基于 XYZ/E 的 CA 系统体系结构描述*

张广泉[1,2], 刘剑云[3], 黄正宝[2]
(1. 重庆师范大学 数学与计算机科学学院, 重庆 400047; 2. 中国科学院 软件研究所, 北京 100080;
3. 无锡商业职业技术学院 信息工程系, 江苏无锡 214063)

摘 要:XYZ/E 是一种面向软件工程的时序逻辑语言,它在统一的逻辑框架下既能表示静态语义又能表示动态语义,在很多领域得到了广泛应用,但是能够同时结合 XYZ/E 的静态语义和动态语义在网络领域具体系统中的应用还不多见。本文用 XYZ/E 对 CA 系统的 CA 组件和 RA 组件之间的体系结构关系进行了抽象描述,并对 CA 和 RA 组件中的部分子组件进行了详细描述,从而为 XYZ/E 在 CA 系统中的应用提供了一个范例。

关键词:XYZ/E;CA;RA;体系结构;组件
中图分类号:TP311 **文献标识码**:A **文章编号**:1672-6693(2005)03-0033-04

Description of the CA System Architecture Based on XYZ/E

ZHANG Guang-quan[1,2], *LIU Jian-yun*[3], *HUANG Zheng-bao*[2]
(1. College of Mathematics and Computer Science, Chongqing Normal University, Chongqing 400047;
2. Institute of Software, the Chinese Academy of Sciences, Beijing 100080;
3. Dept. of Information Engineering, Wuxi Institute of Commerce, Wuxi Jiangsu 214063, China)

Abstract: XYZ/E is a temporal logic language which is oriented toward software engineering. It can represent both dynamic semantics and static semantics under a unified logical framework, so it is applied in a great deal of fields, but there are few applications of XYZ/E in the system of Internet. This article describe the architecture which is composed by CA component and RA component in XYZ/E, and also uses XYZ/E to describe some sub components of CA component and RA component. So this article gives an axample about the application of XYZ/E in CA system.

Key words: XYZ/E; Certification Authority; Registry Authority; system architecture; component

XYZ/E 是一种面向软件工程的时序逻辑语言,它以线性时序逻辑为基础,能在统一的语义框架中表示从形式规范到可执行程序的不同抽象层次的系统描述,即既能表示程序的动态语义,又能表示规范的静态语义。目前,XYZ/E 已经在众多领域得到了应用。这些应用大体可分为两个方面:一方面,主要是利用 XYZ/E 的静态语义特征,进行有关系统的性质证明、推导或者是系统的分析建模等[1~3];另一方面,利用 XYZ/E 的动态语义的特点,对系统中实体间的交互、复杂控制算法等进行描述[4~8]。但是,这些都是侧重于 XYZ/E 的动态语义或静态语义某一个方面的应用,而能够同时结合 XYZ/E 的静态语义和动态语义在网络领域具体系统中的应用则不多见。基于此,本文首先简要介绍了 XYZ/E 和 CA 系统,然后在充分结合 XYZ/E 的动态和静态语义特性的基础上,采用 XYZ/E 描述了 CA 系统的两个关键组件 CA 和 RA 之间的体系结构关系,并对 CA 和 RA 两个组件中的部分子组件进行了详细描述。

1 时序逻辑语言 XYZ/E[9]

1.1 基本语言成分

XYZ/E 的基本语言成分是条件元,有两种形式。

$$LB = y \wedge R = > \$\ O(v1,\cdots,vk) = (e1,\cdots,ek) \wedge \$\ OLB = z \quad (1)$$

* 收稿日期:2005-06-27
资助项目:国家自然科学基金项目(60073020);中国科学院计算机科学国家重点实验室开放课题(SYSKF0303);重庆市教委科学技术研究项目(040803)
作者简介:张广泉(1965-),男,江苏连云港人,教授,博士后,硕士生导师,主要从事软件工程、形式化方法等研究。

$LB = y \wedge R = > @ (Q \wedge \$\ OLB = z)$ (2)

式中"=>"表示蕴含;R 和 Q 表示一阶逻辑公式,分别称为条件元的条件部分和动作部分;y 和 z 分别表示条件元的定义标号和转出标号;符号@为一阶逻辑算子,可以是下一时刻算子 $ O 或最终时刻算子◇。形式(1)条件元定义了程序相邻状态之间的转换关系,用于描述动态语义;形式(2)条件元表示程序抽象规范,用于描述静态语义。

1.2 通信命令

XYZ/E 提供两种通信命令。

输入命令:

$LB_i = y \wedge R = > ChNm?\ x \wedge \$\ OLB_i = w$

输出命令:

