本体概念图的展示过程及技术实现

王诗碕 李伊潇 沈立炜 赵文耘
(复旦大学软件学院 上海 201203) (上海市数据科学重点实验室(复旦大学) 上海 201203)

摘 要 本体建模是语义网研究与建设中的重要工作。在面对一个大规模的领域时,采用众包以及图形化的编辑方式能够吸引更多人参与本体建模的工作。在为此设计的协同建模平台中,本体概念图的展示应满足相应的特征,包括内容正确性、局部化展示以及轻量级的数据传输。针对概念图展示的实际需求,研究并归纳了基于本体文件的概念图展示过程,该过程旨在将后台保存的本体内容传递到前端的概念图编辑器中,包括本体读取、局部本体抽取、本体数据转换、图形模型转换和图形展示一系列步骤。另外,在采用本体查询、模型转换语言等一系列工具的基础上,给出了概念图展示的实现方案,并将实现该展示过程的模块集成在本体协同建模平台中。

关键词 语义网,本体编辑,模型转换
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.019

Display Process and Technique Implementation of Ontology Conceptual Diagram

WANG Shi-qi LI Yi-xiao SHEN Li-wei ZHAO Wen-yun
(Software School, Fudan University, Shanghai 201203, China)
(Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)

Abstract Ontology modeling is critical in the research and construction of semantic Web. Faced with a giant-scale domain, crowdsourcing and graphical editing can attract more people to participate the ontology modeling work. In the platform designed for these, the display of concept mo-del should meet a series of requirements, including the correctness of content, partial display and light-weight data transmission. To satisfy the actual needs in the display of concept model, this thesis studied and concluded the display process of concept map which is based on ontology. This process aims at the transmission of ontology data from backstage to the concept map editor in the foreground. The process includes a series of steps: the loading of ontology model, the localization of ontology, data transmission, model transformation and model display. Besides, based on a set of tools including SPARQL and XSLT, this essay gave an implementation scheme for the display of concept map, and integrated the modules implementing the display process into the cooperative ontology modelling platform.

Keywords Semantic Web, Ontology edit, Model transformation

1 概述

在语义网(Semantic Web)[1]的研究与建设中,对本体进行建模是一项重要的工作。相比于直接使用RDF/RDFS、OWL等本体描述语言来创建本体模型,使用本体建模工具能够提高本体建模的效率,同时保证模型的一致性。当前业界已经开发出多种本体建模工具,例如Protégé[2]、OntoStudio[3]等。

本体模式(Ontology Schema)在抽象的层次上定义了本体中的概念(使用owl:Class表示类)、概念间的关系(使用owl:ObjectProperty表示对象属性)以及概念的属性(使用owl:DatatypeProperty表示值属性)。它可以映射为一张图,将其图形化表示称为"概念图",概念图的范本如图1所示。其中,类与对象属性是同等地位的实体,对象属性的定义域与值域分别与类进行连接。另外,值属性是类特有的,在图形展示中一般可将其隐藏。

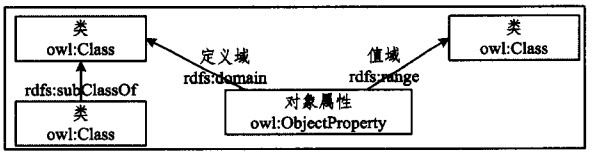

图1 本体概念图的范本

在不同工具中对本体模式进行编辑的方式包括树结构编辑与图形编辑。树结构编辑方式允许本体建模人员在相应视图中创建并维护概念、关系与属性的树状层次结构,如Protégé[2]。然而,这种工具较适合对本体结构具有清晰理解的专业人员使用。相比之下,图形编辑方式在概念图画布上

到稿日期:2015-02-09 返修日期:2015-05-17 本文受863国家重点基金项目(2013AA01A605)资助。
王诗碕(1992-),男,硕士生,主要研究方向为模型转换,E-mail:14212010017@fudan.edu.cn;李伊潇(1990-),女,硕士生,主要研究方向为协同化本体建模技术;沈立炜(1982-),男,博士,助理研究员,主要研究方向为软件产品线、自适应系统等;赵文耘(1964-),男,教授,主要研究方向为软件工程、基于构件的软件开发等。

