计算机科学1999Vol. 26№ 3

Web 上的组件化柔性 CASE 研究*)

A Flexible Component CASE on the Web

蔡智明　刘宗田　袁兆山　鲍敢峰　黄自强

(合肥工业大学计算机与信息学院计算机系　合肥230009)

Abstract　Based on components and the Web, this paper presents a reuse model named as "the social software engineering", and a flexible CASE framework based on network, Agent and domain to support the model. In the framework, software can be repesented on flexible multilevels; Agents are used in component searching, management, etc. The distributed class-liberary is elasticly constructed, according to domain structure. A prototype system of the framework is designed by client/server on WWW, JAVA, HTML, etc.

Keywords　CASE, WWW, Software component, Agent, Client/Server

1　引言

计算机辅助软件工程(CASE)的理论与技术一直是软件工程研究的热点，从工具箱、工作台、集成化 CASE 到开放式 CASE，CASE 一直试图解决困扰软件工程多年的"软件危机"问题，实现软件生产的工程化、自动化、产业化。由于集成化 CASE 存在缺乏对多变的软件、方法过程的适应能力，强制性过强，规模庞大，使用较难，缺乏知识等问题[6,15]，因此当前许多 CASE 研究的重点是在总结集成化 CASE 研究的基础上研究新型开放式、柔性机制的 CASE 方法与机制[1~3]。

软件组件化方法的研究由于 OO 的发展及软件重用，软件开放性、软件产业化的需要而呈现强劲的势头，它将软件生产划分为组件生产和组件集成两个方面。开发者专心于组件的生产，集成者充分利用组件，专心于应用，从而改变软件生产方式，充分体现软件重用的价值[4]。因此，以组件化方法构造开放式柔性体系结构的软件，是实现软件重用，最终解决软件工程方法与技术问题的重要途径，并已应用于数据库等广泛领域[12]。在 CASE 研究中引入组件化方法，既可将组件的重用、开放、灵活、进化特征引入新型开放式 CASE 中来，又可以用开放式 CASE 来支持组件化的软件开发方法，两者相辅相成，将推动软件工程及软件方法学的开放式、组合化发展。

迅猛发展的 Internet/Intranet 上有着丰富的软件资源，但如何充分、有效、规范地组织、重用、搜索 Web 上的资源，是 Web 上亟需解决的重要问题[7]。目前各种开放式 CASE 模型(如烤面包模型、构件-构架模型[9])也尚欠缺分布式、Web 级的开放性软件重用能力。针对这一背景，我们开展了面向 Web 的组件化柔性 CASE 研究，研究目标是试图将组件化方法、CASE、Web 等方面的相互需求与支持结合到一起，以组件化的方法将 Web 上的可重用软件资源组织成可共享的分布式组件，并以相应的柔性 CASE 技术支持组件的描述、搜索、管理、评价、集成、共享等，使 CASE 支持的组件化软件工程方法获得 Web 级的更广泛、深刻的重用意义。

2　社会化软件工程

为此，我们提出一种"社会化软件工程"的复用模式，其基本特征为：

·是全球范围的分布式软件工程，有利于软件资源的大范围共享与调配；

*)本项目研究得到国家教委博士点基金、机械部基金、南京大学国家软件开放实验室基金资助，蔡智明　副教授，博士生，主要从事软件工程、CASE、分布式软件等方面的研究。刘宗田　研究员，博士生导师，主要从事软件工程、人工智能、反编译等方面的研究。袁兆山　教授，主要从事软件工程、CASE 等方面的研究。

·以网络组件的分布式复用为基础，这种复用可跨越组织和地理的界线；

·软件生产分工为组件生产与组件集成两个体系，生产者与集成者相互之间通过 Web 进行通讯与互操作；

·软件过程的基本模型表现为：需求结构分析——组件配置分析——组件网络搜索——组件集成调整——维护进化。基于组件的特征使软件过程具有开放性、主动性；

· 组件的对象模型具有规范性及相互交互能力，并有相应标准技术的支持；

·涉及到非技术的社会问题，如知识产权、信息安全、利益冲突等，但符合世界发展多元化协同、共享、互补的发展潮流，而社会化分工协作是社会生产力发展的重要因素、形式之一；

