计算机科学
COMPUTER SCIENCE

基于自然语言需求的SCADE模型测试用例自动生成方法

邵温欣, 杨志斌, 李维, 周勇

引用本文

邵温欣, 杨志斌, 李维, 周勇. 基于自然语言需求的SCADE模型测试用例自动生成方法[J]. 计算机科学, 2024, 51(7): 29-39.

SHAO Wenxin, YANG Zhibin, LI Wei, ZHOU Yong. Natural Language Requirements Based Approach for Automatic Test Cases Generation of SCADE Models [J]. Computer Science, 2024, 51(7): 29-39.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

面向ARINC653操作系统的综合化航空电子软件代码自动生成方法

Integrated Avionics Software Code Automatic Generation Method for ARINC653 Operating System

计算机科学, 2024, 51(7): 10-21. https://doi.org/10.11896/jsjkx.230600216

基于贝叶斯攻击图的网络资产安全评估模型

Network Asset Security Assessment Model Based on Bayesian Attack Graph

计算机科学, 2023, 50(12): 349-358. https://doi.org/10.11896/jsjkx.221000019

基于图卷积网络的甲型流感H3N2抗原性预测

Antigenicity Prediction of Influenza A/H3N2 Based on Graph Convolutional Networks

计算机科学, 2023, 50(11A): 230100113-6. https://doi.org/10.11896/jsjkx.230100113

基于安全强化学习的航天器交会制导方法

Spacecraft Rendezvous Guidance Method Based on Safe Reinforcement Learning

计算机科学, 2023, 50(8): 271-279. https://doi.org/10.11896/jsjkx.220700210

基于改进Cascade R-CNN的布匹瑕疵检测算法

Fabric Defect Detection Algorithm Based on Improved Cascade R-CNN

计算机科学, 2023, 50(6A): 220300224-6. https://doi.org/10.11896/jsjkx.220300224

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230600126

基于自然语言需求的SCADE模型测试用例自动生成方法

邵温欣[1,2]　杨志斌[1,2]　李　维[3]　周　勇[1,2]
1 南京航空航天大学计算机科学与技术学院　南京 211106
2 高安全系统的软件开发与验证技术工信部重点实验室　南京 211106
3 航空防护救生技术航空科技重点实验室　湖北 襄阳 441003
(wxshao821@163.com)

摘　要　随着安全关键软件规模和复杂性不断增加,模型驱动开发方法在安全关键领域得到了广泛应用。SCADE作为一种重要的建模方法和工具,能够表达确定性并发行为且具有精确时间语义等特性,适用于安全关键软件的建模、测试与验证。目前,已有方法主要采用手工方式构造SCADE模型测试用例,存在需求与测试用例不一致、成本代价高且容易出错的问题。文中提出了一种基于自然语言需求的SCADE模型测试用例自动生成方法。首先,给出了基于模型检测的测试用例自动生成方法,通过自然语言需求处理生成原子命题,用于生成前提假设Assume和观察者模型,同时给出了陷阱性质(Trap Properties)生成规则来生成陷阱性质用于模型检测;其次,给出了基于覆盖分析和变异测试的测试用例质量评估方法,并在SCADE模型上进行变异测试;最后,设计和实现了原型工具,并基于一个工业界案例飞行员弹射座椅控制系统进行了案例分析,验证了所提方法的有效性。

关键词:安全关键软件;模型驱动开发;SCADE;测试用例自动生成;模型检测
中图分类号　TP311

Natural Language Requirements Based Approach for Automatic Test Cases Generation of SCADE Models

SHAO Wenxin[1,2], YANG Zhibin[1,2], LI Wei[3] and ZHOU Yong[1,2]
1 School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
2 Key Laboratory of Safety-critical Software, Ministry of Industry and Information Technology, Nanjing 211106, China
3 Aviation Key Laboratory of Science and Technology on Life-support Technology, Xiangyang, Hubei 441003, China

Abstract　With the increasing scale and complexity of safety-critical software, model-driven development(MDD) is widely used in safety-critical fields. As an important modeling method and tool, SCADE can express deterministic concurrent behavior and has precise time semantics, which is suitable for modeling, testing and verification of safety-critical software. At present, the existing methods mainly use manual methods to construct SCADE model test cases, and there are some problems such as inconsistency between requirements and test cases, high cost and easy to make mistakes. This paper presents an automatic generation method of SCADE model test cases based on natural language requirements. Firstly, an automatic test case generation method based on model checking is presented, which generates atomic propositions by natural language requirements processing to generate the assume and observer models, and provides the rules of trap properties generation to generate trap properties for model checking. Secondly, a test case quality evaluation method based on coverage analysis and mutation testing is presented, and the mutation testing is carried out on SCADE model. Finally, the prototype tool is designed and implemented, and an industrial case of pilot ejection seat control system is analyzed to verify the effectiveness of the proposed method.

Keywords　Safety-critical software, Model-driven development, SCADE, Automatic test case generation, Model checking

1　引言

安全关键软件(Safety-critical Software)[1]指应用于航空、航天、交通、核能等领域的安全关键系统中,且其运行情况可能引起系统处于危险状态,从而导致财产损失、环境破坏或人员伤害的一类软件。这类软件系统对安全性、可靠性、

到稿日期:2023-06-15　返修日期:2023-10-18
基金项目:国家自然科学基金(62072233);国防基础科研项目(JCKY2020205C006);航空科学基金(201919052002);南京航空航天大学科研与实践创新计划(xcxjh20221607)
This work was supported by the National Natural Science Foundation of China(62072233), National Defense Basic Scientific Research Project (JCKY2020205C006), Aeronautical Science Foundation of China(201919052002) and Postgraduate Research & Practice Innovation Program of NUAA(xcxjh20221607).
通信作者:杨志斌(yangzhibin168@163.com)

实时性有极高的要求。

近年来，模型驱动尤其是采用形式化模型驱动的安全关键软件设计与开发方法逐渐受到重视，并被工业界认为是切实可行的重要手段[2]。例如，国际民航领域使用的机载系统适航审定中的软件开发标准 DO-178C[3] 就将模型驱动和形式化方法（即 DO-331[4] 和 DO-333[5]）作为其核心标准的重要技术补充。模型驱动开发方法将模型作为整个软件开发过程的核心元素，在设计阶段就建立软件模型，并尽早进行验证和分析[6]。

目前主流的建模语言有 SysML，AADL，SCADE[7]，Simulink 等。其中 SysML，AADL 等建模语言用来描述体系架构模型，而 SCADE 和 Simulink 则被广泛应用于功能模块设计。SCADE 建模方法的基础是同步语言，可通过一系列严密的数学公式推导，其语言是精确的、无歧义的、确定的，因此适合描述安全关键系统和软件的控制逻辑[8]。同时，SCADE 提供了一套用于基于模型的控制系统的开发工具，包括对 SCADE 模型的验证与测试。针对复杂系统，工业界目前主要还是采用测试手段进行验证活动。

目前，已有研究主要采用手工构造 SCADE 模型测试用例的方法，面临着成本代价高、容易出错、所编写的测试用例与需求之间可能存在不一致等挑战[9-12]。测试用例自动生成（Automatic Test Case Generation，ATCG）技术作为缩短测试用例生成时间、降低测试成本和缩短软件开发周期的主要方法，成为了一个重要的研究方向[13]。

