一种面向控制软件需求分析的方法*

张丽芸，蒲戈光，王　政，李建文
（华东师范大学 软件学院，上海 200062）

摘　要：设计航天控制系统是一个复杂的过程，涉及需求设计、编码、测试等一系列的流程，若能在需求设计阶段发现错误，那么能减少不少的工作量。针对这一问题，提出了一种分析控制软件需求的方法和一个名为 SPARDL 的建模语言，并制作了一套 SPARDL 工具。SPARDL 可以描述周期性的控制系统，首先将需求文档转换为 SPARDL 模型，且提供了图形化的表示方法；然后运用原型生成技术去仿真系统的行为，进一步分析需求的准确性。最后以一个案例表明了用 SPARDL 分析一个简单的航天控制系统需求的有效性。

关键词：控制系统；需求分析；图形化；原型生成；系统仿真
中图分类号：TP311.521　　**文献标志码**：A　　**文章编号**：1001-3695(2013)02-0465-04
doi:10.3969/j.issn.1001-3695.2013.02.040

Analysis method of control system requirement

ZHANG Li-yun, PU Ge-guang, WANG Zheng, LI Jian-wen
(*Software Engineering Institute*, *East China Normal University*, *Shanghai* 200062, *China*)

Abstract: It is a complex process to design a periodic control system which involves design, coding, test and so on. If errors can be found in the period of requirement design, it will reduce a lot of work. In order to solve this problem, this paper proposed an analysis method for control system specification and also provided a modeling language called SPARDL and the SPARDL tool. First SPARDL tool converted a requirement to SPARDL model and provided a graphical interface to show the entire control system. Then it used a prototype generation technique to simulate the system behaviors. It could analyze the accuracy of the requirement. Finally, a case study shows the effectiveness of using SPARDL to analyze the requirement of a simple aerospace control system.

Key words: control system; requirement analysis; graphical; prototype generation; system simulation

0　引言

中国航天工程与技术研究院(CAST)投入了大量的精力设计和开发航天控制器。在一个软件的生命周期内，从需求分析到编码完成，测试维护中任何一个阶段都有可能存在漏洞。因此，软件工程师希望尽可能早地发现软件开发过程中的错误，如果能在需求分析阶段发现错误，那能减少许多不必要的工作。在 CAST 工程中，最常见的需求描述语句是函数调用语句，工程师并没有详细地验证函数调用的结果与自己期望的结果是否一致，这留下了许多不明确的、不正确的或者不完整的潜在的缺陷。面对这一问题，本文提出了一种需求建模语言 SPARDL(space air craft description language)并开发了一个工具，用来辅助 CAST 的软件开发测试人员对周期性的控制系统进行建模。软件需求者按照文档模板书写需求，SPARDL 工具将需求文档转换为 SPARDL 模型。得到了 SPARDL 模型后，使用原型生成技术，即将 SPARDL 的模型转换为 C 程序，这样就可以仿真需求所写的系统的具体行为，大大方便了 CAST 的工程师去验证它们的需求。若在仿真中发现了问题，便可尽早修改需求。

本文阐述了如下的主要工作：a)提出了 SPARDL(适用于描述周期性的控制系统)；b)将软件开发中的需求文档转换为 SPARDL 模型；c)以图形化的方式展现 SPARDL 系统，方便全体软件开发人员对系统的了解，帮助开发者进行沟通；d)使用原型生成技术仿真控制系统的具体行为。

1　方法学的总述

本章讲述如何对一个周期性的控制系统建模以及作相关的分析。图 1 显示了整个方法的工作流。

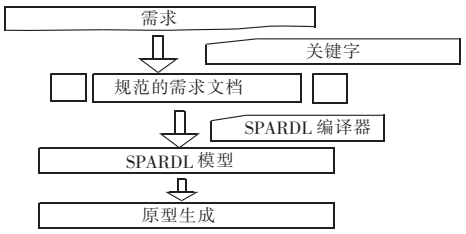

图 1　工作流

首先提供一个需求模板，需求模板中规定了一些关键字，CAST 的软件工程师按照模板书写他们的需求，之后 SPARDL 的编译器将需求文档编译成 SPARDL 模型。SPARDL 模型刻画了组成控制系统的各个模式的基本信息以及其他必要的相

