作“导弹射击”之后，可以预知外部事件“射击结果”（反映“目标消失”、“目标仍然存在”）会到来，因而需要对事件“射击结果”注册其处理方式。

事件注册在模型上的表示方法是：在表示外部事件 E 的控制流上附加构造型《注册事件》，说明在该控制流的源端节点执行完成之后对 E 这一类外部事件进行注册；对外部事件的目标节点附加构造型《动作处理者》，该活动节点等待外部事件的到来。

添加了构造型《注册事件》后，控制流的目标节点就是该事件到来应该执行的动作，完成注册后根据控制流源端节点产生的内部事件进行跳转。

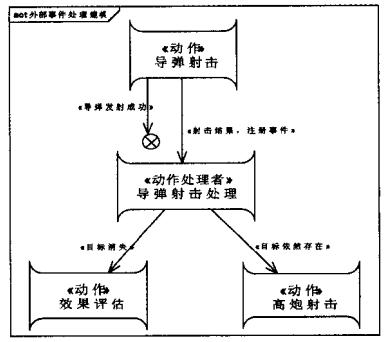

图 3　扩展的活动图外部事件处理建模

如图 3 所示，动作“导弹射击”执行完成之后，对“射击结果”这一类外部事件进行了注册。导弹营“导弹射击”操作成功产生“导弹发射成功”事件，并等待从敌方目标阵地反馈回来的“导弹射击结果”事件的到来。“导弹射击结果”事件到来之后，根据注册的情况，首先交由事件动作“导弹射击结果处理者”，根据导弹射击的结果再执行后续的节点（目标消失则进行效果评估，结束任务；目标依然存在则改用高炮继续射击）。

2.2　扩展的数据传递建模

在数据传递建模方面，活动图提供了活动参数节点（Activity Parameter Ndoe）、数据存储节点（Data Store Node）、输入管脚（Input Pin）、输出管脚（Output Pin）和对象流（Object Flow）等元素用于描述过程中涉及的数据传递。

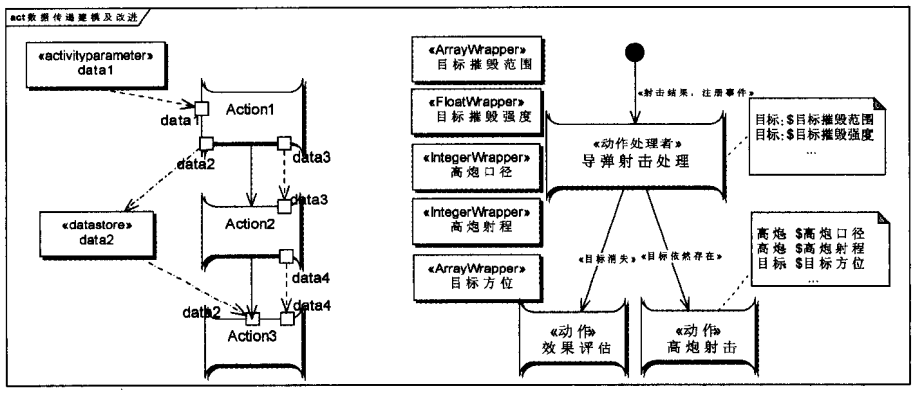

图 4　活动图数据传递建模及改进

图 4 左部分描述了活动图的数据传递建模方法。虽然活动图提供的数据传递建模方法能够描述业务过程中的数据传递，但是应用到系统过程建模中时，仍然存在以下不足：

（1）由于建模元素过多，增加了建模的复杂度，而且搭建的模型也因此变得臃肿，难以维护。如果一个动作执行所需参数和输出数据的数量较多，就必须有相应的输入管脚和输出管脚进行描述，同时，数据存储节点和对象流的数量也会相应增加，这给业务建模人员的建模工作增加了难度。

（2）输入管脚和输出管脚难以和活动执行所需参数进行对应绑定。活动节点上附加的构造型规定了活动执行所需的参数，如何将输入管脚和输出管脚与这些参数进行对应绑定也是一个难题。