ATCG 相关工作可以分为白盒测试和黑盒测试两种。白盒测试主要通过分析模型的结构来覆盖路径，从而生成相应的测试用例。例如 Tian 等[14] 提出了一种自动生成 UML 模型状态图的测试用例方法，以满足 UML 建模需求。他们将 UML 状态图转换为有向图，获得有向图的路径，并基于路径生成逻辑序列。与白盒测试不同，黑盒测试不关心模型的内部结构，主要是基于需求生成测试用例，而基于模型检测的测试用例自动生成是其中的一种重要方法[15]，并且可以为给定的覆盖准则自动生成测试用例[16]。Aniculaesi 等[17] 提出了基于模型检测的 SCADE 模型测试用例自动生成方法。首先，将自然语言需求手工转换为线性时序逻辑 LTL 公式；接着将取反后的 LTL 公式作为陷阱性质（系统性质的否定称为陷阱性质）转换为可以集成到 SCADE 环境中的非确定性 Büchi 自动机；然后使用 SCADE 中的模型检测工具（SCADE Design Verifier，SDV）进行验证得到反例；最后将反例转换为一个测试用例，该测试用例可以在 SCADE 测试环境中的待测系统（System under Test，SuT）上执行。然而，针对复杂安全关键软件的 SCADE 模型测试，该方法存在状态空间爆炸问题。

SCADE 模型检测工具通过遍历模型状态空间进行形式化验证，但是随着模型复杂程度的提高，模型输入参数、控制流程不断增加，模型的状态空间也随之呈指数级增长，可能会出现状态空间爆炸问题导致验证失败，因此需要对状态空间进行削减约束。基于契约的组合验证方法通过为每个构件添加与需求相对应的形式化的 Assume-Guarantee 契约来解决上述问题。Assume 是外部环境对构件的约束，一般体现在对构件的输入空间进行约束，即各个参数的取值范围。通过 Assume 可以约束构件乃至整个系统的状态空间。而 Guarantee则描述了构件应当满足的性质，可以是构件的功能性需求，也可以是构件的安全性需求。

为此，本文提出了一种新的 SCADE 模型测试用例自动生成方法。相比 Aniculaesei 等提出的方法，本文的目标是从自然语言需求自动生成前提假设 Assume 和观察者模型（即 Guarantee）。同时，提出了陷阱性质生成规则，以提高测试用例的 MC/DC（Modified Condition/Decision Coverage）覆盖率。最后通过 MC/DC 覆盖率分析和变异测试（Mutation Testing）评估得到的测试用例的质量。

本文的主要贡献如下：

1）提出了基于模型检测的 SCADE 模型测试用例生成方法。首先，通过自然语言需求处理提取原子命题，并将需求转换为一种中间格式 ICBT；其次，提出了考虑 MC/DC 覆盖的陷阱性质生成规则，并将中间格式转换为若干个陷阱性质；然后，为每个陷阱性质生成对应的前提假设 Assume 和观察者模型；最后，使用 SDV 进行验证，得到的反例即为对应需求的测试用例。

2）给出了基于覆盖分析和变异测试的测试用例评价方法。首先，使用 SCADE TEST 工具自动化分析测试用例的 MC/DC 覆盖率；其次，设计并提出了面向 SCADE 模型的变异算子，并基于变异算子得到 SCADE 变异模型；最后，基于相同测试用例对原始模型和变异模型进行测试，从而评价测试用例的缺陷发现能力。

3）设计并实现了原型工具，并基于工业界案例飞行员弹射座椅控制系统进行了分析，验证了所提方法的可行性和有效性。

本文第 2 章介绍了相关背景知识；第 3 章介绍了测试用例生成及其评价方法；第 4 章介绍了工业界案例弹射座椅控制系统；第 5 章介绍了原型工具并进行实验分析；第 6 章介绍了相关工作；最后总结全文，并对今后的工作进行展望。

2 背景知识

本章概述了 SCADE、MC/DC 覆盖率和变异测试的相关知识。

2.1 SCADE

SCADE 是一个用于基于模型的安全关键软件开发与验证工具。安全关键软件往往采用层次化的软件构件组合架构，在 SCADE 中采用操作符（Operator）来表达一个软件构件，每个操作符包含若干个输入和输出。

SCADE 支持使用数据流方程和安全状态机（Safe State Machine，SSM）来对系统的功能行为进行建模，并且数据流模型和控制流模型可以组合使用。数据流方程描述了输入变量和输出变量之间的逻辑关系，用户可以利用 SCADE 提供的基础操作符进行设计。安全状态机则通过迁移和状态描述离散系统中不同的触发条件下系统状态的变化。

SCADE 提供的模型检测工具 SDV 支持验证带有简单时间属性的功能逻辑，其主要原理如图 1 所示。首先，将验证性质表达成一个观察者（Observer）；然后，将待验证模型和观察

者合并构造成一个验证模型(Proof Operator)，某些时候甚至可以包括前提假设 Assume，验证模型的结果是一个布尔输出参数，它表示验证的结果；最后 SDV 建立一个有限状态转换系统并遍历可达状态空间，若找到不满足验证性质的状态，则返回一个反例，否则验证通过。SCADE 提供了前提假设语句(Assume-state)，用于表示前提假设条件 Assume，另外还提供了时序验证运算符包 libverif，包括多个时间运算符。

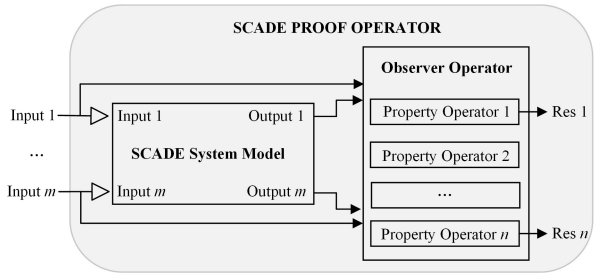

图 1 SCADE 模型验证的原理

Fig. 1 Principle of SCADE model verification

2.2 MC/DC 覆盖

MC/DC 覆盖准则是安全关键软件领域中使用的重要覆盖标准之一，用于满足对复杂布尔表达式进行测试的需要[18]。该准则要求布尔表达式中的每个原子条件独立影响结果，具体包括：

1)模型中的每个入口点和出口点都至少被调用过一次；

2)模型中决策的每一个基本条件都至少对所有可能的结果进行了一次取值；

3)每个基本条件已被证明会独立影响决策结果。

其中，基本条件是不能分解的原子布尔表达式。

例如，公式($A \wedge B$)，当测试用例为(TT，FF)时，表达式结果为(T，F)，满足判定覆盖(Decision Coverage，DC)，即表达式的结果至少一次为真和为假；同时 A 和 B 的取值分别为真和为假一次，满足条件覆盖(Condition Coverage，CC)。但当 A 为假时，整个公式的结果必然为假，B 条件没有独立影响结果。为此，需要考虑 MC/DC 覆盖，增加一个测试用例(TF)，保证 B 对结果的独立影响。

2.3 变异测试

变异测试是一种基于故障注入的测试技术，主要目的是验证测试用例的有效性[19]。实施变异测试时，首先采用变异算子对原程序的某个语句进行合乎语法的微小变动，如改变常数的值等，得到的新程序称为变异体，相应的语法变动规则称为变异算子。其次，基于给定的测试用例集，分别执行原程序和变异体，如果某变异体执行结果不同于原程序，则称该变异体被杀死；如果对于任何测试用例，这两个程序的输出都是相同的，则称该变异体是等价变异体[20]。

3 测试用例自动生成方法 ReqTG

本章介绍了基于自然语言需求的 SCADE 测试用例生成方法 ReqTG，包括基于模型检测的测试用例生成和基于覆盖率分析和变异测试的测试用例质量评估。图 2 给出了所提方法的总体框架。