通过拖曳本体建模元素(包括本体的概念与关系的元素)来创建一个本体概念模型,更自然地将本体概念与关系在一个可视的模型中进行融合并加以展示。在这种方式的支持下,不具备本体建模技术的人员也可以将他们的知识较为便捷地转换成一张本体概念图。

当面对一个大规模的领域,诸如智慧城市领域时,采用众包的技术由一群人员对同一个本体模型进行协同式的编辑是一种有效的方式。这种方式能够吸收并结合群体的智慧,从而保证本体模型内容的完整性。为了支持众包的本体编辑,有必要开发出一个基于 Web 的协同化本体建模平台。它应支持多用户同时登录并在浏览器端通过图形方式对本体模型进行编辑。当概念图成为用户的直接编辑对象时,负责展示概念图的模块也就成为了本体协同建模平台中最核心的功能模块之一。在设计平台及其模块时,我们认为概念图的展示应满足以下一组特性:

1)概念图的内容正确性。需要提供一种机制将本体正确地转换为前端展示的图形。

2)概念图的局部化显示。一个本体模型可能拥有成百上千个概念,而参与本体编辑的单个用户往往仅关心该模型的局部,且在大小有限的屏幕上放置数量合理的概念能够降低视图的复杂程度。因此,需要一种机制来抽取出局部的本体内容,并将其转换为概念图。

3)前后端轻量级的数据传输。在基于 Web 的协同建模平台的使用过程中,后台所保存的本体信息需要经常被传输到前端并展示在概念图中。需要定义更为简单的信息格式,来缩减所需传输的数据包的大小。

本文针对概念图展示的实际需求,研究并归纳了基于本体文件的概念图展示过程,该过程旨在将后台保存的本体内容传递到前端的概念图编辑器中,包括本体读取、局部本体抽取、本体数据转换、图形模型转换和图形展示这一系列步骤。另外,基于采用本体查询、模型转换语言等一系列工具,本文给出了概念图展示的实现方案,并将实现模块集成在本体协同建模平台中。最后,本文还列举了一个本体模型的展示实例,来说明概念图展示的效果。

本文第 2 节简要介绍本体协同建模平台的概念图展示过程;第 3 节详细介绍概念图展示过程中主要步骤的技术实现;第 4 节给出一个使用本文提出方法的概念图展示具体过程,包含对中间制品的列举;最后是对本文的总结与未来工作的展望。

2 本体概念图的展示过程

图 2 展示了应用于本体协同建模平台中的概念图展示过程,该过程中的一组步骤分别在服务器端与客户端中实施。以下首先对这些步骤进行概述。

在后台服务器应用中,使用 Jena 的 TDB 组件读取存储在服务器文件系统中的本体,并将其转换为一个 Jena OntModel,可以通过 Jena 的 API 读取 OntModel 中的内容。

1)在后台服务器应用中,使用 Jena 的 ARQ 引擎对 OntModel 进行局部本体信息的抽取,并存放在相应的列表中,以简化前后端的数据传输负载。

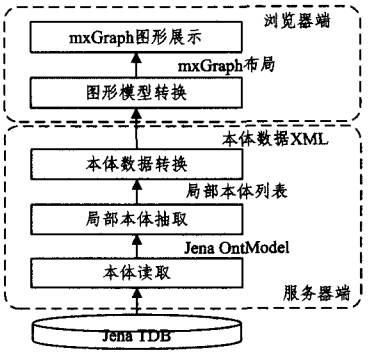

图 2 本体概念图的展示过程

2)在后台服务器应用中,将抽取后得到的列表内容转换为基于预定义 Schema 的 XML,并将该简化的 XML 传送到客户端由客户端脚本处理。

3)在客户端脚本中,将来自于服务器端的简化数据 XML 通过 XSLT 规则转换为符合图形编辑工具 mxGraph 展示格式的 XML 描述,进行图形化展示。

4)通过 mxGraph 的 API 解析 XML 描述,在浏览器端建模平台的 ExtJS 容器中展示该局部的概念图。

下一节将对该过程中的关键步骤(第 1)－3)步)的实现技术进行详细介绍。

3 概念图展示过程中的技术实现

3.1 OntModel 的局部本体抽取

局部的本体信息是在用户指定的中心类或中心对象属性以及给定层次数量条件下的相关类与对象属性的集合。对以一个本体的类为中心的层次给出以下定义。

• 该中心类的父类和子类,即与该类具有 rdfs:subClassOf 关系的其他类;