（3）增加了可执行代码生成的复杂度。出于描述及实现的方便考虑，我们将系统过程中所需要处理的数据全部作为系统过程全局数据进行描述，在系统执行过程中每个活动都随时可以访问处理，这样就大大降低了数据传递建模的复杂度。

实现这种数据传递建模方法的关键有以下几点：

（1）使用活动参数节点描述系统过程中使用到的系统过程全局数据。

（2）动作执行引用全局数据的方法是在其附加构造型的属性中填入“$ + 动作参数节点名称”，从而将动作参数和描述系统过程全局数据的活动参数节点对应绑定起来。

（3）对于事件处理动作来说，其执行所需的参数表示的是对事件处理结束后所得到的事件携带数据。事件携带数据需要保存到系统过程全局数据中，以供后续的动作使用。同样，将事件处理动作的参数和描述系统过程全局数据的活动参数节点进行对应绑定。这样，在事件处理动作结束对事件的处

理后，事件中所携带的数据就能保存到系统过程全局数据中。

由于直接将动作节点附加构造型的参数和活动参数节点进行对应绑定，可以省略数据存储节点、输入管脚、输出管脚和对象流等建模元素，在保持对系统数据传递描述效果的前提下降低了数据传递建模的复杂度。

如图 4 右部分所示，动作节点“导弹射击处理者”对事件“射击结果”进行处理，并将事件中携带的“目标摧毁范围”、“目标摧毁强度”等数据存放到对应的描述系统过程全局数据的活动参数节点“目标摧毁范围”、“目标摧毁强度”中；动作节点“高炮射击”的执行所需的参数“高炮口径”、“高炮射程”和“目标方位”根据“射击结果”事件的参数进行确定，并引用“高炮口径”、“高炮射程”和“目标方位”系统过程全局数据。

2.3 基于能力的需求过程元模型

除了扩展的事件驱动建模和数据传递建模外，扩展的基于能力的需求过程元模型还应包含以下几点：

(1)对于描述动作的构造型来说，其基类为活动图建模元素中的动作节点，以动作的标识作为该构造型的名称，执行动作所需的参数集则对应为构造型的属性，附加了构造型的动作节点表示一个实体动作。

(2)对于描述事件的构造型来说，其基类为活动图建模元素中的控制流，以事件的标识作为该构造型的名称。

(3)对于描述系统数据的构造型来说，其基类为活动图建模元素中的活动参数节点，构造型的名称描述了系统数据的数据类型。

(4)在过程模型中引入了能力概念。

(5)用一个动作处理者结点作为外部事件的目标结点，等待外部事件的到来。

(6)保留了活动元模型中的部分控制节点，如 Initial Node、Activity Final Node、Flow Final Node、Fork Node 和 Join Node。

(7)扩展出 Require、Support、Handle 等用于描述动作与能力、实体与能力、动作与事件等之间的关联关系。

具体方法如表 1 所列。

表 1 能力需求过程扩展的 UML profile

Stereotype/ CRPMM	Base Class/ Activity	Description
Action	Action	完成使命任务实体需要执行的动作
Event	Control Flow	任务执行过程中由动作产生并触发动作
Sys Data	Activity Parameter Node	原始输入数据及动作执行产生的数据
Swimlane	Swimlane	执行动作的实体，包含一个或多个动作
Capability	Class	执行特定使命任务，实现其目标时表现出的功能和效能
Entity	Object Flow	执行动作的主体，实体与泳道相对应，实体拥有能力
Entity Behavior	Object Flow	一个泳道可有多个实体行为，一个实体行对应一个动作
Require	Object Flow	描述活动与能力之间的需求关系，指明活动开展所需要的能力
Have	Association	描述实体与能力间的拥有关系，指明实体所拥有的能力
Support	Object Flow	描述能力与操作间的支撑关系，指明能力所支撑的操作
Produce	Object Flow	描述常规动作与内部事件间的产生关系
Transition	Control Flow	描述活动之间的变迁关系，指明系统执行由一个活动向另一个活动的转换
Trigger	Object Flow	描述事件与动作间的触发关系
Handle	Object Flow	描述事件处理动作与外部事件间的处理关系