·需要相应面向 Web 的组件化 CASE 的支持。

为支持"社会化软件工程"的复用模式，我们提出一种 Web 上的组件化柔性 CASE 构架——AND-CASE，AND-CASE 的研究将组件化 CASE 拓展到 Web 级、分布式的软件复用层次上，在研究策略上运用了 Agent、形式需求规范、Web 上的 Client/Server、分布式构造等方法与技术，以探索一种具有分布性、Web 广泛性及可操作性的新型软件复用方式。AND-CASE 反映了软件工程系统化、专业化、规模化、规范化、产业化的分布、协同的发展趋势，有助于形成既明确分工又全球共享的开放式软件社会和一种以网络组件复用为基础的"社会化软件工程"的复用模式。

3 AND-CASE 构架

AND-CASE 中 AND 的定义为：

AND=Agent+Net+Domain

其中：A 指基于 Agent。Web 上分布式组件重用共享的一个关键问题是如何搜寻网上的组件，将 Agent 机制引入搜索方法中，可以有效解决 Web 上的资源搜索问题，提高搜索的能力与效率；同时，Agent 机制也可有效应用于组件管理等问题中。

N 指基于网络。Internet/Intranet 的软件资源是最大的软件复用库，与集中式的 CASE 环境相比其复用意义是全球性、社会化的，更加深刻、广泛。Web 技术的发展，特别是 Java 及有关技术的出现，则为这种复用提供了可操作的技术支持。

D 指基于领域。组件库的组织层次及组件的生产提取应在领域分析的基础上，对应用领域进行分层归纳、对象建模与过程抽象，提取领域层次框架，加以对象集成与封装，在领域分析基础上组织、提取组件往往层次明确、抽象程度高，稳定、通用、重用能力强[5]。

CASE 所追求的目标一直是试图以一种高层规范描述来刻画软件的需求，然后在 CASE 环境、工具支持下实现软件生产尽可能的自动化、工程化、产业化。作为 Web 上的组件化柔性 CASE，我们将 AND-CASE 构架定义为三个层次（图1）：

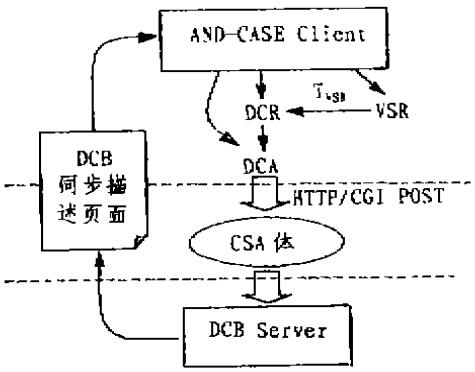

图1 AND-CASE 构架模型

(1)第一层是可视化的软件体系描述 VSR、组件需求规范 DCR 和组件属性集 DCA。VSR 反映的是 AND-CASE 对软件体系的高层抽象表达能力，VSR 是以图形语法为基础，描述软件体系结构、体系端口的二维图形语言；DCR 是描述软件的组件构成、组件连接特性的一维语言；DCA 则为组件特征属性描述集。在 AND-CASE 中，可以用 VSR 描述软件的体系结构需求、过程需求；用 DCR 描述软件的组件结构配置规范需求；用 DCA 则直接描述组件的特征。AND-CASE 提供 VSR 的一个翻译映射 T_{VSR}，将其逐层展开成可为下一层 CSA 体理解的 DCR、DCA 描述。

(2)第二层是搜索体 CSA，CSA 是接收 DCR、DCA 描述请求委托，在 Web 上获取相应分布式组件的搜索体。CSA 是 AND-CASE 中的关键中间体，Agent 机制被引入到 CSA 体的积累式学习和推理引擎中，使 CSA 具有学习、获取、进化搜索知识的能力，能自主、独立、协同地完成分布式组件的搜索、管理等内、外事务，CSA 搜索体不仅能有效地解决网上组件的搜索问题，其智能搜索方法还可扩展到

