第 33 卷 第 8 期 计 算 机 工 程 2007 年 4 月
Vol.33 No.8 Computer Engineering April 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)08—0267—03 文献标识码：A 中图分类号：TP311.52

基于 BISON 的 UML 语法分析器设计

蒋国明[1]，魏仰苏[2]，宋瑾钰[1]
(1. 浙江理工大学信息电子学院，杭州 310033；2. 浙江水利水电专科学校计算机系，杭州 310018)

摘 要：目前基于 UML 类图和状态图做面向对象软件测试方面已有不少研究，因此研究如何实现一个 UML 语法分析器有其现实意义。利用 BISON 设计了一个 UML 语法分析器，通过对 UML 文档的词法语法分析，实现了从 UML 文档中自动提取用于软件测试的信息，提高了软件测试效率。在 VC6.0 环境下对该分析器仿真，实验结果表明，软件能正确提取测试需要的信息。
关键词：软件测试；统一建模语言；语法分析器；类图；状态图

Design of UML Syntax-analyzer Using BISON

JIANG Guoming[1], WEI Yangsu[2], SONG Jinyu[1]
(1. College of Information & Electronics, Zhejiang University of Technology, Hangzhou 310033;
2. Department of Computer, Zhejiang Water Conservancy and Hydropower College, Hangzhou 310018)

【Abstract】 Studying in how to develop a UML Syntax-analyzer is significant for there are many researchs on object-oriented test with UML class diagram and statechart diagram at present. The paper develops a UML analyzer by using BISON, which realizes getting information used to software testing automatically from documents of UML by scanning and analyzing them. It improves the efficiency of software testing. Simulating the software under the environment of VC6.0 and the results show that it can work correctly.
【Key words】 Software test; UML; Syntax-analyzer; Class diagram; Statechart diagram

统一建模语言[1](Unified Modeling Language, UML)是一种图形化建模语言，它为面向对象分析和设计(OOAD)提供了一套强大的建模工具，UML模型也是面向对象软件测试的重要依据，为类状态测试、协议的状态机测试和系统测试提供主要的分析信息。

UML 支持对系统的静态对象和动态行为建模，其中 UML 系统对象模型通过包图、类图和对象图定义系统对象及对象间的静态关系；顺序图、协作图用来描述对象间的交互关系，状态图描述对象的生命周期以及生命周期中对象可能存在的状态和事件发生引起状态间的变迁，它可以作为基于状态测试的依据。

本文通过对Rational Rose2003 的UML文档(*.mdl)分析，利用BISON[4]设计一个UML语法分析器。限于篇幅，本文只讨论对软件测试最有用的类图和状态图的LALR(1)[2]文法和语法分析器。

1 UML 类图的 LALR(1)文法

类图是在面向对象的系统模型中最普遍使用的图，是由一组类、接口、协作以及它们之间的关系构成的。类节点描述类自身的信息，包括类名、属性和操作等；类之间的关系描述类与外部环境的交互，包括泛化、依赖、实现、关联、聚合和组合等关系。例如电梯系统的类图如图 1 所示。

ElevatorControl：电梯控制器类，系统的核心控制类，和系统中所有其它类通信并控制它们。Elevator：电梯类，控制电梯上升和下降，需要时可以停下。Door：门类，系统中有两扇门，可以打开和关闭。Button：按钮类，电梯控制器类控制按钮类，按钮类生成两个子类电梯呼叫按钮类和楼层呼叫按钮类。Indicator：指示器类，系统有两类指示器：电梯位置指示器和电梯方向指示器，提供电梯的当前位置和移动方向。Safety：安全装置类，任何紧急情况时，电梯控制器触发安全装置。