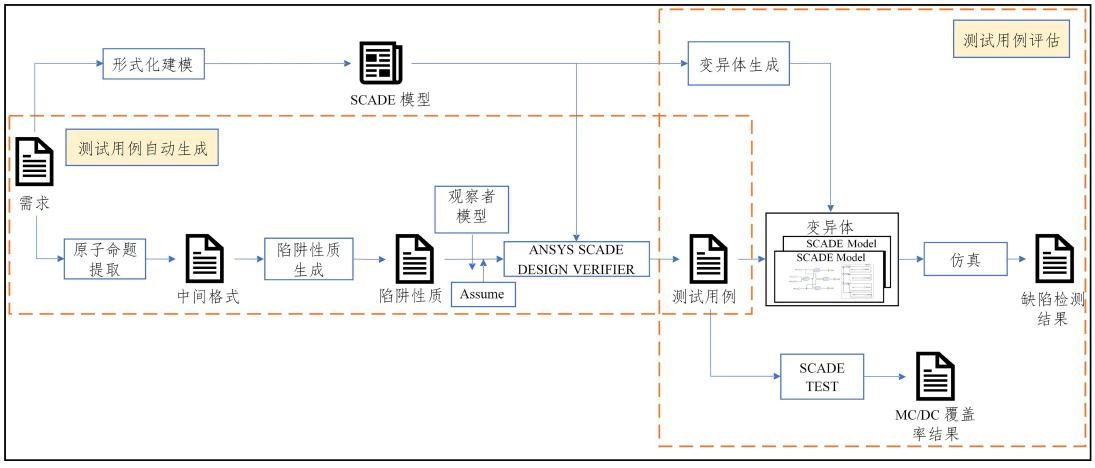

图 2 ReqTG 方法的框架

Fig. 2 Framework of ReqTG

3.1 自然语言需求处理

在航空航天领域，安全关键软件的需求规格说明中，系统需求一般需要显式地给出系统功能的上下文和具体功能处理等描述。例如，火箭加电控制子系统加电 1 控制函数的一条需求如下：

条件：供电 1 指令是单步模式且满足发控表参数；功能处理：系统将向供电版发送加电控制，提取识别码和加电令，并回告系统加电 1 控制函数执行成功。

其中，条件对应前提假设 Assume，功能处理对应 Guarantee。有时系统功能的上下文可能是隐含的，专门有相关工作研究在这类情况下 Assume 的自动生成，例如文献[21-22]研究了基于机器学习的 Assume 自动生成。

为了将自然语言需求转换到逻辑公式，首先需要提取自然语言需求中的原子命题。基于文献[23]的工作，使用自然语言处理方法提取原子命题，涉及需求文本预处理、句子结构解析及生成原子命题 3 个步骤。首先，使用文本预处理工具 HanLP[24] 对需求文本做预处理，得到句子中的依存关系，完成分词、词性标注，以此建立依存句法树；其次，使用基于优先级和依存关系的句子结构提取算法解析句子结构，通过句子结构建立标准句法树；最后，根据标准句法树叶子节点生成原子命题。图 3 为原子命题生成流程图，对于叶子节点对应的简单句，提取其中的主语、谓语、

宾语、时间约束和全域标识符，按主谓宾状的顺序重新组织，生成原子命题。

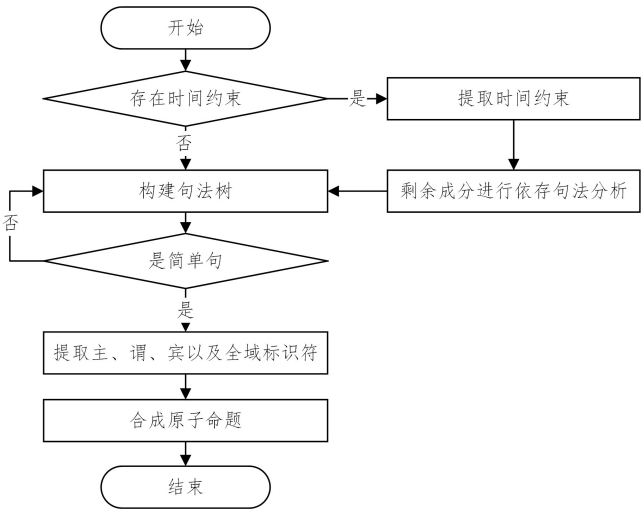

图 3 提取原子命题的流程

Fig. 3 Flowchart of extracting atomic propositions

3.2 中间格式 ICBT

为了方便地从自然语言需求生成前提假设 Assume 和观察者模型，本文提出了一种中间格式 ICBT，即四元组〈*Init*，*Context*，*Behavior*，*Time*〉。*Context* 和 *Behavior* 是 3.1 节中提取出的原子命题集合。其中，*Init* 表示需求的初始条件，来源于需求文档中的数据字典；*Context* 是需求中的前提假设条件，即 3.1 节中提到的上下文条件；*Behavior* 是满足 *Init* 和 *Context* 条件下期望的行为，即 3.1 节中的功能处理；*Time* 是行为发生的延迟时间。对于特定的 SCADE 模型，时间可以作为一个精确的周期来计算。

在初始 *Init* 条件下，如果 $Context_i$ 为 *true*，则 $Behavior_i$ 将发生在 $Time_i$ 时间内，如式(1)所示：

$$Init \wedge Context_i \overset{Time_i}{\Rightarrow} Behavior_i \tag{1}$$

一般，*Context* 和 *Behavior* 是成对出现的。这里给出自然语言需求到中间格式的一个转换示例，如图 4 所示。图 4(a)给出了弹射控制系统的一条需求；图 4(b)给出了提取原子命题的结果，包括生成的句法树和原子命题集合；图 4(c)展示了转换后的中间格式。

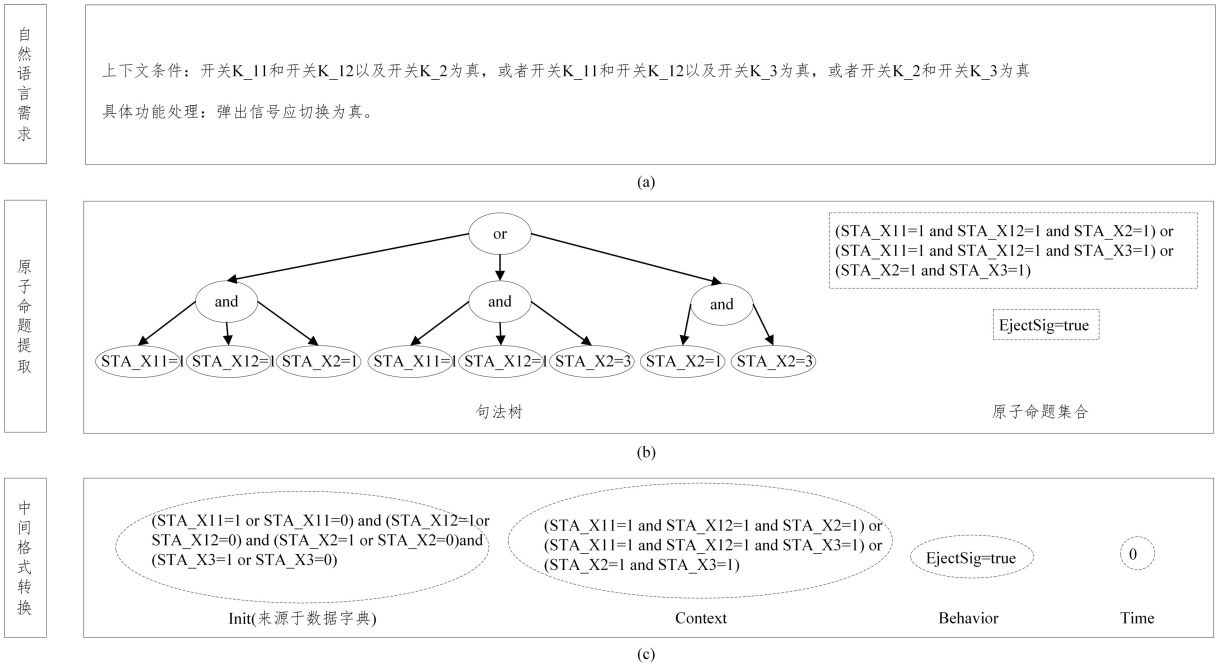

图 4 运行示例

Fig. 4 Running example

3.3 陷阱性质生成

通过构造陷阱性质作为模型检查器的验证条件是生成测试用例的一种常用方法。最简单的陷阱性质构造方式是对验证性质取反。如果陷阱性质不满足并得到反例，则说明该反例满足原始验证性质，即可以作为一条测试用例[25]。

