

基于不动点转移的 SA 动态演化模型

王映辉^{1),2)} 刘瑜³⁾ 王立福²⁾

¹⁾(陕西师范大学计算机学院 西安 710062)

²⁾(北京大学信息科学技术学院软件研究所 北京 100871)

³⁾(北京大学信息遥感与地理信息系统研究所 北京 100871)

摘要 构造性和演化性是软件的两个基本特性。而软件演化包括静态演化和动态演化两个方面。动态演化更为复杂,这种复杂性决定了,对动态演化的研究首先应从宏观层面入手。软件体系结构 SA 作为软件的蓝图和支撑骨架,为人们宏观把握软件的动态演化提供了一条有效的途径。该文描述了构件——连接件组成的 SA 动态语义网络模型,分析了 SA 动态语义网络模型中的浸润过程,给出基于不动点的浸润过程收敛的判定,提出了邻接矩阵过滤和原子过滤的概念,阐明了基于邻接矩阵原子过滤的 SA 动态语义网络浸润步的原子性。指出 SA 动态演化过程可用一系列邻接矩阵原子过滤在时刻上相继的逻辑衔接来描述。最后给出了两个层面上对 SA 动态演化波及效应的分析方法。为基于矩阵变换的 SA 动态演化的进一步研究和计算机自动量化描述奠定了基础。

关键词 软件体系结构; 动态演化; SA 动态语义网; 邻接矩阵; 浸润; 不动点

中图法分类号 TP311

SA Dynamic Evolution Model Based on Static-Point Transition

WANG Ying-Hui^{1),2)} LIU Yu³⁾ WANG Li-Fu²⁾

¹⁾(School of Computer Science, Shaanxi Normal University, Xi'an 710062)

²⁾(Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871)

³⁾(Institute of Remote Sensing and Geographic Information Systems, Peking University, Beijing 100871)

Abstract Construction and evolution are two basic properties of software. Software evolution includes two aspects of static & dynamic evolution. Software dynamic evolution is more complex, this complexity decides that research of software dynamic evolution should be begun with macroscopical level. Software architecture SA acts as blueprint and skeleton of software, it is an availability approach for people to grasp macroscopical software architecture and evolution based on SA. SA dynamic semantic network model based on components and connectors is described. Soak process in SA dynamic semantic network is analyzed, and decision condition about convergence of soak process is addressed. A concept of adjacency matrix atomic filtration and atomic property of soak step based on adjacency matrix atomic filtration is put forward, so SA dynamic evolution is described by a series of continuous adjacency matrix atomic filtrations. At last, an analysis approach of ripple-effect of SA dynamic evolution is described based on two of levels. All the above are credible foundation of SA dynamic evolution research in the future, and are automatic transition foundation based on matrix in computer.

Keywords software architecture; dynamic evolution; dynamic semantic net; adjacency matrix; soak; static-point

收稿日期:2004-04-06;修改稿收到日期:2004-09-29. 本课题得到国家“八六三”高技术研究发展计划项目基金(2001AA113171)、国家“九七三”重点基础研究发展规划项目基金(2002CB312006)、国家博士后基金(20040350251)和陕西省自然基金(2003F35)资助。王映辉,男,1967 年生,博士,副教授,目前的研究方向为软件体系结构与软件演化技术。E-mail: wyh_925@163.com。刘瑜,男,1971 年生,博士,副教授,目前的研究方向为 GIS 理论与 GIS 软件与构件技术。王立福,男,1945 年生,教授,博士生导师,目前的研究方向为软件工程、软件框架和软件体系结构。

1 引言

构造性和演化性是软件的两个基本特性^[1]。软件进行渐变并达到所希望的形态就是软件演化。软件演化由一系列复杂的变化活动组成,所以,对软件演化进行理解和控制显得比较困难。软件演化的复杂性决定了对软件演化研究首先应从宏观层面入手,这样可以避免陷入软件演化研究的复杂细节中。