根据 UML profile 扩展机制及对活动图事件处理机制、数据传递机制的扩展，得到能力需求过程元模型如图 5 所示。

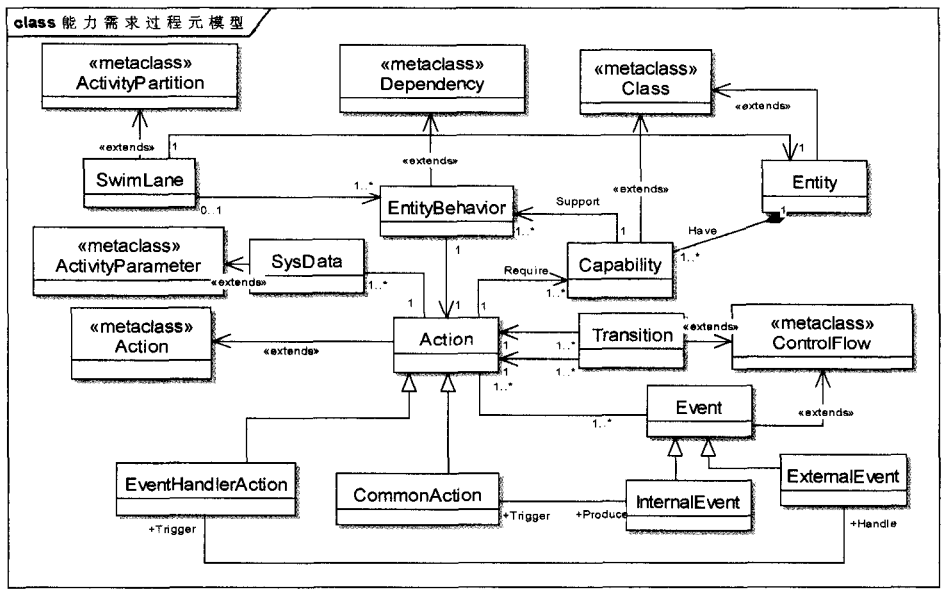

图 5 能力需求过程元模型

3 能力需求过程模型验证

由于 UML 建模语言本身不是纯形式化语言，难以进行完整的逻辑推理验证，因此需要引入一种推理能力较强的形式化描述方法。本文采用本体描述语言 OWL DL[12]将 UML 描述的概念模型自动转换成 OWL DL 描述的本体模型[13]，用本体查询语言 nRQL 定义模型检验规则，最后在 Racer-Porter 环境下采用自动推理引擎 Racer 完成模型一致性检验。

3.1 模型转换

本方法模型转换的主要思路是：将过程元概念模型转换为描述逻辑 Tbox 中的概念知识库；将过程模型的静态语义转换为描述逻辑 Abox 中的概念实例，将过程模型的动态语义用 DL-Safe 规则表示。模型一致性检查工作就成为在

Tbox 约束下的 Abox 中的概念实例查询问题。对于能力需求过程元模型和能力需求过程模型，本方法采用不同的模型转换方法：

(1)过程元模型

①对于元概念模型中出现的各种元类，在 Tbox 中创建相应的概念。

②对于元概念模型中出现的各种元类之间的一般关联关系、聚合关系及其逻辑关系约束，在 Tbox 中创建元概念之间的对应关系以及对关系的约束与限制，对于“动作”与“能力”之间的“需要”关系，应当在 Tbox 中额外创建“需要”的逆关系。

③对于元概念模型中元类之间出现的继承关系，首先在 Tbox 中相应的概念之间创建继承关系，然后在 Tbox 中创建子概念与父概念间的“Subconcept”关系，“Subconcept”关系具有传递属性。

(2)过程模型

过程模型的转换分为静态语义和动态语义的转换。

①静态语义部分，对于过程模型中出现的各种概念，根据其元类型 Sterotype 的类别，在 Abox 中创建对应元概念的实例。如对于活动图中出现的各种动作，在 Abox 中创建对应于 Tbox 中“动作”概念的个体。

