

一种面向列车车载控制软件的需求分析方法

王秀超¹, 缪炜恺², 王以松¹, 包丹珠², 杨 静¹

(1. 贵州大学 计算机科学与技术学院, 贵阳 550025; 2. 华东师范大学 计算机科学与软件工程学院, 上海 200062)

摘 要: 由于缺少面向列车车载控制软件的需求分析工具, 需求分析人员难以自动分析出预期数据。针对该问题, 提出基于模型抽取的需求分析方法, 设计控制软件需求分析工具。通过抽取需求描述文档中的需求条目建立可执行模型, 使用数据流方法对其进行分析, 得到变量影响关系图和状态迁移图, 利用动态分析获得以需求描述文档为依据的预期输出数据, 并将其与软件的真实运行输出数据进行对比, 生成控制软件的可视化分析报告。实验结果表明, 该方法可以帮助需求分析人员快速发现软件实现错误, 为列车安全运行提供保障。

关键词: 列车车载控制软件; 需求分析; 抽象语法树; 数据流分析; 环路检测

中文引用格式: 王秀超, 缪炜恺, 王以松, 等. 一种面向列车车载控制软件的需求分析方法[J]. 计算机工程, 2017, 43(7): 48-53.

英文引用格式: Wang Xiuchao, Miao Weikai, Wang Yisong, et al. A Requirements Analysis Method for On-board Train Control Software[J]. Computer Engineering, 2017, 43(7): 48-53.

A Requirements Analysis Method for On-board Train Control Software

WANG Xiuchao¹, MIAO Weikai², WANG Yisong¹, BAO Danzhu², YANG Jing¹

(1. College of Computer Science and Technology, Guizhou University, Guiyang 550025, China;

2. School of Computer Science and Software Engineering, East China Normal University, Shanghai 200062, China)

[Abstract] Since there is no requirements analysis tool for on-board train control software, requirements analysis researchers are unable to get expected data. Aiming at this problem, this paper proposes a requirements analysis method based on model extraction and designs a Control Software Requirements Analysis (CSRA) tool. It constructs an executable model by extracting the requirement items of requirements description documents, uses data flow method to analyze the model and obtain the Variable Dependency Diagram (VDD) and State Transition Diagram (STD). By using dynamic analysis, it gets the expected output data based on the requirements description documents and compares the data with real output data to generate an visual analysis report of the control software. Experimental results show that the proposed method can help requirements analysis researchers quickly find errors in software implementation, which provides a guarantee for the safe running of trains.

[Key words] on-board train control software; requirements analysis; abstract syntax tree; data flow analysis; loop detection
DOI: 10.3969/j.issn.1000-3428.2017.07.008

0 概述

列车车载控制软件是保证列车运行安全、缩短行车间隔、提高运行效率的重要软件, 任何缺陷都可能导致重大人身伤亡和财产损失, 有较高的安全性和可靠性要求^[1]。在学术界和工业界, 现已投入大量的人员对列车车载控制软件的安全性进行研究。

文献[2]通过形式化方法 VDM++ 分析列车控制软件的需求描述文档, 对控制软件进行一致性证明和可满足性检查。需求描述文档对软件做详细的分析, 包含软件预期功能、定义数据、数据之间的影响关系。因此, 对需求描述文件进行分析能够保证软件的可靠性^[3]。文献[4]以形式化语义作为基础设计了建模语言 SPARDL (Space Air Craft Description

基金项目: 国家自然科学基金“嵌入式控制软件的形式化规格说明构建的工程方法”(61402178); 国家自然科学基金“基于 rCOS 的形式化方法需求分析与验证”(61562011)。

作者简介: 王秀超 (1992—), 男, 硕士研究生, 主研方向为控制系统分析验证; 缪炜恺, 副教授、博士; 王以松, 教授、博士; 包丹珠, 硕士; 杨 静, 教授、博士。

