基于 SSH 框架的 java 代码自动生成①

丁 亮, 许舒人

(中国科学院 软件研究所 软件工程中心, 北京 100190)

摘 要: 利用软件复用技术可以大大提高软件的生产效率, 有效的减轻开发者的负担. 代码自动生成是软件复用常用的一种技术手段. 在 SSH 框架的基础上, 研究了控制层的代码, 从 hibernate 生成的 javaBean 数据对象入手, 利用模板生成技术和配置树的层次方法, 自动化生成控制层的代码, 利用 eclipse 的 SWT 技术做可视化界面, 实现了一个原型系统, 并测试了该系统.

关键词: 软件复用; SSH 框架; 代码自动生成; 模板; SWT 框架

Automatic Generation of Java Code Based on SSH Framework

DING Liang, XU Shu-Ren

(Software Engineering Center, Institute of Software, Chinese Academy of Sciences, Beijing 100090, China)

Abstract: It will improve production rate and release developer's burden by using software reusing technology. Automatic code generation is one of most commonly used methods. Based on SSH framework, this paper studied its control layer. After studying the javabean data which generated by hibernate, this paper auto-generated the control layer codes by using Model generated technology and config tree method. Producing a Demo System and testing it . the Demo System provide a friendly interface based on SWT framework.

Key words: software reusing; SSH framework; code automatic generation; templet; SWT framework

1 引言

由于 60 年代的软件危机导致了有关软件复用的研究, 国内信息化发展较晚, 最早开始研究软件复用的是北大的杨芙清老师领导的项目组. 从 90 年代初期开始[1], 国内的软件复用主要有三个趋势: 一个是将复用的实践惯例化, 第二个是将复用的技术运用到软件开发过程中, 三是最常用的也是最普遍的, 即将分析领域标准化, 开发支持领域分析的方法或者工具, 由于领域内差异小公共框架便于提取, 复用技术发展较快.

软件复用技术可分为组装技术和生成技术[1], 前者利用组装方式来复用软件构件, 类似于函数库或者构件库, 代表有 eclipse 的插件技术, C 函数库, 甚至是自定义的函数都是为了最小化重复的代码. 后者关注点是减少重复工作的开发时间, 节约开发者的时间和精力, 利用程序生成器来获得复用. 利用配置或者提前构思, 相当于种子, 生成后可以获得模块大部分的核心代码, 从而节约了开发者重复开发的时间.

其中, 在代码自动化生成技术领域, 发展较快, 已经有了很多普遍认可的方法. 比如界面的代码自动生成, 现在流行的是基于模型的方法[2], 将对概念模型的描述和陈述通过解释器进行解释, 在经过界面布局算法的优化之后, 通过生成器生成用户界面的布局.

SSH 是 struts+spring+hibernate 的一个集成框架, 是目前较流行的一种 Web 应用程序开源框架. 该框架即是使用了构建重用、提取公共逻辑的方法减少了很多程序员开发的负担. 虽然 SSH 将变量和方法等都进行了简化, 但是 SSH 只注重了代码的语法层简化, 很多功能的业务流虽然在代码部分是有差异的但是逻辑往往是相通的, 然而 SSH 并没有对逻辑进行包装和重用.

如何更加有效快速的开发基于 SSH 架构的代码,

① 基金项目:国家自然科学基金(61170074);国家高技术研究发展计划(863)(2012AA011204);国家科技支撑计划基金(2012BAH14B02)
收稿时间:2014-01-13;收到修改稿时间:2014-02-26

很多研究工作都已经做了. 有些提出的是开发方法和规范, 采用快速迭代的方法进行开发[3], 将常用的生成类, Log 方法等进行包装, 更加方便程序员进行开发, 但是实际上这些新的规范过于复杂, 所以推广并不高. 有些虽然提出了原型, 但是配置文件需要用 XML 去描述[4], XML 配置存在很多冗余的选项, 习惯自己开发风格的程序员往往都很难 XML 配置项中条目的含义, 如果要采用这套方法, 甚至自己要重新学习 XML 规范和原型中对条目配置的意义, 如此复杂的过程还不如直接去编写代码更加高效. 这些不利因素增加了代码自动生成技术的推广难度.