②动态语义部分，过程模型的动态语义指行为模型中特有的模型规则，如活动模型中的动态语义指动作之间的动态执行关系，即一个动作执行结束后如满足条件就执行下一个动作。将包含动态语义的规则抽取出来，然后把抽取出来的规则用 DL-Safe 规则表示[14,15]。为了表示活动图中的动态语义，为动作类添加了一个数据属性 ExecuteState，属性值为 0 表示等待执行，属性值为 1 表示正在执行，属性值为 2 表示执行结束，属性值为 3 表示不能执行，初始化“起始活动”的数据值为 1，其他活动的数据值为 0。通过分析将活动模型的动作语义抽取为以下规则。

规则 1　动作 a 正在执行，如果动作 a、b 之间有转换线，并且满足守护条件 c，则下一步执行动作 b。用 DL-Safe 规则表示为：

$$\text{ExecuteState}(b,1) \leftarrow \text{ExecuteState}(a,1) \wedge \text{Transition}(a,b) \wedge \text{Condition}(c) \wedge \text{Guard}(c,b) \wedge \text{O}(a) \wedge \text{O}(b) \wedge \text{O}(c)$$

规则 2　动作 b 正在执行，如果动作 a、b 之间有转换线，则动作 a 已经结束。用 DL-Safe 规则表示为：

$$\text{ExecuteState}(a,2) \leftarrow \text{ExecuteState}(b,1) \wedge \text{Transition}(a,b) \wedge \text{O}(a) \wedge \text{O}(b)$$

规则 3　动作 a 正在执行，如果动作 a、b 之间为分支关系，则下一步执行动作 b。用 DL-Safe 规则表示为：

$$\text{ExecuteState}(b,1) \leftarrow \text{ExecuteState}(a,1) \wedge \text{Fork}(a,b) \wedge \text{O}(a) \wedge \text{O}(b)$$

规则 4　动作 b 正在执行，如果动作 a、b 之间有分支关系，则动作 a 已经结束。用 DL-Safe 规则表示为：

$$\text{ExecuteState}(a,2) \leftarrow \text{ExecuteState}(b,1) \wedge \text{Fork}(a,b) \wedge \text{O}(a) \wedge \text{O}(b)$$

规则 5　动作集合 Action 聚合到动作 a，如果 Action 中任意一个动作 b 都正在执行，则下一步执行动作 a。用 DL-Safe 规则表示为：

$$\text{ExecuteState}(a,3) \leftrightarrow \text{ExecuteState}(b,1) \wedge \text{Join}(b,a) \wedge \text{O}(a) \wedge \text{O}(b)$$

如果 Action 中任意一个动作 b 在等待执行，那么动作 a 就不能执行。

规则 6　动作集合 Action 聚合到动作 a，如果动作 a 正在执行，则 Action 中任意一个动作 b 都执行结束。用 DL-Safe 规则表示为：

$$\text{ExecuteState}(b,2) \leftrightarrow \text{ExecuteState}(a,3) \wedge \text{Join}(b,a) \wedge \text{O}(a) \wedge \text{O}(b)$$

即如果动作 a 不是不能执行的，那么它就可以执行，Action 中的任意一个动作 b 就可以执行结束。

规则 5 中的 $\neg\text{ExecuteState}(b,1)$ 表示动作 b 的 ExecuteState 属性值不是 1。将以上规则用 SWRL[16]语言表示出来添加到 OWL DL 本体模型中。

3.2　模型验证

模型验证分为模型的一致性(consistency)验证和完整性(integrity)验证[17]。由于 C⁴ISR 系统的能力元概念模型抽取了能力需求建模最基本的概念和关系，系统过程元概念模型则通过扩展 UML Profile 得到，因此所构建的能力需求过程模型受能力元概念模型和系统过程元概念模型的双重约束。行为模型的一致性指模型中的个体不违反能力元概念模型和系统过程元概念模型中所属类的约束。如果在建模时违反了这些约束，那么所建模型必然存在矛盾。

针对一致性问题定义了以下检查规则：

CR1 每个活动图模型有且仅有一个开始节点和一个或多个结束节点以及至少一个 Action 节点和两个 ControlFlow 控制流。

CR2 每个 Action 节点所发出的控制流信号必须符合扩展的 profile 定义。例如动作“地空近程导弹射击结果处理者”，它只能发射出两个扩展的控制流，即版类为“目标消失”和版类为“目标依然存在”的控制流，其他情况都是错误情况。