• 以该中心类为定义域或值域的对象属性,即与该类具有 rdfs:domain 或 rdfs:range 关系的对象属性。

另外,以一个本体的对象属性为中心的层次有如下定义。

• 该中心对象属性的定义域或值域,即与该对象属性具有 rdfs:domain 或 rdfs:range 关系的类。

局部本体的抽取过程即基于此标准。另外,为了提高前后端的本体数据的传递效率,仅抽取与概念图展示相关的本体内容。例如,对本体类来说,只需抽取出其名称与标识符。在浏览器端对一个类进行编辑时,可通过类名与后台进行交互来获取该类的详细信息。这个过程所抽取出的本体信息被分别组织在 5 个列表中。

• classList:存放局部本体中的类名与标识符的列表;

• objectPropertyList:存放局部本体中的对象属性名称与标识符的列表;

• isAList:存放局部本体中的 rdfs:subClassOf 关系,存储为"子类名[is_a]父类名"方式的字符串;

• domainList/rangeList:存放局部本体中的 rdfs:domain/rdfs:range 关系,存储为"对象属性名称[domain/range]类名"方式的字符串。

OntModel 可以被映射为一张本体概念图,本体局部信息的抽取则可被理解为使用类似于广度优先遍历的算法在该图中抽取出符合条件的一组概念与关系。该算法的输入包含 3 个部分,分别为:需要被局部化的本体模型 OntModel、给定的

中心类 CenterClass，以及需要展开的层次数量 Layers。算法的实现过程主要包含 3 个步骤，如图 3 所示。

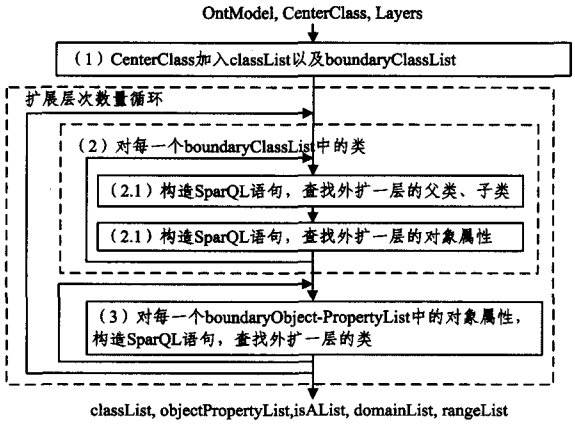

图 3　局部本体抽取算法的过程

(1)将中心类加入 classList 以及 boundaryClassList。算法使用 boundaryClassList 和 boundaryObjectPropertyList 来保存已查找到的节点的最外层类和最外层对象属性，起到类似广度优先算法中队列的作用。

(2)将 boundaryClassList 中的每一个类作为当前节点进行处理。该步骤包括 2 个子步骤。

(2.1)构造查找父/子类的 SparQL 语句，查询与当前类节点之间具有父类与子类关系的节点，并把这些节点合并入 classList 中，同时把它们的父子关系存入 isAList。

(2.2)构造查找对象属性的 SparQL 语句，查询把当前类节点作为定义域和值域的对象属性节点，并把这些节点存入 objectPropertyList 中，同时把它们的定义域关系存入 domainList，值域关系存入 rangeList。

(3)将 boundaryObjectPropertyList 中的每一个对象属性作为当前节点进行处理。构造根据对象属性查找类的 SparQL 语句，查询当前对象属性节点的定义域和值域，并把这些节点存入 classList 中，同时把它们的定义域关系存入 domainList，值域关系存入 rangeList。

基于扩展层次的数目，在每一次循环结束后，需要将新加入 classList 和 objectPropertyList 的类和对象属性分别加入 boundaryClassList 和 boundaryObjectPropertyList，并重新进入步骤(2)。当循环查找的层数与输入 Layers 相等时，算法结束，输出 classList、ObjectPropertyList、isAList、domainList 和 rangeList。

