构件化嵌入式软件设计模型非功能性质验证的工具实现

徐丙凤[1] 胡 军[1,3] 曹 东[2] 黄志球[1] 郭丽娟[1] 张 剑[1]
(南京航空航天大学信息科学与技术学院 南京 210016)[1] (南京航空航天大学自动化学院 南京 210016)[2]
(南京大学计算机软件新技术国家重点实验室 南京 210093)[3]

摘 要 嵌入式软件的非功能性质是系统高可靠性的重要构成部分。传统的嵌入式软件可靠性保障技术主要关注于系统开发后期,缺乏有效工具对系统设计的非功能性质进行分析与验证。对基于接口自动机模型的构件化嵌入式软件设计验证原型工具T-CBESD(Tool for Component-based Embedded Software Designs)进行了资源及能耗等非功能性质验证功能的扩展设计与实现,包括:资源接口自动机和能耗接口自动机模型的输入输出接口设计、UML顺序图模型的预处理、带非功能语义信息的组合系统状态空间数据结构的设计、非实时资源使用性质与实时相关能量消耗特征验证算法的实现,以及一个通信构件组合系统的实例应用分析。

关键词 嵌入式软件设计,非功能性质验证,构件化设计,软件验证工具,接口自动机

Tool Implementation of Non-functional Verification for Component-based Embedded Software Designs

XU Bing-feng[1] HU Jun[1,3] CAO Dong[2] HUANG Zhi-qiu[1] GUO Li-juan[1] ZHANG Jian[1]
(College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[1]
(College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)[3]

Abstract Non-functional properties of the embedded software system are considered as one of the important features for the high reliability assurance of whole system. Traditional reliability methods in embedded computing domain mostly concern the functional implementation and testing phrase, without effective tools supporting the analysis and verification of the system designs, especially for the non-functional properties. In this paper, a prototype T-CBESD(Tool for Component-Based Embedded Software Designs) was extended with analysis and verification capabilities considering both of resource utilization and energy consumption properties, which include the input/output mechanisms for resource interface automata and energy automata respectively, the pre-translation from a UML sequence diagram to a set of message event sequences, the state space data structure designs with non-functional semantics, the implementation issues of several analysis and verification algorithms for resource and energy consumption properties, and an example of a component-based system design analysis.

Keywords Embedded software design, Non-functional property verification, Component-based design, Software verification tool, Interface automata

嵌入式计算系统已经广泛地应用于生活中的各个领域,如交通、能源、医疗、控制、通信、军事等。近年来随着计算机硬件性能的不断提高,嵌入式系统中软件的规模和复杂性不断增加,使软件对整个系统的影响逐渐占据了统治地位。关键系统中的嵌入式软件失效将会导致生命与财产的重大损失。与一般软件系统只考虑功能特征相比,嵌入式软件必须同时还要满足其在非功能方面的性质要求(如:实时性、系统资源的可使用性、能量消耗特征等)。复杂嵌入式软件中各类非功能属性的分析与验证已成为当前嵌入式计算领域的一个研究热点[1-4]。目前工业界常用的嵌入式软件测试和调试方法(如:在处理器中嵌入ICE功能、调试代理软件、JTAG模拟等)主要用于检查系统功能性质;同时,从软件工程角度来看,这些方法都是在系统的开发中后期使用,而在嵌入式软件设计与分析的前期还缺乏有效的方法和工具来对系统设计的非功能性质进行分析与验证。

本文基于接口自动机模型对构件化嵌入式软件设计(Component-Based Embedded Software Designs, CBESD)的分析与验证方法展开进一步研究,对一个原型工具T-CBESD

到稿日期:2009-09-04 返修日期:2009-11-27 本文受航空基金(2007ZD52043),教育部博士点基金(20070287052)项目资助。
徐丙凤(1986-),女,硕士生,CCF学生会员,主要研究方向为软件工程、嵌入式软件建模与分析,E-mail:xbfnuaa@yahoo.cn;胡 军(1973-),男,副教授,CCF会员,主要研究方向为软件分析与验证、嵌入式系统、形式化方法;曹 东(1972-),男,副研究员,主要研究方向为复杂嵌入式系统、航空控制工程、软件工程;黄志球(1965-),男,教授,博士生导师,CCF高级会员,主要研究方向为软件工程;郭丽娟(1986-),女,硕士生,CCF学生会员,主要研究方向为软件工程、UML建模与分析;张 剑(1985-),男,硕士生,CCF学生会员,主要研究方向为嵌入式系统、多核计算。

(a Tool for CBESD)[5]进行了系统资源和能耗等非功能性质验证方面的扩展。T-CBESD是我们设计的一个基于Eclipse开放平台的构件化嵌入式软件设计分析与验证工具,在相关工作[5]中已完成了基于场景式规约的系统非实时以及实时行为分析与验证功能部分的实现。本文工作则主要是进一步完成对系统非实时资源使用性质以及与时间相关的能耗性质这两方面的分析与验证功能的实现,使得T-CBESD的验证框架更为完整。工具最终目的是可以应用于构件化嵌入式软件开发的设计建模阶段;不仅可以对设计者所关心的关键功能性质进行验证,同时也可以对重要的系统资源以及与能耗相关的非功能性质进行严格形式化分析和验证,从而提高系统可靠性的可信度。

本文第1节给出了针对系统设计的资源以及能耗验证的理论基础,包括:描述系统动态行为的资源以及能耗接口自动机模型、基于场景的系统规约描述模型以及形式化分析与验证算法等;第2节中给出了对T-CBESD扩展非功能性质验证模块的基本设计与实现思想,包括:资源及能耗接口自动机模型的输入输出接口设计、状态空间数据接口设计、基于场景的系统规约模型的输入预处理、具体验证算法的设计与实现等;第3节给出了应用实例;第4节给出了相关研究工作;最后对本文工作的意义以及进一步的工作进行了简要讨论。

1 理论基础

软件工程中的构件化设计方法学通过复用和组合软件模块来构造系统,从而提高系统开发效率和可靠性。通常,一个复杂的嵌入式系统由多个计算子系统构成,其软件系统也具有较高的构件化特征,因此,构件化的设计已成为解决嵌入式软件设计复杂性问题的一种手段。与此同时,构件接口之间的交互行为场景也成为体现系统行为复杂性的一个重要方面。

与一般软件系统相比较,嵌入式软件在系统可利用资源方面受到严格约束,使得系统设计开发过程中需要对非功能性质进行合理有效的表达和检验。具体地,在构件化嵌入式系统的设计方法中,人们希望可以通过构件的接口来提供与资源相关的信息,并且能够使用这些资源信息对系统行为进行非功能性质方面的分析和检查。此外,由于许多嵌入式系统都是使用有限能源为系统供电,使得能量成为一类特殊的资源消耗。从本质上看,能量消耗与系统运行时间紧密联系。运行时间越长,能量消耗就越多;并且随着时间的推移,系统中可使用能量逐渐减少,属于不可回收的资源。因此,需要对非实时资源使用相关性质以及与时间相关能耗性质的分析与检验方法分别进行研究。

