基于Event-B的形式化建模关键技术研究

吴 劲，陈志慧

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】软件系统的规模和复杂程度不断提高而传统的需求分析方法难以确保软件的正确性和一致性，为软件系统的质量埋下了隐患。软件工程的实践表明，在开发过程中，错误发现得越早，修复得越早，付出的代价越小。为了确保软件的质量，可在软件开发的早期需求分析阶段，采用Event-B形式化方法描述软件的需求，并验证模型的正确性。以文件系统建模为例，该文讨论了如何利用Event-B方法，采用逐步精化的方式建立并验证模型，确保软件的正确性。

关 键 词 Event-B形式化方法; 形式化建模; 精化; 需求分析

中图分类号 TP311 文献标志码 A doi:10.3969/j.issn.1001-0548.2014.03.015

Research on Formally Modeling Based Event-B

WU Jin and CHEN Zhi-hui

(School of Computer Science & Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract The scale and complexity of software systems continue to increase and the traditional requirement analysis method is difficult to ensure the consistency and correctness of software, which is laying the trouble for the quality of a software system. Practice of the software engineering shows that in the software developing process, the earlier the errors are found and repaired, the less the cost is paid. In order to ensure the quality of software, on the early stage of software development, Event-B can be used to formally describe the software requirements and verify the correctness of the model. In the file system modeling as an example, this paper discusses how to use Event-B method and stepwise refinement to establish and validate the model, so as to ensure the correctness of software.

Key words Event-B formal method; formal modling; refinement; requirement analysis

随着全球信息化的不断深入，软件系统具有规模大且复杂度高的特点，而自然语言描述的软件需求具有不确定性、二义性且缺乏对软件需求进行严格检查的有效途径，因此无法确保软件需求的正确性、完善性和合理性。软件工程的实践表明，在开发过程中，错误发现得越早，修复得越早，付出的代价越小。为了确保软件的质量，在软件开发的早期需求分析阶段，采用形式化方法描述软件的需求，并验证模型的正确性，是确保软件质量的有效方法。

国内外众多学者研究如何有效地将形式化方法应用于实际的软件开发过程，在欧美国家已有将形式化方法应用到实际项目的成功案例。如法国采用B形式化方法开发了高速铁路控制系统，获得成功[1]。而Event-B方法是B方法[2]的简化，并吸取了其他的形式化方法的优点，包括Action Systems[3]、TLA+[4]、UNITY[5]等，适合开发安全性要求较高的大规模高复杂度软件系统。

本文以文件系统建模为例，基于Rodin平台采用Event-B语言，以逐步精化的方式向模型中添加属性和功能达到丰富、完善、细化模型的目的，并验证模型的正确性。

1 Event-B和Rodin平台

Event-B是一种用于进行系统级建模和分析的形式化方法[6]，它基于集合理论，在不同的抽象级构建系统，并逐步精细化，使用数学证明来保证不同精化级别之间的一致性。Rodin是一种用于开发复杂高可信软件系统的开放工具平台，它基于Event-B形式化方法，提供对精化和数学证明的自然支持。

Event-B软件系统模型如图1所示，包含两部分：静态属性和行为属性，分别用Context和Machine进行描述。Context由集合、常量、公理和定理组成，公

收稿日期：2013－04－08；修回日期：2014－03－06

基金项目：中央高校基本科研业务费(ZYGX2011J072)；国家科技重大专项(2011ZX03002-002-03)；国家自然科学基金重点项目(61133016)；四川省科技计划项目(2013GZ0022)

作者简介：吴劲(1972－)，女，博士，副教授，主要从事软件技术与理论方面的研究.