下文给出基于中间格式的陷阱性质构造规则。根据 3.2 节中的式(1) 所示，*Init* 条件和 *Context* 条件是需求中对上下文的约束，在构造陷阱性质时，需要保证其取值为真；而 *Behavior* 条件描述了功能行为，需要对其取反以得到陷阱性质。文献[17]未考虑模型的 MC/DC 覆盖率，为了尽可能满足适航标准对 A 级软件的要求，本文在 Context 条件中应用准则 1 来构建陷阱性质，以达到较高的 MC/DC 覆盖率。

准则 1 对于 *Context* 条件中的命题公式，将其转换为析取范式(Disjunctive Normal Form，DNF)，析取范式由多个合取式析取而成。为了使得 *Context* 条件为真，只需要析取范式中的任意一个合取式取值为真。基于 MC/DC 覆盖要求，我们需要确保每个合取式都独立影响结果，即给定合取式 P_j 取值为真，其余合取式取值为假，如式(2)所示：

$$\neg P_1 \wedge \neg P_2 \wedge \cdots \wedge P_j \wedge \neg P_{j+1} \wedge \cdots \tag{2}$$

另外，考虑到可能出现的异常测试用例，即不满足 *Context* 条件或 *Init* 条件的测试用例，使用准则 2 和准则 3 构造陷阱性质。

准则 2 如果一个需求中所有 *Context* 条件的析取不覆盖取值的全域，则须存在不满足所有 *Context* 条件析取的测试用例，本文使用式(3)来构造这样的陷阱性质。

$$\neg(Context_1 \vee Context_2 \vee \cdots \vee Context_n \Leftrightarrow true) \Rightarrow$$

$$\neg(Context_1 \vee Context_2 \vee \cdots \vee Context_n) \quad (3)$$

准则 3　构造不满足 *Init* 条件的陷阱性质，如式(4)所示：

$$\neg Init \quad (4)$$

最后，为了得到 *Behavior* 的否定构造陷阱性质，使用准则 4 对其取反。

准则 4　对于 *Behavior* 命题公式，其构造的 *TP* 如式(5)所示：

$$\neg Behavior \quad (5)$$

3.4　基于模型检测的测试用例自动生成

如 2.1 节所述，SCADE 提供了前提假设语句(assume-state)，用于表示 *Context* 条件和 *Init* 条件，并基于观察者操作符对要验证的 *Behavior* 进行建模。另外，SCADE 提供了时序验证运算符包 libverif，包括多个时间运算符(Time)。因此，将陷阱性质转换到 SCADE 模型检测所需的验证模型并使用模型检测器 SDV 进行验证。如果陷阱性质不满足并得到反例，则说明该反例满足原始验证性质，即可以作为一条测试用例。

3.5　测试用例评估

通过覆盖率分析和变异测试来评估测试用例的质量。

3.5.1　覆盖率分析

SCADE 提供了 SCADE TEST 工具进行模型覆盖率分析，其方法是：根据选定的覆盖准则和被测模型进行模型插桩，生成新的只读插桩模型，然后在该插桩模型中导入仿真用例进行覆盖分析，并以图形或文本形式展示覆盖分析结果。本文主要通过 SCADE TEST 工具来评估测试用例的 MC/DC 覆盖率。

3.5.2　变异测试与评价

程序级别的变异测试通过对原始程序进行符合语法的微小变动来构造变异程序，而本文通过对 SCADE 模型上的建模元素进行变异来得到变异模型。SCADE 模型通常包含两类元素，即变量和运算符，其中变量包括输入、输出和常量，这些常量具有多种数据类型，如整型和浮点型；而运算符可分为关系运算符、算术运算符、逻辑运算符、时态运算符等。

本文设计了如下变异算子：变量的变异算子包括变量替换变量(VRV)、常数替换变量(VRC)、常量替换常量(CRC)和变量替换常量(CRV)；运算符的变异算子包括算术运算符替换(ARO)、关系运算符替换(RRO)、逻辑运算符替换(LRO)、位运算符替换(BRO)、一元运算符插入(UOI)、一元运算符删除(UOD)和时态运算符参数替换(TPR)。变异操作符如表 1 所列。

表 1　变异操作符

Table1　Mutation operators

操作符	描述	操作对象
VRV	变量替换	同类型的输入
VRC	常量替换变量	同类型的输入和常量
CRC	常量替换(数值型：+/−1；布尔型：否定)	同类型的常量
CRV	变量替换常量	同类型的输入和常量
ARO	算术运算符替换	+，−，*，/，mode
RRO	关系运算符替换	>，>=，<，<=，<>，=
LRO	逻辑运算符替换	and，or，xor
BRO	位操作符替换	land，lor，lxor
UOI	一元运算符插入	unary minus，not
UOD	一元运算符删除	unary minus，not
TPR	时序运算符参数替换	FBY，Init

另外，通过变异分数(Mutation Score，MS)来评估模型 *P* 的一组测试用例 *T* 的质量。其计算式如下：

$$MS(P,T)=\frac{D}{M-E} \quad (6)$$

其中，*D* 是被杀死的变异体的总数；*M* 是所有变异体的总数；*E* 是等价变异体的总数。变异分数越高，测试用例的质量越好。

4　SCADE 模型案例

本章简要介绍弹射座椅控制系统(Ejection Seat Control System，ESCS)及其 SCADE 模型。

4.1　ESCS 系统介绍

ESCS 的主要功能是收集和监测弹射系统的各种弹射离机状态参数和弹射后座椅系统飞行运动参数(如弹射启动开关信号、实时速度、高度、俯仰角、滚转角、加速度、角速度、纬度等)，并根据系统的嵌入式多模式数字弹射控制模型来自动地实时选择弹射控制程序以控制偏转器。根据火箭发动机、姿态火箭、分离引爆机构、降落伞炮等工作机构是否工作以及其工作时间，来实现多模式控制模式，确保飞行员能够在特定条件下与座椅一起弹射出机舱，最终在适当的高度、速度和姿态下实现座椅分离和降落伞打开，并确保飞行员的生命安全，这是一类典型的安全关键软件。

ESCS 软件在两种模式下工作：正常飞行模式和弹射控制模式。在正常飞行模式下，完成系统初始化后，ESCS 软件将持续监测弹射启动开关信号。如果弹射启动开关信号监测结果满足弹射控制模式的进入条件，则系统将进入弹射控制模式。在一系列控制模式选择后，最终点火输出完成。ESCS 软件的工作状态转换图如图 5 所示。

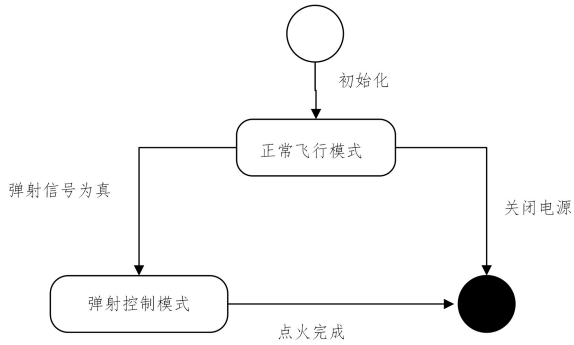

图 5　ESCS 工作状态转换图

Fig. 5　Working state transition diagram of ESCS

4.2　SCADE 建模

基于 SCADE 建模，整个 ESCS 系统由 5 个构件组成：弹射判断模块(EJM)、实时控制模式(RCM)、模式选择模块(MSC)、高空模式(HAM)和表速计算模块(SCM)。EJM，MSM 和 SCM 这 3 个模块使用数据流建模，而 HAM 和 RCM 使用控制流建模。整个系统的顶层操作符是 RCM，其他 4 个模块是独立的，同时集成到 RCM 中。图 6 给出了 ESCS 的 SCADE 模型结构。