近几年,软件体系结构 SA(Software Architecture)已成为软件研究的热点之一,它作为软件设计的高层抽象,为人们宏观把握软件的整体结构提供了一条途径^[2~4];另外,SA 作为软件生命周期的早期产品,着重解决软件系统的结构和需求向实现平坦过渡的问题,是软件生命周期中开发、集成、测试和维护阶段的基础;加之对 SA 检测和修改的相对低代价性^[2],所以深入研究 SA 的演化是非常必要的。

按照演化发生的时间,SA 演化被划分为静态演化和动态演化两个方面。对 SA 在非运行时刻的修改和变更称为 SA 的静态演化(如软件版本的升级等),而软件在运行时刻的 SA 变换称为 SA 的动态演化。按照演化的内容,SA 演化又可分为结构演化和语义演化两个方面。目前部分软件体系结构描述语言 SADL(Software Architecture Description Language)支持软件演化表述,其中 Darwin 和 C2 都支持 SA 的动态结构演化,CHAM, Wright, Rapide 支持 SA 的动态语义演化^[5,6~10]。由于这些以形式化为基础的方法是以“构造”而不是以“演化”为目标的,所以对刻画“演化”存在先天的不足。文献[11]给出了基于可达矩阵的 SA 演化波及效应分析模型,特别对 SA 静态演化中的波及效应进行了描述,并给出了量化界定方法,但未给出具体的 SA 动态演化刻画模型。为此,本文在动态语义网及其浸润理论的基础上^[12,13],提出了基于不动点转移的 SA 动态演化模型。

本文第 2 节根据构件——连接件,给出了 SA 动态语义网模型;第 3 节描述了 SA 动态语义网中的浸润过程和机理;第 4 节对基于动态语义网的 SA 动态演化过程及其波及效应进行了分析;第 5 节总结本文。

2 SA 动态语义网

在文献[11]中,为了研究方便,将 SA 简单地定

义为组成软件系统的构件以及构件之间交互方向关系的抽象。这种简化 SA 模型可以较好地刻画 SA 的静态拓扑结构,为 SA 静态演化中波及效应分析提供良好的支撑。然而,SA 的动态演化,特别是动态语义演化,由于是立足于更为复杂的动态运行环境之中,该简化模型已不能对其提供有效的支持。本文在赋予 SA 更为丰富语义的基础上,结合动态语义网及其浸润理论来描述和刻画 SA 的动态演化。

2.1 SA 网络模型

对构件和连接件的定义参考了文献[12]中的内容。

定义 1. 构件 Com 是系统中承担一定功能的数据或计算单元。 $Com = \langle Ports, Imp_Bs \rangle$, 其中 $Ports$ 是构件接口集合, Imp_Bs 是构件的实现。 $Ports = \{Port_1, Port_2, \dots, Port_n\}$, 而 $Port_i = \langle ID, Publ_i, Ext_i, Prvt_i, Beha_i, Msg_i, Cons_i, NFun_i \rangle$, 其中, ID 为构件标识, $Publ_i$ 是 $Port_i$ 向外提供的功能的集合, Ext_i 是外部通过 $Port_i$ 向构件提供的功能的集合, $Prvt_i$ 是 $Port_i$ 的私有属性集合, $Beha_i$ 是 $Port_i$ 的行为语义描述, Msg_i 是 $Port_i$ 产生的消息的集合, $Cons_i$ 是 $Port_i$ 的行为约束, $NFun_i$ 是 $Port_i$ 的非功能说明。

定义 2. 连接件 Con 是系统中承担构件间交互语义的连接运算单元。 $Con = \langle ID, Beha, Msg, NFun, Cons, Role \rangle$ 。其中, ID 为连接件的标识, $Beha$ 是连接件行为语义的描述, Msg 是构件与各 $Role$ 交互事件产生的消息的集合, $NFun$ 为连接件的非功能描述, $Cons$ 是连接件约束的集合, $Role$ 是连接件与构件交互点的集合。

连接件具有连接的方向性和连接的角色性。前者是指连接件的任何一端可进行单向或双响请求传递,后者是指参与连接一方的作用和地位,有主动和被动或请求和响应之分。从 SA 演化的角度来看,只有一端(接口)服务连接在系统中的连接件是无意义的。