收稿日期：2012-07-10；**修回日期**：2012-08-19　　**基金项目**：国家自然科学基金资助项目(91118007)；国家自然科学基金创新研究群体科学基金资助项目(61021004)；国防科工局“航天器产品软件安全性评价体系及关键技术研究”项目

作者简介：张丽芸(1988-)，女，主要研究方向为程序分析验证(kellyzly@hotmail.com)；蒲戈光(1978-)，男，博导，主要研究方向为程序分析验证、Web 语义；王政(1986-)，男，博士，主要研究方向为程序分析验证；李建文(1987-)，男，主要研究方向为程序分析验证.

关信息,包括模式迁移等。在传统的 CAST 的工作流程中,需求开发工程师写完一份需求后,会根据这份需求文档开发一个可抛弃的快速原型来检验需求是否正确。对此提出了一种原型生成技术,即将 SPARDL 模型转换为 C 程序,需求开发者通过分析 C 程序的执行路径来判断控制系统的行为是否正确或合理。与 CAST 软件工程师传统的方法相比,原型生成的方法更为高效,节省时间。

2 SPARDL 模型

SPARDL 是一种用于控制系统的需求建模语言,尤其适用于基于模式的、含有多种状态和有限周期的控制系统。SPARDL 与状态图比较类似,一个事件引导着一个状态的变迁。相比之下,SPARDL 在控制结构的描述上提出了自己的一套规范而并非像状态图那样着重关注所有状态的变迁关系。对于一个控制系统模型,SPARDL 将用两层结构对它建模,第一层在抽象层次上对模式进行描述,第二层将对模式中的具体每个模块进行描述。

2.1 基于模式的语义

图 2 用符号语言定义了模式。一个控制系统由三个部分组成,即初始化(init)、过程(procs)和迁移(trans)。初始化部分初始化当前模式的状态;过程由一系列的 proc 组成,系统周期性地执行 procs 中的行为。一个 proc 是个二元组,由周期 f 和模块 modu 组成。每间隔 f 时间段,系统会去执行函数模块中的行为,且 $1/f$ 是个严格的整数。例如 $f=1$,一个周期系统执行一次模块中的行为;$f=1/4$,每 4 个周期系统执行一次模块中的行为。模块 modu 是由 pre(前置条件)和 stmts(语句块)组成的,如果在系统执行时,pre 条件为 false,则不会执行模块 modu 中的行为。当系统中的一个模式(mode)执行完它的 procs 部分便会转向迁移部分。迁移部分由 guard、priority、modu、mode! 四部分组成(mode! 代表目标模式),当一个模式转向目标模式前,首先去判断迁移条件 guard 是否满足,若不满足,则不会转向目标模式。当几个 guard 同时满足,此时就需要凭 priority 区分优先级,当前模式转向高优先级的目标模式。模块 modu 主要负责对当前模式中使用到的变量进行赋值。

Guard 即是个 bool 值,是个时间算子。在 SPARDL 中有两个时间算子,即 after 和 duration。其中,after(g,c)是指第 p' 个周期经过 c 个周期到第 p 个周期,条件 g 在第 p' 个周期时能被满足但之后的 c 个周期都不满足;duration(g,c)是指第 p' 个周期经过 c 个周期到第 p 个周期,在这 c 个周期内条件 g 总能被满足。图 2 用执行路径描述了两种时间算子,虚线箭头代表执行路径能够被重复,实心圆代表该状态下条件 g 满足。这两个时间算子能够描述需求中的时间性质,例如当系统在模式 M_1 时,角速度在四个周期内小于 0.08,则转向模式 M_2'',这样一个需求可以使用 duration 算子来描述,即 duration($a<0.08$,4)。

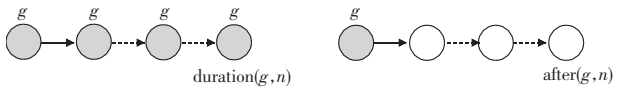

图 2 SPARDL 时间算子

SPARDL 模式语义如下:

$$modes =_{df} \{ mode \mid mode = (init, procs, trans) \}$$

$$init =_{df} modu$$

$$procs =_{df} proc; procs \mid proc$$

$$proc =_{df} (f, modu)$$

$$modu =_{df} (pre, stmts)$$

$$pre =_{df} b$$

$$Trans =_{df} \{ tran \mid tran = (guard, priority, modu, mode!) \}$$

$$guard =_{df} b \mid after(guard, c) \mid duration(guard, c) \mid \neg\ guard \mid guard \vee guard \mid guard \wedge guard \mid guard \rightarrow guard$$