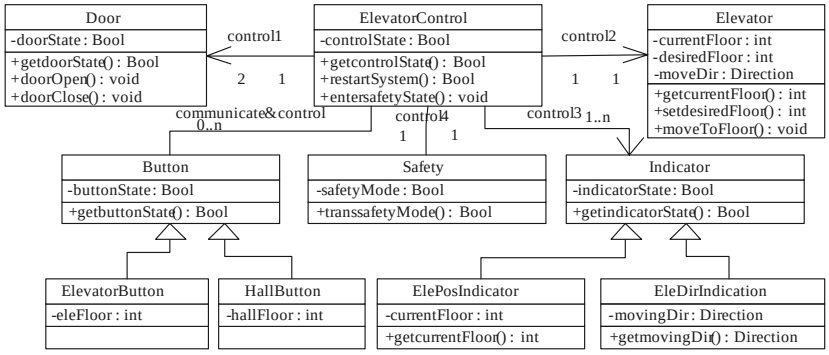

图 1 电梯系统的类图

根据 UML 语义，Rational Rose2003 的*.mdl 中类图的文档结构如下：

(object Class “ClassName”
quid “ID”
stereotype “StereotypeName”
superclasses (list inheritance_relationship_list

基金项目：浙江省科技攻关基金资助重点项目“通信协议一致性测试集自动生成软件”(2005C21027)；浙江省自然科学基金资助项目“协议形式化描述与 ATS 自动生成技术研究”(M603145)
作者简介：蒋国明(1981－)，男，硕士生，主研方向：计算机网络与系统建模；魏仰苏，教授；宋瑾钰，讲师
收稿日期：2006-04-25 **E-mail：**jgm_jobmail@163.com

```
(object Inheritance_Relationship
    …
    supplier “SupplierName”
    …) … )
  used_nodes   (list uses_relationship_list
        (object Uses_Relationship
         … ) … )
  realized_interfaces    (list realize_rel_list
        (object Realize_Relationship
         …)… )
  operations    (list Operations
        (object Operation “OperationName”…
parameters      (list Parameters
      (object Parameter “ParameterName”
        …
      type “ParameterType”
      quidu “ID”) … )
   result “ReturnType”
  …) … )
  class_attributes    (list class_attribute_list
    (object ClassAttribute “ClassAttributeName”
     …
     type “VariableType”
     quidu “ID”)
     …) … )
(object Association “AssociationName”
    …
    roles    (list role_list
    (object Role “RoleName”
    …
  client_cardinality (value cardinality “Value” )
  is_navigable BoolVariable
  is_aggregate BoolVariable)
(object Role “RoleName”
   …)) … )
logical_presentations    (list unit_reference_list
… )
```

上述文档中 Class 表示类，superclasses 表示“泛化”关系，used_nodes 表示“依赖”关系，realized_interfaces 表示“实现”关系，operations 表示“操作”，class_attributes 表示“属性”，Association 表示“关联”关系，roles 表示关系关联的两端点，logical_presentations 表示图形化界面信息。

利用 BISON 构造类图的 LALR(1)文法：

class_list:|class_list L OBJECT list R;

L、R分别代表“(“和”)”，产生式[2]中大写字符为终结符，小写字符为非终结符，class_list是归约[2]文法的开始符号，用左递归定义。list 代表类或类之间关系，产生式：list: class | association;

class 代表类，association 代表“关联”关系，association 产生式：

association: ASSOCIATION assoname quid stereotype roles;

roles:| ROLES L LIST ROLE_LIST role R;

role: | role L OBJECT ROLE rolename quid supplier quidu client_card is_nav is_agg R;

其中 quid、quidu 为 ID，stereotype 为“构造型”，产生式：

quid:QUID qstring;

quidu:|QUIDU qstring;

stereotype: | STEREOTYPE stereotypename;

assoname 是关系名，rolename 是“角色”，supplier 指出关系关联的类，client_card 代表“阶元”，is_nav 代表关系“导航方向”，is_agg 代表“聚合”关系。

class 产生式：

class:CLASS classname quid stereotype superclasses used_nodes realized_interfaces operations class_attributes;

superclasses 代表“泛化”关系，产生式：

superclasses:| SUPERCLASSES L LIST INHERITANCE_RELATIONSHIP_LIST inherit R;