收稿日期: 2016-12-23 **修回日期:** 2017-01-24 **E-mail:** hswangxiuchao@126.com

Language), 以图像化方法表示模式图, 分析需求的准确性, 消除了需求描述的二义性。上述研究都为列车安全运行提供了保障。

近年来, 软件开发者对需求分析的重视程度正逐渐增加。文献[5]分析传统软件开发方法的不足, 提出基于模型的开发方法, 在高安全性应用开发环境(Safety-Critical Application Development Environment, SCADE)下设计列车控制软件, 并对设计的模型进行安全性验证, 可使工程师避免书写具有二义性的需求。文献[6]使用对象精化演算系统(Refinement Calculus of Object System, rCOS)设计列车需求模型, 对模型进行安全验证, 并开发安全验证工具 tMDA(trustable Model-Driven Architecture)。文献[7]基于反馈学习的需求验证方法, 在需求描述中建立有场景问题的查询模型, 以交互的形式验证系统的行为是否需求。文献[8]提出启发式规则方法, 用于更好地理解复杂系统的需求, 以此来获得更详细的需求, 甚至可以得到需求的输入。文献[9]提出基于本体的需求分析方法, 在本体和需求描述之间建立映射关系, 检测语义的完备性和不一致性。尽管研究者已提出上述多种方法, 但对于复杂的周期性控制软件, 对其需求进行分析仍然较为困难。

需求分析是控制软件开发的基础, 为保证列车的安全性, 必须对控制软件的运行数据进行分析, 验证软件是否达到需求所描述的预期结果^[10]。但由于缺少相关工具的支持, 需求分析人员无法自动得到控制软件的预期输出结果。针对这一问题, 本文提出面向列车车载控制软件的需求分析方法, 开发一套控制软件需求分析(Control Software Requirements Analysis, CSRA)工具。通过分析控制软件变量之间的关系, 计算列车运行的预期输出数据, 并与软件数据进行对比, 减少需求分析人员的工作量, 提高其工作效率, 缩短软件开发的周期。

1 控制软件需求分析方法框架

在整个软件开发周期中, 工程师根据需求描述文档进行软件设计, 需求分析人员对需求描述文档的分析是其中一个非常重要的环节。在基于模型抽取的需求分析方法中, 建立可执行模型时首先需要保证需求描述文档的正确性。只有正确的需求文档才能保证控制软件中的变量符合软件的预期设计, 从而使所构建的可执行模型计算出正确的预期输出。因此, 在使用 CSRA 工具对列车车载控制软件运行输出数据和预期输出数据进行分析对比时, 只有得到正确的预期输出数据, 才能保证分析结果的正确性。

本文设计基于 CSRA 工具的需求分析方法, 其基本流程如图 1 所示。

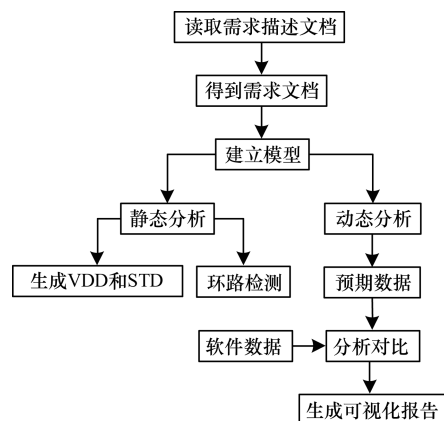


图 1 基于 CSRA 的需求分析方法流程

为建立可执行模型, 需要从需求描述文档中提取需求条目。然而需求条目是人工书写的, 难免出现语法错误和逻辑错误, 因此, 在做进一步处理之前, 需要对需求文档进行语法错误检查。CSRA 工具通过语法分析器解析需求, 根据生成的抽象语法树建立可执行模型。

CSRA 工具在建立可执行模型之后, 对其进行静态分析和动态分析。由于控制软件中的变量之间存在复杂的逻辑关系, 因此需求分析人员难以单纯地通过需求文档分析得到变量之间的关系。CSRA 工具使用数据流分析方法生成变量影响关系图(Variable Dependency Diagram, VDD)和状态迁移图(State Transition Diagram, STD), 并进行环路检测。通过 VDD 和 STD 可以高效地分析变量之间的关系, 发现需求中存在的逻辑错误。CSRA 工具对可执行模型进行动态分析, 将列车运行时的预期输出数据与控制软件运行数据进行对比, 发现预期输出数据和软件运行数据的不一致, 并生成可视化分析报告, 帮助需求分析人员快速定位到具体不一致的需求, 准确地分析出不一致产生的原因。

