基于知识的软件全程生成系统体系结构设计与综合

吴振容

(中科院沈阳计算所，沈阳 110003)

THE DESIGN AND INTEGRATION OF SOFTWARE FULL PHASES GENERATING SYSTEM ARCHITECTURE BASED ON KNOWLEDGE

Wu Zhenrong

(Shenyang Institute of Computing Technology, Academia Sinica, Shenyang 110003)

ABSTRACT

This paper presents a fundamental architecture of software full phases automated/aided generating system (SAAGS) based on software engineering knowledge base and software engineering data base; discusses mainly integritive techniques and processes of SAAGS; describes concisely the design approaches of functional components and facilities of SAAGS.

摘 要

本文提出了一个基于软件工程数据库与知识库的软件全程自动/辅助生成系统 SAAGS 的基本结构，重点讨论了 SAAGS 的综合技术和工作过程，略述了 SAAGS 功能构件及设施的构造方法.

§1. 引 言

随着软件规模和复杂度的日益增大，继续沿用手工式生命周期法已不能满足软件开发效率和软件需求变化的要求. 因此，近年来软件工程领域已逐渐将其着眼点转向软件

1990 年 5 月 23 日收到，1990 年 8 月 28 日定稿. 作者 吴振容，1985 年研究生毕业于北京软件研究中心，现为中科院沈阳计算所助理研究员. 目前，他的主要研究领域是软件工具与环境，数据库理论与技术，分布式算法，网络应用技术和专家系统.

自动辅助生成系统及其开发工具的研制上来，并推出了许多支持这一系统开发环境的自动化工具与技术，初步形成了计算机软件辅助工程学 (CASE). 但问题是当前的 CASE 产品大都面向某一开发阶段或过程，缺乏必要的综合或集成，若能将现有的工具、技术和方法集为一体，并加以适当的改造、补充和完善，使之成为面向软件全程的自动 / 辅助生成系统，无疑是具有重要意义和急待解决的课题. 同时，我们认为软件工程、知识工程、网络技术、数据库技术、声图文处理技术、第四代语言和一系列 CASE 工具已为构造 SAAGS 提供了牢固的理论和技术基础以及支撑环境.

本文提出的 SAAGS 正是一个在上述背景下产生的实验系统. SAAGS 尽可能地综合了现有的工具、方法与技术，其核心是软件工程知识库与数据库. 目前，SAAGS 主要支持 SASD 方法，其设计目标是自动生成软件需求规范；符合第三范式的数据库模式；基于功能需求的事务处理模块；基于有向加权图的软件结构与过程测试用例和各类标准化文档；并通过活动路径和功能层次结构在水平和垂直两个方向上实现项目管理与控制.

应当明确，本文重点讨论 SAAGS 的构造问题，对于 SAAGS 构件和设施所涉及的具体技术细节将在后继的文章中陆续发表.

§ 2. SAAGS 的支撑环境和开发阶段

在充分综合各种可用工具、技术和方法的基础上，我们在 XENIX 操作系统，ORACLE 数据库及其配套设施的支持下，建立了 SAAGS 的基本开发环境，并将在中科院沈阳计算所 CAD/CAM 开放实验室和东北工学院网络数据库实验室的支持下，继续改善 SAAGS 的开发环境与工具，使 SAAGS 最终将以 UNIX 操作系统，APPLICON 图形工作站，ORACLE*, RDB 数据库，DECNET 和 3^+ 乙太网为其工作环境.

当前，SAAGS 在其基本开发环境下将软件生命期的七段式约简为四段式：

1) 概念设计. 其主要任务是进行软件需求定义与分析，收集客体、明确客体关系、数据请求、功能要求和系统自动化边界，生成软件需求规范.

2) 逻辑设计. 其任务是建立系统功能和数据逻辑模型，初步划分功能模块和逻辑数据结构，生成物理设计规范.

3) 物理设计. 其主要任务是建立软件控制结构，初步优化过程模块结构，确定数据存储与查询结构，生成系统原型或试运行软件.

4) 系统测试与评价. 其任务是进行功能测试，数据测试，软件复杂度度量；增补系统容错，故障恢复和例行服务设施，完善整体软件功能.