JAVA 代码的自动生成, 由于java 语言是面向对象的, 考虑到java 语言的特点, 从设计上来说, 复用方法主要有以下几个可以思考的方向: 一是面向对象技术的软件重用方法[5], 这一块最具有代表性的是微软提出的 ActiveX, 现在几乎在主流应用平台上都支持这样的插件方法, 类库是它最核心采用的方法. 二是面向对象但从数据驱动[6], 如果要将对象的代码不再重写, 而能够实现提取方法和成员变量的公共部分, 就要将对象的方法和成员变量放在数据库中, 在对象被第一次实例化的时候, 去实例化这个对象本身的申明. 懒惰式的动态加载一方面减少了内存负担, 另一方面也能够减少对类代码的重复(主要是逻辑部分)定义.

第三种方法是现在主流的研究方向, 基于模型驱动的层次性生成. 由于 SSH 框架就是层次性的, 所以采用这种方法来自动生成该框架上的 java 代码具有优势.

OMEGA 是用基于 UML 扩展进行声明的模型驱动的代码生成方法[7], 模型驱动, 即 Model-driven development,简称 MDD, MDD 依赖于一些公共标准如 XML, MOF, UML 等. 从模型导入后分离出静态代码和动态代码, 静态代码由生成器先生成, 而动态代码在从配置库中解析出来后再由代码生成器生成. 文献[8]模型驱动的方法不仅在代码上实现了分离, 在需求和项目管理上也可以实现分离. UML 设计图由产品经理负责规划, 开发人员从设计流水线上从代码生成器中生成后, 由运营去完成相关的工作. 虽然设计很紧凑, 但是有很多缺点. 一方面是无谓的增加了产品经理的负担, 同时, 在从流水线上拿出模型的时候, 开发人员仍然需要将它用 XML 重新描述, 只会让开发工作更加复杂. 所以推广做的都不是很广泛[8].

这篇论文的关注点将实用提上了高度, 同时基于模型和数据驱动. 由于 SSH 框架中 hibernate 已经将数据库中的对象成员变量生成为 Javabean, 从 Javabean 中直接获得与业务有关的信息, 从而免去了开发人员冗余的配置工作, 会让开发人员更加轻松. 同时, 使用友好的图形化配置界面而不使用 XML 配置描述会让生成工作显得更简单.

2 系统结构

在基于数据驱动的代码自动生成里, 系统必须要将选择 javabean 源文件的方法提供给开发者. 这个 bean 文件在SSH 框架里一般由 Hibernate 自动生成, 开发者不需要为此做额外的工作.

SWT 是一个开源的 GUI 编程框架, 与 AWT/Swing 有相似之处. 如图一, 系统架构图所示, 基于 SWT 的友好的界面体验代替了之前 XML 甚至是 UML 等不友好的配置方法, 界面的部分截图在第五部分有描述, 这样更加简单, 降低了开发者使用的困难. 开发者决定选择执行业务的 javabean.该 javabean 是由 hibernate 自动生成的,主要包括一些成员变量和这些变量相关的基本方法.配置树首先学习这个文件,读出与 action 需要的一部分配置,这些将在后面实现部分具体阐述. 此外, 开发者还需要配置一些其他的参数.如类名、写入文件的目标路径等. 整个配置树的逻辑关系由后面的章节详述.

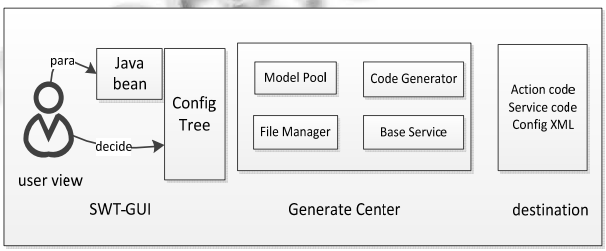

图 1 系统架构

如图中中间部分所示, 配置树将参数写入生成中心.代码解释器根据配置的参数,从模板池中提取相关的代码片段,拼接成目标代码.整个生成过程需要文件管理器和基础服务两个内部类的支持.前者主要负责校对开发提供的输入路径和该目录下文件,负责导向开发者提供的指定输出路径和该目录下的文件.基础服务主要提供置换、正则、格式化、常量变化等服务.

其中, 右边的模块是生成文件的写入路径目录,

主要生成 action 和 service 的代码已经与项目相关的 struts 配置文件. 该路径也是在开发者配置的.