2 需求文档与可执行模型的转换

控制软件的需求描述文档除了包括需求条目外, 还包括中英文注释、图表等, 因此, 需要从需求描述文档中抽取需求条目, 得到需求文档(只含有需求条目)。需求条目是由规范描述语言(Specification Description Language, SDL)书写的。在建立可执行模型之前, 需要对需求条目进行语法检查。本文利用 ANTLR(Another Tool for Language Recognition)工具^[11]进行语法检查, 构造语法分析器, 生成语法抽象树, 之后根据语法抽象树得到可执行模型。

2.1 需求描述文档处理

本文针对列车车载控制软件设计 SDL 语言。

SDL 是一种类似于 Python 语言并包含特殊机制的需求建模语言,和 Python 语言具有相同的缩进规则、控制语句、表达式和数学运算^[12]。列车车载控制软件具有周期控制特性,但在 Python 语言中无法表示周期性的概念。SDL 引用表示周期的规则,其中每个变量后的 (k) 表示当前周期的值,而 $(k-1)$ 表示上个周期或之前的值。

列车车载控制软件的需求由轨道交通特征决定,需要书写成图 2 中的形式。其中,需求条目号是需求描述的唯一标志;中英文注释是对需求描述功能的文字描述;功能设计是使用 SDL 语言对需求进行书写。图中功能设计中第 1 行代码在 Python 语言是一个函数定义,在这里的含义是“获取变量 *TrainControlValid* 在第 k 个周期的值”。需求描述的需求条目号和功能设计称为需求条目,从需求描述文档中抽取出需求条目,组成需求文档。

Train0003	需求条目号
TrainControlValid, Train消息有效标志, 如果超过Constant个周期仍未收到新的Train消息, 则设置为False. TrainControlValid stands for the effectiveness of Trainmessage. If there is no updating Trainmessage past the Constant, TrainControlValid is set as False.	中英文注释
1.def TrainControlValid(k): 2.if (RequestReady(k)): 3.TrainControlValid = True 4.elif (Timer(k-1) < Constant): 5.Timer = Timer(k-1) + 1 6.else: 7.TrainControlValid = False 8.return TrainControlValid	功能设计

图 2 需求描述示例

2.2 可执行模型的建立

ANTLR 是一个基于 Java 开发的语言识别工具^[11],能够识别自定义语言,常用于构建语言的词法分析器和语法分析器。

本文定义 SDL 语言的词法记号和语法规则,使用 ANTLR 自动生成 SDL 的词法分析器和语法分析器,并利用词法分析器和语法分析器对需求条目进行语法检查。其中,词法分析器识别语言中的关键字和严格定义的语法结构;语法分析器分析需求条目的代码是否符合语法规则并生成抽象语法树。

通过遍历抽象语法树建立可执行模型。对不同层次的代码建立不同的模型,用以处理 SDL 语言不同类型的代码,包含表达式模型、简单模型、选择模型、循环模型。

模型层次结构如图 3 所示。其中,表达式模型由代码中的表达式类型的语句生成,用于处理语言

中的表达式;简单模型处理代码段的简单语句,从语句所包含的express式的层次进行执行,是可执行模型的最后一层;选择模型处理代码段中的选择语句,在模拟执行时,判断选择表达式的真假,如果为真,则执行真分支,如果为假,则执行假分支;循环模型处理代码段中的循环语句,在模拟执行时,判断循环条件的真假,如果为真,则进入循环体,如果为假,则跳出。语句模型由表达式模型组成,常用的语句模型有:简单模型,选择模型,循环模型;程序块模型由语句模型组成,每个需求条目建立一个程序块模型;可执行模型由若干个程序块模型组成。

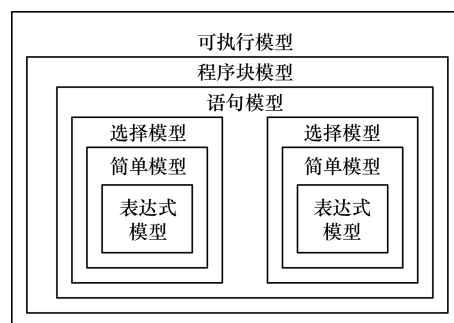


图 3 模型层次结构

由图 2 所示需求条目可以构建一个程序块模型。程序块模块的功能是求 *TrainControlValid(k)* 的值,该模型由一个选择模型和一个简单模型组成,其中选择模型用于处理第 2 行 ~ 第 7 行代码,简单模型用于处理第 8 行代码。在选择模型中,包含处理选择表达式 (*RequestReady(k)*) 以及真分支、假分支的函数。处理选择表达式的函数计算出选择表达式的真假,如果为真,选择处理第 3 行代码的真分支的函数,如果为假,选择处理第 4 行 ~ 第 7 行代码的假分支的函数,处理假分支的函数包含处理第 4 行 ~ 第 7 行代码的选择模型。