2.2 基于模块的语法

SPARDL 模块的语义如下:

$$func =_{df} id\ stmts$$

$$stmts =_{df} stmt \mid stmts; stmts \mid \text{if } b \text{ then stmts else stmts} \mid \text{while } b \text{ do stmts} \mid \text{call id}$$

$$stmt =_{df} x := e \mid \text{skip}$$

以上用符号语言定义了模块,func 由一个独一无二的 id 和一系列的 stmt 组成。

stmt 包含以下两部分:

$x:e$ 赋值语句

skip 不进行任何操作,即 $x := x$

stmt 是一个顺序的组合,首先执行第一句语句,当第一句执行完毕,第二句语句开始执行。

if b then stmts else stmts 是条件结构;

while b do stmts 是循环结构;

call id 是调用指定的函数;

更多的 SPARDL 的语法语义描述请参考文献[1]。

3 需求文档转换成 SPARDL 模型

CAST 的工程师按照本文提供的 Word 模板书写需求文档,图 3 显示了本文提供的 Word 模板。需求文档由数据字典、函数模块、模式组成。数据字典中定义了之后在模块、模式中要用到的变量以及变量的具体信息(位数、初值等),模块中封装了控制系统中经常调用的功能模块以及相关信息(输入、输出、局部变量和公式)。模式描述了控制系统中的具体一个模式以及相关信息(进入条件、初始化、调用过程、模式转换)。通过使用 Microsoft. Office. Interop. Word 类库解析需求文档,从中提取数据字典、模块、模式这三部分。CAST 的工程师使用了 MathType[2] 来书写需求中的数学符号、数学公式,对于如何获取 MathType 的公式信息,本文的解决方法是将文档中的 MathType 公式转换为 LaTeX[3] 公式,之后通过解析 LaTeX 公式获取数学符号和数学公式的相关信息。若在解析过程中发现错误,如在模块、模式中使用的变量在之前的数据字典中没有出现过,工具也会给出相关的错误信息。在解析完需求文档之后,将获取到的控制系统的详细信息传送给 SPARDL 的 compiler。

数据字典

符号	物理意义	位数	…
变量 1	物理意义 1	位数 1	…
…	…	…	…

模块名称: 模块 1
编号:xxx
功能:xxx
输入:xxx
输出:xxx
局部变量:xxx
公式:xxx
…
模式名称:模式 1
进入条件:条件 1
初始化:
调用过程:
　周期任务: 周期 xxx 秒
　　a)调用模块 1
模式转换
　优先级:xxx
　满足:条件 xxx
　目标:xxx

图 3　SPARDL 提供的 Word 模板

本文使用 ANTLR[4] 这个工具来生成 SPARDL 的词法解析器、语法解析器。通过定义 SPARDL 的词法记号和语法规则,ANTLR 自动生成 SPARDL 的词法解析器(SPARDL lexer)和语法解析器(SPARDL parser)。

