面向系统能力的形式化分析和测试方法

陈 平 梁启明 孙 伟
(工业和信息化部电子第五研究所 广州 510610)

摘 要 国内软件业界实施系统测试时,大部分采用对系统规格说明描述的功能点进行逐一测试的方法,很少从系统能力的角度进行测试,难以充分说明系统软件产品满足系统能力需求的要求。同时,系统规格说明使用自然语言进行描述,存在语义不准确的现象,直接影响系统测试的质量。针对上述问题,提出了一种面向系统能力的形式化分析和测试方法。通过该方法,测试工程师可得到语义清晰的系统能力需求描述,并实施面向系统能力的系统测试,有效地提高系统测试的充分性和准确性,从而提高系统软件质量。

关键词 系统测试,系统能力测试,形式化方法,软件需求分析
中图法分类号 TP306+.2 **文献标识码** A

System Capability-oriented Approach for Formalized Software Requirements Analysing and Testing

CHEN Ping LIANG Qi-ming SUN Wei
(China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou 510610, China)

Abstract In domestic software industry, system testing is based on only the function items in system requirements specifications, rarely based on the system capabilities. Consequently, the result of system testing can not fully testifies that the SUT(System Under Rest) meets the system requirements. Besides, there is ambiguity in system requirements specifications which is caused by natural language descriptions. These problems directly decrease the efficiency of system testing. In order to deal with these problems, a system capability-oriented approach for formalized software requirements analysing and testing was proposed. With this approach, software testing engineers can get clear descriptions about requirements of system capabilities, and conduct system testing for system capabilities. It will efficiently improve the adequacy and accuracy of system testing, which consequently increase the quality of SUT.

Keywords System testing, System capabilities testing, Formalized approach, Softwre requirements analysis

1 引言

随着信息产业的高速发展,软件行业进入了大系统时代。现代商用系统软件规模大,包含的软件构件数量繁多,多个软件构件必须通过协作和交互才能完成指定的系统功能。面对如此复杂的软件系统,既要保障系统软件质量,又要确保开发出的软件满足用户需求,已成为软件开发项目的核心目标。现今软件工程业界达到这一目标的普遍方法是软件测试,而其中关键一环就是系统测试。但软件业界进行系统测试时,普遍存在两个问题:1)测试工程师只按照系统规格书说明或软件需求规格说明对系统软件需求进行逐条测试验证,很少考虑面向系统软件能力或用户使用需求来进行贯穿式的系统测试,于是对于软件是否满足用户需求、系统能力是否得到正确实现,这样的系统测试未能给出有说服力的结论;2)测试工程师依据系统软件需求提取测试需求时,由于使用自然语言进行描述,导致后续系统测试的执行容易出现理解歧义以及测试不充分的情况。

针对以上两个业界普遍存在的问题,本文提出一种面向系统能力的形式化分析和测试方法。该方法使用形式化语言描述系统能力,并依据系统能力的形式化表述进行贯穿式的系统测试。

2 国内系统测试现状

系统测试过程的首要工作是进行软件需求分析,并依据分析结果得到测试需求,进而设计测试用例。国内商用现货软件("COTS")系统需求文档主要依据 GB/T 8567-2006《计算机软件文档编制规范》[1]和 GJB 438B-2009《军用软件开发文档通用要求》[2]的要求来编写,并形成系统规格说明。国标和国军标要求系统规格说明文档中逐条描述系统软件的功能点,包括输入、处理过程和输出,因此国内软件业界普遍将系统功能需求描述章节按如下结构进行描述。

"3.4.1 系统功能 1
输入:……
处理:……
输出:……
3.4.2 系统功能 2
输入:……
处理:……
输出:……
…
3.4.x 系统功能 x
输入:……

本文受核高基国家科技重大专项(JAZ1472010)资助。
陈 平(1984-),男,工程师,主要研究方向为软件测试,E-mail:bolama@163.com;梁启明 男,硕士,工程师,主要研究方向为软件测试、需求形式化分析,E-mail:lqming@mail2.sysu.edu.cn(通信作者)。

处理：……

输出：……”

基于上述的系统软件需求组织结构，软件测试工程师逐条对系统功能点使用等价类划分、场景法[3]、边界值分析法、UML 顺序图分析法[4]、因果图法进行测试用例设计，并依据测试用例执行系统测试。