1.1 资源接口自动机模型

接口自动机(Interface Automata,IA)[6,7]是用来刻画软件构件接口交互行为时序特征的一种形式化语言。它描述了一个构件被使用时其对环境的输入假设和输出保证。资源接口自动机(Resource Interface Automata,RIA)[8]对接口自动机模型进行了系统资源语义信息的扩展。考虑到嵌入式系统中常见的资源如:内存、消息队列、缓冲区、传感器以及感应器等都具有可量化、独占使用以及可回收再使用等特征,在RIA模型中引入了资源特征向量来描述构件接口在状态变化过程中对资源使用的数量以及资源动态使用归还的特征。目前,在模型设计抽象层次上,我们主要关心的问题包括:整体行为的资源可满足性,即在给定的资源约束条件下,组合系统中所有的交互行为是否都满足资源限制;指定行为的资源可满足性,即给定资源约束,系统某个重要交互行为是否满足限制。

资源接口自动机模型的形式定义如下。

定义1 一个资源接口自动机是一个多元组:$P=(V_P, A_P, F_P, W_P, \Gamma_P)$,其中:

1)V_P是有穷的状态集,每一个状态$v_i \in V_P$,且P的初始状态表示为v_P^{Init};

2)A_P是一个有穷的动作集,包括Input,Output和Internal 3种动作集合,即$A_P=A_P^I \cup A_P^O \cup A_P^H$,且$A_P^I$,$A_P^O$和$A_P^H$互不相交;

3)F_P是一个有穷的映射集,$F_P=(f_P^1, f_P^2, \cdots, f_P^K)$($K$为正整数,表示系统中可使用资源的种类数);其中每一个映射$f_P^k: V_P \mapsto R^+ \cup \{0\}$是将$V_P$中任意一个状态都映射为一个非负整数。用$F_P(v_i)$表示有序集$\langle f_P^1(v_i), f_P^2(v_i), \cdots, f_P^K(v_i)\rangle$;

4)W_P是一个有穷集,每一元素形如$w_i=\{(v_i, F_P(v_i)) \mid v_i \in V_P\}$,简记为$W_P=(V_P, F_P(V_P))$,其中$w_P^{Init}=\{(v_P^{INIT}, F_P(v_P^{Init})) \mid v_P^{Init} \in V_P\}$;

5)Γ_P是转换集,$\Gamma_P \subseteq W_P \times A_P \times W_P$。

称W_P中的每一个w_i为一个资源状态,有序数对集$F_P=(f_P^1, f_P^2, \cdots, f_P^K)$就是状态$v_i$上的资源特征向量。

我们使用资源接口自动机网络(RIA-Networks)作为构件化嵌入式软件系统中带资源约束信息的交互行为模型。系统中每一个构件的资源使用行为使用一个相应的资源接口自动机来表示,整体行为通过构件接口之间的共享动作进行同步组合。其形式定义如下。

定义2 RIA-Networks是一个二元组$N=(Q, Z)$,其中:

1)$Q=\{P_1, P_2, \cdots, P_n\}$为可组合的自动机集;

2)$Z=\{shared(P_i, P_j) \mid 1 \leqslant i, j \leqslant n, i \neq j\}$为所有的共享动作集,其中任何$P_i$和$P_j$的共享动作集为$shared(P_i, P_j)=A_{P_i} \cap A_{P_j}=(A_{P_i}^O \cap A_{P_j}^I) \cup (A_{P_i}^I \cap A_{P_j}^O)(1 \leqslant i, j \leqslant n, i \neq j)$。

RIA-Networks的状态集、动作集等定义如下。

定义3 设$N=(Q, Z)$是一个RIA-Networks,其中$Q=\{P_1, P_2, \cdots, P_n\}$,且$P_i=(V_{P_i}, A_{P_i}, W_{P_i}, \Gamma_{P_i})(1 \leqslant i \leqslant n)$,则$N$中的组合状态集和动作集定义如下:

1)N的每一个不带资源向量的组合状态都形如$\bar{v}=(v_1, v_2, \cdots, v_n)$,其中,$v_i \in V_{P_i}$,$1 \leqslant i \leqslant n$。组合状态的集合为$V_N=V_{P_1} \times V_{P_2} \times \cdots \times V_{P_n}$,其中,无资源向量的初始组合状态为$\bar{v}_N^{Init}=(v_{P_1}^{Init}, v_{P_2}^{Init}, \cdots, v_{P_n}^{Init})$;

2)N的每一个资源组合状态都形如$\bar{w}=(\bar{v}, F_N(\bar{v}))$,其中,$F_N(\bar{v})$是一个资源向量标记$\langle f_N^1(\bar{v}), f_N^2(\bar{v}), \cdots, f_N^K(\bar{v})\rangle$,对于每一个$i(1 \leqslant i \leqslant K)$,有$f_N^i(\bar{v})=\sum_{j=1}^n f_{P_j}^i(v_j)$;初始的组合状态为$\bar{w}_N^{Init}=(\bar{v}_N^{Init}, F_N(\bar{v}_N^{Init}))$;$N$中所有的资源组合状态集记为$W_N$;

3)N的动作集为$A_N=A_N^I \cup A_N^O \cup A_N^H$,其中,输入动作集为$A_N^I=(\bigcup_{1 \leqslant i \leqslant n} A_{P_i}^I)/Z$,输出动作集为$A_N^O=(\bigcup_{1 \leqslant i \leqslant N} A_{P_i}^O)/Z$,内部动作集为$A_N^H=(\bigcup_{1 \leqslant i \leqslant n} A_{P_i}^I) \cup Z$。

定义4 $N=(Q, Z)$是一个RIA-Networks,其中$Q=\{p_1, p_2, \cdots, p_n\}$,$P_i=(V_{P_i}, A_{P_i}, F_{P_i}, W_{P_i}, \Gamma_{P_i})(1 \leqslant i \leqslant n)$;$\bar{w}=(\bar{v}, F_N(\bar{v}))$和$\bar{w}'=(\bar{v}', F_N(\bar{v}'))$是$N$中两个不同资源的组合

状态，其中，$\bar{v}=(v_1,v_2,\cdots,v_n)$；$\bar{v}'=(v_1',v_2',\cdots,v_n')$；$F_N(\bar{v})$与$F_N(\bar{v}')$是相应的资源向量标记。当满足以下条件之一时，系统可以通过动作a从状态$\bar{w}$到达状态$\bar{w}'$，用$\bar{w}\xrightarrow{a}\bar{w}'$来表示：

1)对于一个动作$a\notin Z$，在P_i中存在一个转换$(w_k,a,w_k')\in\Gamma_{P_i}$，其中$w_k=(v_k,F_{P_i}(v_k))$，$w_k'=(v_k',F_{P_i}(v_k'))$，且对于任何其它的$l(l\neq k,1\leqslant l\leqslant n)$，有$w_l=w_l'$；

2)对于一个动作$a\in shared(P_i,P_j)(1\leqslant i,j\leqslant n,i\neq j)$，在$P_i$中存在一个转换$(w_i,a!,w_i')\in\Gamma_{P_i}$($a!$表示$a$是输出动作)；同时，在$P_j$中存在一个转换$(w_j,a?,w_j')\in\Gamma_{P_j}$($a?$表示$a$是输入动作)，且对于任何的$k(k\neq i,j,1\leqslant k\leqslant n)$有$w_k=w_k'$。