以上对构件和连接件的描述充分体现了它们的动态行为和动态语义特性。

定义 3. SA 是由构件通过连接件及其之间的语义约束形成的拓扑网络 $N_{SA} = \langle Coms, Cons, Const \rangle$ (称 SA 网络或 SA 网络模型), 其中 $Coms$ 是构件的集合, $Cons$ 是连接件的集合, $Const$ 是构件与连接件之间的动态语义约束。

推论 1. SA 网络模型 N_{SA} 能用有向图刻画。

在文献[11]中为了重点研究 SA 的静态演化,

将 SA 模型简化成了构件及其构件间交互方向关系的组合, 并将构件间的约束赋予了连接件, 较好地描述了系统的静态结构。由于此结构实质上是一个有向图, 所以用有向图对应的矩阵变换运算对 SA 静态演化中波及效应进行了较好的刻画。但是, 描述 SA 动态演化必须依赖于 SA 更丰富的动态语义。也就是说在 SA 的动态演化研究中, 要求 SA 模型不仅具有刻画静态结构特性的能力, 还应具有描述构件状态变化、构件间通过连接件的相互作用和局部网络的集群行为等动态特性的能力。因此, 需要将构件间的相互作用与约束细化为构件与连接件间的相互作用与约束。可见, 定义 3 与文献[11]中 SA 的定义实质上是一致的。

定义 4. 在 SA 的动态演化中, 可以设想, 信息流(数据流或控制流)是通过 N_{SA} 中的连接件在构件之间流动的, 当这种流动到达某一构件时, 继续驱动目标构件, 进而辐射出去或终止。这种反复出现的过程称为 N_{SA} 浸润。

2.2 SA 动态语义网

定义 5. 对被考察的系统 S 形式化后的动态语义网^[5]是一个有向图 $G_S = \langle V(G_S), E(G_S), F_S \rangle$, 其中 $V(G_S)$ 是策动源; $E(G_S)$ 是浸润传播途径集; F_S 是 $V(G_S)$ 到 $E(G_S)$ 的有序集合上的函数, 表示某种可预定义的形式化语义联系。

定理 1. N_{SA} 与 G_S 同构。

证明. 略。

定义 6. 在 SA 的动态演化中, 将 N_{SA} 称为 SA 动态语义网, 记为 G_{SA} 。

可见, SA 动态语义网也是一个有向图 $G_{SA} = \langle V(G_{SA}), E(G_{SA}), F_{SA} \rangle$, 其中 $V(G_{SA})$ 是策动源和被激构件集; $E(G_{SA})$ 是浸润传播途径集; F_{SA} 是 $V(G_{SA})$ 到 $E(G_{SA})$ 的有序集合上的函数, 表示某种可预定义的形式化语义联系。

3 SA 动态语义网中的浸润过程

动态语义网的浸润过程^[13]可以抽象为有向图结点间的数据或控制信息的驱动, 即是一个由源结点策动的传播。

定义 7. 浸润域是浸润发生过程的载体, 它是一个有向图 $G = \langle V(G), E(G), F \rangle$, 其中 $V(G)$, $E(G)$, F 含义类似于定义 5。

推论 2. 在 G_{SA} 上同样可以定义与定义 7 完全相同含义的浸润域。

以下概念的描述是在的浸润域 G_{SA} 上进行的, 不再做特别说明。

定义 8. 策动源 $Act_Nodes^{(t)} = \{V_i | V_i \in V(G) \wedge S_i = 1 \wedge 0 \leq i < |V(G)|\}$, 即 t 时刻 $V(G)$ 中状态为 1 的所有结点构成的集合。 S_i 表示 V_i 的状态, 定义为

$$S_i = \begin{cases} 1, & F_i(x_1, x_2, \dots, x_n) > \theta(x_1, x_2, \dots, x_n) \\ 0, & \text{其它} \end{cases},$$

其中, $F_i(x_1, x_2, \dots, x_n) \in R$ 是 V_i 的激励变换函数; $\theta_i(x_1, x_2, \dots, x_n) \in R$ 是 V_i 的门限函数。