但是，现今规模大、协作软件繁多、复杂度高的系统软件令上述传统方法面临一个问题，系统用户使用需求或系统能力可能由多个系统功能点逻辑组合完成，这种系统功能逐条测试的方法将系统能力逐块隔断，使得无法对整个系统软件使用需求进行覆盖测试，从而无法达到保证系统软件满足用户使用需求的质量目标。

3　国内外软件需求形式化分析方法简介

主流软件需求形式化分析方法包括 Z 语言[5]、VDM[6]、LOTOS[7]、B 方法[8]等形式化方法，以命题逻辑、一阶逻辑、高阶逻辑、代数、状态机等逻辑语言描述软件需求，使用逻辑推理的方法验证软件需求模型的一致性和完备性。Z 语言使用一阶谓词和集合论描述软件需求，利用模式和模式演算对软件结构和行为进行描述，其中模式包括状态模式和操作模式，状态模式描述软件的结构，而操作模式描述软件的行为。VDM 由 IBM 公司维也纳实验室提出，该形式化方法使用一阶谓词以及预先建立的抽象数据类型对软件结构化功能进行描述，该方法简明清晰地描述软件行为和功能点，但缺乏对复杂软件行为时序的描述方法，并且使用了艰深的数学符号和运算，限制了大部分软件从业人员对其进行使用[9]。LOTOS 是国际化标准组织提出的一种软件需求形式化描述技术，该技术的特点是以事件的时序关系对系统行为进行形式化描述，包括两个部分：1）进程代数 CCS；2）抽象数据类型描述语言 ACT ONE。LOTOS 是适应协议工程、分布处理和并行处理要求而产生的语言，能够有效地验证协议的正确性、测试协议的一致性[10]。B 方法基于集合、关系和谓词逻辑等数学理论书写规范，通过伪程序的形式来描述软件需求规格说明，主要通过不变式和证明义务来保证规范正确性。

上文概述的 Z 语言、VDM、LOTOS、B 方法等形式化方法与 GB/T 8567-2006《计算机软件文档编制规范》以及 GJB 438-2009《军用软件开发文档通用要求》的系统规格说明描述内容存在较大差异，难以应用于国内的大规模系统软件测试工作。因此，本文提出的系统需求形式化分析方法需建立系统规格说明与流行形式化方法的应用联系。由于 LOTOS 方法是国际化标准组织提出的一种软件需求形式化描述通用技术，其状态迁移属性与系统功能逻辑组合存在相似性，因此将本文提出的面向系统能力的形式化建模方法与 LOTOS 语言建立连接，使得建模结果更通用，而且可使用 CADP[11]等工具进行形式化验证和测试用例生成。

4　面向系统能力的形式化分析方法

4.1　系统能力及系统能力路径定义

定义 1（系统能力）　为了实现用户使用需求，系统内各软件构件功能按照一定时序运行，形成系统能力。在功能方面，整个系统软件用户的使用需求就是所有系统能力的集合。系统能力将系统软件划分成单个独立但相互联系的功能区域。

定义 2（系统能力路径）　每一个系统能力包含一条或若干条基本运行路径，这些运行路径只与特定的软件功能、运行条件和运行时序相关联。在系统软件结构方面，系统软件功能实现就是所有系统能力路径的集合。

4.2　系统能力路径的形式化定义

定义 3（定义系统软件）　S 由软件构件 $\{C_1, C_2, \cdots, C_i\}$ 组成。

定义 4　定义每个软件构件包含的软件功能集合。

$$\begin{cases} C_1: F_1^{C_1}, F_2^{C_1}, F_3^{C_1}, \cdots, F_{|C_1|}^{C_1} \\ C_2: F_1^{C_2}, F_2^{C_2}, F_3^{C_2}, \cdots, F_{|C_2|}^{C_2} \\ \quad \cdots \\ C_i: F_1^{C_i}, F_2^{C_i}, F_3^{C_i}, \cdots, F_{|C_i|}^{C_i} \end{cases}$$

其中，$|C_i|$ 为软件构件 C_i 的软件需求规格说明中定义的功能个数。

定义 5　定义系统能力集合为 $\{A_1, A_2, \cdots, A_i\}$。

每一个系统能力 A_i 由若干条系统能力执行路径以及该执行路径的前提条件组成。

系统能力 A_i 的执行路径集合为：$\{Path_1^{A_i}, Path_2^{A_i}, \cdots, Path_{|A_i|}^{A_i}\}$，其中 $|A_i|$ 为系统能力 A_i 的能力路径数目。