N为一个资源接口自动机(RIA-Networks)，N的一个行为就是一个资源状态转换序列，形如$\bar{w}_0\xrightarrow{a_0}\bar{w}_1\xrightarrow{a_1}\cdots\xrightarrow{a_{m-1}}\bar{w}_m$，其满足：1)$\bar{w}_0=\bar{w}_N^{init}$；2)对于每一个$i(0\leqslant i\leqslant m)$，有$(\bar{w}_i,a_i,\bar{w}_{i+1})\in\Gamma_N$。

1.2 能耗接口自动机模型

软件系统的能耗本质上是运行过程中所依赖的底层物理部件的能量消耗，通常的能耗分析都基于底层特定的硬件架构。由于人们最终关心的是当嵌入式软件完成某功能时的能耗性质，因此我们希望可以从设计模型这个抽象层次来描述、分析和检验系统的一些重要能耗性质，而不用涉及底层种类繁多的硬件平台信息。能耗接口自动机模型(Energy Interface Automata，EIA)[9,10]在实时接口自动机(Real-Time Interface Automata，RTIA)[11]的基础上进行了系统能量消耗语义的扩展。考虑到目前嵌入式系统中的许多硬件设备都具备多种能耗运行模式(如：忙碌、空闲、休眠等)，则从模型设计层面上可以认为嵌入式软件在不同的系统运行状态上能够具有不同的单位时间能量消耗率。因此，在EIA模型中引入了状态功耗因子来描述系统状态变化过程中的能量消耗特征。在本文工作中主要关心能耗的边界值分析与验证问题，包括：最少能耗计算，即系统到达某个特定状态时至少需要消耗多少能量；最大能耗验证，即在给定的时间约束下，到达某个特定状态时，是否超过预设的能耗限制。

能耗接口自动机模型的形式定义如下。

定义5 一个能耗接口自动机是一个五元组：$p=(V_P,F_P,A_P,I_P,\Gamma_P)$，其中：

1)V_P是有穷的状态集，每一个状态$v\in V_p$，且P的初始状态表示为$v_P^{init}\in V_P$；

2)F_P是一个映射(函数)：$V_P\mapsto R^+$；

3)A_P为有穷的动作集合，包括输入、输出和内部动作集：A_P^I，A_P^O和A_P^H，且三者交集为空；

4)I_P为有穷的时间区间集，每个时间区间都形如$[x,y]$，其中x,y为非负整数，且$x\leqslant y,y\neq\infty$；

5)$\Gamma_P\subseteq V_P\times A_P\times I_P\times V_P$为有穷的转换集，其中每一个元素都形如$(v_i,a_i,[x_i,y_i],v_j)$。

上述定义中的映射(函数)F_P将EIA中的每一个状态v映射到正实数集合中的某一个数e，即：$F_p(v)=e,e\in R^+$，其语义为每一个状态都赋予一个相应的功耗因子(单位时间能量消耗率)。如上所述，每个状态的功耗因子可以是不一样的。

与RIA类似，可以给出EIA-Networks的组合状态、组合行为等的形式定义，详细内容见参考文献[9,10]。

1.3 基于场景的交互行为规约

在原型工具T-CBESD中主要使用UML的顺序图模型及其扩展形式的交互概观图模型[12]来描述系统设计中的场景式交互行为规约说明。通常，一个相对独立的系统功能模块建模为一个场景描述；这个场景表达了参与其中的各构件之间如何进行交互的特征。在相关工作中[13,14]我们详细讨论了不同种类的场景式规约，本文以下只简要给出基本顺序图模型行为的形式化定义。

定义6 一个顺序图是五元组$D=(C,E,M,L,V)$，其中：

1)C：有穷的构件集；

2)E：有穷的事件集；

3)M：有穷的消息集；对每一个消息$m\in M$，分别使用$m!$和$m?$来表示消息m的发送和接收事件；

4)$L:E\to C$是一个标记函数，将每一个事件$e\in E$分配给一个构件$L(e)\in C$；

5)V：元素形式为(e,e')的有穷集，其中$e,e'\in E$，且$e\neq e'$，表示D中的每一个可视序列对。

任意一个事件$e\in E$，都对应一个消息$m\in M$的发送和接收事件，即$\tau(e)=m!$或者$\tau(e)=m?$。顺序图刻画了系统运行的一个场景，其运行过程就表现为一个消息事件的序列$e_0^{\wedge}e_1^{\wedge}\cdots e_m$，其中事件$e_{i+1}$在事件$e_i$之后发生$(1\leqslant i\leqslant m-1)$。此外，由于在顺序图中可能存在不确定的接收消息的先后次序，因此一个顺序图场景可能会包含多个不同的消息事件序列。

在顺序图形式化定义的基础上，可以通过附加消息事件的时间标记和时间布尔表达式来进一步描述系统实时行为的时间约束需求，其中布尔表达式形如$c_0(t_{e_0}-t_{e'_0})+c_1(t_{e_1}-t_{e'_1})+\cdots+c_n(t_{e_n}-t_{e'_n})\sim b;t_{e_0},t_{e'_0},t_{e_1},t_{e'_1},\cdots,t_{e'_n}$表示每个消息事件发生时的时间值，$c_0,c_1,\cdots,c_n,b$为实数，且$b\neq\infty$，$\sim\in\{\leqslant,<\}$。

1.4 模型分析与验证算法

基于以上给出的资源接口自动机模型、能耗接口自动机模型以及交互场景规约模型，在相关工作[9,10]中分别针对非实时资源使用以及实时相关能量消耗两方面的问题进行了严格形式化定义和分析，设计了相关的验证算法。算法的基本思想是对带有不同语义信息的系统组合行为的状态空间进行搜索；将每一个可能的系统行为与基于场景的交互规约进行比较，判断设计模型是否满足各类场景式系统功能规约；然后对符合要求的系统行为的资源使用以及能量消耗特征进行进一步的分析与验证。例如，对于系统整体行为的资源使用可满足性问题，其相应的抽象验证算法框架如图1所示。

```
current_path:={s0}; B:=⟨b1,b2,...,bk⟩;
X:=⟨0,0,...,0⟩;abnormal:=Φ;satisfied:=true;
repeat
    node:=取current_path中的最后一个节点
    if node的后继节点已访问过
    then删除current_path中的最后一个节点
    else begin node:=取node的一个未访问的后继节点
              for i=0 to K-1
                  if    then satisfied:=false;
                  if satisfied=false then将node加入到abnormal集中;
                  将node加入到current_path中;
         end
until current_path={};
if abnormal:=Φ then retrun  true else return false  .
```

图1 检验资源可满足性算法

此外，由于能耗自动机模型是建立在实时自动机模型的

基础上，因此，对于系统能耗方面的验证算法，还需要考虑由于时间与能耗语义引入所带来的将连续时间进行整型化处理的方法，以及带时间约束与能耗率的投影路径的建立。图 2 为 EIA-Network 的最少能耗路径计算的抽象算法框架。其中所提到的投影路径则是为了处理状态空间中因环路的出现而导致所检验的系统行为路径可能是无穷长度的问题。

```
min_omega := ∞; ε:= 0; //min_omega记记录最小能耗
min_omega _path:= Φ; //记录最少能耗路径信息，初始集为空
current_path:={s0};
repeat
    node=current_path的最后一个结点;
    if node 没有先的后继结点then 删除current_path中的最后一个节点;
    else begin node:=node的下一个新的后继节点
        if node=sj
        then begin 计算current_path的能耗值
            if current_path的能耗值<min_omega
            then begin
                min_omega:= current_path的能耗值
                min_omega_path:={ };
                min_omega_path:=current_path ∪ {sj}
            end
        end
    else 将node加入到current_path中;
end
until current_path={ }
```