§ 3. SAAGS 的基本结构与特点

SAAGS 主要由八类设施综合而成，即 SAAGS={U, I, O, M, S, T, R, P}. 这里，U 是用户接口，它由客体定义语言 ODL 和交互式客体定义会话语言 IOSL 组成. 客体定义语言以非过程化方式向系统导入客体定义，它是系统分析员进行系统描述的主要工具，

其功能类似于 PSL(Problem Statment Language). 交互式客体定义会话语言由数据操纵语言 DML 和图形编辑语言 GEL 组成，并统称为客体操纵语言 OML. 前者用于数据客体查询、增删与更新，后者用于图形文档编辑. 同时，客体操纵语言可以弥补客体定义语言无交互性的弱点.

I 是需求解释器，它用于客体解释，捕捉设计关键字，将客体定义语言和交互式会话语言翻译成客体目录，数据流程图，E-R 图和处理规范，生成系统需求分析文档，其功能类似于 PSA(Problem Statment Analyer).

O 是系统客体基，它由环境客体，数据客体和知识客体组成，并被分别装入软件工程数据库 SEDB 和软件工程知识库 SEKB, 由目录系统提供管理与控制信息. 具体地说，在软件工程数据库中装有环境数据客体，应用数据客体，元数据客体，需求规范，技术文档，软件版本，设计进程说明，测试用例和统计信息等. 它们大都是由客体定义语言导入系统的，或是由系统分析器自动生成的. 在软件工程知识库中装有应用领域知识与规则，系统设计知识与规则，项目管理知识与规则，自动化工具映射知识与规则. 其中，仅应用领域知识及规则与具体系统或应用对象有关，且是由系统分析员通过知识客体描述器与解释器导入系统的；而系统设计知识与规则，项目管理知识与规则，工具映射知识与规则则是由 SAAGS 设计者预先设置或定义的.

M 是基于知识的客体管理与作业调度设施，其核心构件是系统元客体目录 MOD, 它含有大量用于系统设计、管理与控制的描述信息. 利用这些信息，SAAGS 可将高层系统规范映射为低级活动，并充当 SAAGS 各构件的公共接口. 在系统元客体目录的支持下，客体解释器可将已导入系统的应用客体及其间关系的描述再次加工成数据模型和功能模型，选择与之匹配的设计模式，支撑软件，可复用模块与算法以及自动化工具. 作业调度设施根据系统元客体目录给出的作业特征描述，确定作业的优先队列，并进行作业调度. 一旦完成当前作业的初始化，便自动标识相应的自动化设计工具，抽象数据类型，应用知识与规则，设计知识与规则；提取设计模式；控制和实施当前作业；进而进行设计质量评价与复审. 一旦完成当前作业，便自动启动下一作业.

S 是系统软件集，它包括数据库管理系统 DBMS, 知识库管理系统 KBMS, 图形库管理系统 GBMS, 文件管理系统 FS, 分布式处理器 DPE, 通讯网络系统 CNS 和操作系统 OS. 综合利用和合理选择商品化系统软件所能提供的服务可大幅度降低 SAAGS 的设计成本和复杂度.

T 是自动化工具库，它由一组商品化的和自行开发的自动化工具组成，其中主要包括数据流程图自动生成工具，实体关联图自动生成工具，屏幕菜单自动生成工具，用户视图自动生成工具，数据库概念与存储模式自动生成工具，授权与保护自动生成工具，屏幕格式与报告自动生成工具，活动字典自动生成工具，功能模块与软件结构自动生成工具，测试用例自动生成工具，系统评价自动生成工具，代码原型自动生成与综合工具等. 客体分析器，作业管理与控制器，原型发生器和质量评审器根据作业阶段和对象，自动地选择和运用这些工具，并完成相应的动作.

R 是报告生成与管理器，它在标准图形库和基本文档库的支持下，按不同的需求灵活地输出各种格式的报告和标准化系统规范图文及项目管理与技术文档.

P 是常用算法库，它含有各种可复用软件过程模块和抽象数据类型. 通过语言转换

器和选择器，可将它们映射为当前作业所使用的高级语言或其它程序设计语言.

SAAGS 的总体结构如图 1 所示.