系统能力 A_i 的各条系统能力执行路径的前提条件集合为：

$\{Condition_{Path_1^{A_i}}, Condition_{Path_2^{A_i}}, Condition_{Path_3^{A_i}}, \cdots, Condition_{Path_{|A_i|}^{A_i}}\}$

令第 n 条系统能力路径为：

$$P_n^{A_i} = \exists Condition_{Path_n^{A_i}} Path_n^{A_i} \tag{1}$$

系统能力 A_i 可表示为 A_i 的全部系统能力执行路径以及前提条件的并集，即：

$$A_i = \exists Condition_{Path_1^{A_i}} Path_1^{A_i} \cup \exists Condition_{Path_2^{A_i}} Path_2^{A_i} \cup \exists Condition_{Path_3^{A_i}} Path_3^{A_i} \cup \cdots \cup \exists Condition_{Path_{|A_i|}^{A_i}} Path_{|A_i|}^{A_i} \tag{2}$$

系统能力 A_i 的其中一条系统能力执行路径 $Path_n^{A_i}$ 由系统内各软件构件功能按时序完成实现，由定义 4 可得：

$$Path_n^{A_i} = \langle F_a^{C_1} \rightarrow F_b^{C_2} \rightarrow F_c^{C_3} \rightarrow \cdots \rightarrow F_m^{C_j} \rangle \tag{3}$$

4.3　系统能力路径细化

使用系统能力路径对系统能力进行形式化分析的目的在于更准确和更全面地生成系统测试需求和测试用例，提高系统测试对系统能力和用户使用需求的测试准确性和充分性。依据式(2)和式(3)，对于系统能力 A_i 实现该系统能力的某一条运行路径可形式化描述为：$\exists Condition_{Path_n^{A_i}} \langle F_a^{C_1} \rightarrow F_b^{C_2} \rightarrow F_c^{C_3} \rightarrow \cdots \rightarrow F_m^{C_j} \rangle$。对软件构件功能 $F_m^{C_j}$ 继续向下进行细化，可得到系统能力路径测试需求的形式化描述。

定义 6　定义 Act_x 为 x 所需要的外部操作和内部处理。

定义 7　定义 $Para_x$ 为 x 所需要的输入参数。

首先对系统能力运行路径的前提条件 $Condition_{Path_n^{A_i}}$ 进行细化，依据定义 6 和定义 7 可得：

$$Condition_{Path_n^{A_i}} = \exists Para_{Condition_{Path_n^{A_i}}} : Act_{Condition_{Pa}} \tag{4}$$

依据式(3)和定义 6 可将系统能力运行路径 $Path_n^{A_i}$ 细化为：

$$Path_n^{A_i} = \langle Act_{F_a^{C_1}} \rightarrow Act_{F_b^{C_2}} \rightarrow Act_{F_c^{C_3}} \rightarrow \cdots \rightarrow Act_{F_m^{C_j}} \rangle \tag{5}$$

根据式(4)和式(5)即可得到系统能力 A_i 的全部运行路

径 $\exists Condition_{Path_n^{A_i}}\langle F_a^{C_1}\rightarrow F_b^{C_2}\rightarrow F_c^{C_3}\rightarrow\cdots\rightarrow F_m^{C_j}\rangle$ 的完整细化操作序列，得到系统能力路径的测试需求和测试用例。

4.4 LOTOS 语言形式化描述

使用 LOTOS 语言对上述面向系统能力的功能需求形式化分析结果进行描述。

定义每一个系统能力 A_i 为规范体(Specification)，系统能力包含 $|A_i|$ 条系统能力路径。系统实现某一种系统能力时，有且只有一条系统能力路径 $P_{x^i}^{A_i}$ 运行，于是 $P_{x^i}^{A_i}$ 可定义为进程(Process)。按照 LOTOS 语法，依据式(2)、式(4)和式(5)可得：

Specification $A_i[Para_{Condition_{Path_1^{A_i}}}, Para_{Condition_{Path_2^{A_i}}}, Para_{Condition_{Path_3^{A_i}}},\cdots, Para_{Condition_{Path_{|A_i|}^{A_i}}}]$:exit:=behaviour