CR3 模型中外部事件的目标结点(即动作处理者结点)所产生的事件必须是已经注册过的事件。

CR4 模型中的任何 Action 节点都应该处于开始节点到结束节点的某条路径上。即模型实例中的节点均可达，不存在死节点。

CR5 模型中的环路不存在死循环。即模型中的每个动作必须可以到达结束动作。

CR6 系统执行过程中使用到的所有数据必须在活动参数节点中找得到。

CR7 模型中的任何一个 Action 最多被一个实体执行。

CR8 模型中的任何 Action 必须在某个 SwimLane 里。

由于 C⁴ISR 系统的能力需求建模是针对军事电子信息系统，属于特定领域建模。在建模过程中，有些领域规则建模人员必须遵守，如果所建模型不满足这些规则，那么模型是不完整的。

针对完整性问题定义了以下检查规则：

IR1 所有任务目标必须由活动完成，不存在与活动无关

的任务目标。

IR2 模型中，如果动作 a 是由实体 e 开展的且需要能力 c 的支持，则 e 必须拥有 c。

IR3 如果某能力 c_1 依赖于另一个能力具体 c_n，且 c_1 和 c_n 分别支持动作 a_1 和 a_n，则在模型中 a_n 必然存在且优先于 a_1 开展，即 c_n 优先于 c_{n-1}…优先于 c_2 优先于 c_1。

IR4 在规则 3 中，如果 c_1 是复杂能力 c_k 所包含的一个能力，且 c_k 支持动作 a_k，则在活动模型中 a_n 必然存在且优先于 a_k 开展。

IR5 在规则 3 中，如果 c_n 是一个抽象能力，有一个具体能力 c_k 继承 c_n 且支持活动 a_k，则在活动模型中只要 a_k 存在且优先于 a_1 开展，模型就是完整的。

用查询语句对每条规则进行形式化描述，建模人员使用本体推理工具和上述查询规则可完成对过程模型的一致性、完整性检验。

4 实例分析

项目组根据本文提出的方法开发了一套支持 C⁴ISR 系统的能力需求行为建模工具——基于本体的能力需求获取与分析工具(Ontology Based Capability Requirements Elicitation and Analysis Tools，OBCREAT)，C⁴ISR 系统开发人员可以使用该工具进行能力需求行为建模和模型验证。图 6 就是使用该建模工具构建的一个区域防空系统的导弹拦截活动模型。如果建模人员在构建导弹拦截活动模型时，由于疏忽遗漏了“导弹射击”和“导弹射击处理者”之间的转换线，那么“导弹部队”将无法等待到外部事件“射击结果”的到来，从而无法开始“高炮射击”或“效果评估”活动，那么整个区域的防空活动都将失败。此时模型中“导弹射击”和“威胁评估”之间的活动都是不能终止的，“导弹射击处理者”以后的活动都是开始活动无法到达的，即无法执行的，因此该模型是不一致的。