图 2　搜索最少能耗路径算法

2　非功能性质验证的工具实现

基于以上的理论基础，本文对原型工具 T-CBESD 进行了非功能性质分析方面的扩展，设计并实现了非实时相关资源性质以及实时相关能量消耗这两方面特征的验证功能。其目的是使得在构件化嵌入式软件开发的设计建模阶段，不仅可以对设计者所关心的关键功能性质进行验证，也可以对重要的系统资源与能耗相关的非功能性质进行严格形式化的分析和验证，从而提高系统可靠性的可信度。T-CBESD 的基本设计原则包括两个方面：

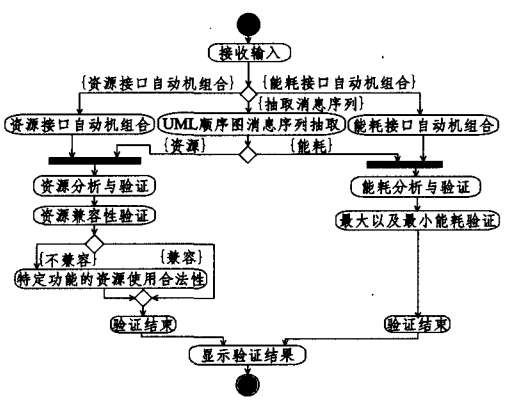

图 3　T-CBESD 的模型分析与验证流程框图

1）跨平台运行、易扩展：即工具应该尽可能在多种不同运行平台上运行。因此，选择了面向对象程序设计语言 Java 作为工具的实现语言。Java 具有良好的跨平台运行特征以及丰富的类库资源，并可以使用面向对象程序设计思想中的类继承等方法对工具进行方便可靠的扩展。

2）易使用、易维护：用户可以比较方便地使用工具，或进行调整；因此，选择了工业界广泛使用的开放集成开发环境 Eclipse 作为工具的运行平台，即使用 Eclipse 的插件（plug-in）技术来设计和开发。用户可以很容易在 Eclipse 环境中通过插件技术来安装、配置和使用工具；同时，在 T-CBESD 的输入输出接口中所使用的 XML 语言在 Java 和 Eclipse 环境中也得到了完全的支持。

T-CBESD 中与非功能性质相关的模型分析与验证的基本处理流程如图 3 所示。主要的逻辑处理框架包括：输入输出接口；UML 顺序图模型的预处理；自动机组合模型的建立；资源相关的分析与验证算法的实现；能耗相关的分析与验证算法的实现等。以下分别给出详细说明。

2.1　输入输出接口设计

T-CBESD 的输入输出都是以 XML 文件形式来描述系统设计模型、系统需求规约以及验证结果信息等。其中，工具的输入包括：描述系统设计的接口自动机模型的 XML 文件和描述系统规约的消息交互序列的 XML 文件；输出则包括：描述系统组合行为的接口自动机组合模型的 XML 文件和包含验证结果信息的 XML 文件。这里，最核心的部分是资源与能耗接口自动机模型的 XML 文件格式的设计。在图 4 中给出了一个资源接口自动机模型的 XML 文件示例说明；通过 XML 的树形标签格式，分别定义了自动机名、自动机个数（如果这是一个组合自动机）、资源种类、资源名称以及数量、状态个数、状态名、后继状态名、所需资源的种类以及数量、转换个数、转换名、转换的出发和到达状态名、转换的出发和到达状态所需的资源种类和数量、动作个数、动作名、动作类型等数据信息，用来完整准确地保存接口自动机模型的语义信息。此外，对于扩展的能耗接口自动机模型，其相应的 XML 文件格式定义中还包含与动作相关联的时间区间约束标记以及能耗标记。

```
<?xml version="1.0" encoding="UTF-8"?>
<automata>
  <automataInfor number="1">                          <!--自动机个数-->
    <name>ReAutomata</name>                           <!--自动机名-->
  </automataInfor>
  <numberOfSourceTypeInfor number="1">                <!--资源信息-->
    <numberOfSourceType>2</numberOfSourceType> <!--自动机资源种类数-->
  </numberOfSourceTypeInfor>
  <sumOfSourceInfor number="1">                       <!--资源和信息-->
    <sumOfSource>[t4],[m4]</sumOfSource>              <!--资源的种类以及个数-->
  </sumOfSourceInfor>
  <statesInfor number="6">                            <!--自动机状态信息-->
    <state>
    <name>s0</name>                                   <!--自动机名-->
    <isInitial>true</isInitial>                       <!--是否为初始状态-->
    <nextStates>s1</nextStates>                       <!--后继状态-->
    <sourceNeed>[t0],[m0]</sourceNeed>                <!--所需资源的种类和数量-->
    </state>
    <state>...</state>
  </statesInfor>
  <transitionsInfor number="8">                       <!--转换的信息-->
    <transition>
    <from>s0</from>                                   <!--出发状态的状态名-->
    <to>s1</to>                                       <!--到达状态的状态名-->
    <action>a?</action>                               <!--转换名-->
    <sourceOfFromState>[t0],[m0]</sourceOfFromState> <!--出发状态资源需求-->
    <sourceOfToState>[t4],[m2]</sourceOfToState>      <!--到达状态资源需求-->
    </transition>
    <transition>...</transition>
  </transitionsInfor>
  <!-- InputAction information of automata  -->       <!--输入动作信息-->
  <inputActionInfor number="3">                       <!--输入动作个数-->
    <inputAction>a</inputAction>                      <!--输入动作动作名-->
    <inputAction>...</inputAction>
  </inputActionInfor>
  <!-- outputAction information of automata  -->      <!--输出动作信息-->
  <outputActionInfor number="2">                      <!--输出动作个数-->
    <outputAction>d</outputAction>                    <!--输出动作动作名-->
    <outputAction>e</outputAction>
  </outputActionInfor>
  <!-- internalAction information of automata  -->    <!--内部动作信息-->
  <internalActionInfor number="0">                    <!--内部动作个数-->
  </internalActionInfor>                              <!--内部动作动作名-->
</automata>
```

图 4　资源接口自动机模型的 XML 文件示例

在上述定义的 XML 文件基础上，就可以使用 Java 类库中的 DOM（文档对象模型）方法很方便地对自动机模型进行解析及生成。例如，在 T-CBESD 中设计了 parseReXmlDocument()和 parseEnXmlDocument()两个类方法来分别对资源接口自动机模型 XML 文件和能耗接口自动机 XML 文件进行解析，并根据 ReAutomata，ReTransition 以及 ReState 等类定义在内存空间创建相应的接口自动机对象。其中，资源接