$P_1^{A_i}[]P_2^{A_i}[]P_3^{A_i}[]\cdots[]P_{|A_i|}^{A_i}$

where

Process $P_1^{A_i}$

$[Para_{Condition_{Path_1^{A_i}}}]$:exit:=

…

endProc

Process $P_2^{A_i}$

$[Para_{Condition_{Path_2^{A_i}}}]$:exit:=

…

endProc

…

Process $P_{|A_i|}^{A_i}$

$[Para_{Condition_{Path_{|A_i|}^{A_i}}}]$:exit:=

…

endProc

endSpec

由式(4)和式(5)可以继续得到任意一条系统能力路径 $Path_{x^i}^{A_i}$ 的 LOTOS 形式化描述：

Process $Path_{x^i}^{A_i}[Para_{Condition_{Path_{x^i}^{A_i}}}]$:exit:=

$i_{Condition_{Path_{x^i}^{A_i}}}\gg i_{F_a^{C_1}}\rightarrow i_{F_b^{C_2}}\rightarrow i_{F_c^{C_3}}\rightarrow\cdots\rightarrow i_{F_m^{C_j}}$;exit

endProc

其中，软件构件实现某一个功能的操作步骤 $i_{F_a^{C_1}}$，$i_{F_b^{C_2}}$，…，$i_{F_m^{C_j}}$ 可以从软件构件的测试需求或用例复用得到。

5 面向系统能力的测试方法及应用实例

5.1 面向系统能力的测试方法

面向系统能力的测试方法核心是为每一条系统能力路径生成系统测试需求或测试用例。生成的测试需求或测试用例兼顾各软件构件功能与内部处理逻辑和时序，实现了系统能力测试覆盖。本文提出的面向系统能力的形式化分析方法使用 LOTOS 形式化语言描述，最终利用 CADP[11] 的 OPEN/CAESAR 和 TGV[12] 工具进行形式化验证和生成测试需求或测试用例。

面向系统能力的形式化分析和测试方法应用步骤如下：

步骤 1 依据系统规格说明抽象软件系统能力；

步骤 2 依据系统规格说明和各软件构件需求规格说明，定义每一条系统能力路径；

步骤 3 分析每一条能力路径的运行路径和前提条件；

步骤 4 对每一条系统能力运行路径和前提条件进行细化，得到形式化分析结果；

步骤 5 依据形式化分析结果，将系统能力路径使用 LOTOS 形式化语言进行描述；

步骤 6 直接利用 LOTOS 描述结果得到测试需求或设计测试用例，或者利用 CADP 的 OPEN/CAESAR 和 TGV 工具生成测试需求和测试用例。

5.2 银行 ATM 系统软件需求形式化分析和测试应用实例

现代银行 ATM 系统软件基础结构包括 ATM 终端软件(ATMC)、ATM 前置处理软件(ATMP)和业务主机软件(ServiceHost)。

银行 ATM 系统软件 S={ATMC,ATMP,ServiceHost}。

ATMC 的基本业务功能包括持卡人验证、查询、存款、取款、转账、修改密码和提示信息显示，所以 ATM 系统软件中 ATMC 软件构件 C_{ATMC} 需求规格说明中的功能项为：

$$C_{ATMC}=\{F_{\text{持卡人验证}}^{C_{ATMC}},F_{\text{查询}}^{C_{ATMC}},F_{\text{存款}}^{C_{ATMC}},F_{\text{取款}}^{C_{ATMC}},F_{\text{转账}}^{C_{ATMC}},F_{\text{修改密码}}^{C_{ATMC}},F_{\text{提示信息显示}}^{C_{ATMC}}\}\tag{6}$$

ATMP 将 ATMC 的用户交易操作转换为符合标准的请求报文，并转发至业务主机软件，接收业务主机软件回送的交易业务响应报文，并转发至 ATMC。所以 ATM 系统软件中 ATMP 软件构件 C_{ATMC} 需求规格说明中的功能项为：

$$C_{ATMP}=\{F_{\text{验证信息转发}}^{C_{ATMP}},F_{\text{查询交易转发}}^{C_{ATMP}},F_{\text{存款交易转发}}^{C_{ATMP}},F_{\text{取款交易转发}}^{C_{ATMP}},F_{\text{转账交易转发}}^{C_{ATMP}},F_{\text{修改密码转发}}^{C_{ATMP}},F_{\text{交易响应转发}}^{C_{ATMP}}\}\tag{7}$$

ServiceHost 接收到 ATMP 转发的交易请求后，进行业务交易处理，并返回交易处理响应。所以 ATM 系统软件中 ServiceHost 软件构件 C_{SH} 需求规格说明中的功能项为：

$$C_{SH}=\{F_{\text{持卡人验证处理}}^{C_{SH}},F_{\text{查询交易处理}}^{C_{SH}},F_{\text{存款交易处理}}^{C_{SH}},F_{\text{取款交易处理}}^{C_{SH}},F_{\text{转账交易处理}}^{C_{SH}},F_{\text{修改密码处理}}^{C_{SH}}\}\tag{8}$$