inherit:| inherit L OBJECT INHERITANCE_RELATIONSHIP quid stereotype supplier quidu R;

supplier 是“继承”的父类，产生式：

supplier: SUPPLIER suppliername;

used_nodes 和 realized_interfaces 分别代表“依赖”和“实现”关系，产生式：

used_nodes:| USED_NODES L LIST USES_RELATIONSHIP_LIST uses R;

uses: | uses L OBJECT USES_RELATIONSHIP quid stereotype supplier quidu R;

realized_interfaces:|REALIZED_INTERFACES L LIST REALIZE_ REL_LIST realize R;

realize:| realize L OBJECT REALIZE_RELATIONSHIP quid stereotype supplier quidu R;

operations 代表“操作”，产生式：

operations:| OPERATIONS L LIST OPERATIONS operation R;

operation:| operation L OBJECT OPERATION opname quid pars result opexp uid quidu R;

pars:| PARAMETERS L LIST PARAMETERS par R;

par:| par L OBJECT PARAMETER parname quid type quidu R;

其中 opname 是操作名，pars 是参数列表，result 是操作的返回值，opexp 描述操作的可见性，parname 是参数名，type 表示参数的数据类型。

class_attributes 代表“属性”，产生式：

class_attributes:| CLASS_ATTRIBUTES L LIST CLASS_ATTRIBUTE_LIST attribute R;

attribute:| attribute L OBJECT CLASSATTRIBUTE attrname quid type quidu R;

attrname 是属性名，type 表示属性的数据类型。

2 UML 状态图的 LALR(1)文法

状态图由状态和迁移组成，迁移又由事件、监视条件和动作等组成，用来描述整个系统、子系统或类的动态行为。例如电梯系统的状态图如图 2 所示。

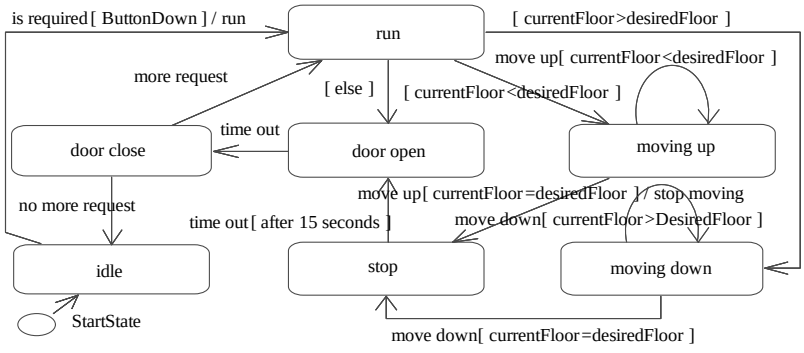

图 2 电梯系统的状态

电梯开始处于空闲状态(idle)，当有人按下按钮要求使用

电梯时(is required 发生)，进入运行状态(run)，而后根据对电梯位置和用户要求楼层的判断，进入状态变迁，没人使用时进入空闲状态。状态图文档结构如下：

```
statemachine (object State_Machine “State Model”
quid            “ID”
states      (list States
(object State “StateName”
quid            “ID”
transitions   (list transition_list
(object State_Transition
…
supplier “TargetState”
quidu “ID”
Event (object Event “EventName”
…
parameters “ParameterList”)
condition “ConditionExpresion”
action   (object action “Action”
…)
sendEvent   (object sendEvent
…)) …)
type      “StateType”) … )
  … …
  statediagrams   (list StateDiagrams
        …))
```

状态图的 LALR(1)文法：

```
statemachine: STATEMACHINE L OBJECT STATE_MACHINE modeltype quid states R;
```

其中 modeltype 为视图类型，states 为状态列表。

```
states: STATES L LIST STATES state R;
state:| state L OBJECT STATE startstatename quid transitions type R;
```