3 可执行模型分析

需求分析人员对需求描述文档进行分析,需要确保正确地反映预期需求,对需求条目进行检查,保证需求无逻辑错误,同时需要分析软件运行输出数据的正确性。本文提出的需求分析方法,运用数据流分析方法生成 VDD 和 STD,检查是否存在环路,以此来保证需求描述文档无逻辑错误,模拟执行可执行模型,计算出控制软件的预期输出数据,产生可视化分析报告,用于判断控制软件运行输出数据的正确性。

3.1 静态分析

数据流分析收集程序的语义信息,确定变量的

定义和使用, 可以不必实际运行程序就能够得到程序运行时的行为。通过扫描代码, 可以提取出各模块之间的数据来往关系, 检测可能存在的模块结构缺陷、异常数据流等情况。本文使用数据流分析方法, 需要寻找到程序的控制结构, 并利用控制流图描述该控制结构, 识别程序中的基本块, 用必经点方法来识别程序中的环^[13]。

VDD 描述的是可执行模型中变量之间的依赖关系, 如果数据从变量 A 流向变量 B , 则称变量 A 影响变量 B 。STD 描述的是可执行模型中某个变量不同状态之间的转换以及状态转换所需要满足的条件。VDD 和 STD 的示例如图 4 和图 5 所示。

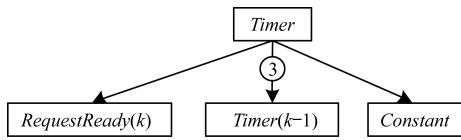


图 4 变量 $Timer$ 的变量影响关系图

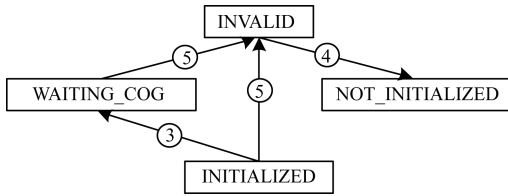


图 5 变量 $OdometerState$ 的状态迁移图

图 4 是关于 $Timer$ 的 VDD, 其中父节点 $Timer$ 值由孩子节点的值决定。在变量影响关系图中存在 2 种影响: 条件影响和赋值影响。如图 4 所示, 处于 if 或者 else 条件中的变量, 如 $RequestReady(k)$, $Timer(k-1)$, $Constant$, 对其父节点 ($Timer$) 影响关系则称为条件影响, 图中数字 3 表示变量 ($Timer$) 受变量 $Timer(k-1)$ 影响需要满足的条件涉及 3 个变量。处于赋值语句中变量 ($Timer(k-1)$) 对父节点 ($Timer$) 的影响关系称为赋值影响。

图 5 是关于变量 $OdometerState$ 的 STD, 其中包括 4 种状态: $INVALID$, $INITIALIZED$, $WAITING_COG$, $NOT_INITIALIZED$, 图中显示变量 $OdometerState$ 状态之间的转换关系, 数字 5 表示变量 $OdometerState$ 在处于 $INITIALIZED$ 时, 需要由 5 个相关变量决定能否迁移下一状态 $INVALID$ 。

环路是指数据流从某一个变量流经过若干个变量流向其自身, 即存在某个变量的变量影响关系图存在环。变量影响关系图是一个有向图, 利用深度优先访问算法即可检测可执行模型中是否存在环路。

一个可能导致环路存在的表达式如下:

$$TrainVaria(k) = TrainTimer + 1$$

$$TrainTimer = TrainVaria(k)$$

由上述表达式可以看出, 变量 $TrainVaria(k)$ 受变

量 $TrainTimer$ 的影响, 变量 $TrainTimer$ 也受变量 $TrainVaria(k)$ 的影响, 这样就在变量 $TrainVaria(k)$ 的影响关系图中形成了一个环路。