可见 $S_i \in \{0, 1\}$, 其中 0 表示抑制, 1 表示活跃; $x_i (1 \leq i \leq n)$ 表示结点 V_i 的输入。

另外, 如果定义 $Y_i = F_i(x_1, x_2, \dots, x_n) / S_i$ 作为结点 V_i 的输出; 可见, 当 S_i 抑制时 Y_i 是无意义的。

定义 9. 图 G 的邻接矩阵

$$\mathbf{X} = (x_{ij}), 0 \leq i, j < |V(G)|,$$

其中,

$$x_{ij} = \begin{cases} 1, & S_i = 1 \wedge \langle i, j \rangle \in F_G(E(G)) \\ 0, & \text{其它} \end{cases};$$

$F_G(E(G))$ 表示图 G 对应的边集 $E(G)$ 中对应的所有边的两端结点之间的某种可预定义的语义联系的集合。

t 时刻图 G 的全部结点状态的集合表示为

$$S^{(t)} = \{S_i | V_i \in V(G), 0 \leq i \leq |V(G)|\}.$$

定义 10. F_i 对整个矩阵 \mathbf{X} 的操作等价于对矩阵 \mathbf{X} 中每个元素的操作。同样 θ_i 对整个矩阵 \mathbf{X} 的操作等价于对矩阵 \mathbf{X} 中每个元素的操作。

定义 11. 浸润算子 Soak 是在时刻 t 起作用的、限定 F_i 和 θ_i 对 $X^{(t)}$ 进行一个离散时间步长的、逻辑上并行运算的操作。

定义 12. 一个浸润步是指浸润算子 Soak 作用在三元组 $\langle Act_Nodes^{(t)}, X^{(t)}, t \rangle$ 上的结果:

$$\langle \bigcup_{V_i \in Act_Nodes^{(t)}} \{V_j | \langle i, j \rangle \in F_G(V(E)) \wedge F_i(X^{(t)}, S^{(t)}) > \theta(X^{(t)}, S^{(t)})\}^{(t+1)}, X^{(t+1)}, t+1 \rangle,$$

记为 $Soak_Step^{(t+1)}$ 。

性质 1. $Soak_Step^{(t+1)} = Soak(Soak_Step^{(t)})$, $t=0, 1, \dots, N$.

定义 13. 浸润过程是有限个时刻相继的浸润步的逻辑衔接。

定义 14. 在浸润过程中, 当 t 到达某一时刻 T 时, 如果出现了 $X^{(T+1)} = X^{(T)}, S^{(T+1)} = S^{(T)}$, 则称此时刻的点为不动点。

定义 15. 存在不动点的浸润过程称为收敛的, 否则就称为不收敛的。

定理 2^[13]. 一个浸润过程收敛的充要条件是浸润中的每个结点 V_i 的 F_i 当 $t=T$ 足够大时收敛于 θ_i .

证明. 参阅文献[13].

定义 16. 在 G_{SA} 的浸润过程中, 当 t 足够大到 T 时, 如果所有结点的状态和对应的邻接矩阵的元素全不为 0, 则称 SA 是最优的.

可见, 在最优的 SA 中, 各构成元素之间“紧密”合作, 表现为内聚性非常高.

4 SA 动态演化分析

SA 的动态演化分析比静态演化分析复杂, 建立 SA 动态演化过程模型是动态分析的关键. 本文对 SA 动态演化过程模型的建立满足两个要求, 一是与文献[11]中的静态模型相一致, 二是采用与文献[11]中不同的层面(基于 G_{SA} 的浸润过程)和更为丰富的语义进行刻画.

4.1 SA 动态演化的基本活动

定义 17. SA 特定条件域 U_i 是指与 SA 相关的软件系统在一次运行中所对应的环境和约束条件集.

在 SA 特定条件域 U_i 下, SA 对应的软件系统在运行中数据和控制信息的流动总是在若干个确定的构件和连接件之间传播, 这若干个确定的构件和连接件组成了新的 SA, 称为 SA 特定条件域 U_i 下的 SA, 记为 SA_i .