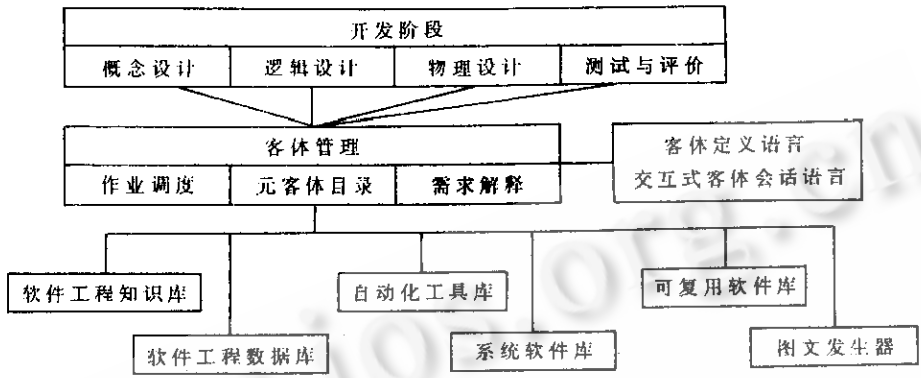

图 1

从 SAAGS 的总体结构上看，它是一个以软件工程知识库和数据库为基础，以客体管理器为核心，全面综合了各种自动化工具，系统软件和可复用模块的一体化系统. 其重要特点是： 1) 面向软件全程分析与设计. 2) 支持 SASD 方法，作业选择与设计模式匹配主要依托于系统目录和数据流程图. 3) 自动进行工具选择、支撑软件选择、可复用模块选择、程序设计语言转换及设计模式匹配；自动生成软件需求规范、数据库逻辑和存储模式、数据流程图和实体关联图；自动 / 辅助进行软件结构设计，模块优化设计和系统测试与评价.

§ 4. SAAGS 的知识提取过程

SAAGS 设有四个相对独立且分别与四个开发阶段相对应的软件工程知识库. 知识库装有软件需求规范生成规则，应用领域通用与专用规则，数据库模式组织规则，处理模块分解与重组规则，软件控制结构生成规则，图形生成与布局规则，物理限制规则，设计模式与过程选择规则，设计复杂度控制规则，抽象数据类型与算法引用规则，作业调度与作业并行性规则，综合实例与原型生成规则，系统测试、优化与评价规则等. 每一知识库的客体均按三级进行组织，第一级为元级，它包括面向领域的客体、活动及其间的关系描述，用于构造顶层系统模型. 中间级给出特定作业的知识类型描述，用于生成中层系统模型. 第三级为作业所涉及的具体事实、推理规则和技术，用于生成低层执行模块. 客体管理与作业调度器是 SAAGS 的核心，而系统元客体目录又是客体管理与作业调度器的核心. 系统元客体目录设有设计模式目录，作业优先级目录，元数据目录和公共断言目录. 设计模式目录表示数据流加工与变换，其中，数据流给出了用于变换的数据类型及其性质的描述，每一数据类型又设有一个与元数据目录相连的指针及上下文信息，以便提供设计模式的约束条件. 设计模式中含有知识提取和限定规则，提取规则用于生成数据流层次模型，限定规则给出特定的上下文条件，以便引导数据流层次的

下延. 元客体目录装有各个抽象层次的客体定义，每一客体定义和在其上下文中所要引用的限定规则均经指针引向处理该数据类型的设计模式，交叉引用元客体目录和设计模式目录有益于设计模式的选择. 作业优先目录提供作业优先队列和可能的并行操作，为作业调度提供必要的初始化信息. 断言目录按超类和子类进行组织，以便形成一个具有抽象层次的断言定义.

设计模式选择是 SAAGS 的技术关键. 我们知道，通常的程序变换系统大都采用名字映射方法，这时，用户必须给出有关设计模式匹配的信息描述，当系统含有大量程序模块时，此方案一般是不可行的. 为此，SAAGS 采用了面向领域的客体描述方法. 初始，用户仅需给出有关系统高层规范的描述，SAAGS 可根据这一描述自动建立顶层系统模型，而后，SAAGS 再根据上下文信息和进一步给出的规范信息生成中级系统模型，同时选择知识提取规则. 需求解释器通过试探各种信息类型进行客体解释，析取设计关键字和上下文关键字. 一旦数据流规范获得了满意的解释，便可通过客体定义查询设计模式，以期获得设计模式候选集. 最后，调度模块负责实施当前作业和完成设计精化以及作业复杂度控制. 作业优先队列是将数据流程图转化成 pert 图后，按拓扑排序形成的，但对于同一层次的数据流程图，若其所对应的 pert 图存在着无交叉的分支路径，则令其并行地执行设计作业. 整个过程可视作始于高层规范和由知识库制导的精化加工过程.

§ 5. 当前结果与未来的工作

在完成了 SAAGS 体系结构设计及其开发环境建设的基础上，我们还在逻辑级和物理级实现了若干基础设施. 本节仅简述这些设施的框架和设计原理，更详细的讨论将以专题形式陆续发表.