本文通过对变量影响关系图进行遍历, 以深度优先访问作为基础算法设计环路检测算法, 检测可执行模型中是否存在环路, 具体见算法 1。其中, $Graph$ 用于存储所有变量; $v.visit$ 表示变量 v 是否已经被访问过; $v.inPath$ 表示变量 v 是否在当前路径上; $PathStack$ 的数据结构类型是栈, 用于记录路径上的变量; $cycle$ 的数据结构类型是栈, 用于存储找到的环; $FindCycle$ 是一个寻找环的函数。

算法 1 环路检测

输入 变量的影响关系图 $Graph$

输出 影响关系图中存在的环 $cycle$

```

1. for v in Graph do
2.   if (! v.visit) then
3.     FindCycle(v)
4.   end if
5. end for

```

下面给出环路检测算法中 $FindCycle$ 函数的实现代码。

```

1. if v.visit && ! v.inPath then
2.   return
3. end if
4. PathStack.push(v)
5. if (v.inPath) then
6.   for v in PathStack do
7.     cycle.Push(v)
8.   end for
9. else
10.  v.visit = true;
11.  v.inPath = true
12.  for v in Edges[v] do
13.    FindCycle(v)
14.  end for
15.  v.inPath = false
16. end if
17. PathStack.pop(v)

```

3.2 动态分析

动态分析通过运行软件得到程序的动态行为信息^[14-15], 本文通过模拟执行可执行模型对程序进行动态分析, 得到预期输出数据, 对比软件运行输出数据与可执行模型的预期输出数据。

可执行模型依次模拟执行程序块模型。CSRA 工具对需求条目的程序段模块根据不同的模型进行不同的模拟执行, 可执行模型模拟执行单个周期的算法见算法 2。其中, 如果模型属于简单模型, 则直接计算模型的表达式; 如果模型属于选择模型, 则判断选择条件的值的真假。若选择条件的值为真, 执行选择模型的真分支。若选择条件的值为假, 执行选择条件的假分支; 如

果模型属于循环模型,则判断循环条件的值的真假。若循环条件的值为真,执行循环体,并且更新循环条件。若循环条件的值为假,循环结束。

算法 2 程序块模型模拟执行

输入 模拟执行的模块 *model*

输出 模块的输出 *ret*

```

1. if model is Simplemodel then
2.   ret = executeExpr( model. expr)
3. else if model is Choicemodel then
4.   if executeExpr( model. condition). value == true then
5.     ret = execute( model. trueBlock)
6.   else
7.     ret = execute( model. falseBlock)
8.   end if
9. else if model is Loopmodel then
10.  while executeExpr( model. condition). value == true do
11.    ret = execute( model. body)
12.    execute( model. iterator)
13.  end while
14. return ret

```

CSRA 工具根据已经建立的可执行模型,以上一周期的输出数据作为下一周期的输入数据,计算每个周期的输出数据。将列车车载控制软件的一组运行输出数据记为 A_{op} ,每组输出数据有 n 个周期,用 $A[i]$ 表示第 i 个周期输出数据。将可执行模型的一组预期输出数据记为 S ,用 $S[i]$ 表示第 i 个周期预期输出数据。初始状态时,读取列车车载控制软件的一组运行输出数据 A_{op} ,以 $A[0]$ 作为可执行模型模拟执行的第一个周期的输入数据 $S[0]$,循环 $(n-1)$ 个周期,在每个周期内依次执行可执行模型中的每个程序块模型。模拟执行所有周期的算法见算法 3。其中, R 表示可执行模型; r 表示程序块模型; $r.in$ 表示程序块模型的输入数据; $r.out$ 表示程序块的输出数据。

算法 3 可执行模型模拟执行

输入 第一周期输入数据 $S[0]$

输出 预期输出 S

```

1. for i = 1 ; i < n ; i ++ do
2.   for r in R do
3.     S[i][r.out] = excute(r. body, i, S[i-1][r.out])
4.   end for
5. end for

```

CSRA 工具将可执行模型计算得到的预期数据与软件运行数据进行对比,是分析软件正确性的主要过程,其读取软件运行数据每周期中的变量值,与可执行模型计算得到的输出数值进行比对,分析对比结果,产生可视化分析报告。