其中 state 为状态节点，采用左递归定义。startstatename 为事件的发起状态，type 为该状态的类型，可以为 StartState、Normal 和 EndState，transitions 为迁移列表。

```
transitions:| TRANSITIONS L LIST TRANSITION_LIST transition R;
transition:|transition L OBJECT STATE_TRANSITION quid supplier quidu event condition action sendevent R;
supplier: SUPPLIER endstatename;
event:| EVENT L OBJECT EVENT eventname quid parameters R;
parameters:| PARAMETERS paramname;
condition:| CONDITION condname;
action:| ACTION L OBJECT ACTION actname quid R;
sendevent:| SENDEVENT L OBJECT SENDEVENT quid event parameters target R;
```

其中 transition 代表一个迁移分支，supplier 是结果状态，event 是输入事件，parameters 是事件的参数列表，condition 是监视条件，action 是输出动作，sendevent 是发送事件，target 为发送事件的接受对象。

3 UML 语法分析器实现

利用FLEX[5]和BISON，该语法分析器由两个模块yylex()和yyparse()组成。yyparse()调用yylex()扫描*.mdl文档，yylex()返回Token，yyparse()进行“移进—归约”[2]，完成语法分析，执行语义动作，提取模型信息。

3.1 UML 类图的语法分析器

(1) classlex()设计 FLEX 文档包括 3 部分：声明部分，规则部分和辅助程序部分。声明部分定义：

```
%{   #include <cstring>
#include <cstdlib>
#include "class.tab.h" %}
%option prefix="class"
```

class.tab.h 是由 classparse()生成的头文件，classlex()从中查询 Token 值；%option prefix="class"是 FLEX 系统功能，把所有以”yy”为前缀的全局变量和函数的前缀”yy”替换为 prefix 定义的字符串。

规则部分定义正则表达式[2]并加入动作，将Token送给classparse()，如：Class {return CLASS;}

即 classlex()匹配 Class，若匹配，将 CLASS 返回给 classparse()。

辅助程序部分，如：int classwrap(){return 1;}

classwrap()用于判断本次扫描是否结束，返回非零值表示扫描结束。

(2) classparse()设计 BISON 文档包括 3 部分：声明部分，规则部分和辅助程序部分。声明部分定义：

```
%{   #include <cstring>
#include <cstdlib>
#include "ClassDef.h" %}
%union{   int digit; char *str; }
```

头文件"ClassDef.h"中定义了 5 个结构链表(以下形式定义中均省去链表的“next”指针)： Operation、Attribute、ClassNode、RoleNode 和 AssoNode。Operation 定义为五元组(char *opName, char *result, char **pars, char **parstype, char * opexp)；Attribute 定义为二元组(char *attrName, char *type)；ClassNode 定义为七元组(char *name, char *stereotype, char **depence, char **inherit, char **realize, Operation *op, Attribute *attr)；RoleNode 定义为五元组(char *rolenode, char *rolename, char *client_card, char *is_nav, char *is_agg)；AssoNode 定义为三元组(char *name, char *stereotype, RoleNode *rolenode)。

ClassNode 保存节点信息，Operation 保存操作信息，Attribute 保存属性信息，AssoNode 保存类之间关联关系信息。共用体 union 定义了 Token 的所有可能数据类型。

规则部分定义 LALR(1)文法并加入语义动作，提取类图信息，例如：

```
uses: | uses L OBJECT USES_RELATIONSHIP quid stereotype supplier quidu R
{ Insert_used_nodes (ClassNode,$7); };
```

表示某个类的某个依赖关系指向的类名存入 ClassNode 的 depence 分量中。

3.2 UML 状态图的语法分析器

(1) statelex()设计

statelex()设计与 classlex()类似，这里就不再赘述。

(2) stateparse()设计

声明部分定义如下：

```
%{   #include <cstring>
#include <cstdlib>
#include "StateDef.h" %}
%union{ char *str; }
```

头文件"StateDef.h"中定义了一个结构链表 StateNode，StateNode 为一个七元组：

```
(char *StartState, char *EndState, char *event, char *condition, char *action, char * sendevent , char *type)。
```

规则部分提取状态迁移信息并插入到 StateNode 中，例如：

(下转第 272 页)