在以上过程的步骤(2.1)、步骤(2.2)和步骤(3)步骤中，都需要自动地构造 SparQL 查询语句来查找出与指定类或对象属性相关的其他类和对象属性。我们为这些查询定义了如表 1 所列的 SparQL 语句模板，其中的占位符(userOnt：当前类、userOnt：当前对象属性)要由具体的类名在运行时进行代替。

在以上的查询语句模板中，常量 Prefix 由一组默认的命名空间地址组成，表示默认加载 XMLS、RDF、RDFS 和 OWL 的词汇表，以保证解释 SparQL 语句的 ARQ 引擎可以寻找到查询中所使用的这些命名空间中的词汇。userOnt 代表编辑器默认的命名空间。参数“当前类”和“当前对象属性”是给定中心类和属性的标识符，变量“?父类”、“?子类”分别代表中心类的父类和子类，“?定义域对象属性”、“?值域对象属性”分别表示以中心类为定义域和值域的对象属性，“?定义域类”、“?值域类”则分别代表给定中心对象属性的定义域与值域。

表 1　用于局部本体抽取的 SparQL 查询语句模板

```
String querySubClsStr =Prefix +
  Select ?子类 ?父类
  WHERE { { ?子类 rdfs:subClassOf userOnt:当前类} UNION {userOnt:当前类 rdfs:subClassOf ?父类} };
  String queryObjectPropertyByClassStr=PREFIX+
  Select ?定义域对象属性 ?值域对象属性
  WHERE { {?定义域对象属性 rdfs:domain userOnt:当前类} UNION {?值域对象属性 rdfs:range userOnt:当前类} };
String queryDomainRangeByPropertyStr=PREFIX+
  Select ?定义域类 ?值域类
  WHERE { {userOnt:当前对象属性 rdfs:domain ?定义域类} UNION {userOnt:当前对象属性 rdfs:range ?值域类} };
```

3.2　本体数据转换

局部本体抽取过程得到的本体信息(5 个列表)需要被转换为中间的数据格式，传递到浏览器端并通过脚本处理。定义专门的中间数据格式符合关注点分离[4]的原则。这意味着当浏览器端需要使用不同的图形展示方式时，不需要修改服务器端的本体抽取与数据转换的过程，仅需在前端为中间数据定义新的处理方式即可。

表 2 展示了中间数据格式的 XML 模式(部分)。由于篇幅所限，仅以表示本体类以及子类/子属性的关系为例，来展示相关本体数据的保存结构。除这些元素类型之外，标签还包括〈ObjectProperty〉、〈ObjectProperties〉、〈domain〉、〈domains〉、〈range〉、〈ranges〉，分别表示对象属性、定义域类、值域类及其有序的集合。

表 2　中间数据格式的 XML 模式(部分)

```
〈xs:element name="LocalName" type="xs:string"/〉〈! ——存放类或对象属性的唯一标识符——〉
〈xs:element name="Label" type="xs:string"/〉〈!——存放类或对象属性的标签|命名——〉
〈xs:element name="r" type="xs:string"/〉〈!——存放关系的起始点标识符——〉
〈xs:element name="t" type="xs:string"/〉〈!——存放关系的目标点标识符——〉
〈!——Class 表示本体类，具有 LocalName 与 Label 两个属性——〉
〈xs:element name="Class"〉
 〈xs:complexType〉
  〈xs:sequence〉
   〈xs:element ref="LocalName"/〉
 〈xs:element ref="Label"/〉
 …
〈!——Classes 表示一组本体类的有序集合——〉
〈xs:element name="Classes"〉
 〈xs:complexType〉
  〈xs:sequence〉
   〈xs:element ref="Class" maxOccurs="unbounded"/〉
…
〈!——isA 表示子类/子属性关系——〉
〈xs:element name="isA"〉
 〈xs:complexType〉
  〈xs:sequence〉
   〈xs:element ref="r"/〉
   〈xs:element ref="t"/〉
…
〈!——isAs 表示一组子类/子属性关系的有序集合——〉
〈xs:element name="isAs"〉
 〈xs:complexType〉
  〈xs:sequence〉
   〈xs:element ref="isA" maxOccurs="unbounded"/〉
…
```