4　图形化展示系统

为了方便 CAST 的软件工程师更直观地了解控制系统,SPARDL 工具用不同的符号展现了 SPARDL 所描述的控制系统。图 4 从上层到下层逐步描述了工具对整个系统的图形表达。第一层描述了模式间的转换关系,如图中有四个模式 $m_0 \sim m_3$,m_0 可以在一定的条件下转向 m_1,m_2。在本系统中,双击具体一个模式,就可以进入第二层。第二层描述一个模式的主要组成部分:init,procedure,transition,在系统中用不同的符号表示了 init,procedure,transtion(图 4(b))。双击 init 部分(也可以双击 procedure,transition),就可以进入第三层图 4(c),它描述了具体一个 init(或者 procedure,transition)的行为。Init 或 procedure 部分主要采用流程图的方式描述 init 和 procedure 中的行为、支持顺序、条件、循环结构。条件结构的图形符号中菱形代表条件,沿着 Y 标签走向 true 分支,沿着 N 标签走向 false 分支,最后两条分支归并到 end 终节点。循环结构的图形符号由一个具有循环条件 condition 标签的虚线框的矩形构成,虚线框中描述了满足条件的具体的执行流程。Transition 中描述了迁移模式,当监听条件(guard)满足时,当前模式转向目标模式,如图 4 中,当 guard0 满足时,当前模式 m_0 转向 m_1,当 guard1 满足时,当前模式 m_0 转向 m_2。图 5、6 是一个控制系统的模式层的表示和具体一个模块内的控制流的表示。图 5 用 SPARDL 界面工具显示了一个控制系统的模式层,图中显示了模式名、模式之间的转换。选中两个模式间的转换条件(ZCC1-2),工具会自动打开 properties 视图显示模式迁移的源模式(from)、目标模式(to)和监听条件(guard)。

5　SPARDL 原型生成

为了验证控制系统的行为,本文使用了原型生成方法来进行基于模型的测试,整个方法的流程如图 6 所示。a)将 SPARDL 模型转换为 C 程序,SPARDL 中的控制结构(判断、循环)将对应 C 语言的控制结构(if、while);b)将 C 代码和物理环境仿真器的 DLL 文件、head 文件一起编译。其中数据字典被转换为 C 文件(Data. h、Data. c)中的全局变量,并在文件中进行全局变量的赋初值操作,在模式和模块中需要用到的变量也需要定义在这些文件中。封装着控制算法的模块将转换为 C 文件(Modules. h、Modulesxxx. c)中的函数,SPARDL 模型中的模式将转换为 C 文件(System. h、System. c、Modexxx. c…)中的具体的模式函数,模式函数将会调用相应的模块函数。

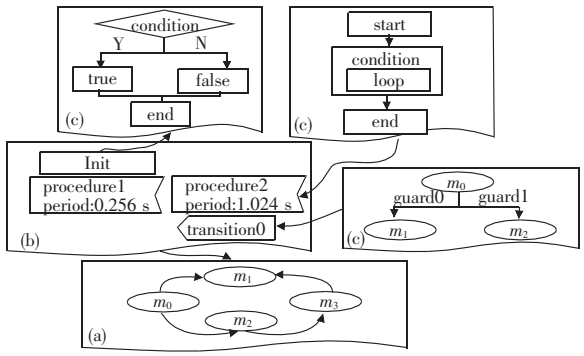

图 4　SPARDL 图形化表示

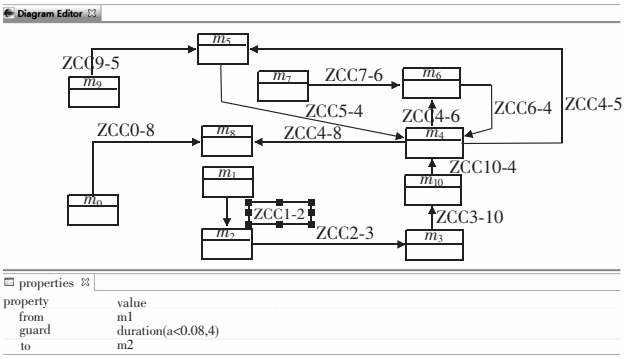

图 5　SPARDL 工具中的模块

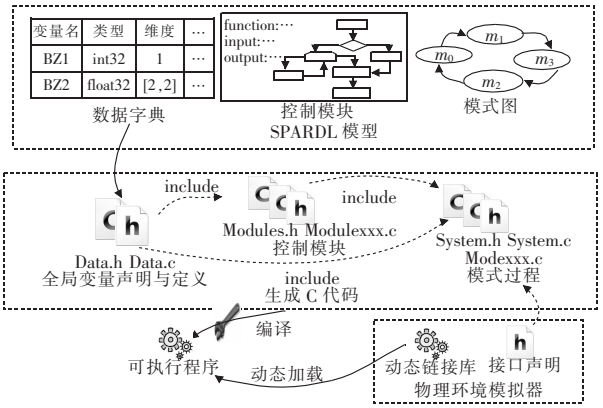

图 6　原型生成方法

6　示例

本章给出的两个例子讲述了如何从需求文档构建 SPARDL 模型,软件需求撰写者按照含有关键字的模板书写宇宙飞船的需求(例 1),通过 SPARDL 的编译器将之编译成宇宙飞船控制系统中的一个模式 mode1(例 2),mode1 由 name、guard、initialization、procedure、transition 几部分组成。其中,guard 是进入该模式的条件;initialization 则是进入模式后初始化变量的语句集合;procedures 是一系列周期性的任务的集合,在例子中