理用于描述集合和常量之间的关系，Context可以被继承，也可以被Machine引用。Machine由状态、不变式、事件和定理组成，其中状态是用变量进行定义的在模型中必须保证无论变量的值如何改变，不变式都成立，这一性质必须以证明义务的方式进行证明[7]。一个Machine可以包含多个原子事件，原子事件代表模型发生改变的方式。

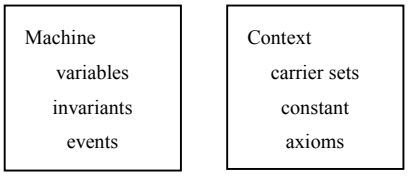

图1 Event-B模型

建模的过程就是一个逐步精化的过程，精化方式有两种：精化Machine的状态和精化Machine的事件，两种方式可同时使用。通常采用多个具体事件精化一个抽象事件，把多个抽象事件合并成一个抽象事件或引入新事件的方式来对Machine的事件进行精化。通过模型验证来确保软件需求模型的正确性，Rodin平台为Event-B模型验证提供了支持。

2 基于Event-B的形式化建模

本文基于Rodin平台采用Event-B语言对文件系统进行建模，首先建立文件系统的树型抽象模型，然后采用逐步精化的方式向模型中添加更多的设计细节，达到扩大模型的目的，并证明其正确性。

2.1 文件系统的初始模型

首先建立文件系统的初始模型，在这个抽象级别中将建立一个树型文件系统的初始模型，其需求描述如表1所示(Req代表需求)。

表1 初始模型的需求描述

初始模型	需求描述
Req1.1	树型结构有一个根节点
Req1.2	除根节点外的所有节点都有一个父节点
Req1.3	树型结构中没有环
Req1.4	可以从根节到达每个节点

2.1.1 Context的定义

首先创建树型文件系统初始模型的静态部分CTX01，定义集合OBJECT用于描述树型结构中的所有节点，定义常量root、objrel、tcl、objfn分别表示根节点、OBJECT到OBJECT的有序对的幂集、传递闭包、子节点与父节点的对应关系，它们必须满足以下公理：

axm1: root∈OBJECT

axm2: objfn=OBJECT\{root}⇸OBJECT

axm4:objrel=OBJECT↔OBJECT

axm3: tcl∈objrel→objrel

因为objfn是objrel的子集，所以定义定理：

thm5:objfn⊆objrel

2.1.2 Machine定义

创建树型文件系统初始模型的动态部分MCH00，引用CTX01，定义变量objects、parent，其中objects表示树型结构中的节点，parent表示树型结构中节点的父子对应关系，定义以下不变式：

inv1:objects∈ℙ(OBJECT)

inv6:root∈objects

inv8:parent∈objects\{root}→objects

inv10:∀s•(s⊆parent~[s] ⇒s=ϕ)

inv1表示objects是OBJECT的子集。inv6表示根节点是objects的一个元素，在这个抽象模型中，初始化事件将objects初始化为只包含root的集合，parent初始化为空集，规约了需求Req1.1。inv8表示parents是一个全函数，这个全函数定义了除根节点外的子节点到父节点的映射，实际表示除根节点外任何节点都有一个父节点，规约了需求Req1.2。inv10规约需求Req1.3，确保在树型文件系统中没有环，这个不变式的定义方式由文献[2]提出，parent~[s]得到的是集合s的直接子节点，如果s⊆parent~[s]且s不为空，则表示parent关系中存在环，因此这个不变式表示s为空集，即parent关系中没有环。

定义以下定理：

thm2:∀T•root∈T∧parent~[T]⊆T⇒objects⊆T

theorem

thm3:objects\{root}∪(tcl(parent))~[{root}]

theorem

thm4:(objects\{root})⊆(tcl(parent))~[{root}]

theorem

本文通过定理thm4对于需求Req1.4进行规约，确保从根节点能够到达每个节点，定理thm3用来证明thm2，定理thm4用来证明thm3。

在MCH00中，定义了5个抽象事件：创建(newobj)、删除(delete)、删除子树(deltree)、复制(copy)和移动(move)，其中事件copy和move操作类似，以copy为例说明事件的定义和规约方法，其定义如下：

grd1:obj∈objects\{root}

grd9:des⊆objects

grd7:des=(tcl(parent))~[{obj}]

grd3:to∈objects
grd11:to∉des∪{obj}
grd6:objs=des∪{obj}
grd5:nobjs⊆OBJECT\objects
grd10:corres∈objs↣↠nobjs
grd2:nobj=corres(obj)
grd4:subparent =des◁ parent
grd8:replica =corres~ ; subparent ; corres
THEN
act1:parent ≔parent∪replica∪{ nobj ↦ to }
act2:objects ≔objects∪nobjs