图 7 给出了 EJM 模块的详细信息。EJM 中有 4 个布尔信号 STA_X11，STA_X12，STA_X2，STA_X3 作为输入，一个布尔信号 EjectSig 作为输出，其中 STA_X1 的值是 STA_

X11 信号量和 STA_X12 信号量相与的结果。如果 3 个开关信号量中 2 个(STA_X1,STA_X2,STA_X3)的值为真,则输出信号EjectSig 为真,这意味着弹射开关信号满足弹射控制模式的进入条件,系统将进入实时控制模式。

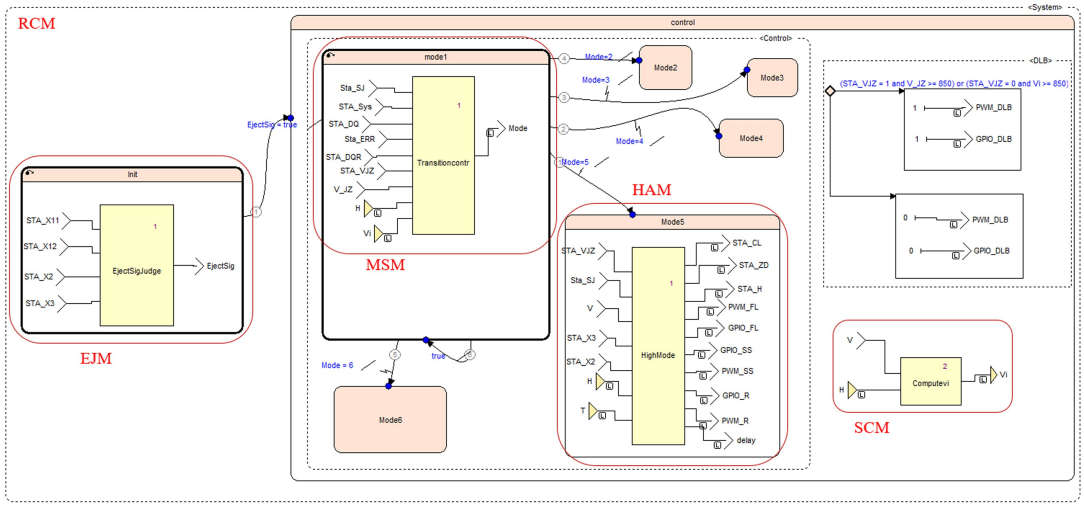

图 6 ESCS 系统的 SCADE 模型结构

Fig. 6 SCADE model structure of ESCS

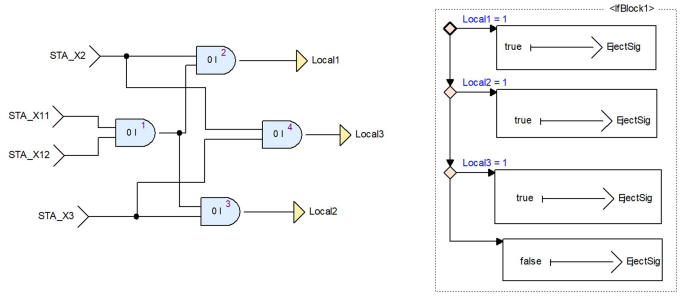

图 7 EJM 模块

Fig. 7 EJM module

图 8 给出了控制流建模的 HAM 的具体细节。在高空模态中,包含 7 个输入和 10 个输出信号量。整个构件由一个安全状态机建模,共有 15 个状态和相应的迁移条件,该构件需要对座椅姿态、速度、高度进行调整,保证飞行员能尽快以合适的速度下降到合适的高度再进行人椅分离和开伞动作。

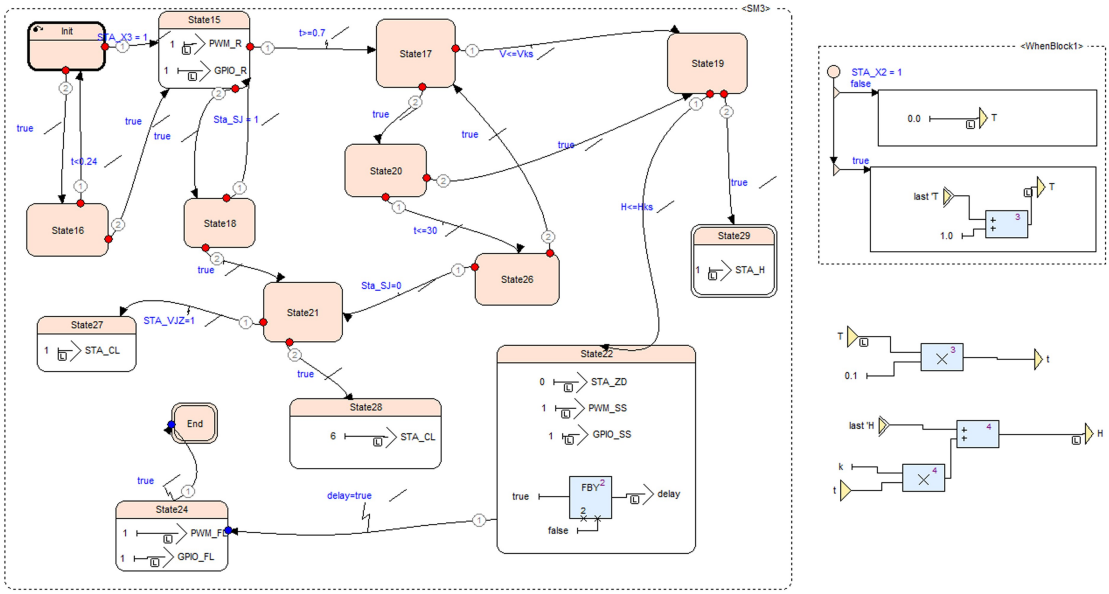

图 8 HAM 模块

Fig. 8 HAM module

5 原型工具和案例分析

本章主要介绍 SCADE 测试用例生成工具的设计与实现及案例分析。

5.1 原型工具

ReqTG 原型工具主要包括 3 个模块，即上传需求文件和 Init 文件、陷阱性质生成，以及测试用例生成，各模块的功能如下。

1）上传需求和 Init 文件：基于自然语言需求提取原子命题并转换到中间格式 ICBT。

2）陷阱性质生成：生成陷阱性质并展示在工具界面。

3）测试用例生成：将陷阱性质转换为 SCADE 验证模型并调用 SDV 进行验证，使用 Dom4j 解析验证得到的反例 XML 文件，得到测试用例并生成对应的测试脚本文件。

5.2 案例分析

5.2.1 自然语言需求转换为中间格式

本文以弹射判断模式（EJM）为例，其中一条自然语言需求如下。

Req1：如果开关 K_11 和开关 K_12 以及开关 K_2 为真，或者开关 K_11 和开关 K_12 以及开关 K_3 为真，或者开关 K_2 和开关 K_3 为真，则弹出信号应切换为真。

根据自然语言需求 Req1 转换的中间格式如表 2 所列。其中，*Init* 条件提取自需求文档中的数据字典。此外，*Req*1 的 *Time* 参数为零，因为当满足 *Init* 条件和 *Context* 条件时，*Behavior* 行为立即发生。