3 生成方法和技术

生成方法和技术将会描述模板生成的方法, 由模板决定的配置树的构建、由配置决定的生成器对代码逻辑的约定, 和局部变量约定等.

3.1 模板生成

从若干个代码文件中提取他们公共的部分, 并将其抽象成模板文件, 称之为模板生成. 模板生成首先要考虑的是如何提取公共的部分. 在各个不同样本中, 有时候代码逻辑是相同的, 可以抽象成一个公共模板; 有些地方的代码是依据某一个关键字或者变量的, 可以抽象成若干个不同的模板碎片; 有些地方的代码具有自己的特殊性又不太容易被抽象, 最好是能够舍去. 基于相似结构的片上系统提供了一种基于相似性抽象定义成图[9]的方法将它抽取出来, 同理, 可以通过下面这个方法来提取 java 样本文件中的公共模板, 并且能够导出一些决策性变量.

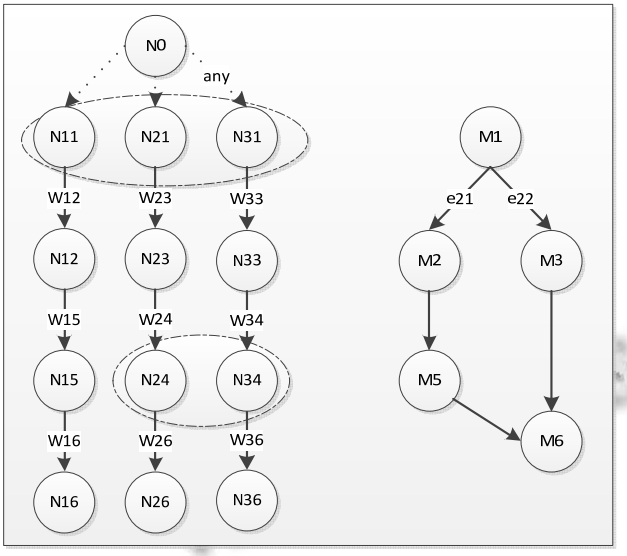

图 2　模板生成示例

定义一个森林 $F=(N,W)$来表示输入的样本, 如图 2 所示, 从树竖直方向上表示一个样本, 其中每个节点 N_{ij} 表示一段代码片段, 在 N_{ij} 之间的指向性线段表示到这个语句代码段的条件, 这些条件或者是局部变量或者是私有变量. 现在为这个森林在最开始端加上一个虚拟节点 N_0, 并且在 N_0 指向所有的开始代码段之间加上虚拟条件, 不妨称之为 *any*, 这样得到一棵树:

$$T=F\cup G',\ G'=\{N0,W0i \mid Nik \in N\}$$

按照下面的规则对这棵树做这样的映射操作:

$$V(T,Nij)=\begin{cases} \mathrm{M_j} & \exists \mathrm{I,kW_{ij}=W_{kj}} \\ \mathrm{NULL} & \mathrm{Arb.i,kW_{ij}\neq W_{kj}} \end{cases}$$

这样将所有的代码节点映射成可以抽象的模板, 如图中右侧的树(准确的说已经不是树了, 只是保持了树形的结构)所示, 由于 $W_{23}=W_{33}$, 得到了一个合并的模板 M_3, 由于 $W_{24}\neq W_{34}$,设置为 NULL, 这个私有的片段便不再考虑. 相同的部分提取成一个公共的模板. 这种方法会对由不同变量条件产生的模板生成不同的分枝, 分枝条件由下面的公式确定:

$E(T,W_{ij})=e_{ju},M_j$ 存在, u 为分枝数目

很明显图中 M_2 和 M_3 产生了一次分枝, 分枝条件 e_{21} 和 e_{22} 便是边的条件, 按照上面的公式产生. 决定这个分枝走向的值可以设置成一个变量, 比如 e_2 是一个变量条件, 当 $e_2=e_{21}$ 时分枝从左边走, 用模板 M_2 来拼接目标代码, 反之从右边. 该变量可以在配置树中配置.

3.2 配置树

配置树镶嵌在 SWT 平台之上, 用 java 开发, 这种开源框架提供了比较友好轻量的界面操作. 描述配置树从层次的角度展现配置的关键信息. 在构建系统的时候, 配置树描述法被广为采纳. 它的原型是 XML 文件或者是类 html 文件, 因为可以通过 DOM 轻松的解析他们. 如在文献[10]中, 用配置树的方法来描述 DCLITTA 框架, 该框架是一种将分布在各个不同地域的构件库进行关键字匹配并重装的一个方案.