WWW 其它资源的搜索上，从而解决目前 Web 搜索中，依赖 Search engine 和 Robot 所带来的一次搜索返回信息过多及网络过载等问题[5]。

(3)第三层是 AND-CASE 站点服务器上的描述领域框架及组件特征属性的对象类库 DCB，通过 DCB 将分布于 Internet/Intranet 上的各领域、组件链接到一起，形成一个分布式领域框架、组件的有向无环网络图 DCBG，CSA 体依赖 DCBG 进行漫游、操作，处理 DCR、DCA 请求委托。AND-CASE 同时提供与 DCB 保持动态同一的 HTML(VRML)[11]页面文档，客户可通过 WWW 浏览器浏览。

AND-CASE 运行的基本模式是：AND-CASE 客户通过 Web 浏览器访问 AND-CASE 站点，获得 VSR、DCR、DCA 描述工具，用其中之一表达自己的软件需求，然后将请求事件以 HTTP/CGI POST 方式委托给 CSA 体，CSA 体搜索服务器上的 DCB，取得搜索结果后，将描述页面的 HTML(VRML)文档传回客户端。AND-CASE 客户根据页面对组件进行评价，决定下载与否；AND-CASE 客户也可以直接以超文本浏览方式浏览 DCB 类库的同步描述页面，下载所需组件；并可以向 DCB 服务器提交自己的符合规范的组件供其他客户共享。

4 VSR、DCR 与 DCA

VSR 提供描述软件体系结构的多种视图，包括静态结构关系和动态结构关系，目前主要针对静态关系。由于 VSR 是使用二维图形来表示的，因而需要采用二维语法。VSR 采用上下文无关从语法，将一个软件体系结构看作由许多个 NAPE(n-attaching-point-entity)互相连接的从结构，以下给出 VSR 描述软件结构流程的从语法(参见图2)：

G=(N,T,P,S,I,IO)
N={⟨Flow Chart⟩,⟨Compound Statement⟩,
⟨Rest of Compound Statement⟩,⟨Statement⟩}
T={⟨Start⟩,⟨Begin⟩,⟨End⟩,⟨Simple Statement⟩,⟨Predicate⟩}
S=⟨Flow Chart⟩
I={0,1,2,3}
IO=0
P={⟨Flow Chart⟩::=⟨Start⟩⟨Compound Statement⟩⟨End⟩(110,021)
⟨Compound Statement⟩(1,2)::=⟨Begin⟩⟨Rest of Compound Statement⟩(21)(10,02)
⟨Rest of Compound Statement⟩(1,2)::=⟨Statement⟩⟨Rest of Statement⟩(21)(10,02)
⟨Rest of Statement⟩::=⟨End⟩
⟨Statement⟩::=⟨Simple Statement⟩
⟨Statement⟩(1,2)::=⟨Predicate⟩⟨Compound Statement⟩(12,21)(10,03)
⟨Statement⟩(1,2)::=⟨Predicate⟩⟨Compound Stsaement⟩⟨Compound Statement⟩(210,301)(100,022)}

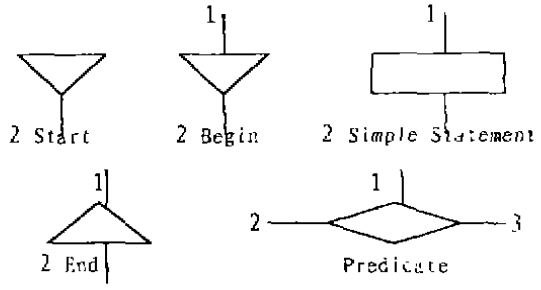

图2 终结 VSR 符号

关于软件体系结构描述语言的研究，目前国内外都处于刚刚起步阶段[6]，VSR 描述也不可能是完备的，二维语法的分析工具也比较缺乏，因而需要一种一维语言作为中间语言；而对于 AND-CASE 来说，对软件结构的需求描述最终要表现为对组件的规范需求描述，从这两个方面考虑，可以再定义一描述软件构成到领域组件配置需求的一维规范语言 DCR，使得 VSR 可以容易地转换为 DCR，以下给出 DCR 语法的简化定义：