有两个周期性的任务,第一个任务的的周期是 8 ms,第二个任务的周期是 16 ms;transition 描述模式间的迁移,在例子中,当 $\omega>0$ 时,mode1 转向 mode2;当 $\phi>0$ 时,mode1 转向 mode7,当 $\varphi>0$ 时,mode1 转向 mode9。

例 1 spacecraft

```
entry condition:modeflag = 1
initialization
1.    set voltage q as zero
2.    set angular velocity ω,ϕ,φ as zero
procedure
period is 8 ms
1.  call 1.1
2.  if(q>0) call 1.1.2
period is 16 ms
1.  call 1.14
transition
        1.  优先级:1
            进入条件:ω>0
            目标:mode 2
        2.  优先级:1
            进入条件:ϕ>0
            目标:mode7
        3.  优先级: 1
            进入条件:φ>0
            目标:mode9
```

例 2 mode 1

```
mode name:spacecraft
guard:modeflag = 1
initialization:q=0;ω=0;ϕ=0;φ=0;
procedures:
period:8 ms
    call 1.1.1
    if(q>0) call 1.1.2
period 16 ms
    call 1.1.4
transitions:
  { priority:1
    condition:ω>0
    target:2        }
  { priority:1
    condition:ϕ>0
    target:7        }
  { priority:1
    condition:φ>0
    target:9        }
```

将需求文档转换为 SPARDL 模型后,再运用原型生成的技术将模型转换为 C 代码(图 7)对模型进行仿真。

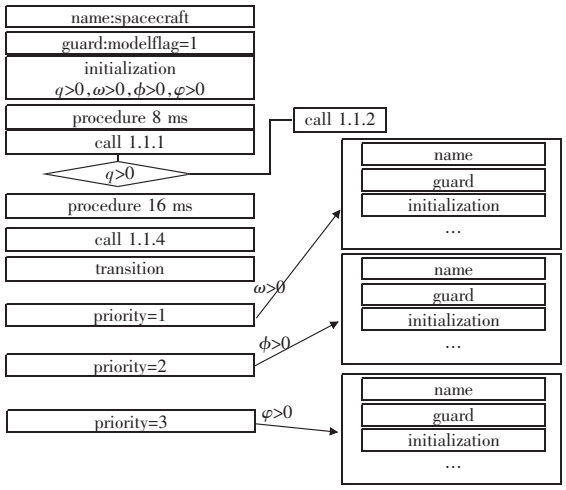

图 7 SPARD 转换为 C 代码

仿真代码如下:

```
if(ω>0)
  transferTo(mode2);
if(ϕ>0)
  transferTo(mode7);
if(φ>0)
  transferTo(mode9)
```

当输入不同的测试数据后,发现当 $\omega>0$,mode1 总是转换不到 mode2。通过分析仿真代码的执行路径发现,当 $\omega>0$ 时,$\phi>0$,$\varphi>0$ 也成立,所以此时 mode1 先转向 mode2,再转向 mode7,最终转向 mode9。而需求开发者的原本意图是希望当 $\omega>0$ 时,mode1 转向 mode2 即可,不需要再转向其他模式。检查出需求文档中的这个二义性问题后,发现需要重新修改三种转换模式的优先级就可以解决该问题,修改后的需求文档如下:

```
transition
      a)优先级:1
          进入条件:ω>0
          目标:mode2
      b)优先级:2
          进入条件:ϕ>0
          目标:mode7
      c)优先级:3
          进入条件:φ>0
          目标:mode9
  if(ω>0)
    transferTo(mode2);
  else if(ϕ>0)
    transferTo(mode7);
  else if(φ>0)
    transferTo(mode9)
```