ATM 系统软件的常见用户使用需求为查询余额、存款、取款和转账，于是可以定义 ATM 系统软件的系统能力为：

$\{A_{\text{查询余额}},A_{\text{存款}},A_{\text{取款}},A_{\text{转账}}\}$

以系统能力 $A_{\text{存款}}$ 为例，系统能力路径示意图如图 1 所示。

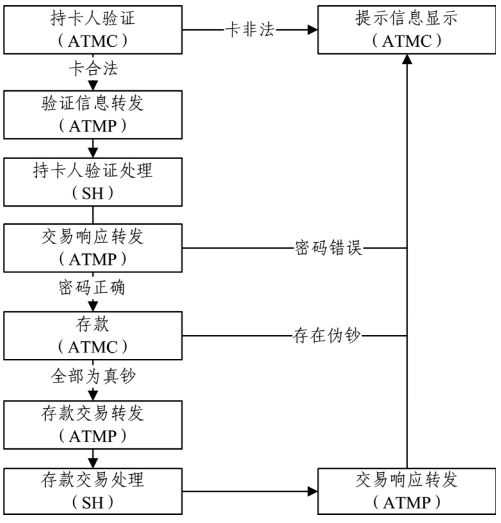

图 1 ATM 系统软件存款系统能力路径示意图

由图 1 得到系统能力 $A_{\text{存款}}$ 的 4 条系统能力路径为：

$P_1^{A_{\text{存款}}}=\exists$ 卡合法 and 密码正确 and 全部为真钞 $\langle F_{\text{持卡人验证}}^{C_{ATMC}}\rightarrow F_{\text{验证信息转发}}^{C_{ATMP}}\rightarrow F_{\text{持卡人验证处理}}^{C_{SH}}\rightarrow F_{\text{交易响应转发}}^{C_{ATMP}}\rightarrow F_{\text{存款}}^{C_{ATMC}}\rightarrow F_{\text{存款交易转发}}^{C_{ATMP}}\rightarrow F_{\text{存款交易处理}}^{C_{SH}}\rightarrow F_{\text{交易响应转发}}^{C_{ATMP}}\rightarrow F_{\text{提示信息显示}}^{C_{ATMC}}\rangle$

$P_2^{A_{存款}}$ = ∃ 卡非法〈$F^{C_{ATMC}}_{持卡人验证} \rightarrow F^{C_{ATMC}}_{提示信息显示}$〉

$P_3^{A_{存款}}$ = ∃ 卡合法 and 密码错误〈$F^{C_{ATMC}}_{持卡人验证} \rightarrow F^{C_{ATMP}}_{验证信息转发} \rightarrow F^{C_{SH}}_{持卡人验证处理} \rightarrow F^{C_{ATMP}}_{交易响应转发} \rightarrow F^{C_{ATMC}}_{提示信息显示}$〉

$P_2^{A_{存款}}$ = ∃ 卡合法 and 密码正确 and 存在伪钞〈$F^{C_{ATMC}}_{持卡人验证} \rightarrow F^{C_{ATMP}}_{验证信息转发} \rightarrow F^{C_{SH}}_{持卡人验证处理} \rightarrow F^{C_{ATMP}}_{交易响应转发} \rightarrow F^{C_{ATMC}}_{存款} \rightarrow F^{C_{ATMC}}_{提示信息显示}$〉

于是系统能力 $A_{存款} = P_1^{A_{存款}} \cup P_2^{A_{存款}} \cup P_3^{A_{存款}} \cup P_4^{A_{存款}}$。

按照式(4)和式(5)对 4 条系统能力路径继续进行细化。由于系统测试是黑盒测试，细化结果中只给出人工的外部输入参数和操作步骤，软件构件的内部处理使用 i 来表示。

$P_1^{A_{存款}}$ = ∃ 合法的卡片 and 正确的密码 and 真钞〈插入卡片 → 输入密码 → $i^{C_{ATMC}}_{持卡人验证}$ → $i^{C_{ATMP}}_{验证信息转发}$ → $i^{C_{SH}}_{持卡人验证处理}$ → $i^{C_{ATMP}}_{交易响应转发}$ → 装入钞票 → $i^{C_{ATMC}}_{存款}$ → $i^{C_{ATMP}}_{存款交易转发}$ → $i^{C_{SH}}_{存款交易处理}$ → $i^{C_{ATMP}}_{交易响应转发}$ → $i^{C_{ATMC}}_{提示信息显示}$ → 完成存款〉