表 2 Req1 的中间格式

Table 2 Intermediate format of Req1

四元组元素	内容
Init	(STA_X11＝1 or STA_X11＝0) and(STA_X12＝1or STA_X12＝0) and(STA_X2＝1 or STA_X2＝0) and (STA_X3＝1 or STA_X3＝0)
Context	(STA_X11＝1 and STA_X12＝1 and STA_X2＝1) or (STA_X11＝1 and STA_X12＝1 and STA_X3＝1) or (STA_X2＝1 and STA_X3＝1)
Behavior	EjectSig＝true
Time	0

5.2.2 从中间格式生成陷阱性质

根据 3.3 节中提出的准则，得到 Req1 相应的陷阱性质 TPs，如表 3 所列。其中，TP1，TP2 和 TP3 的 *Context* 条件由准则 1 构造，*Behavior* 条件由准则 4 构造；TP4 的 *Context* 条件由准则 2 构造；TP5 的 *Init* 条件由准则 3 构造。

表 3 Req1 的陷阱属性

Table 3 Trap properties of Req1

TP 名称	*Init*	*Context*	*Behavior*	*Time*
TP1	(STA_X11＝1 or STA_X11＝0) and(STA_X12＝1 or STA_X12＝0) and(STA_X2＝1 or STA_X2＝0) and (STA_X3＝1 or STA_X3＝0)	STA_X11＝1 and STA_X12＝1 and STA_X2＝1 and STA_X3＝0	EjectSig＝false	0
TP2	(STA_X11＝1 or STA_X11＝0) and(STA_X12＝1 or STA_X12＝0) and(STA_X2＝1 or STA_X2＝0) and (STA_X3＝1 or STA_X3＝0)	STA_X11＝1 and STA_X12＝1 and STA_X2＝0 and STA_X3＝1	EjectSig＝false	0
TP3	(STA_X11＝1 or STA_X11＝0) and(STA_X12＝1 or STA_X12＝0) and(STA_X2＝1 or STA_X2＝0) and (STA_X3＝1 or STA_X3＝0)	STA_X11＝0 and STA_X12＝0 and STA_X2＝1 and STA_X3＝1	EjectSig＝false	0
TP4	(STA_X11＝1 or STA_X11＝0) and(STA_X12＝1 or STA_X12＝0) and(STA_X2＝1 or STA_X2＝0) and (STA_X3＝1 or STA_X3＝0)	not[(STA_X11＝1 and STA_X12＝1 and STA_X2＝1) or(STA_X11＝1 and STA_X12＝1 and STA_X3＝1) or(STA_X2＝1 and STA_X3＝1)]	EjectSig＝true	0
TP5	not[(STA_X11＝1 or STA_X11＝0) and(STA_X12＝1 or STA_X12＝0) and(STA_X2＝1 or STA_X2＝0) and(STA_X3＝1 or STA_X3＝0)]	(STA_X11＝1 and STA_X12＝1 and STA_X2＝1) or (STA_X11＝1 and STA_X12＝1 and STA_X3＝1) or (STA_X2＝1 and STA_X3＝1)	EjectSig＝true	0

5.2.3 从陷阱性质生成测试用例

首先，ReqTG 工具将陷阱性质转换成 SCADE 以验证模型的前提假设 Assume 和观察者模型。如图 9 所示，红色文字部分为前提假设 Assume，TP1_Behavior 为所生成的观察者模型。

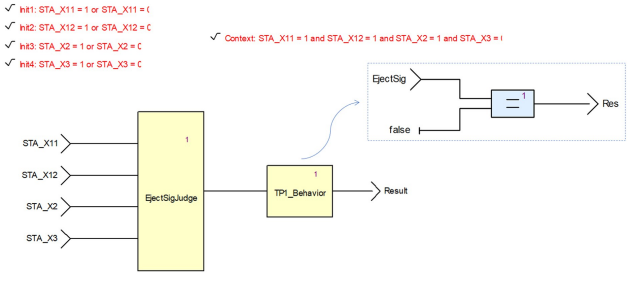

图 9 TP1 的验证模型（电子版为彩图）

Fig. 9 Validation model of TP1

其次，使用 SDV 进行验证得到反例，即测试用例脚本（.sss 文件），每个 TP 对应生成一个测试用例。图 10 给出了验证 TP1 时获得的反例，即信号量 STA_X11，STA_X12，STA_X2 的值为 1，STA_X3 的值为 0。

```
SSM::set EjectSigJudge/STA_X11 1
SSM::set EjectSigJudge/STA_X12 1
SSM::set EjectSigJudge/STA_X2 1
SSM::set EjectSigJudge/STA_X3 0

SSM::cycle
```

图 10 TP1 生成的对应测试用例

Fig. 10 Test case generated by TP1

为了便于理解本文方法的工作流程，展示的案例需求是整个系统中较为简单的一条，并且针对的是数据流模型。而对于控制流模型，生成的测试用例往往是多个周期的，较难通过人工方式编写。图 11 给出了一个通过 ReqTG 方法生成

的 HAM 模块的测试用例。该测试用例给出了 HAM 模块的 7 个输入信号每个周期的取值。

```
周期1
SSM::set HighMode/STA_VJZ 1
SSM::set HighMode/Sta_SJ 1
SSM::set HighMode/V 0
SSM::set HighMode/STA_X3 0
SSM::set HighMode/STA_X2 1
SSM::set HighMode/Horigin 4096
SSM::set HighMode/currentT 0

SSM::cycle

周期2
SSM::set HighMode/STA_VJZ 1
SSM::set HighMode/Sta_SJ 0
SSM::set HighMode/V 5.90043063e-39
SSM::set HighMode/STA_X3 0
SSM::set HighMode/STA_X2 1
SSM::set HighMode/Horigin 3521
SSM::set HighMode/currentT 3.95451979e+19

SSM::cycle

周期3
SSM::set HighMode/STA_VJZ 1
SSM::set HighMode/Sta_SJ 1
SSM::set HighMode/V 5.00000095
SSM::set HighMode/STA_X3 0
SSM::set HighMode/STA_X2 1
SSM::set HighMode/Horigin 1273
SSM::set HighMode/currentT 4.05693138e+19

SSM::cycle

周期4
SSM::set HighMode/STA_VJZ 1
SSM::set HighMode/Sta_SJ 1
SSM::set HighMode/V 512
SSM::set HighMode/STA_X3 0
SSM::set HighMode/STA_X2 0
SSM::set HighMode/Horigin 531
SSM::set HighMode/currentT 1.71136623e+16

SSM::cycle

周期5
SSM::set HighMode/STA_VJZ 1
SSM::set HighMode/Sta_SJ 1
SSM::set HighMode/V 1024
SSM::set HighMode/STA_X3 0
SSM::set HighMode/STA_X2 1
SSM::set HighMode/Horigin 2.90625
SSM::set HighMode/currentT 5.73333454

SSM::cycle
```

图 11 HAM 模块的一条测试用例

Fig. 11 Test case of HAM

5.2.4 测试用例的 MC/DC 覆盖分析

使用 SCADE TEST 来测量测试用例的覆盖率，结果如图 12 所示。生成的测试用例覆盖了模型的 43 个观测点，达到了 100%的 MC/DC 覆盖率。

Covered or justified / Total: 43 / 43 (100.00 %)

Covered or justified / Coverable: 43 / 43 (100.00 %)

Operator	Observed		Not observed				Total	% of total	% of coverable
			Coverable		Not coverable				
	Justified	Not justified	Justified	Not justified	Justified	Not justified			
EjectSigJudge	0	43	0	0	0	0	43	100.00 %	100.00 %
Total	0	43	0	0	0	0	43	100.00 %	100.00 %

图 12 Req1 的测试用例的 MC/DC 覆盖率

Fig. 12 MC/DC Coverage of test cases generated for Req1

5.2.5 变异测试评估

使用 3.5.2 节提出的变异算子，手工对原始模型注入变异，得到变异体模型。图 13 给出了一个引入了 CRC 变异算子后的变异体模型，将原始模型中的 true 常量变换为 false。