抽取得出的本体信息的数据结构与该 XML 模式具有清晰的对应关系。通过编程可以基于 5 个列表中的内容创建一段包含本体信息的 XML 中间数据。例如，通过遍历 ClassList 中的各个元素可创建多个〈Class〉标签节点，而这组〈Class〉节点可最终组合为一个〈Classes〉上层标签节点。

3.3 图形模型转换

由服务器端传递而来的局部本体信息与实现语言无关，也与具体的前端图形展示方式无关。当在浏览器端选择某一种特定的图形工具来展示本体概念时，就需要将这些中间数据转换为符合工具要求的特定数据格式。

在基于 Web 的协同式本体开发平台中，浏览器端采用 mxGraph 图形工具[5] 支持概念图的编辑与展示。mxGraph 是一个基于 JavaScript 语言的绘图组件，它可以被嵌入在前端的页面文件中，并通过编写 JavaScript 代码配置绘图画布的尺寸、背景、工具栏项目等一系列工具参数。在图形展示方面，mxGraph 可读取一个 XML 格式的布局文件，按照布局文件中定义的信息绘制出相应的图形元素。

在用于展示本体概念图的布局文件模式中，定义了 2 类节点与 3 类连线，如表 3 所列。其中，第一列表示 XML 标签，第二列是对该标签的描述，第三列则列举其图例。

表 3 mxGraph 的布局 XML 模式

标签	描述	图例
〈Class〉	表示本体类的节点	类
〈ObjProperty〉	表示对象属性的节点	对象属性
〈subClass〉	表示父类与子类之间的关联	——是一种——→
〈domain〉	表示对象属性与其定义域之间的关联	——关系主体——→
〈range〉	表示对象属性与值域之间的关联	——关系客体——→

使用 XLST 将中间数据格式的 XML 自动转换为符合 mxGraph 数据模式的 XML。由于篇幅的限制，表 4 中仅列出中间数据中的〈Classes〉节点与〈isAs〉节点被转换至相应 mxGraph 元素的 XLST 代码。

表 4 XLST 代码示例

```
〈xsl:for-each select="/root/Classes/*"〉
  〈Class〉
    〈xsl:attribute name="id"〉〈xsl:value-of select="LocalName"/〉〈/xsl:attribute〉
    〈xsl:attribute name="name"〉〈xsl:value-of select="Label"/〉〈/xsl:attribute〉
    〈xsl:attribute name="cellid"〉〈xsl:value-of select="LocalName"/〉〈/xsl:attribute〉
    〈mxCell parent="parent" vertex="1" style=""〉
      〈mxGeometry as="geometry" /〉
...
〈xsl:for-each select="/root/isAs/*"〉
  〈subClass cellid="is_a" name="是一种"〉
    〈xsl:attribute name="id"〉
      〈xsl:value-of select="s"/〉〈xsl:text〉[is_a]〈/xsl:text〉〈xsl:value-of select="t"/〉
    〈/xsl:attribute〉
    〈mxCell edge="1" parent="parent"〉
      〈xsl:attribute name="source"〉〈xsl:value-of select="s"/〉〈/xsl:attribute〉
      〈xsl:attribute name="target"〉〈xsl:value-of select="t"/〉〈/xsl:attribute〉
      〈mxGeometry as="geometry" /〉
...
```

在该转换过程中，基于预定义的转换规则，使用 XPath 对中间数据中的相应内容进行定位，并将这些内容转换为相应的 mxGraph 布局文件的元素。其中，〈Classes〉标签的各子节点通过 xsl:for-each 被搜索并转换为 mxGraph 中的〈Class〉节点，其属性 id、cellid 均被设值为该节点的名称，name 设为其标识符，〈ObjectProperties〉标签同理。〈isAs〉、〈domains〉和〈ranges〉3 个节点的子节点分别被转换为〈subClass〉、〈domain〉和〈range〉，其 id 被设置为“关系起始点[is_a/has_domain/has_range]关系目标点”的形式。

在转换本体元素基本信息的同时，转换过程还将为 mxGraph 的布局文件自动添加默认的图元布局信息，如表 4 中所列的〈mxCell〉与〈mxGeometry〉标签所示。当 mxGraph 绘图工具打开该布局文件时，可调用 mxGraph 的 mxFastOrganicLayout 提供的布局算法来为这些图元进行自动布局，此时这些图元的位置坐标也就相应地改变了。