$P_2^{A_{存款}}$ = ∃ 非法的卡片〈插入卡片→输入密码→$i^{C_{ATMC}}_{持卡人验证}$→$i^{C_{ATMC}}_{提示信息显示}$→退出存款〉

$P_3^{A_{存款}}$ = ∃ 合法的卡片 and 错误的密码〈插入卡片→输入密码〉→ $i^{C_{ATMC}}_{持卡人验证}$ → $i^{C_{ATMP}}_{验证信息转发}$ → $i^{C_{SH}}_{持卡人验证处理}$ → $i^{C_{ATMP}}_{交易响应转发}$ → $i^{C_{ATMC}}_{提示信息显示}$ →退出存款〉

$P_4^{A_{存款}}$ = ∃ 合法的卡片 and 正确的密码 and 伪钞数量大于 1〈插入卡片 → 输入密码 → $i^{C_{ATMC}}_{持卡人验证}$ → $i^{C_{ATMP}}_{验证信息转发}$ → $i^{C_{SH}}_{持卡人验证处理}$ → $i^{C_{ATMP}}_{交易响应转发}$ →装入钞票→ $i^{C_{ATMC}}_{存款}$ → $i^{C_{ATMC}}_{提示信息显示}$ →退出存款〉

最后使用 LOTOS 形式化语言对 $A_{存款} = P_1^{A_{存款}} \cup P_2^{A_{存款}} \cup P_3^{A_{存款}} \cup P_4^{A_{存款}}$ 进行描述：

```
Specification A存款[In_Card, In_Password, In_Cash]: exit:=
behaviour
  P1^A存款 [] P2^A存款 [] P3^A存款 [] P4^A存款
where
  Process P1^A存款[In_Card, In_Password, In_Cash]: exit:=
  In_Card ?Card:Card_Set [Card∈LegalID_Set];
  In_Password !Correct_Password;
  In_Cash ?Cash:Cash_Set [Cash∈Real];
  插入卡片[In_Card];输入密码[In_Password];i;装入钞票[In_Cash];i;完成存款;exit
  endProc
  Process P2^A存款[In_Card]:exit:=
  In_Card ?Card:Card_Set [Card∈LegalID_Set];
  插入卡片[In_Card];i;退出存款;exit
  endProc
  Process P3^A存款[In_Card, In_Password]:exit:=
  In_Card ?Card:Card_Set [Card∈LegalID_Set];
  In_Password ?Password:6 位数字字符串 [Password != Correct_Password];
  插入卡片[In_Card];输入密码[In_Password];i;退出存款;exit
  endProc
  Process P4^A存款[In_Card, In_Password, In_Cash]:exit:=
  In_Card ?Card:Card_Set [Card∈LegalID_Set];
  In_Password !Correct_Password;
  In_Cash ?Cash:Cash_Set [Cash∈Fake];
  插入卡片[In_Card];输入密码[In_Password];i;装入钞票[In_Cash];i;退出存款;exit
  endProc
endSpec
```

6　与其他方法的比较

本节从 3 个方面将本文提出的面向系统能力的形式化分析和测试方法与现有方法进行比较：1)系统软件需求形式化分析的实用性；2)对系统软件使用需求覆盖程度；3)测试用例自动化生成的程度。

6.1　系统软件需求形式化分析的实用性

国内软件业界系统规格说明和软件需求规格说明大部分遵循 GB/T 8567-2006《计算机软件文档编制规范》和 GJB 438B-2009《军用软件开发文档通用要求》，现有的软件需求分析方法(UML、Z 语言、VDM、LOTOS、B 方法，以及执行路径分析方法(Execution Path Analysis)[13])与本文方法的比较如表 1 所列。