定义 18. SA 条件域 U 是指软件系统所有可能的 SA 特定条件域 U_i 的并集, 即 $U = \bigcup_{i=1}^N U_i$, N 为 U 的非空子集的个数.

定义 19. $\forall U_i \in U, \exists SA_i \subseteq SA (i=1, 2, \dots, N, N$ 为 SA 条件域 U 的非空子集的个数), 称为不动点软件体系结构 SA_i , 简称不动点 SA_i . 其中 $SA_i \subseteq SA, N_{SA_i} = \langle Coms, Cons, Const \rangle, N_{SA_i} = \langle Coms_i, Cons_i, Const_i \rangle$, 且 $Coms_i \subseteq Coms, Cons_i \subseteq Cons, Const_i \subseteq Const$.

定义 19 说明, 当 SA 特定条件域确定之后, 对应的 SA_i 也是确定的. 从 G_{SA} 浸润的全过程来看, $SA_i \subseteq SA$ 相当于 SA 在此特定条件域下的一个稳态.

定义 20. 在一个 SA 中, 自一个不动点 SA_i 到另一个不动点 $SA_j (i, j=1, 2, \dots, N, N$ 为 SA 条件域 U 的非空子集的个数) 之间的变换称为 SA 的不动点转移.

可见, 软件运行环境条件组合的变化决定了 SA 的不动点转移, 即 SA 条件域 U_i 的变化决定了 SA 的不动点转移. 另外, SA 的动态演化过程可用在 G_{SA} 的浸润过程来刻画, 而浸润步是 SA 动态演化的基本活动.

通过不动点的转移, SA 由一种形态变为另外一种形态. 实际上是: 软件在特定的环境条件下(包括运行的硬件环境、输入和使用人员等), 软件只有部分成分(构件和连接件)在起作用. 也就是说, 数据或信息只在部分成分中流动, 且决定了在这种特定环境下的局部的 SA, 即不动点 $SA_i (i=1, 2, \dots, N, N$ 为 SA 条件域 U 的非空子集的个数).

定义 21. SA_k 对应的邻接矩阵为 $X_k = (x_{ij})$, $0 \leq i, j < |V(G_{SA})|$, 其中

$$x_{ij} = \begin{cases} 1, & S_i = 1 \wedge \langle i, j \rangle \in F_G(E(G_{SA})) \wedge \\ & Con_i \in SA_k \wedge Con_j \in SA_k \\ 0, & \text{其它} \end{cases};$$

$F_{G_{SA}}(E(G))$ 含义与定义 9 中相同, $Con_i \in SA_k$ 和 $Con_j \in SA_k$ 分别表示构件 Con_i 和 Con_j 都属于 SA_k 中的构件. 而 $k=1, 2, \dots, N, N$ 为 SA 条件域 U 的非空子集数.

设不动点 SA_i 对应的邻接矩阵为 X_i , 则 $X = \bigvee_{i=1}^N X_i$, \vee 表示矩阵元素的“逻辑或”运算. 例如, 当 $N=3$ 时, 设

$$X_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X_2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}, X_3 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

则

$$X = \begin{bmatrix} 1 \vee 0 \vee 1 & 1 \vee 0 \vee 1 & 1 \vee 1 \vee 0 \\ 1 \vee 1 \vee 1 & 0 \vee 0 \vee 0 & 0 \vee 0 \vee 1 \\ 0 \vee 1 \vee 1 & 0 \vee 0 \vee 1 & 1 \vee 1 \vee 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

定义 22. SA 条件域 U 的每一个非空子集 $U_j (j=1, 2, \dots, N)$ 对应的一个 X_j 称为 X 的一个过滤, 简称 X 过滤, 且 X_j 和 X 的阶数相同, $X = \bigvee_{i=1}^N X_i$, \vee 表示矩阵元素的“逻辑或”运算, N 为 SA 条件域 U 的非空子集的个数.

按照定义 22 中“逻辑或”运算规则, 将 X 进行“拆分”, 可得 M 个 X 过滤.