$LB_i = y \wedge R = > ChNm!\ z \wedge \$\ OLB_i = w$

其中,"ChNm"是通道名,x 是变量,z 是表达式。"ChNm? x"表示由通道 ChNm 接收信息送入变量 x 中,数据与 x 的类型必须一致;"ChNm! z"表示由通道 ChNm 送出表达式 z 的值,数据与 z 的类型必须一致。

2 基于 XYZ/E 的 CA 系统体系结构

由于电子邮件、电子商务、资源发布等 Internet 和 Intranet 应用的发展,经常需要在大型异种开放网络中不明身份的实体之间进行通信。为了在这种环境中提供机密性、认证、数字签名和完整性等服务,通常将公开密钥与其拥有者之间的信息绑定到一条电子记录上,称为数字证书,简称证书。此证书由一个可信任的权威机构签发,此权威机构称为 CA[10]。CA 系统从逻辑上可以分成 3 层,分别是用户层 CL(Client Layer),应用层 AL(Application Layer)和系统层 ML(System Layer),其结构如图 1 所示。其中应用层处理有关证书验证,查询等事务,系统层则处理证书签发、密钥管理等。

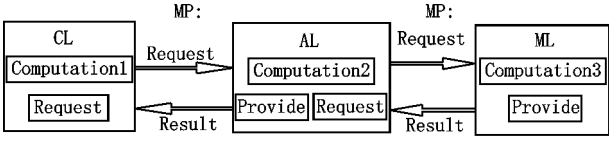

图1 CA系统的组件模型

2.1 体系结构描述

CA 系统通常由注册审核部门 RA(Registry Authority)和 CA 认证中心组成。注册审核部门 RA 负责接收用户申请,对证书申请者进行资格审查,审查通过后将信息发送给 CA,由 CA 来生成数字证书,然后将证书发往数字证书库和 RA 等处。其中数字证书库主要基于 LDAP 构建,轻量目录访问协议 LDAP(Lightweight Directory Access Protocol)作为一个开放的、独立于任何厂商的标准,为目前在分布式应用系统和服务中所要求的信息集中存储和管理提供了一个可扩展的结构[11]。

在本文中,把 CA 系统的注册审核部门 RA 和认证中心 CA 看作两个组件,CA 和 RA 组件之间是采用管道-过滤器风格构建的,可用 XYZ/E 描述如下。

过滤器 sender 调用 SeInput 过程输入,调用 SeOutput 过程输出。

```
Sender = =[LB = m = > SeInput ∧ $ OLB = m1;
LB = m1 = > Seprocess ∧ $ OLB = m2;
LB = m2 = > SeOutput ∧ $ OLB = y]
SeInput = =[LB = n = > connector. source ∧ $ OLB = RETURN]
SeOutput = =[LB = n = > connector. sink ∧ $ OLB = RETURN]
```

管道 connector 有输入端口 Source 和输出端口 sink.

```
connector = =[source;sink]
%ALG[
 %PROS[
    PROSSource(%CHN ASYN ch1(*,soprocess1:NM):String;
               %CHN ASYN ch2(soprocess2:NM,*):String;)
               = =[
                 %ALG[
                   LB = START = > $ O(LB = y0);
                   LB = y0 = > Protocol. send;
                   ]]
    PROSSink(%CHN ASYN ch3(*,siprocess1:NM):String;
             %CHN ASYN ch4(siprocess2:NM,*):String;)
             = =[
               %ALG[
                 LB = START = > $ O(LB = m0);
                 LB = m0 = > Protocol. receiver;
                 ]]
 %ALG[
   LB = START = > y0;
   LB = y0 = > %O(sender1 = PROSSource{1} ∧ receiver1 = PROSSink{1} ∧ LB = y1);
   LB = y1 = > ‖[sender(%CHN ch1(*,receiver1)|ch1(*,soprocess1);
                %CHN ch2(receiver,*)|ch2(soprocess2,*);
                receiver(%CHN ch3(*,sender1)|ch3(*,siprocess1);
                %CHN ch4(sender1,*)|ch2(siprocess2,*))]
```