表 1　实用性比较

	与 GB/T 8567-2006 和 GJB 438B-2009 的符合性	形式化描述程度
UML	符合性强。国内软件业界软件需求规格说明和设计说明广泛应用 UML 语言。	形式化描述程度差，多使用自然语言表达。
Z 语言	符合性差。使用一阶谓词和集合论描述软件需求，与国标和国军标差距较大。	形式化描述程度高。
VDM	符合性差。使用一阶谓词以及预先建立的抽象数据类型对软件结构化功能进行描述，与国标和国军标差距较大。	形式化描述程度高。
LOTOS	符合性一般。使用进程代数和抽象数据类型描述语言对事件的时序关系对系统行为进行形式化描述。时序关系容易从软件设计说明中提取，但未能很好地与系统软件功能点和软件功能点进行关联。	形式化描述程度高。
B 方法	符合性差，基于集合、关系和谓词逻辑等数学理论书写规范，通过伪程序的形式来描述软件需求规格说明，与国标和国军标差距较大。	形式化描述程度高。
Execution Path Analysis	符合性差。对软件进行代码级的关键路径分析，与国标和国军标差距很大。	未使用形式化描述。
本文方法	符合性较好。可以从系统规格说明和软件需求规格说明中提取软件功能和抽象出系统能力，并能从软件设计说明中总结抽象出系统能力路径。	形式化描述程度较高，并结合 LOTOS 形式化语言。

6.2　与其他系统测试分析方法的比较

本文方法与场景法、基于 UML 顺序图分析方法、因果图分析方法[14]进行的比较如表 2 和表 3 所列。本文方法重点考虑被测系统软件中各软件构件在系统能力实现过程中的协作性和交互性，突出系统能力处理逻辑的运行条件，使得生成的测试需求或测试用例能更充分地覆盖系统能力的各种实现逻辑和情况，确保开发出来的系统软件符合用户使用需求，且能处理各种异常情况。同时，本文提出的测试方法使用 LOTOS 形式化语言进行描述，可利用 OPEN/CAESAR 和 TGV 工具进行测试需求或测试用例自动化生成，以及形式化验证。

表 2 系统软件使用需求覆盖程度比较

测试方法	需求覆盖程度
场景法[15]	从系统规格说明和软件需求规格说明中分析提取场景流程图，但场景流程图未能很好地反映系统软件内部各软件构件的协作性和交互性，并且未突出场景的运行条件。
基于 UML 顺序图的测试用例设计方法	对软件设计说明中 UML 顺序图进行测试需求或测试用例设计分析。但顺序图往往描述的是系统规格说明中的某一个功能下的处理逻辑，未能完全描述整个系统使用需求或系统能力。
因果图法系统测试用例设计方法[16]	基于系统状态的转换进行用例设计和生成，未能很好地反映系统软件内部各软件构件的协作性和交互性，并且较难从系统状态关联到各软件构件的具体功能。
本文方法	将系统内部各软件构件功能、功能运行逻辑和运行条件进行关联，反映软件构件功能在系统能力实现过程中的协作性和交互性，突出系统能力处理逻辑的运行条件，使得生成的测试需求或测试用例能更充分地覆盖系统能力的各种实现逻辑和情况，而且能在测试用例生成时复用各软件构件功能测试时的用例步骤，提高测试设计效率。

表 3 测试需求或测试用例自动化生成程度比较

测试方法	测试需求或测试用例自动化生成程度
场景法系统测试用例设计方法	未使用形式化语言描述软件需求，难以进行测试用例自动化生成
基于 UML 顺序图的测试用例设计方法	未使用形式化语言描述软件需求，难以进行测试用例自动化生成
因果图法系统测试用例设计方法	未使用形式化语言描述软件需求，难以进行测试用例自动化生成
本文方法	使用形式化语言描述软件需求，可使用 OPEN/CAESAR 和 TGV 工具进行测试用例的设计和生成

结束语 本文提出了一种面向系统能力的形式化分析和测试方法，并给出应用实例。相较于现行的 Z 语言、VDM、LOTUS、B 方法、UML 等软件需求形式化分析方法，该方法可从符合 GB/T 8567-2006《计算机软件文档编制规范》和 GJB 438B-2009《军用软件开发文档通用要求》要求的系统规格说明和软件需求规格说明中抽象出系统能力，并总结出系统能力运行路径和运行条件的形式化描述结果，在国内软件业界环境下具有较好的国标、国军标符合性和实用性，并解决了系统需求描述不准确、存在二义性的问题；相较于场景法和因果图法等系统测试方法，本文方法将系统软件内部各软件构件功能点、功能运行逻辑和运行条件串联起来，整体地描述了系统使用需求和系统能力，使得系统测试能更充分地验证系统能力的各种实现逻辑和场景。同时，使用 LOTOS 形式化语言描述系统能力，为测试需求和测试用例自动化设计和生成提供了可行性，有助于提高系统测试效率。

在后续工作中，可研发基于 OPEN/CAESAR 和 TGV 等工具的测试用例自动化生成工具，基于本文系统能力需求形式化分析结果，自动生成测试用例（预置条件-输入步骤-预期输出），从而提高面向系统能力测试方法的工程应用效率。

参 考 文 献

[1] GB/T 8567-2006. 计算机软件文档编制规范[S]. 北京：中国标准出版社，2006.