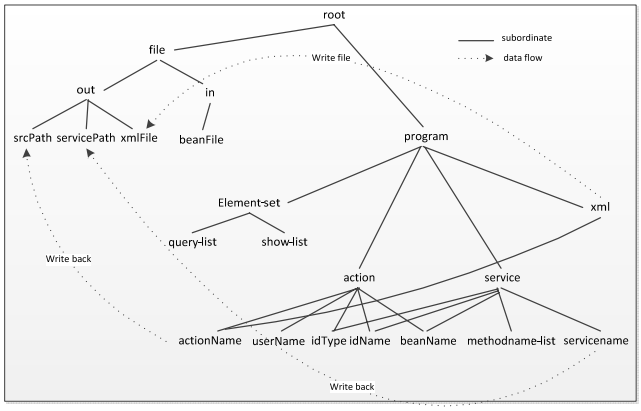

图 3　配置树

如图 3 所示, 由于本论文只针对单业务模型的

java 代码生成, 所以不需要配置与模型相关的信息. 该原型系统可以生成 action、service 等 java 代码. 在图中, 用户配置项已经被标识, 其中, 配置参数分成文件配置选项和程序运行需要的选项. 以生成器为中心, 文件配置又分为 out 和 in, out 表示生成器生成代码输出的路径和文件名, 输入路径就是在第二章提到的系统结构中的用户界面中的 java bean 文件, 从这个文件中读出与程序有关的变量填写到右侧"程序需要"的配置中, 右子树程序分为元素集合和 action 文件、service 文件、xml 文件四类. 元素集合是该对象的成员变量集合, 可以导出 *id* 和 *username* 等. 其他在各子树下的变量分别是生成该文件所需要的配置选项. 而这些选项一些是从 java bean 中学习读出的, 一些则是由开发者在生成初填写的.

3.3 生成规则的约定

为了隐藏冗余的配置信息, 系统将很多局部变量命名进行了隐藏. 尊重以下约定:

baseService: 这个 service 文件为系统创建, 写入用户定义的 servicePath 路径. 这个文件将很多 action 文件中使用到的公共部分提取出来, 支持了很多的公共方法.

Id: javabean 对象都需要有一个 id, 这个 id 是在 hibernate 生成时候导出的. 该 id 可能是用户自定义的 String 类型(唯一), 也可能是由数据库控制的自增的 Integer 类型. 系统从 javabean 中读出带有关键字 id 或者 Id 且不带有关键字 user 的成员变量供用户选择确定哪一个是使用 id, 然后记录 id 的类型. Idtype 决定内容包括: 一、action 中 toadd 方法是否要使用 baseService 为系统生成一个随机且唯一的 String 码作为 id; 二、传参给 service 的方法中带有 id 变量的类型; 三、在校对 id 是否合法有效时候, 使用 null 判断或者 string 空判断.

UserId: 由于 SSH 是基于业务的, 每一个 javabean 对象都需要考虑对当前对象实体数据的合法性授权. 举个例子, 如果这个是一份财务表单, 拥有者才有权利确定名称, 修改, 删除等操作. *Userid* 可能是一个 String 串, 也有可能是根据系统原来权限管理里生成的 *username* 对应的 *id* 号. 系统从 javabean 中读出关键字 *userid*(不区分大小写), 并供用户选择使用哪个, 然后记录 *userid* 的类型. *Userid* 确定以下内容: 一、在 add 操作、update 操作自主为对象加上这样的数据, 登录 id 通过 baseService 的 getloginUserId 方法获得, 写入这样的登录 id 确保对象的用户所有权; 二、在 delete 操作的 service 层, 对 userid 进行判断, 确保是拥有者才有权利进行删除操作; 三、取消了 get 和 list 方法中的登录 id 校对, 即对不是拥有者放开读的权限.

beanName:beanName 在生成代码中经常使用, 但是 beanName 由提供的 beanName-file 直接确定, 不再供用户选择. 示例在一个 javabean 名为"SampleBean"对应的 action 中使用申明定义"SampleBean sbean;"那么代码中就可以使用 sbean 进行操作了.