在上面,对管道 connector 的输入端口 source 和输出端口 sink 采取的是抽象描述,利用 XYZ/E 的特点,也可以对它们进行求精,例如,对输出端口 sink 的求精结果如下。

```
%PROS[
  %PROSSink(%CHN ASYN ch3(*,siprocess1:NM):STRING;
            %CHN ASYN ch4(siprocess2:NM,*):STRING;
    ==[
      %ALG[
        %LOC[next,messnum:INT;
             sinkwindow:ARRAY[sws2:STRING];
             message:STRING;]
             LB=START=>$ OLB=m1;
             LB=m1=>$ O(next=1)∧$
                 OLB=m2;
             LB=m2=>$ Osinkwindow.shift
                 (1)∧$ OLB=m3;
             LB=m3=>$ OLB=m4;
             LB=m4=>$ O ch4? message
                 ∧$ OLB=m5;
             LB=m5=>$ O(messnum=get-
                 seq(message)∧LB=m6);
             LB=m6∧(next>=messnum)
                 and(messnum<=next+
                 sws2)=>$ OLB=m7;
             LB=m6∧(next<messnum)and
                 (messnum<=next+sws2)
                 =>$ OLB=m4;
             LB=m7∧sinkwindow.buffer
                 [messnum]=empty=>$
                 OLB=m8;
             LB=m7∧¬sinkwindow.buffer
                 [messnum]=empty=>$
                 OLB=m14;
             LB=m8=>$ Osinkwindow.
                 buffer[messnum]=message
                 ∧$ OLB=m9;
             LB=m9∧¬sinkwindow.buffer
                 [next]=empty=>$ OLB
                 =m10;
             LB=m9∧sinkwindow.buffer[next]
                 =empty=>$ OLB=m13;
             LB=m10=>$ Onext=next+1
                 ∧$ OLB=m11;
             LB=m11=>$ Osinkwindow.
                 shift(next)∧$ OLB=m12;
             LB=m12=>$ OLB=m9;
             LB=m13=>$ Omessage=
                 merge(next,"ack")∧$
                 OLB=m14;
             LB=m14=>$ O ch3! message
                 ∧$ OLB=m15;
             LB=m15=>$ OLB=m4;
        ]]]
```

2.2　CA 组件描述

CA 组件是一个复合组件,它还包含很多子组件,限于篇幅,本文只对其中最重要的证书签发组件进行描述。签发组件具体实现过程为:CA 服务器从证书申请表中逐条读取已经审核过的证书申请信息,对应每一条证书申请信息,CA 首先为申请人生成 RSA 密钥对,并将私钥加密后写入密钥库的密钥表中;而后 CA 为申请人生成证书申请,接着 CA 为该申请人生成证书,并将证书保存在证书库的证书表中;同时将证书的一个副本发布到 LDAP 目录服务器;之后通过 RA 服务器向用户发送一封电子邮件,通知用户证书已经发行成功,并且把证书序列号告诉用户;最后将该条证书申请信息从证书申请表中删除。

```
%PROS[
  %PROSCAcountersign(%CHN ASYN c1(*,process1:NM):String;
                     %CHN ASYN c2(process2:NM,*):String;)
    ==[
      %ALG[
        %LOC[appmessage,pubkey,prikey,tellmessage,
             delmessage:string;
             x,y,m,n,o,p:string;
             certino:integer;]
             LB=START=>$ OLB=k1;
             LB=k1=>$ O appmessage=0∧$ OLB=k2;
             LB=k2∧c2?=>$ OLB=k3;
             LB=k2∧¬c2?=>$ OLB=k1;
             LB=k3=>$ O c2? appmessage∧$ OLB=k4;
             LB=k4=>$ O x=appmessage∧$ OLB=k5;
             LB=k5=>create_key(%iop pubkey|
                 pu;%iop prikey|pr);
             LB=k6=>$ O y=prikey∧$ OLB=k7;
             LB=k7=>encrpt_key(%iop y|epri);
             LB=k8=>$ O c1! y∧$ OLB=k9;
             LB=k9=>create_certi_app(%iop x|cap);
             LB=k10=>$ O(m=x∧LB=k11);
             LB=k11=>create_certi(%iop m|cc);
             LB=k12=>$ O n=get_certino(m)∧
                 $ OLB=k13;
             LB=k13=>$ O o=copy_certi(m)∧
                 $ OLB=K14;
             LB=k14=>$ O c1! m∧$ OLB=
                 k15;
             LB=k15=>$ O c1! o∧$ OLB=k16;
             LB=k16=>$ O tellmessage=1∧$
                 OLB=k17;
             LB=k17=>$ O tellmessage=tellmes-
                 sage+n∧$ OLB=k18;
```

```
LB = k18 = > $ O c1! tellmessage ∧ $
     OLB = k19;
LB = k19 = > $ O delmessage = 1 ∧ $
     OLB = k20;
LB = k20 = > $ O delmessage = delmessage
     + appmesssage ∧ $ OLB = k21;
LB = k21 = > $ O c1! delmessage ∧ $
     OLB = k22;
LB = k22 = > $ OLB = k1;
]]]
```