[2] GJB 438B-2009. 军用软件开发文档通用要求[S]. 北京：中国人民解放军总装备部，2009.

[3] 盛晓娟，贾丽娟，姬鹏超. 场景法在系统测试用例设计中的应用[J]. 计算机工程与设计，2016，37(7)：1798-1804.

[4] 李志强，邵培南，朱辉. 基于 UML 顺序图的测试用例生成[J]. 计算机工程，2010，36(22)：58-60.

[5] SPIVEY J M，ABRIAL J R. The Z notation[M]. Hemel Hempstead：Prentice Hall，1992.

[6] JONES C B. Systematic software development using VDM[M]. Englewood Cliffs，NJ：Prentice-Hall，1986.

[7] LOTOS ISO. A formal description technique based on the temporal ordering of observational behaviour[S]. DIS 8807，1987.

[8] LANO K. The B language and method：a guide to practical formal development[M]. Springer Science & Business Media，2012.

[9] 邹盛荣，郑国梁. B 语言和方法与 Z，VDM 的比较[J]. 计算机科学，2002，29(10)：136-138.

[10] 罗铁庚，陈火旺，齐治昌，等. 协议形式化开发环境的规范语言[J]. 软件学报，1997，8(11)：817-823.

[11] GARAVEL H，MATEESCU R，LANG F，et al. CADP 2006：A toolbox for the construction and analysis of distributed processes[C]// International Conference on Computer Aided Verification. Springer Berlin Heidelberg，2007：158-163.

[12] CLAUDE J，THIERRY J. TGV：Theory，principles and algorithms：A tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems[J]. Software Tools for Technology Transfer，2002，7(4)：297-315.

[13] YANG C Q，MILLER B P. Critical path analysis for the execution of parallel and distributed programs[C]// 8th International Conference on Distributed Computing Systems，1988. IEEE，1988：366-373.

[14] 穆建成，辛未，马连川，等. 基于因果图法的 CTCS-3 级列控系统测试案例完备性验证方法[J]. 中国铁道科学，2016，37(1)：124-131.

（上接第 528 页）

[4] XIONG J J，ZHENG E H. Position and attitude tracking control for a quadrotor UAV[J]. ISA Transactions，2014，53：725-731.

[5] 林旭梅，王婵. 四旋翼飞行器的自适应鲁棒滑模控制器设计[J]. 仪器仪表学报，2015，36(7)：1522-1527.

[6] 杨立本，章卫国，黄得刚. 基于 ADRC 姿态解耦的四旋翼飞行器鲁棒轨迹跟踪[J]. 北京航空航天大学学报，2015，41(6)：1026-1033.

[7] 李劲松，杨炼，王乐天. 小型四旋翼无人直升机自适应优化控制[J]. 上海交通大学学报，2015，49(2)：202-207.

[8] LIU H，BAI Y Q，LU G，et al. Robust Tracking Control of a Quadrotor Helicopter[J]. Intell Robot Syst，2014，75：595-608.

[9] 杨立本，章卫国，黄得刚，等. 欠驱动四旋翼飞行器反演模糊自适应控制[J]. 西北工业大学学报，2015，33(3)：495-499.

[10] 贺有智，刘同其. 四旋翼飞行器时延积分反演容错控制[J]. 系统工程与电机技术，2015，37(10)：2341-2346.

[11] 赵元伟，卢京朝. 四旋翼飞行器的建模及基于反步法的控制[J]. 科学技术与工程，2013，34(13)：10425-10430.

[12] LARA D，ROMERO G，SANCHEZ A，et al. Robustness marginfor attitude control of a four rotor minirotorcraft：Case of study[J]. Mechatronics，2010(20)：143-152.

[13] ASHFAQ A，WANG D B. Modeling and backstepping based nonlinear control strategy for a 6DOF quadrotor helicopter[J]. Chinese J of Aeronautics，2008(21)：261-268.

[14] CAI Z，DE QUEIROZ M S，DAWSON D M. A sufficiently smoothprojection operator[J]. IEEE Trans. on Automatic Control，2006，51(1)：135-139.