1) 客体定义语言设施 ODL. 它是 SAAGS 与用户进行通讯或对活的工具，由一组用 C 语言编写的过程组成，对用户提供高级非过程化命令. 客体定义语言由字处理模块，语法、语义检验模块，交互式图形编辑模块，客体定义及完整性检验模块，功能需求规范模块，客体操纵模块组成. 系统分析员通过用客体定义语言编写的源程序向 SAAGS 提供关于目标系统的描述信息.

2) 需求解释器 RIR. 它由客体及其关系解释模块，目录生成模块，数据流程图生成模块和处理描述生成模块组成. 其输入是用客体定义语言书写的源程序，输出是活动数据目录，数据流程图，实体关联图和功能描述文档或处理说明.

3) 数据流程图与客体目录间的双向转换器 D/DFC. 初始，系统分析员通过客体定义语言向系统导入软件需求描述信息，然后通过客体解释器生成符合数据库模式设计规则的客体目录，并按布局规则，源目（客体的来源和去向）完整性规则，无损分解规则，接口边界规则，合理冗余规则，模块互连规则、加标规则和援引规则提取图形库中的标准图素和图表格式，将客体目录转化成数据流程图，同时将各类检验结果信息反馈给用户. 系统分析员根据已形成的客体目录或数据流程图决定是否需要进一步更改或补充. 若需要，他即可以修改用客体定义语言书写的源程序，又可以通过修改客体目录，还可以通过修改数据流程图来实现上述目的，同时完成同步转换. 具体地说，若修改的是客

体目录，则要将修改部分逆映射到数据流程图上，反之亦然，并将修改命令转化成客体定义语言，同时填补到原有的用客体定义语言书写的源程序中．这样，只要保留源程序的各个版本便会获得整个软件需求的精化过程．可见，在需求定义阶段，客体目录与数据流程图之间要发生频繁的双向转换．在一般情况下，客体目录主要用于系统规范的内部表示，数据流程图作为系统的外部表示（主要用于通讯和人工评审目的）．为简化设计，客体操作语言采用 SQL．图形编辑命令则是针对数据流程图自行编写的，其功能完备性与 SQL 相等价．图形编辑命令主要有数据流走向更新，数据流名称更新，模块名及其互连更新，数据源目更新，模块与数据流分解与组合等．在进行客体目录与数据流程图间的互转时，SAAGS 可提供一致性，相容性检验信息．总之，D/DFC 提供了两种修改系统逻辑模型的手段．为提高效率，系统分析员最好先采用客体定义语言来进行系统描述，生成初步客体目录和数据流程图，然后再通过图形编辑命令语言或 SQL 语言来完善需求描述．

4) 基于客体目录的数据库模式生成器 DSG．它根据客体目录所提供的客体及其关联的信息导出初步实体关联图，然后通过最小覆盖过程将其转化成基本实体关联图，最后根据依赖保持规则、分解和无损联接规则将基本实体关联图转化成符合 3NF 或 BC-3NF 的关系模式，从而完成概念模式设计．同理，根据客体目录所提供的用户或用户组对数据的需求信息可以获得数据库的外模式；根据客体目录所提供的数据查询特征（如响应速度，发生频度，执行效率等）可确定数据的存储结构，查询结构．在分布式环境下，还要进一步划分逻辑数据片段，确定网上分布及分布连接策略．当数据库的三级模式确定之后，要随之生成数据模式目录．

5) 基于有向加权图的软件测试工具 ST．它将软件结构和模块结构分级转化成块结点有向加权图．图中的每一结点代表一个无分支的程序段，边值为分支条件和边界测试数据．然后按深度优先遍历分级进行程序路径测试．静态测试用例与规则由软件结构和模块结构目录提供，动态测试用例和规则由功能规范目录与专用测试用例发生器联合生成．测试过程递归地进行，先在软件结构级，然后在程序模块级，最后在程序结构级．测试结果为无用变量、未定义变量、重复定义变量、不可达分支和过程、无用语句、软件模块含量、变量个数、模块尺寸、模块间数据传输类别与总量，耦合级别和软件复杂度．