展现列表: 读取 javabean 的成员列表供用户选择, 确定某几项为展现项. 展现项的含义是需要在首页简要的显示这几个属性, 如一个财务表单有很多信息, 但是可能首页显示的列表中只要简要的显示 "收账人、出纳日期"这两项, 则勾选这两项, 展现列表将会决定: 一、action 中对输出往前端页面的表单中, 只展现 *id*、操作和展现列表规定的项目, 表单题目显示为成员变量名(这样是为了减少用的配置负担); 二、service-list 方法在读取时, 除必要信息外, 必须要读取展现列表规定的项目.

查询列表: 读取 javabean 的成员列表供用户选择, 确定某几项为查询项. 查询项的含义是需要在首页提供几个条件查找项目, 这些项目一般在 web 端用户条件选择表列中展现的信息时候使用, 比如出纳日期在 1 号到 2 号之间的. 对于查询项目, 有以下规定: 1)一定把 javabean 实例作为参数向下传递, 2)如果成员是 String 类型, 则条件记录在 javabean 内, DAO 语句描述为 "xxx like '%javabean.getxxx()%' ; 3)如果成员是可以条件判断上下界的, 比如 Integer, Double, Timestamp 等, 则将下界保存在 javabean 实例内, 这样可以节约一个传参变量, 上界新建一个传参变量 maxXXX, DAO 语句描述为" xxx >=javabean.getxxx() and xxx <=maxXXX", 实际代码生成时候, 需要判空, 校验, 参数确定类型等具体工作要考虑. 如果工作再深入一点, 当然还有更多的数据类型支持工作还需要考虑.

3.4 局部变量名、方法名

局部变量全部都使用内部规定的方法, 对用户屏蔽, 不影响逻辑.

action 内使用的 javabean 名称由用户定义的 javabean-file 文件决定. 在 bean 前加一个 a 表示, 如" SampleBean aSampleBean"; action 使用的 service 名称

由用户定义，对应实体前加一个 a 表示，如”SampleService aSampleService”. action 内使用的与对象 id 对应的辨别码统一定义为 id, 其类型由读取的 javabean 文件中 id 类型确定，如”private Integer id”. Action 内使用的所有 service 方法名(当然也是 service 自身定义的方法)可以由用户配置，但是建议默认，增删改查列表方法都默认为 “add,delete, update, getById, getList”.

Service 内使用的 DAO 默认为 hibernate 生成的 DAO 名，即用 javabean 名+DAO，实例使用 d+javabean 名+DAO，如 “private SampleBeanDAO dSampleBeanDAO” ; Service 内使用的 list 中查找的方法是使用在 BaseService 中定义的 findall(String hql)方法，该 hql 由查询项的条件拼接而成，在该查询项对应的部分如果为 null 或者为空，则忽略这个查询条件.

4 模块功能

如在第二章里系统架构里描述的一样，生成算法主要由交互模块、生成中心两个模块构成，这里用算法语言描述一下交互模块即配置中心的算法和生成中心的算法.

4.1 交互模块

交互使用 SWT 轻型组件，在第五章将会截图示例，这里介绍算法:

① 选择 javabean 文件;

② 选择确定后，调用 generator.readBean 方法，学习显示 id、idtype、username、记录其类型，列表显示所有的成员变量，前端显示给用户操作选择;

③ 确定 userid, id，查找表项、展现表选，配置其他参数

④ 确定后，将前台参数写入 generator 配置中心，调用 generator 高层方法 generate_action; generate_service;generate_xml 三个方法，实现代码生成.

交互逻辑很简单，符合系统结构中的前台逻辑，只需要将配置写入就可以了，特色是需要先选择 javabean 文件，这一步会在配置库中产生很多配置，从而省去了开发者手动配置的负担.

4.2 生成模块

生成模块由模板池、生成器、文件系统和基础服务四个部分构成.

文件系统可以检查输入文件的可读性，读出的数据是否合法，输出文件路径是否可写，在指定路径生成文件，检查文件的大小等等功能.

基础类服务被生成器调用，比如用来正则匹配，大小写转换，关键字替换等等.

模板池包含了很多的模板，这些模板是经过几十个 java 文件经过 3.1 节描述的算法学习推导出来的，生成器根据生成算法读出这些模板进行拼接产生目标 java 文件.

生成器对 action、service、xml 文件的生成算法很类似，这里以 action 为例作说明.