Program::=PROGRAM⟨Programname⟩;DOMAIN⟨Domain⟩;INTERFACE⟨Interfaces⟩;
COMPONENT ⟨Components⟩; CONNECTOR⟨Connectors⟩;CONFIG⟨Config⟩
⟨Program name⟩::=⟨Identify⟩
⟨Domain⟩::=⟨Identify⟩{;SUB⟨Identify⟩{;SUPER⟨Identify⟩}}
⟨Interfaces⟩::={⟨Interface⟩}
⟨Components⟩::={⟨Component⟩:{CATEGORY⟨Category⟩;KEY⟨Keywords⟩;DOMAIN⟨Domain⟩; CONNECTOR ⟨Connectors⟩;
…;}}
⟨Connectors⟩::={⟨Connector⟩.{⟨Role⟩}}
⟨Config⟩::={⟨Component⟩.⟨Interface⟩:⟨Connector⟩.⟨Role⟩;
⟨Interface⟩::=⟨Identify⟩
⟨Component⟩::=⟨Identify⟩
⟨Connector⟩::=⟨Identify⟩
⟨Role⟩::=⟨Identify⟩
⟨Keywords⟩::=⟨String⟩
⟨Categoty⟩::=Active X Control|OCX|Java Beans|Applet|Delphi VCL|Plug-in|…

DCR 有关组件描述的终结符(V_{Tdcr})，为 CSA 体可理解的领域组件特征集 DCA 包含，DCA 包括组件标识、组件类型、父子类关系、描述关键字、超链地址、概述、详述等，DCA 描述可以包含多种组件的分层与表示方法[10]。VSR、T_{VSR}、DCR、DCA 之间的关系可描述为：

$$VSR \xrightarrow{T_{VSR}} DCR \quad V_{Tdcr} \subseteq V_{DCA}$$

T_{VSR}则定义为：

$$T_{VSR}=(V_N, V_{Tvsr}, V_{Tdcr}, R, S)$$

其中：V_N 为非终结符的非空有限集；V_{Tvsr} 为

VSR 终结符的非空有限集；V_{Tdcr}为 DCR 终结符的非空有限子集；$R=\{A::=\alpha,\beta|\alpha\in(V_N\cup V_{T_{VSR}})^*,\beta\in(V_N\cup V_{Tdcr})^*\}$；S 为开始符号。

DCA 词汇表 V_{DCA}的范畴参见6节中 DCB 的 JAVA 描述。

由此，AND-CASE 具有了多层次的软件需求柔性表达能力，可为客户提供三种粒度的软件规范[14]描述，从而将形式化与非形式化手段结合起来：

a)用 VSR 描述软件的体系结构，T_{VSR}将之转化为 DCR 描述；

b)用 DCR 描述软件构成配置的领域组件形式规范，V_{Tdcr}则为组件特征集 CDA 包含；

c)用 DCA 描述向 CSA 提出组件特征搜索请求委托或直接以超文本浏览方式浏览 DCB，获得所需组件。

AND-CASE 客户可依据本身的软件需求层次及 VSR、DCR、DCA 的不同表示能力和可操作度，选择相应的表达、交互手段。

5 CSA 体的构造

CSA 体的构造引入了 Agent 机制，如图3所示。其自主性表现为 GA、LA、SA、MA 以相互关联的多线程方式各自根据内部的状态及外部事件自主决定自己的行为；其独立性表现为 GA、LA、SA、MA 各为范畴明确，可被独立引用的逻辑、过程、数据实体；其可动性表现为在 DCBG 中的遍历、漫游；其协同性则表现为各 Agent 之间的消息通讯、协作工作及知识交互学习。

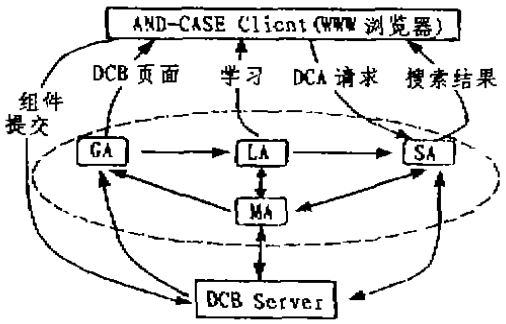

图3 CSA 体构造

GA 为 HTML(VRML)文档构造体，GA 按照 DCB 的当前组织层次动态生成相应领域框架(DF)和领域组件(DC)的描述页面。AND-CASE 客户可通过浏览器浏览页面进行观察、评价、获取，也可通过页面向 DCB 提交自己的组件。管理体 MA 一旦发生对 DCB 的操作事件，即与 GA 通讯，GA 即刻重新生成相应 DF 与 DC 的描述页面，从而保持页面与 DCB 结构、内容的同步一致。GA 对客户的有效浏览行为将保持记录，并与 LA 交互，使得 LA 及时获取有关搜索知识。