6) 软件模块结构生成与优化器 STG．它先根据数据流程图和/或客体目录生成初步处理模块及其控制结构，并构造模块关联矩阵（用于计算模块间的数据传输量，评价模块间的耦合度），先驱处理模块矩阵（用于计算一模块的先行处理模块），偏可达矩阵（用于计算一模块的高阶先驱模块，以便提供模块的可重组信息，即判定模块间是并行关系还是直接线性关系），可重组矩阵（它由先驱处理模块矩阵和偏可达矩阵导出，用于计算实际上可重组的模块）和时序矩阵（用于计算模块的最早开工时间和最迟开工时间）．根据上述矩阵和模块耦合度度量准则（经形式化定义和量化）确定具高内聚和低耦合的模块结构，使模块具有高独立性，可复用性和信息隐藏性．最后，根据模块共享规则和控制域大于作用域规则以及软件层次深度控制规则简化和调整软件结构．

在下一阶段，我们将进一步完善上述设施，使之商品化，同时改善 SAAGS 的支撑环境和人力投入，有计划地完成下述工作：

1) 改造数据流程图为信息流程图，即在数据流程图中扩展控制流分量，丰富数据流程图操纵命令，为实现系统逻辑设计向物理模型的平滑过渡建立基础条件.

2) 改进实体关联图为增强型实体关联图，为建立面向客体的高级数据模型奠定基础，使 SAAGS 可以面向更广泛的应用领域.

3) 将 SAAGS 扩展到网络数据库环境，增设数据分布处理器，事务原子性控制器，生成可在 3+ 乙太网和 ORACLE/RDB 环境下运行的系统原型.

4) 增设 SAAGS 与 APPLICON CAD 图形工作站和分布式处理器 ORACLE* 的接口，并使 SAAGS 运行在 UNIX 操作系统下，以改善和拓宽 SAAGS 的运行与应用环境.

5) 研制基于属性的客体基设施，使原有的异构应用软件能够方便地集成到网络数据库环境，以便构成功能更为强大的联合式系统.

6) 增设限定性自然语言理解器与客体描述语言的接口，建立更为理想的人机界面.

7) 改进 SAAGS 的体系结构，使其成为可以面向多种设计方法学的开放式系统.

结语： SAAGS 是一较大的软件实验工程，其所涉及的理论与方法一直是国内外的热门课题，有些尚处于应用研究阶段. 尽管 SAAGS 从应用基础研究的角度是一可喜的阶段成果，但真正达到实用水平尚有一定距离. 因此，我们诚恳地希望工作在这一领域的软件工程专家给予支持和指导，使 SAAGS 最终走出实验室，成为实用的大型应用软件开发工具系统. 两年前，这一工作曾得到英国城市大学 P. D. Robert 教授的好评和关注，提出了许多建设性意见，在此表示衷心感谢.

参考文献

[1] Doker, T W G and Tate, G, "Executable Data Flow Diagrams", Software Engineering, 1986.

[2] Albano, A, Gardelli, L and Orsini, R. Galileoia, "Strongly Typed, Interactive Conceptual Language"ACM Trans, Database System, Vol. 10, No (June 1985).

[3] Mccabe, T and Schulmeyer, CG, "System Testing Aided by Strucured Analysis: A Practical Experience", IEEE Trans. Software Engin. Vol. SE–11, No. 9.

[4] Trenkel, K. A, "Toward Automating the Software-Development Cycle", Comm, ACM Vol. 28, No. 6(June 1985).

[5] N. Giddings and T. Colbum, "An Automated Software Design Evaluater", in proc, Annu, conf. ACM 84 Sun Francisco, CA, Oct.,1984.

[6] Balzer, R., T. E Cheatham, and c. Green, "Software Technology in the 1900's: Using a New Paradigm", IEEE Computer, Vol. 16, No. 11, November, 1983.

[7] Pangalos. G., "Information-Oriented Approach to Structured Analysis and Design", Inf. Age Vol. 8, No. 1, (January 1986).

[8] Mehdi. T. Harandi and Mitchell. D. Lubars, "Knowledge-Based Software Development: A Paradigm and Tool", AFIPS Conf proc Vol. 55, 1986. National Computer conf 1986.

[9] Wasserman, "AI Automated Tools in the Information System Development Environment" from Trends in Information System North-Holland (1986).

[10] Hoffnagie, G F and Beregi, W E, "Automating the Software Development Process", IBM Syst. J. Vol. 24, No. 2 (1985).

[11] Wu zhenrong, Zheng huaiyuan, "A Model of Software Full Scale Automated & Aided Generating System (SAAGS)", International 88 Conferance Modelling & Simulation Nov.,7–9. 1988.