口自动机的核心类数据结构 ReAutomata，ReTrasition 以及 ReState 的定义如图 5 所示。

```
public class ReAutomata {
        public String name ;                                    //自动机名
        public Vector <String>inputActionVector ;  //输入动作集
        public Vector <String>outputActionVector ; //输出动作集
        public Vector <String>internalActionVector ; //内部动作集
        public Vector <ReState> stateVector;                    //保存自动机状态集合
        public Vector <ReTransition > transitionVector ; //保存自动机转换集合
        int numberOfSourceType ;                        // 定义资源接口自动机的资源的种类
        public Vector <SourceNode> sumOfSource; //定义资源总数，资源名称以及对应的数量
}
public class ReTransition {
        public String actionName ;                      //转换名
        public String from ;                            //出发结点
        public String to ;                              //到达结点
        public Boolean inputAction ;                    //输入动作
        public Boolean outputAction ;                   //输出动作
        public Boolean internalAction ;                 //内部动作
        Vector <SourceNode > sourceOfFromState; //描述出发状态所需资源的数量
        Vector<SourceNode > sourceOfToState ;   //描述到达状态所需的资源数量
}
public class ReState {
        public String name ;                    //状态的名字
        public Boolean isInitial ;              //标志该状态是否为初始状态
        public Vector <String>enabledActions ;  //记录由该状态出发的所有转换的action名
        public Vector <ReState> nextStateVector; //保存该状态的下一状态
        Vector <SourceNode > sourceOfNeed;      //定义所需资源名称以及对应的数量
```

图 5　RIA 的 ReAutomata，ReTransition 和 ReState 类定义

2.2　顺序图模型的输入预处理

虽然 T-CBESD 的输入输出定义为标准 XML 文档格式，但在工具中加入了从 UML 建模环境 Rational Rose 的顺序图模型到 T-CBESD 的 XML 输入文件（描述消息交互序列集）的自动转换处理。其原因有二：

一，现在工业界已存在较为成熟的图形化建模工具，可以快速方便地绘制 UML 模型图，可以利用这些工具作为 T-CBESD 的前端，而不用在 T-CBESD 中重新设计复杂的用户接口来支持图形化建模设计。

二，在 1.3 节中提到，一个顺序图场景可能会包含多个不同的消息事件序列；显然，如果让系统设计与分析人员从每一个顺序图中手动地生成所有可能的消息事件序列，并不是件容易的事情。因此，需要提供一种从顺序图模型自动化生成所有可能的消息事件集合的方法。

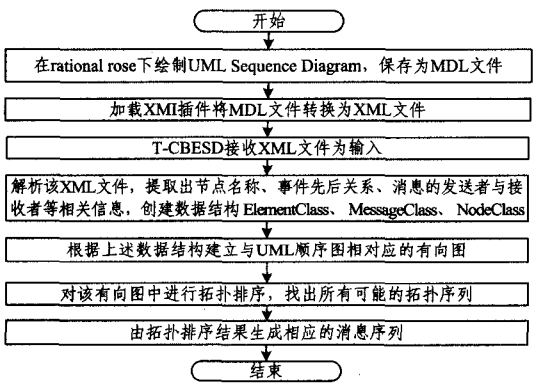

图 6　UML 顺序图消息序列抽取流程

在 Rational Rose 中所生成的顺序图模型文件是 MDL 格式，需要先转换成 XML 格式文件，然后进行相应消息序列的抽取。其处理过程如图 6 所示，首先通过在 Rational Rose 中加载 XMI 插件将 MDL 格式的文件转换为 XML 格式；然后对 XML 文件进行解析，建立文档解析树，提取消息事件节点，并根据顺序图中的事件发生先后顺序构造一个相应的有向无环图（在此，定义了顺序图的参加者类（Element Class）、消息类（Message Class）以及结点类（Node Class）用于图的构造）；最后设计了一个拓扑排序算法，对该有向图中的消息事件节点进行拓扑排序，从而得到一个顺序图中所有可能的消息事件序列的集合。

2.3　组合模型状态空间的建立

接口自动机组合过程与一般自动机组合的语义存在不同之处。在两个接口自动机组合的状态空间中，有可能存在两个构件接口之间交互不同步的所谓的“非法状态”，在应用验证算法之前必须将这些非法状态找出来并从状态空间中去掉。文献[7]中给出了一个识别非法状态集合的基于不动点（Fixpoint）的抽象算法框架，其基本思想是先构造出所有可能的组合状态的空间，然后逆向搜索非法状态集。在 T-CBESD 的实现中则采用正向的合法状态集合构造方法，其好处是避免了需要首先生成所有的状态空间。在资源接口自动机组合的过程中，还需考虑两个状态所需要的资源的种类以及数量的计算。图 7 给出了工具中资源接口自动机组合算法流程图。此外，在能耗接口自动机组合过程中需要考虑时间约束以及能耗率的计算。

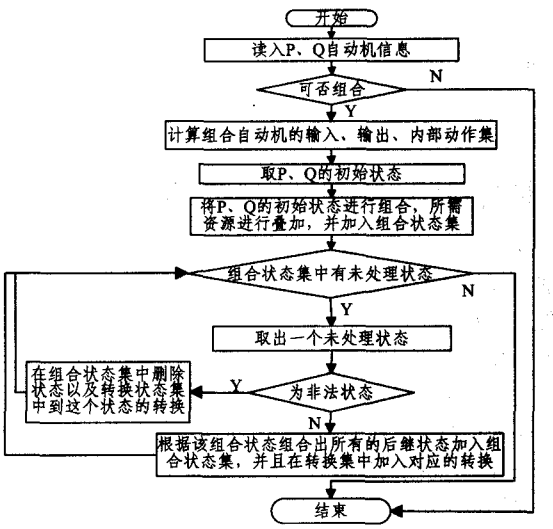

图 7　资源接口自动机组合算法流程图

2.4　资源验证算法的实现

在建立以上的接口自动机组合状态空间的基础上，T-CBESD 中实现了包括资源可满足性验证以及特定功能行为的资源可满足性验证算法。图 8 给出了工具中资源验证模块的类图框架，主要包括两大部分：一部分是自动机模型核心类，包括 ReAutomata class，ReTransition Class，ReState Class 以及组合模型的 ReComposition Class；另一部分则是与验证算法相关的类，包括 ReSatisfiabilityChecking Class，FucSpeSatisChecking Class，以及辅助类 ReActionString Class，ReAdjacentMatrix Class，ReTransitionNode Class。

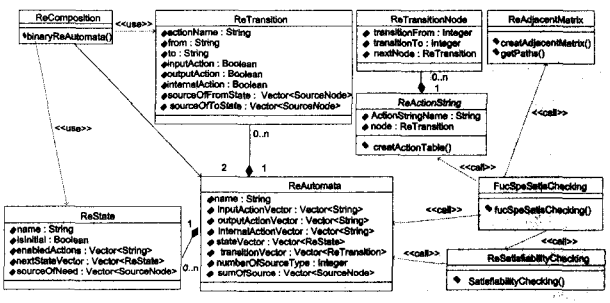

图 8　T-CBESD 中资源验证模块的类图

在 FucSpeSatisChecking 类的实现过程中，一个关键问题是：当在系统组合状态空间中搜索与 UML 顺序图交互序列所对应的投影路径时，有可能出现一个满足条件的投影路径