可视化分析报告把需求条目划分为 3 类:语法错误,匹配,不匹配。根据需求文档建立模型时,需求条目会出现常见的语法错误,如缺失冒号、缩进错误等。

CSRA 工具对比每一周期的模型计算预期输出数据和软件运行输出数据,当一条需求条目所有周期的数据一致时,称为需求条目匹配,否则称为不匹配。

4 实验与结果分析

需求分析人员在对控制软件的需求进行分析时,使用 CSRA 工具自动分析出需求条目的语法错误,建立可执行模型,通过分析变量之间的影响关系图和状态迁移图,检测变量使用是否存在错误及变量之间存在的环路。当软件运行数据和可执行模型计算的预期数据不一致时,需求分析人员可以通过分析 VDD 和 STD 检查需求条目是否存在设计错误,快速定位出需求文档的功能错误或者平台实现错误。

在实验中,通过 CSRA 工具对控制软件的需求描述文档进行建模,计算出可执行模型的预期输出数据,然后与软件运行输出数据对比分析,软件运行输出数据包含 1 500 个周期的数据。需求描述文档和软件运行输出数据均由国家可信嵌入式软件工程技术研究中心提供。

通过 CSRA 工具对一个由 455 个需求条目组成的需求文档进行分析,语法错误分析结果如图 6 所示。可以看出,需求文档存在 123 个语法错误,需求条目号为 Train06372 的需求在第 1 行出现语法错误,错误类型为缺少冒号。

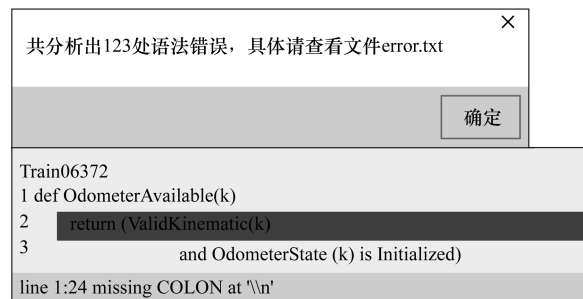


图 6 语法错误分析界面

通过 CSRA 工具在建立可执行模型后进行静态分析,检查需求文档中是否存在环路,如图 7 所示。可以看出,需求文档中存在 2 个环路。

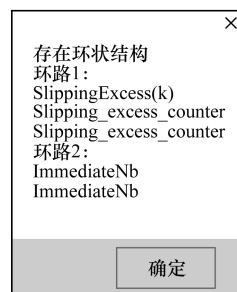


图 7 环路检测界面

需求分析人员人工和通过 CSRA 工具分别对需求条目数量为 455 的需求描述文档进行分析,结果

如表 1 所示。可以看出,通过 CSRA 工具可比人工检查出更多的语法错误和环路,明显减少了使用时间(分析语法错误和环路的总时间)。

表 1 使用 CSRA 和人工分析需求描述文档的实验结果

分析方法	语法错误数目	环路数目	耗时/min
CSRA	123	4	3
人工	99	3	28 800

通过 CSRA 工具对需求描述文档进行动态分析,结果如图 8 所示。可以看出,需求条目匹配率达到 70.8%,123 个需求条目存在语法错误,10 个需求条目对比结果存在不匹配,322 需求条目在 1 500 个周期中预期数据和运行数据相同。1 个需求条目不匹配的可视化报告显示如图 9 所示。可以看出,需求条目 Train0287 在第 34 周期出现数据不相同,第 34 周期软件运行输出数据的值为 160,可执行模型的预期输出结果数据为 150,导致需求条目不匹配。需求分析人员通过进一步分析,得到不匹配的原因如下:1)需求描述文档的描述错误,导致工具建立错误的可执行模型,得出错误的预期输出;2)软件在实现过程中的编程错误,导致软件的运行数据的真实值错误。

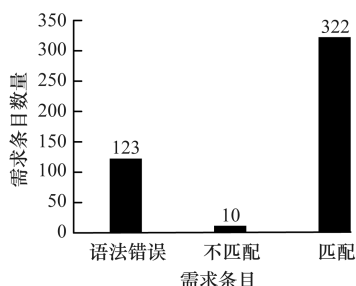


图 8 可视化报告统计结果

Train0287
1 def RadSpeed(k):
2 if DrsValid:
3 return DrsSpeed
4 else:
5 return MAXSPEED
失败周期: 34 分析值: 150 真实值: 160