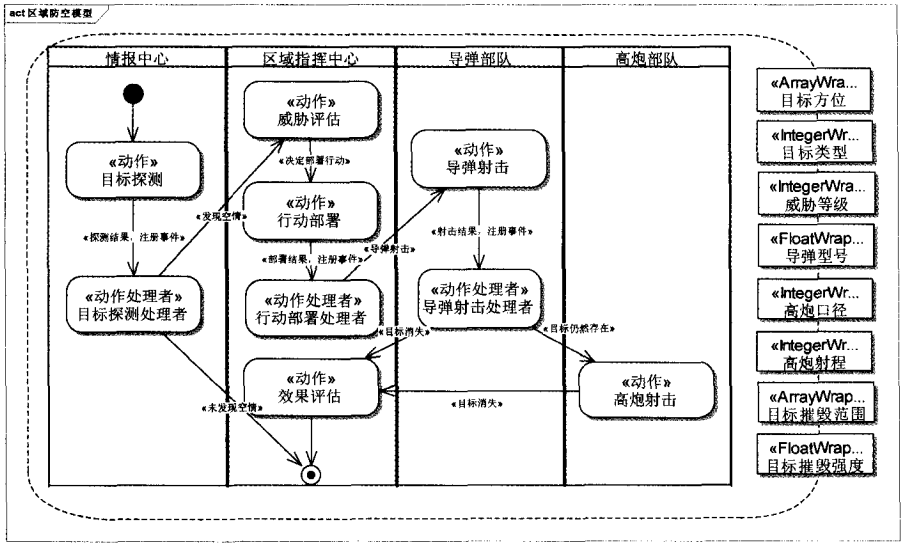

图 6 区域防空模型

在将过程模型形式化到 OWL DL 本体时，活动“目标探测”、“目标探测处理者”、“威胁评估”、“行动部署”、“行动部署处理者”、“导弹射击”、“导弹射击处理者”、“高炮射击”和“效果评估”的数据属性 ExecuteState 都为 0，表示它们都在等待执行。使用规则 1 对 OWL DL 本体进行推理后，活动“目标探测”、“目标探测处理者”、“威胁评估”、“行动部署”、“行动部署处理者”、“导弹射击”的数据属性 ExecuteState 都为 1，表示它们可以执行。但由于遗漏了“导弹射击”和“导弹射击处理者”之间的转换线，活动“导弹射击处理者”、“高炮射击”和“效果评估”的数据属性 ExecuteState 仍为 0，表示它们不能正常执行，违反了规则 CR4。使用规则 2 对 OWL DL 本体进行推理后，活动“导弹射击处理者”、“高炮射击”和“效果评估”的数据属性 ExecuteState 为 2，表示它们可以执行结束，但“导弹射击”和“威胁评估”之间的活动的数据属性 ExecuteState 仍为 0，表示它们不能正确执行结束，违反了规则 CR5。工具 OBCREAT 检查结果如图 7 所示，项目组已经将本体推理工具 Pellet 和验证规则集成到了开发的建模工具中，从而实现了模型的自动验证。

结束语 过程分析是可执行体系建模与仿真的基础，构建基于活动图的能力需求过程模型有益于对可执行体系建模与仿真的研究。本文研究了一种基于活动图的能力需求过程模型分析与验证方法。首先给出了系统过程模型的定义，然后在 C⁴ISR 系统能力元概念模型的指导下通过扩展 UML 活动图，引入能力、系统参数等概念，扩展针对 C⁴ISR 系统过程的事件处理机制、数据传递机制得到系统的能力需求过程元模型。然后用本体表示能力需求过程元模型语义，通过对本体的逻辑推理实现对 C⁴ISR 系统能力需求过程元模型一致性、完整性的验证。

参 考 文 献

[1] 孙严，戴浩. 基于能力的军事需求方法简介[J]. 科学技术与工程，2007，9(7)：2170-2176

[2] Joint Chief of STAFF. CJCI3170. 01D Joint Capability Integration and Development System[S/OL]. 2005-8-63. http://www.dtic.mil/cjcs_directives/index.htm

[3] MOD Partner. MOD Architecture Framework Overview Version1. 0(MODAF-M09-002) [S/OL]. 2005-8-31. http://www.modaf.org.uk/

(下转第 507 页)

```
if(y>0)
cout<<" **** ";
cout<<* x;
```

根据 weiser[5] 提出的程序静态切片的基本思想，得到程序中变量 x 的切片代码如下：

```
int* x,y;
x=new int;
* x=10;
cout<<* x;
```

5.2 对变量切片中变量所进行的操作进行标记

在获取变量切片的基础上，根据第 3 节中关于普通变量和指针变量的操作的分析，对切片中所涉及到的变量的操作进行标记，通过对变量的操作标记确定变量的操作。对 5.1 节中关于变量 x 的切片，针对涉及到变量 x 的操作进行如下标记：

```
1. int* x,y;
2. x=(int* )malloc(sizeof(int));[HA]
3. * x=10;[RM]
4. cout<<* x;[RM]
```

5.3 利用变量状态转换模型对变量切片中所涉及的变量的状态转换过程进行分析

下面以 5.2 节中的变量切片中的代码为例，利用第 3 节中的指针变量状态转换模型进行分析。

从初态出发，进行 HA 操作，根据变量的状态转换模型，确定后继状态为 PV_H；在 PV_H 状态下，进行 RM 操作，进入 PVUA_U 状态，该状态为状态转换模型中的终态，此时意味着程序中存在着动态内存未分配引用的错误，造成这种错误的原因是在动态内存分配之后没有进行分配结果的检查；在 PVUA_U 状态下，继续进行 RM 操作进入 PVUA_U 状态。在变量切片中的关于指针的所有操作都执行结束之后，指针变量的状态没有进入 PV_F 状态，意味着程序中存在着内存泄露的错误。