其一部分出现在某个环内部，而另一部分却出现在此环路的外部路径上的情形[9,10]。如果只采用经典的深度优先遍历或广度优先遍历方法对组合状态空间进行搜索判定，将会遗漏这种情况。为此，设计了动作名表(ReActionString Class)和邻接矩阵(ReAdjacentMatrix Class)。其中，动作名表是以 RIA 的动作名作为表头向量，并以执行该动作名的转换作为表结点的一张哈希表，其定义如图 9 所示(注：未包含类方法说明)。

```
public class ReActionString {                    //表头结点
        public String ActionStringName ; //动作名
        public ReTransitionNode node ;  //执行该动作的一个转换
}
class ReTransitionNode {                         //表结点
        public int transitionFrom ;              //出发状态在状态集中的序号
        public int transitionTo ;                //到达状态在状态集中的序号
        public ReTransitionNode nextNode ;//下一执行该动作的转换
}
```

图 9　动作名表(ReActionString Class)的定义

基本思想为：对于所给出的一个消息交互序列，先根据消息名从动作名表中依次取出与消息名所对应的表头结点以及表结点，构成一张与消息序列中消息次序对应的消息名表。然后遍历这张消息名表来搜索投影路径，搜索过程中需要根据邻接矩阵来判断两个结点之间是否可达。资源验证算法执行的流程框架如图 10 所示。

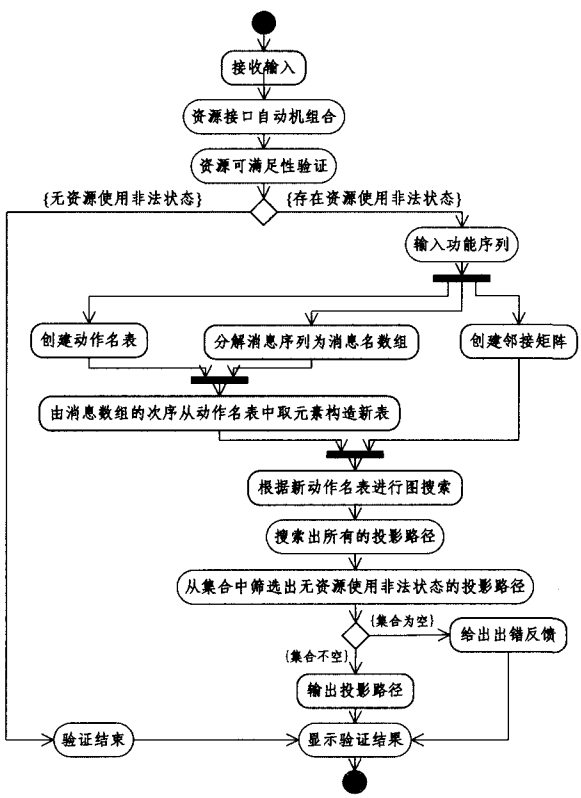

图 10　资源验证活动图

2.5　能耗验证算法的实现

在能耗验证算法的实现中，需要首先搜索出所有与给定的消息序列对应的投影路径。在此基础上对所有搜索到的投影路径进行筛选，找出能耗最大值路径以及能耗最小值路径。能耗验证模块的类图结构与资源验证模块的类图结构类似，验证过程实现的活动图如图 11 所示。图 12 则给出了能耗验证算法中搜索所有投影路径的具体算法。

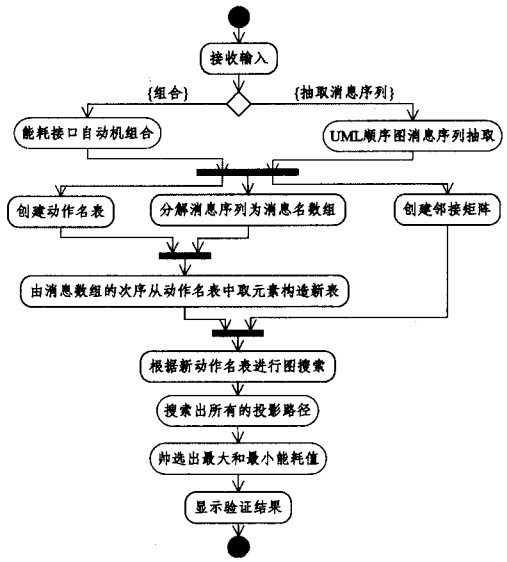

图 11　能耗验证活动图

```
N为消息序列中消息个数
数组A(A为Vector<ActionString>类型)保存消息序列中的动作名在动作名表中所对应的元素
数组T(T为TransitionNode 类型)保存表头元素A[j]的第一个表结点(0≤j<N);
i:=0;
repeat
    if N=1
    then 保存当前动作名对应的所有转换break;
        if i=N-1 /*表明搜索到一条与消息序列对应的投影路径*
        then  将搜索到的路径片段连接为完整的搜索路径并继续找下一条路径
    else begin
        repeat
            if i=0
            then 保存该转换的出发结点
            repeat
                if T[i].transitionTo ,T[i+1].transitionFrom 之间无合法可达路径
                then T[i+1]:=T[i+1].nextNode;
                else begin
                    path[i]保存该合法可达路径
                    break;
                end
            until T[i+1]=null
            if T[i+1]=null
            then begin
                将已保存的部分路径删除
                T[i+1]:=A[i+1]的第一个表结点
                T[i]:=T[i].nextNode;
                if i>0 then i--;
            end
            else begin
                i++;
                if i=N-1
                then 将搜索到的路径片段连接为完整的搜索路径继续找下一条路径
            end
        until T[i]=null
        if T[i]=null then break;
    end
until i>=N
```

图 12　能耗验证实现算法

3　实例应用

T-CBESD 的设计开发和运行环境是：Windows XP 操作系统平台，Eclipse SDK 3.4.0，Java SDK 1.6，所引用的 MDL 文件是由 Rational Rose 2003 生成的。本节中分别给出 T-CBESD 的与时间无关的资源性质验证和与时间相关的能耗性质验证两方面的实例应用说明。

首先，对文献[7]中所给出的两个通信构件接口自动机模型增加资源使用语义信息的描述，如图 13 所示。图 13(a)的接口自动机模型描述了构件 Communication 与环境交互的行为特征：当 msg 被外部调用时，构件将先调用 send 方法与其他的通信信道构件交互，并获得从信道返回的 ack；然后向调用者返回通信成功：ok；若连续两次 send 动作都收到 nack，则返回通信失败：fail。Communication 一共拥有两类资源，资源名称分别为 r1 和 r2，r1 和 r2 的总数量都为 4，用资源向量描述为〈[r1,4],[r2,4]〉。状态 s1 上的资源向量〈[r1,4],[r2,2]〉表示在状态 s1 上需要 4 个 r1 类资源和 2 个 r2 类资源；图

13(b)的接口自动机模型描述了一个 Communication 构件的使用者:User 的行为特征,User 拥有 r1 和 r3 两类资源,总数量分别为 2 和 3,用资源向量描述为〈[r1,2],[r3,3]〉;图 13(c)的顺序图模型则给出了这两个构件的组合系统的一个交互行为规约说明。