4 概念图展示过程实例

本节以一个实际的本体模式为例，展示在概念图展示过程中所生成的一系列中间制品。同时，结合已开发的前端本体图形化展示工具，列出概念图的展示效果。

示例本体是我们所研究的智慧城市领域核心本体中的组成部分，它一共包括 10 个本体类以及 4 个对象属性，这些类和对象属性的名称如表 5 所列。

表 5 示例本体的类和对象属性

类名	对象属性名称	定义域	值域
角色、实体角色、事件角色、协作角色、系统角色、服务角色、实体基本类、协作基本类、事件、服务	承担	实体基本类	角色
	实现	协作基本类	服务
	服务参与	服务角色	服务
	协作参与	协作角色	协作基本类
	触发	事件	服务

当以角色类为中心类，仅设置一层扩展层次来展示该本体概念图时，得到方法实施过程中的一组中间制品。根据本体局部信息抽取的算法，需要被展示的本体类被缩减至 6 个（角色、实体角色、事件角色、协作角色、系统角色、服务角色）。同时，还将展示一个对象属性（承担）。由于篇幅原因，仅列举出与角色类相关的中间数据 XML 以及它在浏览器端转换而成的 mxGraph 布局文件中的数据内容，如表 6 所列。两者间的转换是通过预定义的 XLST 脚本实现的。

表 6 中间数据 XML 部分（左）和 mxGraph 布局文件部分（右）

```
〈?xml version="1.0" encoding="utf-8"?〉
〈root〉
  〈Classes〉
    〈Class〉
      〈LocalName〉Role〈/LocalName〉
      〈Label〉角色〈/Label〉
    〈/Class〉
    ……
  〈/Classes〉
  ……
〈/root〉
```

```
〈mxGraphModel〉
  〈root〉
    〈mxCell id="root"/〉
    〈mxCell id="parent" parent="root"/〉
    〈Class name="角色" cellid="Role" id="Role"〉
      〈mxCell style="fillColor=#7D7DFF;gradientColor=white;shadow=true;" parent="parent" vertex="1"〉
        〈mxGeometry as="geometry" height="30" width="120" y="0" x="0"/〉
      〈/mxCell〉
    〈/Class〉
    ……
  〈/root〉
〈/mxGraphModel〉
```

浏览器端的 mxGraph 读取布局文件后展示出如图 4 所示的本体概念图。

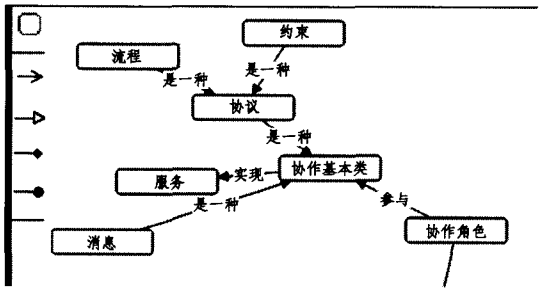

图 4　示例本体的概念图展示效果

结束语　提供一套具有较高可用性且符合语义网诸多语言标准的本体编辑器是本体研究领域的重要实现工作。已有的诸多本体编辑器均提供了强大的本体编辑与推理等功能，但它们仍然需要在展示本体概念图的实现机制上更加贴近实际用户的需求，并且在概念图内容正确性、概念图局部化显示以及轻量级的前后端数据传输方面体现其优势。

本文提出一个应用于协同本体编辑平台的本体概念图展示过程，该过程包含一系列步骤：本体读取、局部本体抽取、本体数据转换、图形模型转换和图形展示。这些步骤分别应用于建模平台的服务器端与客户端，涉及了抽取算法，并且建立在模型转换等通用技术的基础之上。

当然，本体概念图的展示在许多方面仍然具有扩展与改善的潜力，它们可作为我们今后的工作内容。这些潜在的改进包括扩展对数据属性（DatatypeProperty）和限制（Restriction）的展示、改善概念图的自动布局，以及实现选择性隐藏以支持概念过多的情况下的图形简化等诸多方面。

参 考 文 献

[1] Berners-Lee T, Hendler J, Lassila O. The Semantic Web[J]. Scientific American, 2001, 284(5): 28-37