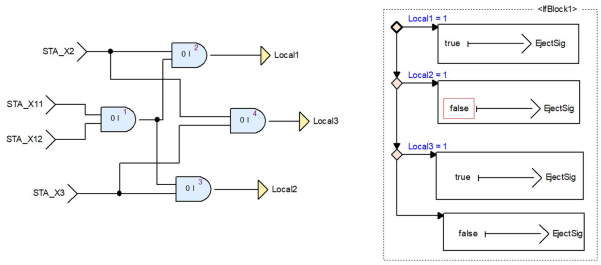

图 13 变异体模型

Fig. 13 Mutation model

5.2.6 实验统计结果及分析

本小节给出整个弹射座椅控制系统案例分析结果。表 4 列出了每个模块的类型、输入和输出数量。

表 4 实验参数

Table 4 Experiments parameters

模块	类型	输入数量	输出数量
EJM	数据流	4	1
HAM	控制流	7	10
MSM	数据流	9	1
RCM	控制流	13	14
SCM	数据流	2	1

表 5 列出了测试用例覆盖率分析结果，包括具体模块、对应的陷阱性质数量、生成的测试用例数量、平均生成时间和 MC/DC 覆盖率。

表 5 MC/DC 覆盖率的实验结果

Table 5 Results of MC/DC coverage

模块	TPs	测试用例	时间/s	MC/DC 覆盖率/%
EJM	5	5	0.40	100.00
HAM	18	18	186.65	98.01
MSM	12	12	0.50	100.00
RCM	5	5	73.40	96.92
SCM	5	5	51.60	100.00

从表中可以看出，除了 SCM 模块之外，数据流模型(EJM，MSM) 的验证过程通常比控制流模型(HAM，RCM) 更快。在 SCM 模块中使用了求幂和平方根运算的复杂计算，这是导致验证速度较慢的原因。

表 6 列出了变异测试分析结果，包括变异体数量、变异体使用的变异算子、被杀死的变异体数量、等价变异体数量以及变异分数。

表 6 变异测试实验结果

Table 6 Results of mutation testing

模块	变异体数量	使用的变异算子	杀死的变异体	等价变异体	变异分数
EJM	23	VRV，CRC，RRO，BRO，UOI	19	1	0.8632
HAM	60	VRV，VRC，CRC，CRV，ARO，RRO，LRO，TPR	53	2	0.9138
MSM	53	VRV，VRC，CRC，RRO，LRO，UOI	48	1	0.9231
RCM	70	VRV，VRC，CRC，CRV，ARO，RRO，LRO，UOI	51	9	0.8361
SCM	60	VRV，VRC，CRC，ARO，RRO，LRO，UOI	50	2	0.8621

从实验结果可以看出，ReqTG 方法生成的测试用例能够达到 96% 的 MC/DC 覆盖率，可以有效地减少测试工程师手工设计测试用例的成本，手工添加少数测试用例就可以达到 100% 的 MC/DC 覆盖率。同时，平均变异分数也高达 0.87，说明测试用例的缺陷发现能力较强。

生成的测试用例未达到 100% 的 MC/DC 覆盖率可能是因为以下外部风险：

1)测试用例不能完整执行：使用 SDV 验证时，一旦找到一个不满足性质的反例，SDV 就会立即终止验证并将反例返回。对于控制流模型来说，返回的反例是一条输入信号的跟踪(Trace)，给出了每个周期的输入值，并且在最后一个周期出现了不满足验证性质的状态。在这种情况下，可能会出现测试用例不能完整执行进而导致 MC/DC 覆盖率未能达到 100%。

2)在对案例进行建模时，某些输出信号量同时被用作状态转换的激活条件。然而，在覆盖率分析中，这些作为转换条件的输出变量被认为是无法覆盖的。

6 相关工作

6.1 基于模型的测试用例自动生成

近年来，基于模型的测试技术(MBT) 凭借高测试效率成为软件测试领域中一种主流的测试方法。与传统的测试方法相比，MBT 可以以更快、更便宜、更高效的方式维护和实现测试职责[25]。使用 MBT 方法，可以在软件开发的早期阶段就开始测试活动，缩短了整体的测试时间[26]；并且在设计阶段可以识别和修复需求规格说明和设计文档中的不一致和歧义，从而降低开发成本，因为错误不会进一步传播和提前消除[27]。常用的 MBT 技术有 UML 建模方法和有限状态机 FSM 建模方法。Li 等[28]在面向状态识别的测试方法中，考虑约束集对所生成状态区分序列的影响，基于带约束的测试生成树产生相应的特征集、状态识别集和 UIO 序列，提出或者改进了相应的算法。同时，他们将测试方法扩展到了 NFSM 的情形下，提出了 NFSM 模型中前缀序列的生成算法和状态识别集的构建算法；结合状态识别矩阵与有限状态机同步乘积，提出了在 NFSM 模型中的适应性测试方法，扩展了 FSM 应用于测试理论的完备性。

最近有几项研究专门用于模型驱动开发工具的测试用例的自动生成[29]。Minj 等[30]基于 Simulink 的 Stateflow 模型的 XML 文档创建了一个节点链表(包括源、目标和条件节点)，将该模型转换为有限状态机模型，并基于获得的有限状态机生成测试用例。

6.2 基于需求的测试用例自动生成

Moitra 等[31]介绍了由通用电气开发的工具 ASSERT，该工具基于形式化需求自动生成测试用例。ASSERT 从需求捕获环境(Requirement Catch Environment，RCE) 开始，该环境允许工程师以人类可读和计算机可分析的明确方式编写需求。然后，使用需求分析引擎(Requirement Analysis Engine，RAE) 对需求进行分析，以消除需求中的冲突、不完整性和其他错误，并根据测试策略自动生成测试目标，随着测试目标的过滤、优化和简化，生成最小有效测试用例。Yang 等[32]利用模型驱动的思想，阐述了测试需求的概念并定义了一种测试需求的元模型和测试需求建模方法。利用测试需求建模方法可以得到测试需求模型，从而得到相应的测试目标，生成对应的测试用例，并给出了对 SIP 协议进行测试的实验分析。

模型检测是一种基于需求构建测试用例的自动化方法，Durrieu 等[33]分析了基于形式化技术的自动测试用例生成在实践中是可行的。Aniculaesei 等[34]于 2018 年提出了一种使用 NuSMV 模型检查器的自适应巡航系统自动测试用例生成方法，并考虑了 Whalen 等[35]于 2006 年提出的 3 个基于需求的覆盖标准，即 RC(Requirement coverage)、AC(Antecedent coverage) 和 UFC(Unique first coverage)，最后通过变异测试评价生成的测试用例的质量。鉴于 SCADE 工具链的成熟，Aniculaesi 等考虑将这种方法集成到 SCADE 工具链[36]中，使用 SCADE 工具中的 SDV 作为模型检查器，他们在自动驾驶系统的门锁系统中进行实验，得到了较好的效果。为了与在 NuSMV 上实现的结果进行比较，Aniculaesei 等也在自适应巡航系统上使用了 SCADE 工具链方法，但仅使用了 RC 标准。

结束语 本文提出了一种基于自然语言需求的 SCADE 模型测试用例自动生成方法。首先，给出了基于模型检测的 SCADE 模型测试用例生成，通过自然语言需求处理提取原子命题并将需求转换为一种中间格式 ICBT，再将中间格式转换为陷阱性质，为每个陷阱性质生成对应的前提假设 Assume 和观察者模型，并使用 SDV 进行验证，得到的反例即为对应需求的测试用例；其次，给出了基于 MC/DC 覆盖分析和变异测试的测试用例质量评价方法；最后，设计和实现了原型工具，并基于工业界案例飞行员弹射座椅控制系统进行了案例分析，验证了本文方法的有效性。

在未来的工作中，首先考虑更为复杂的安全关键软件需求句式并扩充陷阱性质生成准则；其次，研究基于 SMT 求解器的 SCADE 模型检测方法，以提高 SCADE 模型的形式化验证能力；最后，在对 SCADE 模型进行变异测试时，目前依赖手动方法进行变异注入。开发自动化模型变异测试工具是我们未来工作的重点。

参 考 文 献

[1] YANG Z B, YUAN S H, XIE J, et al. A synchronous language multi-threaded code automatic generation tool[J]. Journal of Software, 2019, 30(7): 23.