图 9 需求条目不匹配的可视化报告

需求条目的统计数据可以帮助需求分析人员针对具体需求进行分析。需求分析人员通过可视化分析报告可以诊断出软件潜在的 inconsistency 漏洞。这些检查可以在 3 min 内完成,而人工检查则需要 3 个月。

5 结束语

本文提出面向列车车载控制软件的需求分析方法,开发需求文档自动分析工具 CSRA。目前该工具已被应用于列车车载控制软件的需求分析中,解决了需求分析人员无法分析变量之间的影响关系和状态迁移情况,以及判断软件输出数据是否正确的问题。

需求分析人员根据生成的可视化报告可以找到软件预期输出中的错误数据,进而发现列车车载控制软件的实现错误。由于本文只对需求条目进行了语法检查,而未考虑其安全性检测的问题,因此下一步将对此进行研究。

参考文献

- [1] Friman B, Andreioux T. Automated System Testing of an Automatic Train Protection System[C]//Proceedings of the 12th International Conference on Computers in Railways. Beijing, China: [s. n.], 2010: 71-80.
- [2] Xie Guo, Hei Xinhong, Mochizuki H, et al. Formal Analysis of Automatic Train Protection and Block System for Regional Line Using VDM++[J]. International Journal of Railway, 2012, 5(2): 65-70.
- [3] 王继成, 高 珍. 软件需求分析的研究[J]. 计算机工程与设计, 2002, 23(8): 18-21.
- [4] Wang Zheng, Li Jianwen, Zhao Yongxin, et al. SPARDL: A Requirement Modeling Language for Periodic Control System[M]//Margaria T, Steffen B. Leveraging Applications of Formal Methods, Verification, and Validation. Berlin, Germany: Springer, 2010: 594-608.
- [5] Wang Haifeng, Gao Chunhai, Liu Shuo. Model-based Software Development for Automatic Train Protection System[C]//Proceedings of 2009 Asia-Pacific Conference on Computational Intelligence and Industrial Applications. Washington D. C., USA: IEEE Press, 2009: 463-466.
- [6] Xiong Xijiao, Liu Jing, Zhang Miaomiao, et al. Modeling and Verification of an Automatic Train Protection System[C]//Proceedings of IEEE Computer Software and Applications Conference Workshops. Washington D. C., USA: IEEE Press, 2010: 226-231.
- [7] Daniel A, Hyunsook D, Lee Seok-Won. SQ²E: An Approach to Requirements Validation with Scenario Question [J]. Proceedings of the Royal Society of London A: Mathematical Physical and Engineering Sciences, 2010, 161(906): 33-42.
- [8] Chand M G, Reddy K N, Rao A A, et al. An Approach to Requirements Elicitation and Analysis Using Goal[C]//Proceedings of the 2nd International Conference on Software Technology and Engineering. Washington D. C., USA: IEEE Press, 2010: 218-221.
- [9] Kaiya H, Saeki M. Ontology Based Requirements Analysis [J]. Technical Report of IEICE. SS, 2005, 105(25): 223-230.
- [10] Backes J, Cofer D, Miller S, et al. Requirements Analysis of a Quad-redundant Flight Control System [M]//Havelund K, Holzmann G, Joshi R. NASA Formal Methods. Berlin, Germany: Springer, 2015: 82-96.
- [11] Parr T. The Definitive ANTLR Reference: Building Domain-specific Languages [M]. [S. l.]: Pragmatic Bookshelf, 2007.
- [12] 包丹珠. 轨道交通系统需求分析与一致性测试[D]. 上海: 华东师范大学, 2016.
- [13] 李兰英, 张 滇, 崔林海, 等. 一种使用控制块消除流图中回边的算法[J]. 计算机工程, 2008, 34(20): 74-76.
- [14] 吴 昊, 毋国庆. 程序的动态完整性: 模型和方法[J]. 计算机研究与发展, 2012, 49(9): 1874-1882.
- [15] 黄荷洁, 康 维, 舒 辉. 基于动态数据流分析的虚拟机保护破解技术[J]. 计算机工程, 2014, 40(9): 59-65.

编辑 金胡考