①调用文件服务和基础服务，检查生成路径的文件状态，补偿默认一些没有配置的参数使用默认值;

②打开写入文件 tmp 文件的通道，写入头部 Model;

③判断 id 类型，选择性写入 id 对应的 Model;

④根据 id 类型，选择性匹配增删改查的模板并写入;

⑤根据查询条件，拼接 query 模型的模板，并循环写入;

⑥写入 tmp 文件的尾部，关闭流;

⑦将 tmp 作为输出流，经过基础服务的正则匹配解码器，将变量规则加入，将对应的 beanName 等替换成配置的值，过滤后导出文件为 actionName + “action.java”.

Service 和 xml 与此类似，不过 service 需要拼接的是 list 条件，xml 没有这个选项. 在最后都需要将 tmp 文件重新导出，用正则匹配过滤掉应该替换的项，完成代码生成.

5 生成实例和应用

出于篇幅的考虑，只截图一部分，下面是生成一个示例代码的界面和目录图:

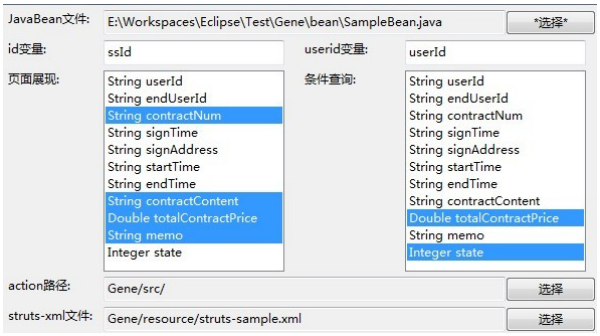

图 4　用户界面

图 4 所示为交互界面, 除必要配置外, 其他都是用默认配置,

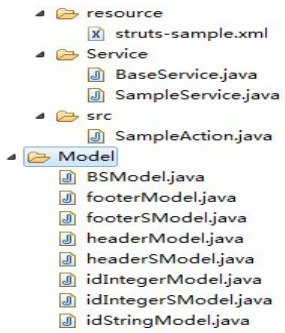

图 5 代码目录截图

图 5 所示为原型系统的部分代码目录截图, 上面的文件夹在 GENE 里, 是生成代码的目标路径, 包含 xml、service、action 等代码. Model 目录中保存的是模板, 构成系统中的模板池模块.

在项目“新发地供应链系统”中的保理模块, 使用这样的方法生成了采购合同、订单票据等对象的 java 代码, 有效的提高了工作效率.

6 结语

随着软件行业的蓬勃发展, 软件重用技术越来越受到重视. 在 J2EE 这一块, 很多重用的技术都得到了实践. 随着软件发展方向的多样化, 给重用技术带来的困难越来越多. 其中, 基于模型的和面向对象两种方法将会成为下一轮代码自动生成领域的主流方法. 但由于模型的难于提取, 面向对象缺乏统一的标准, 因而现行的技术都不能满足软件重用的要求. 在软件重用领域, 仍然需要不断的探索和研究.

参考文献

1 杨芙清,朱冰,梅宏.软件复用.软件学报,1995,6(9):525–533.
2 徐龙杰,万建成.基于模型的用户界面代码自动生成.计算机工程与应用,2004.12:112–115.
3 杨涛,周志波,凌力.基于 Struts 和 Hibernate 的 J2EE 快速开发框架的设计与实现.计算机工程,2006,32(10):83–85.
4 肖寒.J2EE 平台下代码自动生成技术研究.电脑知识与技术,2009,5(20):5421
5 陈岳鹏,戴永成,崔静.基于面向对象的软件重用技术.河北工业科技,2009,26(5):434–437
6 王忠群.面向对象技术和基于数据驱动的软件重用.计算机应用,1999,19(8):1–5.
7 Gitzel R, Schwind M. Experiences With Hierarchy-Based Code Generation in the j2EE Context. ACM, 2006.
8 Han GY, Liu H, Zhang ZJ, Liu ZD, Du ZY. Analysis and design of auto-matic code generation system based on J2EE. IEEE, 2011.
9 韩睦华,刘雷波,魏少军.基于相似结构自动提取的SoC划分方法.计算机工程,2010,36(1):4–6.
10 李琰,李田,谢冰,张路,孙家.一种基于 P2P 支持检索条件重构的构件库互联技术.计算机研究与发展,2007,44(12):2122–2129.