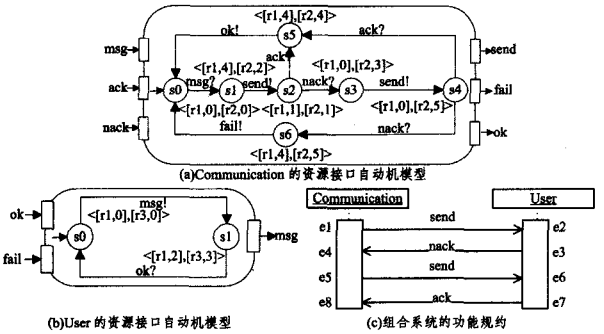

图 13 通信构件资源接口自动机模型及系统功能规约

在 T-CBESD 中所构造的构件 Communication 和 User 的合法组合状态空间(去除了非法状态集)包括 6 个组合状态:(s0|s0),(s1|s1),(s2|s1),(s3|s1),(s4|s1),(s5|s1)。组合系统所拥有的资源种类以及数量应该为两个资源接口自动机资源种类与数量的叠加,用资源向量描述为〈[r1,6],[r2,4],[r3,3]〉。基于该组合空间进行资源方面的验证,首先进行资源可满足性验证,验证结果显示有一个资源使用非法状态:s4|s1,其资源需求量为[r1,2],[r2,5],[r3,3]。s4|s1 状态对资源 r2 的需求量是 5,而组合系统中 r2 资源的总数量为 4,判定该状态不满足资源限制。因此,从整体行为来观察,系统不满足资源约束。其次,挑选特定功能行为来进行资源使用合法性验证。示例中的特定功能使用图 13(c)UML 顺序图所描述的消息交互序列来描述。验证结果显示在组合状态空间中确实存在一条与该功能序列相符合的投影路径,但该路径中仍然存在资源使用非法状态。这意味着该交互行为在组合系统中是不满足资源限制的,需要设计者进行系统设计模型修改。图 14 为 T-CBESD 的资源验证模块的界面。界面左边部分为操作区,主要提供组合、查看、工具输入、验证类型等功能选择,界面右边为分析与验证过程中工具所反馈的数据信息。

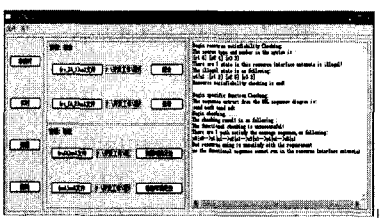

图 14 T-CBESD 资源验证界面

图 15 中给出了一个上述通信构件接口自动机的带能耗信息的版本,其中 Communication 和 User 的每一条转换边上都添加了一个时间区间标记,用以表达系统状态转换的时间约束需求;并且在每一状态上都添加了一个浮点数,用以标记状态上的能量消耗率。例如,状态 s0 上的 1.2 表示在该状态上的能耗消耗率为 1.2 个单位。s2→s3 上的时间标记[1,2]表示构件在 s2 上至少停留 1 个时间单位,至多停留 2 个时间单位,即从状态 s2 到状态 s3 的转换必须在此时间区间发生;其他依次类推。经过T-CBESD中能耗接口自动机的组合构造,去除掉交互行为的非法状态集以及与时间相关的非法状态集之后,可以得到 22 个带时间标记的合法组合状态。

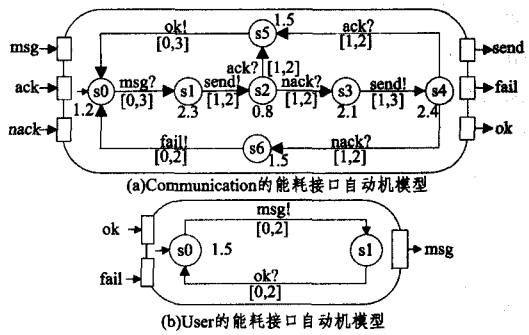

图 15 通信构件能耗接口自动机模型

根据图 13(c)所给出的功能规约,T-CBESD 在验证的过程中,首先搜索出符合非实时功能行为的路径一共有 19 条。然后对这 19 条路径当中的能耗值进行计算比较。验证结果显示,其中能耗最大的为 45.2,对应的投影路径为 s0|s0(0 0)2.7-->s1|s1(0 0)5.4-->s2|s1(0 2)3.9-->s3|s1(0 4)5.2-->s4|s1(0 7)5.5-->s5|s1(0 9)4.6,能耗值最小的为 29.1,对应的投影路径为 s0|s0(0 0)2.7-->s1|s1(0 0)5.4-->s2|s1(0 1)3.9-->s3|s1(0 3)5.2-->s4|s1(0 5)5.5-->s5|s1(0 9)4.6。

图 16 所示为 T-CBESD 的能耗验证模块插件的界面。类似地,界面左边部分为操作区,主要提供组合、查看、工具输入、验证类型等功能选择,界面右边部分为分析与验证过程中工具所反馈的数据信息。

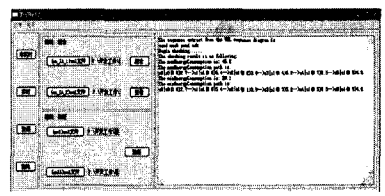

图 16 T-CBESD 能耗验证界面

4 相关研究工作

与接口自动机相关的工具包括 Chic[15],SPIN,UPPAAL 和 OpenEmbeDD。Chic 是第一个基于接口自动机理论的原型工具。它是作为 Jbuilder 集成环境下的一个插件模块来设计开发的,其目的只是用于对相关理论工作的一个初步验证。Chic 中包括了接口自动机基本模型的一个扩展模型 Resource Interface(RI)[16]的实现,但 RI 模型中只是通过在非实时 IA 的状态上附加整型数值来表达状态的资源消耗数量,语义描述能力不够;在后续的一个工具 Ticc[17,18]中已经没有 RI 的相关工作。目前 Chic 已经被 Ticc 所替代。Ticc 的理论基础是接口自动机的一个最新扩展版本:Sociable Interface[19],其基本思想仍然是检验构件接口组合中非实时以及实时交互行为是否兼容。Ticc 中并未包括资源和能耗等非功能性质的验证。Ptolemy II[20,21]中也实现了基本接口自动机模型的组合兼容性分析工作,不过 Ptolemy II,一个包含了多种不同工具集的混成系统建模、分析、合成和代码生成的开发环境。此外,SPIN[22]是一个经典的分布式系统模型检验工具。系统设计模型是否满足规约性质是通过组合系统和时序公式相对应的 Buchi 自动机进行同步积,然后检验其结果是否为空。目前 SPIN 在工业界硬件设计以及通信协议规约的验证领域得到了较广泛的应用,但对同时具有功能和非功能需求的嵌入式软件验证领域,SPIN 并未提供相应的支持。UPPAAL[23,24]是一个基于时间自动机理论[25]的实时系统仿真和验证工具。其基本思想是将实时系统的行为建模为一个实时自动机网络,并进行了数据类型的扩展,采用时间 μ-算