2.3 RA 组件描述

在 RA 组件当中也包含很多子组件,本文只对其中的证书申请组件进行形式化描述,其实现过程如下:用户填写申请人信息及其 PIN 值,提交给子 RA 服务器。子 RA 服务器将申请人信息保存在证书申请库的证书申请表中,并将其状态值标记为"未审核"。RA 管理员审核申请人真实身份,若提交的申请人信息与其真实物理身份相符,则审核通过;否则审核不能通过。若用户的证书申请审核通过,则 RA 将证书申请表中该申请人状态值更新为"已审核",将该申请转交给安全服务器,经过安全检查后,转交给系统的 RA 服务器,等待 CA 发放证书;反之,如果未通过审核,则子 RA 服务器将该申请人的信息从证书申请表中删除,并且将其对应的用户信息从用户表中删除。最后子 RA 服务器将审核结果以电子邮件的方式发送给用户。

```
% PROS[
  % PROSCAapplication(% CHN ASYN m1( * ,chout:NM):STRING;
% CHN ASYN m2(chin:NM, * ):STRING
= =[
% ALG[
% LOC[ client _ message,exam _ value:STRING;
     tellmessage,delmessage,cm,ev,sh:STRING;]
LB = START = > $ OLB = y1;
LB = y1 = > $ O cm = 0 ∧ $ OLB = y2;
LB = y2 ∧ m2? = > $ OLB = y3;
LB = y2 ∧ ¬ m2? = > $ OLB = y1;
LB = y3 = > $ O m2? cm ∧ $ OLB = y4;
LB = y4 = > $ O client _ message = cm ∧ $ OLB = y5;
LB = y5 = > $ O exam _ value = 0 ∧ $ OLB = y6;
LB = y6 = > $ O cm = cm + exam _ value ∧ $ OLB = y7;
LB = y7 = > $ O client _ message = cm ∧ $ OLB = y8;
LB = y8 = > $ O m1! cm ∧ $ OLB = y9;
LB = y9 = > $ O ev = exam _ vale ∧ $ OLB = y10;
LB = y10 ∧ (sh = 1) = > $ OLB = y11;
LB = y10 ∧ (sh = 0) = > $ OLB = y15;
LB = y11 = > $ O exam _ value = 1 ∧ $ OLB = y12;
LB = y12 = > $ O cm = cm + exam _ value ∧ $ OLB = y13;
LB = y13 = > $ O client _ message = cm ∧ $ OLB = y14;
LB = y14 = > $ O m1! cm ∧ $ OLB = y1;
LB = y15 = > $ O tellmessage = 1 ∧ $ OLB = y16;
LB = y16 = > $ O tellmessage = tellmessage + sh ∧ $ OLB = y17;
LB = y17 = > $ O m1! tellmessage ∧ $ OLB = y18;
LB = k18 = > $ O delmessage = 1 ∧ $ OLB = y19;
LB = k19 = > $ O delmessage = delmessage + cm ∧ $ OLB = y20;
LB = k20 = > $ O m1! delmessage ∧ $ OLB = y21;
LB = k21 = > $ OLB = y1;
]]]
```

3 结束语

本文采用 XYZ/E,对 CA 系统中的 CA 和 RA 两个关键组件的体系结构构成以及它们当中的部分子组件进行了形式化描述,从而为 XYZ/E 在网络领域实际系统的初步应用提供了一个范例。下一步将采用 XYZ/E 对 CA 系统进行全面地描述与求精,并在此基础上对系统进行形式化分析和验证工作。

参考文献:

[1] 郭亮,唐稚松. 基于 XYZ/E 描述和验证容错系统[J]. 软件学报,2002,13(5):916-919.

[2] 韩俊刚,王岩冰,沈武威. 用 XYZ/E 语言描述和验证硬件的行为[J]. 软件学报,1996,7(11):676-677.

[3] 费丽娟,胡金柱,李敏. 使用时序逻辑检测软件需求阶段的特征干扰[J]. 计算机应用,2004,24(3):109-111.

[4] 唐小平,唐稚松,马华东,等. XYZ 系统在动画设计中的应用[J]. 软件学报,1998,9(1):1-6.

[5] 王国意,史元春,徐光佑. 计算机支持的协同工作系统的时序逻辑模型[J]. 软件学报,1998,9(3):169-170.

[6] YAN An,TAHG Zhi-song. Building Hybrid Real-Time Systems in XYZ/E-Implementation of the Steam-Boiler Control Specification Problem[J]. Journal of Software, 2000, 11(6):711-719.

[7] 张广泉,戎玫,晏荣杰. 基于时序逻辑语言描述的监控系统的软件体系结构求精[J]. 计算机工程与应用,2003,39(31):14-17.

[8] YAN An,TANG Zhi-song. Hybrid Systems in XYZ/E[J]. Journal Software,2000,11(1):1-7.

[9] 唐稚松. 时序逻辑程序设计与软件工程[M]. 北京:科学出版社,2002.

[10] 汪立东,余祥湛,方滨兴. PKI 中几个安全问题的研究[J]. 计算机工程,2000,26(1):14-15.

[11] 张辉,杨岳湘,汪诗林. 数字校园中基于 LDAP 的统一用户身份管理技术研究[J]. 计算机工程与科学,2005,17(1):14-16.

(责任编辑 游中胜)