[2] SINGH P, SINGH L K. Reliability and Safety Engineering for Safety Critical Systems: An Interview Study With Industry Practitioners[J]. IEEE Transactions on Reliability, 2021, 70(2): 643-653.

[3] RIERSON L. Developing safety-critical software: a practical guide for aviation software and DO-178C compliance[M]. CRC Press, 2017.

[4] DO-331, Model-based development and verification supplement to DO-178C and DO-278A[S]. RTCA: Washington, DC, USA, 2011.

[5] DO-333, Formal methods supplement to DO-178C and DO-278A[S]. RTCA: Washington, DC, USA, 2011.

[6] WAKANKAR A, BHATTACHARJEE A K, DHODAPKAR S D, et al. Automatic test case generation in model based software design to achieve higher reliability[C]// 2010 2nd International Conference on Reliability, Safety and Hazard - Risk-Based Technologies and Physics-of-Failure Methods (ICRESH). 2010: 493-499.

[7] SCADE[OL]. https://www.ansys.com/products/em-beddedsoftware/ansys-scade-suite.

[8] JING H, SHEN Y Y. ANSYS SCADE Suite Modeling Foundation [M]. Beijing: China Water Resources and Hydropower Press, 2018.

[9] ELKHOLY W, EL-MENSHAWY M, BENTAHAR J, et al. Model checking intelligent avionics systems for test cases generation using multi-agent systems[J]. Expert Systems with Applications, 2020, 156: 113458.

[10] KADRY S. A New Proposed Technique to Improve Software Regression Testing Cost[J]. International Journal of Security & Its Applications, 2011, 5(3): 21-39.

[11] BRILLOUT A, HE N, MAZZUCCHI M, et al. Mutation-Based Test Case Generation for Simulink Models[C]// Formal Methods for Components and Objects. Berlin, Heidelberg: Springer, 2010: 208-227.

[12] WANG W X, HU J, HU J C, et al. Test Case Generation Method Oriented to Tabular Form Formal Requirement Model[J]. Computer Science, 2021, 48(5): 16-24.

[13] DAI X, GONG W, GU Q. Automated test case generation based on differential evolution with node branch archive[J]. Computers & Industrial Engineering, 2021, 156: 107290.

[14] TIAN Y, YIN B, LI C. A Model-based Test Cases Generation Method for Spacecraft Software[C]// 2021 8th International Conference on Dependable Systems and Their Applications (DSA). 2021: 373-382.

[15] RAYADURGAM S, HEIMDAHL M P E. Coverage based test-case generation using model checkers[C]// Proceedings of the Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS 2001). 2001: 83-91.

[16] CHANDLER R E. Model Checking[M]. MIT Press, 2005.

[17] ANICULAESEI A, VORWALD A, RAUSCH A. Using the SCADE Toolchain to Generate Requirements-Based Test Cases for an Adaptive Cruise Control System[C]// 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). 2019: 503-513.

[18] CHILENDKI J J, MILLER S P. Applicability of modified condition/decision coverage to software testing[J]. Software Engineering Journal, 1994, 9(5): 193-200.

[19] DEMILLO R A, LIPTON R J, SAYWARD F G. Hints on Test Data Selection: Help for the Practicing Programmer[J]. Computer, 1978, 11(4): 34-41.

[20] YANG H, HONG M, QU Y Y. Approach of Mutation Test Case Generation Based on Model Checking[J]. Computer Science, 2018, 45(11A): 488-493.

[21] GAALOUL K, MENGHI C, NEJATI S, et al. Combining Genetic Programming and Model Checking to Generate Environment Assumptions[J]. arXiv: 2101.01933, 2021.

[22] ZHANG Z L, YANG Z B, LI X J, et al. Machine Learning Based Environment Assumption Automatic Generation for Compositional Verification of SCADE Models[J]. Computer Science, 2023, 50(6): 297-306.

[23] YANG Z B, YANG Y Q, YUAN S H, et al. Terminology Recommendation and Requirement Classification Method for Safety-critical Software[J]. Computer Science, 2021, 48(5): 32-44.

[24] HE H, CHOI J D. The Stem Cell Hypothesis: Dilemma behind Multi-Task Learning with Transformer Encoders[J/OL]. https://arxiv.org/abs/2109.06939.

[25] WHALEN M W, RAJAN A, HEIMDAHL M P E, et al. Coverage metrics for requirements-based testing[C]// Proceedings of the 2006 International Symposium on Software Testing and Dnalysis. New York, NY, USA: Association for Computing Machinery, 2006: 25-36.

[26] MA C, PROVOST J. Design-to-Test Approach for Programmable Controllers in Safety-Critical Automation Systems[J]. IEEE Transactions on Industrial Informatics, 2020, 16(10): 6499-6508.

[27] MOHD-SHAFIE M L, KADIR W M N W, LICHTER H, et al. Model-based test case generation and prioritization: a systematic literature review[J]. Software and Systems Modeling, 2022, 21(2): 717-753.

[28] LI Y P, LI H, ZHAO J L. Research about FSM Test Sequence Generation Algorithm[J]. Computer Science, 2016, 43(Z11): 474-481.

[29] HUANG S Q, LIU Y B, HUANG X S. Research on Automatic Testing Technology of Model Driven Development Tools[J].

Computer Science,2021,48(6A):568-571.

[30] MINJ B. Generation of Test Cases based on Analysis of Simulink Stateflow Models[C]// IJCA Proceedings on National Conference on Knowledge, Innovation in Technology and Engineering(NCKITE 2015). Foundation of Computer Science(FCS), 2015.

[31] MOITRA A, SIU K, CRAPO A W, et al. Automating requirements analysis and test case generation[J]. Requirements Engineering, 2019, 24(3): 341-364.

[32] YANG B, WU J, XU L, et al. A Method for Modeling Software Testing Requirements and Generating Test Cases [J]. Chinese Journal of Computers, 2014, 37(3): 17.

[33] DURRIEU G, LAURENT O, SEGUIN C, et al. Formal Proof and Test Case Generation for Critical Embedded Systems Using Scade[C] // Building the Information Society. Boston, MA: Springer US, 2004: 499-504.

[34] ANICULAESEI A, HOWAR F, DENECKE P, et al. Automated generation of requirements-based test cases for an adaptive cruise control system[C]// 2018 IEEE Workshop on Validation, Analysis and Evolution of Software Tests(VST). 2018: 11-15.

[35] WHALEN M W, RAJAN A, HEIMDAHL M P E, et al. Coverage metrics for requirements-based testing[C]// Proceedings of the 2006 International Symposium on Software Testing and Analysis. New York, NY, USA: Association for Computing Machinery, 2006: 25-36.

[36] ANICULAESEI A, VORWALD A, RAUSCH A. Automated Generation of Requirements-Based Test Cases for an Automotive Function using the SCADE Toolchain[C]// The 11th International Conference on Adaptive and Self-Ada Ptive Systems and Application. 2019.

SHAO Wenxin, born in 1998, postgraduate. Her main research interests include safety critical-system and formal verification.

YANG Zhibin, born in 1982, Ph.D, professor, is a member of CCF (No. 08632M). His main research interests include safety-critical system, formal verification and AI software engineering.

(责任编辑:喻藜)