7 讨论

SPARDL 模型的思想来源于领域建模语言[5]和可执行的需求分析规划说明[6]。SPARDL 的总体结构与 Statechart[7,8]相似,在 SPARDL 中,控制系统里的一个模式就如状态图里的一个状态,模式间的迁移就如状态间的转换;但是在状态图里,一个状态的子结构仍然是状态图,而在 SPARDL 里面一个模式的子结构是个活动图。SPARDL 的目标是对控制系统提供一个精简的抽象,这与 Gitto[9]很相似。Gitto 也是个周期性的建模语言,在 Gitto 中,一个模式内的任务都是并行被处理的,相比之下,SPARDL 扩展了语义,提出了自己的时间算子。SPARDL 模式着重于获取和验证系统的需求文档。也有许多其他的模型建模语言,例如,RCAT[10]介绍了一种获取需求文档中的概念,用 spin[11]这个工具将它们转成自动机;CHARON[12]是一种描述分层的混合模型和嵌入式软件的建模语言。相比之下,SPARDL 着重于对需求文档建立模型之后,验证需求文档而非自动生成软件。

8 结束语

本文提出了一种对周期性控制系统需求建模分析的方法,将需求文档转换为 SPARDL 模型。SPARDL 可以用图形化的方式展示整个控制系统,这些能帮助了 CAST 的工程师(需求分析人员、开发人员、测试人员)更深刻地理解整个系统。接下来笔者还将进一步完善本文的工具,使之能便捷地被 CAST 的工程师们使用。

致谢 最后再一次给本文资助的国家自然科学基金委重点项目和国防科工局“航天器产品软件安全性评价体系及关键技术研究”项目致谢。

(下转第 475 页)

3.3.3　去重率对比

对 NAF 去重的效果与其他常见方法进行了对比,去重率情况如图 5 所示。由于采用了声学指纹,NAF 的鲁棒性更强,去重率比其他几种方法有了明显的提升。

3.3.4　文件平均网络流量

各种方法下,平均每个文件产生的网络流量情况如图 6 所示。

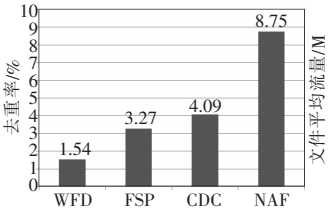

图 5　去重率对比

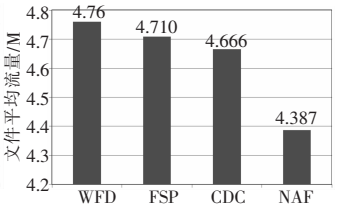

图 6　平均网络流量对比

由于 NAF 提高了去重率,减少了重复文件的上传,而且 NAF 指纹产生的网络流量相比于 MP3 文件而言非常小。所以 NAF 方法中每个文件的平均网络流量有了明显下降。

4　结束语

本文从海量音乐分享场景出发,针对 Hadoop 不能有效处理小文件,以及传统的文件去重方法去重率不高、不适应海量数据在线处理的问题,结合音乐文件自身的声学特性和 MP3 文件包含的元信息,提供一种采用声学指纹去重的海量 MP3 文件存储架构,通过索引、在线归并和去重,在保证性能的前提下,很好地解决了 Hadoop 处理小文件时的内存瓶颈问题,同时提供了更好的去重效果。另外,通过离线归并和副本调整,可以根据系统的运行不断地优化存储,具备良好的性能和扩展性,能够支撑海量数据处理,具有良好的实用价值。本文的方法针对 MP3 文件进行,如何将其扩展到其他文件类型,值得进一步的研究;另外,离线归并和副本调整算法目前仍相对简单,后续仍需研究适用于 MapReduce 框架的优化算法。

参考文献:

[1] 王福林. 新技术对音乐产业的冲击[J]. 辽宁行政学院学报,2008,10(1):185-186.

[2] 中国宽带用户调查[EB/OL]. (2011). http://www.dcci.com.cn/.

[3] WHITE T. Hadoop: the definitive guide[M]. Sebastopol: O'Reilly Media,2009:150-190.

[4] WHITET small files problem[EB/OL]. (2009-02-02). http://blog.cloudera.com/blog/2009/02/the-small-files-problem/.