搜索知识学习体 LA 可以对客户浏览搜索的有效知识(例如某领域的某些关键词与某个超链之间通过浏览建立的联结)，进行学习和记忆，并与搜索体 SA 通讯，在下次遇到类似的搜索事件时，将利用记忆的知识直接推理获得匹配的组件。因而，除第一客户(即第一次浏览某领域，某组件的客户)外，其它客户均可避免类似的多层复杂浏览。LA 对学习到的搜索知识拟人化地自主进行积累、分类、筛选和进化，对积累的搜索知识按照"新频度"——N 度分级为多种"记忆态"动态记忆。

N 度的计算规则是：①随着使用次数的递增，N 度递增；②随着(当前时间—最近使用时间)的递增，N 度递减。

当知识 K 的 N 度大于权值 B_i时，LA 将 K 的索引相应保存在"活动态记忆体索引 I_i"中，搜索体 SA 在搜索中，依 $I_n,I_{n-1},\cdots,I_1,I_0$的次序引用索引，因而 N 度较高的 K，将较为优先、快速地被回忆、匹配。LA 自主对各 K 的 N 度刷新，对某些逐渐"陈旧"的 K，其 N 度将不断降低，当某个 K_i的 N 度$<B_0$时，K_i即被进化出"记忆索引"而被 LA"遗忘"。

搜索体 SA 接收组件搜索的请求委托事件，依据 LA 学习的搜索知识，对在 LA 记忆体中保存的满足 DCA 要求的组件，将直接从记忆体中回忆、匹配；若记忆体中满足 DCA 特征的组件有多个，则 N 度较高的将被优先匹配，供客户选择；若记忆体搜索失败，则按领域分类、N 度分级，由大到小，由新到旧逐层推理搜索。考虑 SA 搜索的效率及 DCB 的层次结构，组件按领域从属关系组织在各领域框架 DF 对象内，每个 DF 对象保存该领域的属性描述，DC 对象保存组件的属性描述，DF、DC 的对象操作则由管理体 MA 提供方法。DF、DC 按领域分支、交叉关系构成一有向无环网络图 DFCG，如图4所示。

SA 搜索中按有向无环网络图的分布式遍历算法在 DFCG 中遍历，遍历中将领域属性满足 DCA 领域特征的领域，确定为搜索范围；对某些跨领域的子 DF、DC，保存有多个父领域链(如图4中的 $D_{1k/m}$)，可以从任一父链进入子链；对在搜索范围内的组件，则按组件的特征匹配度、N 度进行搜索匹

配。特征匹配中还采用了模糊匹配及同义词辨别的处理[3]，但为降低模糊与二义距离，SA 通过与 MA 通讯获取领域的特征词典供客户表达时选择，从而使 DCA 请求描述与 DCB 中的 DF、DC 特征描述尽量一致，减少模糊、二义、同义处理的消耗，提高搜索效率与精确度。