结束语 本文首先对应用程序中的变量的状态进行了分析，在此基础上，根据程序中变量引用的方式，建立了普通变量和指针变量的状态转换关系，并讨论了利用变量切片进行软件错误检测的方法。该方法在对程序进行静态分析的基础上进行，克服了动态检测方法中由于执行分支的限制而造成的路径覆盖不完整而引起的不能暴露所有错误的问题，为方便地检测程序中潜在的由于变量的异常使用导致的软件错误提供了帮助。

参 考 文 献

[1] 李普曼，拉乔伊，等. C++ Primer 中文版(第 5 版)[M]. 王刚，杨巨峰，译. 北京：电子工业出版社，2013：23-50

[2] Sattar H, Bajwa I S, et al. Automated DD-path testing: A challenging task in software testing[C]// Ninth International Conference on Digital Information Management. Phitsanulok, IEEE, 2010：230-236

[3] XuZhen-bo, Zhang Jian, Xu Zhong-xing. Memory Leak Detection Based on Memory State Transition Graph[C]// 18th Asia Pacific Software Engineering Conference, 2011. Ho Chi Minh, IEEE, 2011：33-40

[4] Sor V, Ou P, et al. Improving Statistical Approach for Memory Leak Detection Using Machine Learning[C]// 29th IEEE International Conference on Software Maintenance, 2013. Eindhoven, IEEE, 2013：544-547

[5] 李必信. 程序切片技术及其应用[M]. 北京：科学出版社，2006：3-4

(上接第 478 页)

[4] 王智学，董庆超，陈剑. 基于能力的复杂系统需求分析[C]// 江苏省系统工程学会军事系统工程委员会第十届学术年会. 2008：115-121

[5] 王智学，董庆超，陈彬，等. 基于 UML 模型的 C4ISR 系统能力需求分析与验证[J]. 系统工程与电子技术，2009，31(9)：2167-2171

[6] 董庆超，王智学，陈剑，等. 基于描述逻辑的能力需求模型验证方法[J]. 系统工程与电子技术，2010，32(3)：533-539

[7] Vincent C, Matthieu R. Interoperability constraints and requirements formal modelling and checking framework [C]// Ternational Federation for Information Processing. 2010：219-226

[8] Wagenhals L W, Haider S, Levis A H. Synthesizing executable models of object oriented architectures[C]// Workshop on Formal Methods Applied to Defence Systems. Adelaide, Australia, 2002

[9] 张炜钟. 指控系统能力需求的可执行建模及仿真评估技术研究[D]. 南京：解放军理工大学，2012

[10] 陈章耀. 模型驱动的业务生成技术中业务过程模型的研究与应用[D]. 北京：北京邮电大学，2008

[11] Group C4ISR Architecture Working. C4ISR architecture framework version2. 0 [R]. The United States: Department of Defense, 1997

[12] Brockmans S, Volz R, Eberhart A. Visual modeling of OWL DL ontologies using UML[M]// Lecture Notes on Computer Science 3298. 2004：198-213

[13] Van Der Straeten R. Inconsistency management in model-driven engineering: An approach using description logics [D]. Vrije Universiteit Brussel, 2005

[14] W3C Working Draft. SWRL: a semantic web rule language combining OWL and RuleML[EB/OL]. 2009-07-02. http://www.w3.org/Submission/SWRL/

[15] Motik B, Sattler U, Studer R. Query answering for OWL-DL with rules[C]// Proc. of the 3rd International Semantic Web Conference. 2004：549-563

[16] Glimm B, Horridge M, Parsia B, et al. Asyntax for rules in OWL 2[R]. Oxford: University of Oxford, 2008

[17] 朱雪峰，金芝. 关于软件需求中的不一致性管理[J]. 软件学报，2005，16(7)：1221-1231