[5] BOLOSKY W J,CORBIN S,GOEBEL D,*et al*. Single instance storage in Windows® 2000[C]//Proc of the 4th Usenix Windows System Symposium. Berkeley:USENIX Association, 2000:13-24.

[6] BOBBARJUNG D R,JAGANNATHAN S,DUBNICKI C. Improving duplicate elimination in storage systems[J]. ACM Trans on Storage,2006,2(4):424-448.

[7] DENEHY T E,HSU W W. Duplicate management for reference data, RJ 10305(A0310-017)[R].[S.l.]:IBM Research Division, 2003.

[8] BRODER A Z. Identifying and filtering near-duplicate documents[C]//Proc of the 11th Annual Symposium on Combinatorial Pattern Matching. London:Springer-Verlag,2000:1-10.

[9] LIU Chuan-yi,LU Ying-ping,SHI Chun-hui,*et al*. ADMAD: application-driven metadata aware de-duplication archival storage systems[C]//Proc of the 5th IEEE International Conference on Storage Network Architecture and Parallel. Piscataway:IEEE Press,2008:29-35.

[10] MEISTER D,BRINKMANN A. Multilevel comparison of data deduplication in a backup scenario[C]//Proc of Israeli Experimental Systems Conference. New York:ACM Press,2009.

[11] 赵晓永,杨扬,孙莉莉,等. 基于 Hadoop 的海量 MP3 文件存储架构[J]. 计算机应用,2012,32(6):1724-1726.

[12] 布隆过滤器[EB/OL]. http://zh.wikipedia.org/wiki/布隆过滤器.

[13] BLOOM B. Space/time trade-offs in hash coding with allowable errors[J]. Communications of the ACM,1970,13(7):422-426.

[14] 声学指纹[EB/OL]. http://zh.wikipedia.org/wiki/声学指纹.

[15] HAITSMA J,KALKER T. A highly robust audio fingerprinting system[C]//Proc of International Symposium on Music Information Retrieval. 2002:107-115.

[16] CHARIKAR M S. Similarity estimation techniques from rounding algorithms[C]//Proc of the 34th Annual Symposium on Theory of Computing. New York:ACM Press,2002:380-388.

[17] CouchDB[EB/OL]. (2011). http://couchdb.apache.org/docs/overview.html.

(上接第 468 页)

参考文献:

[1] WANG Zheng, LI Jian-wen, ZHAO Yong-xin, *et al*. SPARDL: a requirement modeling language for periodic control system[C]//Proc of the 4th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Berlin: Springer, 2010:594-608.

[2] Design Science. MathType. http://www.dessci.com/en/products/mathtype[EB/OL].

[3] LaTeX. http://www.latex-project.org/[EB/OL]. (2010-01-10).

[4] ANTLR. http://www.antlr.org/[EB/OL].

[5] HAMMOND K,MICHAELSON G. Hume: a domain-specific language for real-time embedded systems[C]//Proc of the 2nd International Conference on Generative Programming and Component Engineering. New York:Springer-Verlag,2003:37-56.

[6] HEITMEYER C. Using the SCR* toolset to specify software requirements[C]//Proc of the 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques. Washington DC: IEEE Computer Society,1998:12-13.

[7] HAREL D. Statecharts: a visual formalism for complex systems[J]. Science of Computer Programming,1987,8(3):231-274.

[8] MARANINCHI F,RÉMOND Y. Mode-automata: a new domain-specific construct for the development of safe critical systems[J]. Science of Computer Programming,2003,46(3):219-254.

[9] HENZINGER T A,HOROWITZ B,KIRSCH C M. Giotto: a time-triggered language for embedded programming[J]. Proceedings of the IEEE,2003,91(1):84-99.

[10] SMITH M,HAVELUND K. Requirements Capture with RCAT[C]//Proc of the 16th IEEE International Requirements Engineering Conference. Washington DC: IEEE Computer Society, 2008: 183-192.

[11] Spin model checker[EB/OL]. http://spinroot.com/.

[12] ALUR R,IVANCIC F,KIM J,*et al*. Generating embedded software from hierarchial hybrid models [J]. ACM SIGPLAN Notices, 2003,38(7):171-182.