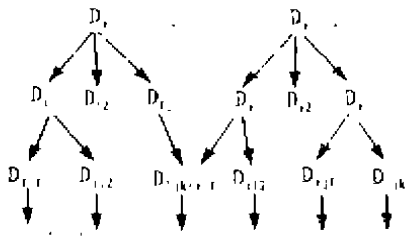

图4 DFCG/DCBG 网络图

6 DCB 层次结构及其管理体 MA

由于领域类别和组件数量的庞大，AND-CASE 作为网上分布式组件的集结站点，不可能将所有领域及组件全部包揽在 DCB 类库中，为使 AND-CASE 客户能通过 DCB 在尽可能广的范围内找寻到相应领域组件，DCB 被设计为分布式有向网络图 DFCG 的一个同构映射。DCB 中并不保存每个领域组件本身，而仅保存描述 DF、DC 各 DCA 特征属性的对象类及其子类和组件的超链，构成一个同构于 DFCG 的 DCB 分布式有向网络图 DCBG(见图4)，CSA 体搜索、遍历、漫游的动作实际上是在 DCBG 中进行的。下面是用 JAVA 语言描述的 DCB 中 DF、DC 的部分 DCA 特征属性的类形式定义：

```
/ * DCB 类结点定义 * /
class DcbNode{
    String identify; // 类标识
    String name; // 领域框架名或组件名
    URL hyperlink; // 子类或组件的超链，可有一个或多个
    String Category; // 结点类型，DF 或 DC，何种 DC
    String superclassid; // 父类名，可以有多个
    String subclassid; // 子类名，可以有多个
    int subclassnumber; // 子类数
    int accesstime; // 访问次数
    Date lastvisit; // 最近访问时间
    String keyword; // 描述关键字，可以有多个
    String outline; // 概述
    URL documentlink; // 详述文档超链
    ……
    }
```

其中，Category 表明该类描述的是 DF 还是 DC；若为 DC，又为何种 DC(如 Active X Control，Java Beans，Applet 等)，从而在多种对象标准技术[1](如 OLE，CORBA，OpenDoc)并存的情况下，DCB 具有容纳、表达多种类型组件的能力；各 DF、DC 通过 hyperlink 构成有向网络图 DCBG；outline 描述 DF 或 DC 的主要特性，功能，性能；同样是考虑容量问题，DCB 类中并不保存对象的详述文档本身，而只保存详述文档的存放位置，需要时可通过 documentlink 超链获得详述文档。

DCB 的管理体 MA 提供 DCBG 中各结点的增加、修改、删除等操作方法，并保持类描述的同一(如修改一个类后，其子类的属性也将重新予以计算；有依赖某类的类存在，则该类不许删除等)，MA 的操作方法只有具有 AND-CASE 站点类库管理权限的用户方可操作，一般客户只能通过 MA 或 GA 生成的同步页面在 DCBG 中浏览。

MA 除按 DCB 管理员的要求操作外，还采用类似 LA 管理 K 的办法自主对保存在 DCB 中的各 DF、DC 对象类依 N 度计算规则计算各类的 N 度，并在服务器空闲时依 N 度动态调整各类在库中的位置，N 度较大的，将安排在优先被浏览和搜索的位置上。采用类似内存页面淘汰的近似 LRU 算法，最近较久较少使用的类将被淘汰出 DCB，以保持 DCB 的总体 N 度及控制类数量的总体增长，保持进化与活力。同时对新提交的对象类，MA 将与同领域的原有类似对象进行比较，若特性不强于原有类，将拒绝接收。

当 MA 接收了经检验的新提交类后，及时与 LA，GA 交互，通知有"新类到达"事件，LA 即将新类索引以较高 N 度记入"记忆体索引"前端，以便 SA 及时采纳；GA 则及时生成新类的描述页面并修改相关页面(如父、子类页面)，以保持浏览页面与 DCB 内容与结构的同一。

结束语 AND-CASE 模型支持了分布式的"社会化软件工程"的复用模式，拓展了一种面向 Web 的多层次柔性 CASE 工作模式，AND-CASE 原型已在 Windows95/NT 下用 JAVA，HTML，SQL Server 等相关技术实现，从而表明了 AND-CASE 的如下特点：①支持广域网范围的分布式大型软件复用和基于组件的软件过程，②可将 Internet 上的软件资源组织为规范可复用方式。③将 WWW 的 C/S，超文本，超媒体技术应用于 CASE 中。④软件需求规范的多层次柔性表达能力。⑤Agent 机制的应用使组件搜索、管理保持效率与活力。⑥面向领域的 DCB 组织自然、精巧，具有容纳能力。

感谢汪巨涛、孙惠杰、董钢、俞晓虎、张凌铃等为本项目所做的工作。

(下转第37页)