2.2 第一次精化

本节对初始模型进行第一次精化，对初始模型中的节点进行了区别，引入了文件和目录，第一次精化模型的需求描述如表2所示。

表2 第一次精化模型的需求描述

精化模型	需求描述
Req2.1	树型结构中的对象集合分成文件和目录
Req2.2	根节点root是目录
Req2.3	树型结构中每个对象的父节点一定是目录

创建MCH01对MCH00进行精化，在MCH01中定义了变量files、directories，继续使用了MCH00中的变量parent。变量files描述了树型文件系统中的所有文件的集合，变量directories描述了树型文件系统中的所有目录的集合。定义了以下不变式：

inv1:ran(parent)⊆directories
inv2:files⊆objects
inv3:directories⊆objects
inv4:files∩directories = ϕ
inv5:objects=files∪directories
inv6:root∈directories

不变式inv2定义了变量files的数据类型，表示files是objects的子集，描述的是树型文件系统中的文件。不变式inv3定义了变量directories的数据类型，表示directories是objects的子集，描述的是树型文件系统中的目录。不变式inv4表示files和directories没有交集，即不存在即是文件又是目录的节点，inv5表示文件系统中只有文件和目录这两种实体，inv4和inv5共同规约了Req2.1。不变式inv6表示root是一个目录，即规约了Req2.2。不变式inv1表示在parent关系中的父节点都是目录类型，即规约了Req2.3。

在初始化事件中，files初始化为空集表示，没有任何文件存在，directories初始化为只含有根目录root，因为初始化情况下只有一个root目录，所以也就不存在相关的parent关系，即parent等于ϕ。在machine MCH01中，不变式inv5使用了machine MCH00中的变量objects，所以inv5是一个联接不变式，且在inv5将抽象变量objects定义为files∪directories，所以machine MCH01中的所有objects都可以用files∪directories进行替代。

在此次的事件精化的步骤是：事件mkdir和crt_file共同精化抽象事件newobj；事件move精化抽象事件move；事件delfile和rmdir共同精化抽象事件delete；事件copy精化抽象事件copy；事件deltree精化抽象事件deltree。

2.3 第二次精化

在本次精化阶段，为模型引入了文件内容，文件缓冲区和意外掉电处理，根据前面的描述，第二次精化模型的需求描述如表3所示。

表3 第二次精化模型的需求描述

第二次精化模型	需求描述
Req3.1	任何一个文件都有文件内容(内容可以为空)
Req3.2	在读写操作前要先打开文件
Req3.3	每个打开的文件都分配一个缓冲区，关闭文件时释放缓冲区
Req3.4	当掉电时任何操作都无法执行

2.3.1 Context的精化

创建继承CTX01的CTX02，增加3个集合DATA、NAME、DATE，其中DATA表示数据块，NAME表示名字，DATE表示时间。它们必须满足以下公理：

axm1:CONTENT = ℕ ↛ DATA
axm2:∅ ∈CONTENT
axm3:rname∈NAME
axm4:dfdate∈DATE
axm5:nowdate∈DATE
axm7:∀c·c∈CONTENT⇒ finite(c)

CONTENT表示文件内容，axm1说明是CONTENT是从N映射到DATA的部分函数；axm2表示文件的内容可以为空；axm7表示文件内容的长度是有限的。

2.3.2 事件的精化

创建machine MCH02对machine MCH01进行精化，引用CTX02。MCH02的不变式定义如下：

inv1:fcontent∈files → CONTENT not theorem
inv2:w_opened_files⊆files not theorem

inv3:r_opened_files⊆files not theorem

inv4:w_opened_files∩r_opened_files=ϕ

inv5: wbuffer∈w_opened_files→ CONTENT

inv6:rbuffer∈r_opened_files → CONTENT

inv7:power_on∈BOOL not theorem

inv8:power_on= FALSE ⇒(w_opened_files=ϕ ∧r_opened_files=ϕ∧rbuffer=ϕ∧wbuffer= ϕ)