[2] Protégé[OL]. http://protege.stanford.edu

[3] OntoStudio [OL]. http://www.semafora-systems.com/en/products/ontostudio/

[4] Dijkstra E W. On the Role of Scientific Thought[M]// Selected Writings on Computing: A Personal Perspective. Springer New York, 1982: 60-66

[5] mxGraph[OL]. http://www.jgraph.com/mxgraph.html

（上接第 86 页）

Web_Ser 的可靠性变化对系统可靠性最大值和最小值的影响比较小，资源 *NetWork* 和 *ClientWorkstation* 的可靠性变化对系统可靠性最大值的影响比较大，资源 *ClientWorkstation* 的可靠性变化对系统可靠性最小值的影响比较大。

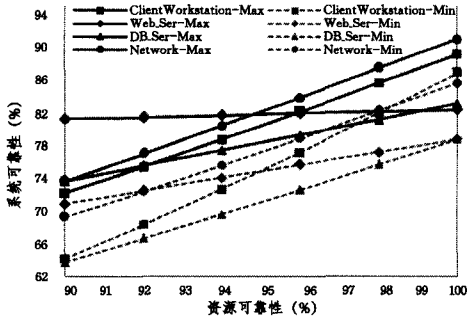

图 9　图 1 模型的系统可靠性预测

结束语　本文提出了基于 MARTE 模型的系统可靠性预测方法。该方法不仅能够预测系统可靠性的最大值和最小值，还能通过调整各个资源的可靠性值，考察其对系统可靠性的影响，为设计人员的进一步工作提供参考。下一步将在现在的方法上考虑资源的故障传播性和资源的冗余优化问题。

参 考 文 献

[1] Gokhale S S, Trivedi K S. Analytical models for architecture-based software reliability prediction: A unification framework [J]. IEEE Transactions on Reliability, 2006, 55(4): 578-590

[2] Booch G, Rumbaugh J, Jacobson I. Unified Modeling Language User Guide[M]. New York: Addison Wesley, 2005

[3] OMG. UML Profile for MARTE, Beta 2[OL]. http://www.org/cgi-bin/doc?ptc/2008-06-08

[4] Cortellessa V, Singh H, Cukic B. Early reliability assessment of UML based software models[C]// Proceedings of the 3rd International Workshop on Software and Performance. Rome, Italy: ACM, 2002: 302-309

[5] Genaina N R, David S R. Sebastian Uchitel. Reliability Prediction in Model-Driven Development[C]// Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems. Berlin: Springer, 2005: 339-354

[6] Cheung R C. A User-Oriented Software Reliability Model[J]. IEEE Transactions on Software Engineering, 1980, 6(2): 118-125

[7] Majzik I, Patariczа A, Bondavalli A. Stochastic dependability analysis of system architecture based on UML Models[J]. Architecting Dependable Systems, 2003, 2677: 219-244

[8] 刘毅，麻志毅，何啸，等. 一种从 UML 模型到可靠性分析模型的转换方法[J]. 软件学报，2010，21(2)：287-304
Liu Yi, Ma Zhi-yi, He Xiao, et al. Approach to Transforming UML Model to Reliability Analysis Model[J]. Journal of Software, 2010, 21(2): 287-304

[9] Forejt V, Kwiatkowska M, Norman G, et al. Automated Verification Techniques for Probabilistic Systems[C]// Proceedings of 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM 2011). Bertinoro, Italy, Springer, 2011: 53-113

[10] Kwiatkowska M, Norman G, Parker D. PRISM 4.0: Verification of Probabilistic Realtime Systems[C]// Proceedings of 23rd International Conference on Computer Aided Verification. Berlin: Springer, 2011: 585-591

[11] FMPAer[OL]. http://lcs.ios.ac.cn/~zxy/tools/fmpaer.htm

[12] Gérard S, Dumoulin C, Tessier P, et al. Papyrus: A UML2 Tool for Domain Specific Language Modeling[C]// Proceedings of Model-Based Engineering of Embedded Real-Time Systems. Germany, Springer, 2010: 361-368

[13] Jouault F, Kurtev I. Transforming models with ATL[C]// Proceedings of the 2005 International Conference on Satellite Events at the MoDELS. Italy, Springer, 2005: 128-138