假设-验证策略来排除伪解。本文工作利用点特征对应求解上述问题，其中深度信息嵌入比例因子中。

设 $I_{ai}(u_i,v_i)$ 为图象特征点，$M_{ai}(x_i,y_i,z_i)$ 为模型特征点，$i=1,2,\cdots,n$。且 I_{ai} 与 M_{ai} 相对应，问题具体化为：确定绕坐标轴 x、y、z 的旋转角 α、β、γ，沿坐标轴 x、y 的平移 d_x、d_y 以及比例因子 s，由图象特征点与模型特征点对应关系，可建立如下方程：

$$T_x\cdot T_y\cdot T_z\cdot R_z\cdot R_y\cdot R_x\cdot\begin{bmatrix}x_i\\y_i\\z_i\end{bmatrix}=\begin{bmatrix}x_i'\\y_i'\\z_i'\end{bmatrix}$$

其中：R_x、R_y、R_z 分别为绕坐标轴 x、y、z 的旋转矩阵，T_x、T_y 分别为沿坐标轴 x、y 的平移矩阵，T_s 为比例矩阵。上式可进一步写为：

$$T\cdot\begin{bmatrix}x_i\\y_i\\z_i\end{bmatrix}=\begin{bmatrix}x_i'\\y_i'\\z_i'\end{bmatrix}$$

该式中，$T=T_x\cdot T_y\cdot T_z\cdot R_z\cdot R_y\cdot R_x$，参数 α、β、γ、d_x、d_y、s 这六个未知因子包含于矩阵 T 中。

本文工作中，考虑到透视投影在目标距相机较远（或目标尺寸与该距离相比可以忽略）时可用平行投影近似，故此处取平行投影。即

$$P\cdot\begin{bmatrix}x_i\\y_i\\z_i\end{bmatrix}=\begin{bmatrix}-u_i\\-v_i\end{bmatrix}$$

其中，P 为投影变换矩阵。在平行投影下，$u_i=x_i'$，$v_i=y_i'$。

至此，问题转化为：确定矩阵 T 中的六个参数。因每一个特征点给出两个方程，故六个未知数共需三个对应的特征点，但这样建立的关于六个未知数的方程为非线性方程，考虑到线性代数方程求解的有效性，此处引入一额外对应特征点，通过四个特征点对应，建立八个方程，从而完成求解，为了减小误差影响，又引入了附加的对应特征点，利用最小二乘法对求解结果进行优化。实验结果表明，该种方法具有很好的三维位姿测定能力。（参考文献共19篇略）

（上接第81页）

参 考 文 献

1 King S, et al. CASE 2000: the future of CASE technology. Software Engineering Journal, 1994, 9(4): 138～140

2 扬芙清. 青鸟工程现状与发展. 96全国软件工程会议论文集. 1～8

3 冯玉琳，等. 面向对象的组合软件工程研究. 计算机学报，1996，19(3)：237～241

4 Mili M, et al Reusing software: issues and research directions. IEEE Transactions on Software Engineering, 1995, 21(6). 528～562

5 Gacek C. Exploiting domain architectures in software reuse. ACM Software Engineering, 1995, 20(8): 229～232

6 Garlan D. First international workshop on architectures for software systems workshop summary. ACM Software Engineering, 1995, 20(3) 84～89

7 Koster M. Robots in the web: threat or treat? ConneXtions, 1995, 9(4): 38～45

8 Adler R M. Emerging standards for component software. IEEE Computer, 1995, 18(3): 68～77

9 蔡智明，等. CASEE 的集成与开放. '96全国软件工程会议论文集. 79～84

10 蔡智明，等. CASE 构件的表示与分层. 合肥工业大学学报，1996，19(4). 60～63

11 蔡智明，等. 多媒体虚拟现实技巧. 北京，机械工业出版社，1996

12 刘宗田 通用历史数据库管理系统 GHDBC 的设计与实现 计算机研究与发展，1997，34(4). 281～286

13 鲍敬峰，等. 软件复用库的模糊表示和查询方法. 小型微型计算机系统，1997，18(4). 27～31

14 鲍敬峰，等. 软件规格说明综述. 软件，1996，(6)：14～19

15 袁兆山，等. 软件工程环境 EPOS 应用基础. 中国科技大学出版社，1994