不变式inv1表示fcontent是一个从filles映射到CONTENT的全函数，规约了需求Req3.1。inv2、inv3、inv4规约了需求Req3.2。inv5、inv6规约了需求Req3.3。inv7、inv8规约了需求Req3.4。

在此次精化过程中，添加了新事件w_open、writefile、r_open、readfile、close、power_loss、power_on。对事件mkdir、crt_file、move、delfile、rmdir、copy、deltree分别精化相应的抽象事件。

2.4 第三次精化

本次精化的目标就是把名字、创建时间、修改时间以及文件大小这4个属性引入模型中。第三次精化模型的需求描述如表4所示。

表4 第三次精化模型的需求描述

第三次精化模型	需求描述
Req4.1	文件和目录都有一个名字
Req4.2	文件和目录都具有创建时间
Req4.3	文件和目录都具有修改时间
Req4.4	文件都有文件大小属性

创建MCH03，对MCH02进行精化。在MCH03中，增加了4个变量，其中变量oname表示文件或目录的名字，变量dateCreated表示文件或目录的创建时间，变量dateLastModified表示文件或目录的最后修改时间，变量file_size表示文件的大小。定义了以下不变式：

inv1:oname∈(files∪directories)→NAME

inv2:dateCreated∈(files∪directories)→DATE

inv3:dateLastModified∈(files∪directories)→DATE

inv4:file_size∈ files → N

不变式inv1规约了需求Req4.1，inv2规约了需求Req4.2，inv3规约了需求Req4.3，inv4规约了需求Req4.4。

在此次精化过程中，增加了新事件rename，对MCH02中的相应事件mkdir、crt_file、move、delfile、rmdir、copy、deltree、writefile分别进行了精化。

3 模型验证

通过3次精化文件系统的模型已经建立，然而工作并没有结束，用形式化方法建立的模型要经过严格地数学验证才可以确保模型的正确性，即模型生成的所有证明义务都得以证明才表示建立的模型是正确的。Rodin平台不但为建立模型提供了开发环境而且为模型的验证提供了支持，Rodin为开发者提供了一套自动化模型验证工具，简化了复杂且繁琐的验证过程。本文建立的树型文件系统模型生成的所有证明义务都得以证明，证明结果如图2所示。

图2 文件系统模型成功证明截图

4 结 束 语

软件系统的规模和复杂程度不断提高而传统的需求分析方法难以确保软件的正确性和一致性，为软件系统的质量埋下了隐患。本文以文件系统建模为例，在软件开发的早期需求分析阶段，采用Event-B形式化方法描述软件的需求，采用逐步精化的方式建立并验证模型，确保了软件的正确性，对复杂软件系统的开发具有较好的借鉴作用。

参 考 文 献

[1] ABRIAL J R. Formal methods: Theory becoming practice[J]. Journal of Universal Computer Science, 2007, 13(5): 619-628.

[2] ABRIAL J R. The B-book: Assigning programs to meanings[M]. Cambridge: Cambridge University Press, 1996.

[3] BACK R J, KURKI-SUONIO R. Distributed cooperation with action systems[J]. ACM Transaction on Programming Languages and Systems, 1988, 10(4): 513-554.

[4] LAMPORT L. Specifying systems: the TLA+ language and tools for hardware and software engineers[M]. Boston: Addison-Wesley, 1999.

[5] CHANDY K M, MISRA J. Parallel program design, a foundation[M]. Boston: Addison-Wesley, 1988.

[6] ABRIAL J R. Modelling in Event-B: System and software engineering[M]. Cambridge: Cambridge University Press,

2010.
[7] HALLERSTEDE S. On the purpose of Event-B proof obligations[J]. Formal Aspects of Computing, 2011, 23(1): 133-150.

编 辑 漆 蓉