子作为系统的规约语言，主要对系统进行安全性和活性等性质的检验。UPPAAL 具有良好的图形化编辑和模拟功能。目前，已有一些工具以 UPPAAL 为核心作进一步扩展，如：TIMES[26,27]是以时间自动机模型验证为基础的一个工具集环境，可以进行建模、可调度分析、系统合成以及特定平台上的代码生成；Save-IDE[28]则是基于构件模型 SaveCCM[29]所建立的一个支持构件化嵌入式系统开发的工具集，等等。OpenEmbeDD[30]则是以法国 INRIA 为中心的欧洲多个研究机构正在构建的一个以 Eclipse 为开放平台的模型驱动嵌入式系统开发工具集。这是一个庞大的开源工具组合环境，可提供嵌入式系统设计（包括软件和硬件）、模拟、验证、合成以及测试等各个阶段的开发支持。

结束语 本文对一个构件化嵌入式软件设计分析与验证的原型工具 T-CBESD 进行了非功能性质验证方面的扩展，包括：系统非实时资源使用性质以及与时间相关的能耗性质两方面的分析与验证功能的设计与实现；使得 T-CBESD 的验证框架更为完整。论文主要内容包括：资源属性验证以及能耗属性验证的理论基础，T-CBESD 的基本设计思想，工具的输入输出接口、状态空间数据结构、验证算法等的设计与实现，以及应用实例分析。

进一步的工作包括以下几个方面：

1）扩展工具的输入和输出接口形式。使用 Rational Rose 等图形化建模环境作为前端工具来方便用户进行系统各类接口自动机模型的设计。资源与能耗接口自动机与 UML 状态机的语法和语义存在不同之处，需要重新设计一个中间转换过程来处理。输出方面，将设计更为完整的验证结果信息 XML 文件格式，并考虑与软件测试技术相结合，利用验证结果给出的系统出错（验证失败）行为轨迹来指导生成相应的测试用例。

2）应用复杂的多构件系统设计实例，来对工具的性能进行检验和提高。目前所运行的实例相对比较简单，主要目的是为了检查实现算法的正确性，还需要在复杂多构件系统模型情形下对算法性能作进一步的检验和改进。现在我们正在对某飞行控制软件系统进行模型分析和抽取工作，准备将其作为 T-CBESD 的一个复杂实例验证。

3）进一步完善原型工具的操作界面。目前本文的工作主要关注于基本的输入输出处理和核心算法的设计与实现，还需要考虑工具用户界面接口设计的实用性和有效性。此外，我们正在整理相关的文档和源码，准备建立一个 Wiki 网站以开放源码的形式来对工具进行维护和进一步开发。

参 考 文 献

[1] Lee E A. Cyber physical systems: design challenges[C]// Proceedings of 11th IEEE International Symposium on Object-oriented Real-time, Distributed Computing (ISORC 2008). 2008: 363-369

[2] Lee E A. Embedded software[J]. Journal of Advances in Computers, 2002, 56: 56-97

[3] Lee E A. What's ahead for embedded software? [J]. IEEE Computer, 2000, 33(9): 18-26

[4] Peled D A. Software reliability methods[M]. Berlin: Springer-Verlag, 2001

[5] 徐丙凤，胡军，曹东，等. T-CBESD：一个构件化嵌入式软件设计模型验证工具[J]. 小型微型计算机系统，2009

[6] Alfaro D L, Henzinger T A. Interface theories for component-based design[C]// Proceedings of Embedded Software, First International Workshop, EMSOFT 2001, LNCS 2211. Berlin: Springer-Verlag, 2001: 148-165

[7] Alfaro D L, Henzinger T A. Interface automata [C]// Proceedings of the Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (ESEC/FSE 01). 2001: 109-120

[8] 胡军，黄志球，曹东，等. 网构软件的资源自适应性的形式化分析与验证[J]. 软件学报，2008, 19(5): 1186-1200

[9] Hu J, Li X D, Zheng G L, et al. Modeling and analysis of power consumption for component-based embedded software[C]// Proceedings of the EUC workshops, LNCS 4097. Berlin: Springer, 2006: 795-804

[10] 曹东，胡军，徐丙凤. 构件化嵌入式软件设计的能耗性质分析与验证[J]. 南京理工大学学报，2009, 33(1): 26-31

[11] Hu J, Yu X F, Zhang Y, et al. Checking component-based embedded software designs for scenario-based timing specification [C]// Proceedings of the EUC 2005, LNCS 3824. Berlin: Springer, 2005: 395-404

[12] OMG. UML 2.0 Superstructure specification[OL]. OMG document ptc/05-07-04. http://www.uml.org, August 2005

[13] Hu J, Yu X F, Zhang Y, et al. Scenario-based verification for component-based embedded software systems[C]// Proceedings of 2nd embedded computing workshop (ICPP-EC05). IEEE Computer Society Press, 2005: 240-247

[14] 胡军，于笑丰，张岩，等. 基于场景规约的构件式系统设计分析与验证[J]. 计算机学报，2006, 29(4): 513-525

[15] Chic. http://www.eecs.berkeley.edu/~arindam/Chic/. 2006-08-13

[16] Chakrabarti A, de Alfaro L, Henzinger T A, et al. Resource Interfaces[C]// Proceedings of Third International conference on embedded software, LNCS 2855. Berlin: Springer, 2003: 117-133

[17] Ticc. http://dvlab.cse.ucsc.edu/Ticc. 2008-05-15

[18] Adler B T, Alfaro L D, da Silva L D, et al. Ticc: A tool for interface compatibility and composition[R]. ucsc-crl-06-01. School of Engineering, University of California, Santa Cruz, 2006

[19] Alfaro L D, da Silva L D, Faella M, et al. Sociable interfaces[C]// FROCOS 2005: 5th International Workshop on Frontiers of Combining Systems, LNCS 3717. Berlin: Springer-Verlag, 2005: 81-105

[20] Ptolemy. http://ptolemy.eecs.berkeley.edu/ptolemyII/index.Htm. 2008-02-07

[21] Cheng C P, Fristoe T, Lee E A. Applied verification: The ptolemy approach[R]. UCB/EECS-2008-41. EECS Department, University of California, Berkeley, 2008

[22] Holzmann G J. Software model checking with SPIN[J]. Journal of Advances in Computers, 2005, 65: 78-109

[23] UPPAAL. http://www.uppaal.com/. 2006-09-08

[24] Behrmann G, David A, Larsen K G, et al. UPPAAL 4.0[C]// Proceedings of Third International Conference on the Quantitative Evaluation of Systems (QEST 2006). IEEE Press, 2006: 125-126

[25] Alur R, Dill D L. A theory of timed automata[J]. Journal of Theoretical Computer Science, 1994, 126: 183-235

[26] TIMES. http://www.timestool.com/. 2008-11-06

[27] Amnell T, Fersman E, Mokrushin L, et al. TIMES: A tool for schedulability analysis and code generation of real-time systems [C]// Proceedings of Formal Modeling and Analysis of Timed Systems: First International Workshop, FORMATS 2003, LNCS 2791. Berlin: Springer-Verlag, 2003: 60-72

[28] Sentilles S, Pettersson A, Nyström D, et al. Save-IDE-A tool for design, analysis and implementation of component-based embedded systems [C]// Proceedings of International Conference of Software Engineering (ICSE 2009). IEEE Press, 2009: 607-610

[29] Kerholm M A, Carlson J, Fredriksson J, el al. The SAVE approach to component-based development of vehicular systems [J]. Journal of System and Software, 2007, 80(5): 655-667

[30] INRIA. OpenEmbeDD. http://openembedd.inria.fr/home_html. 2008-10-22