性质 2. 互异的 X 过滤的个数 M 最大为 $2^{X \text{ 中元素个数}}$. 当 SA 条件域 U 非空子集的个数 N 大于 $2^{X \text{ 中元素个数}}$ 时, 对应的 X_1, X_2, \dots, X_N 中有相同的.

性质 2 也意味着, 通过 X “拆分” 获取的每一个 X 过滤并不都是有意义的. 也就是说, 这种“拆分”下

的每个 \mathbf{X} 过滤对应的 SA 不总是实际存在的.

定义 23. 设 G_{SA} 对应的邻接矩阵 $\mathbf{X} = (X_{ij})$, $0 \leq i, j < N$, $N = |V(G_{SA})|$, 称 $X_{11}, X_{12}, \dots, X_{1n}, X_{21}, X_{22}, \dots, X_{2n}, X_{n1}, X_{n2}, \dots, X_{nn}$ 为 \mathbf{X} 的一个序列, 其中 $X_{ij} \in \{0, 1\}$.

定义 24. 对 \mathbf{X} 的一个序列按 0 和 1 的组合值从小到大进行编码, 形成一个编码序列, 称为 \mathbf{X} 的编码序列. 在 \mathbf{X} 的编码序列中, 除第一个元素之外的每个元素都有前驱, 除最后一个元素之外的每个元素都有后继. 对应的 0 和 1 的组合值称为 \mathbf{X} 编码序列元素的序号.

如, 对于 \mathbf{X} 的一个序列 $X_{11}, X_{12}, X_{21}, X_{22}$, 即 \mathbf{X} 为 2×2 的矩阵, 按 0 和 1 的组合从小到大编码序列为: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 则元素 1001 的前驱为 1000, 而后继为 1010, 序号为十进制的 9; 其它依次类推.

定义 25. 如果将 \mathbf{X} 编码序列中元素的序号当作 \mathbf{X} 的下标, 并用此元素中的 0 或 1 代替矩阵 \mathbf{X} 中相应元素的值时, 形成了一个矩阵系列 $\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_{2^{N \times N}-1}$ (N 为 \mathbf{X} 矩阵的阶数) 称为 \mathbf{X} 的过滤系列. 每个 \mathbf{X}_i 称为一个 \mathbf{X} 原子过滤, 简称原子过滤 \mathbf{X}_i .

例如, 上例中 \mathbf{X} 的编码序列中的元素 1001, 对应的 $\mathbf{X}_9 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, 元素 1110 对应的 $\mathbf{X}_{14} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$.

可见, \mathbf{X} 过滤系列中的元素是矩阵, 除了第一个和最后一个元素之外, 每个元素也有前驱和后继.

推论 3. SA 动态演化过程可用一系列在时刻上相继的 \mathbf{X} 原子过滤的逻辑衔接来刻画.

推论 3 给出了 SA 动态演化过程的量化矩阵描述模型. 由此可见, 通过矩阵的变换运算^[14], 可以对 SA 动态演化过程进行较好的量化分析和描述.

另外, 由于每一个特定的 \mathbf{X} 原子过滤对应一个特定的 SA. 所以, 对此特定的 SA 静态演化波及效应分析可利用文献[11]中相关的方法.

此外, 从不动点转移的角度看, SA 静态演化是 SA 动态演化的特例; 但从本文中对演化过程的描述层次来看, SA 静态演化是 SA 动态演化的一个子过程. 所以, 文中的 SA 动态演化分析模型和文献[11]中的静态演化分析模型具有一致性.

4.2 SA 动态演化中的波及效应

文献[11]基于矩阵变换和运算, 对 SA 静态演化中的波及效应进行了较好的分析, 它是在保持 SA 在结构上稳定(基于一个不动点)的基础上, 对

SA 演化的波及效应进行分析的. SA 动态演化中的波及效应比静态演化中的波及效应更为复杂, 所以应在不同的层面上进行描述.

为了宏观地把握 SA 演化的动态性, 假定被研究系统的 SA 静态结构是不变的, 且构成了稳定的网状通道, 在系统运行时, 信息经过这个网状通道传播, 即信息的流动范围随着运行状态的变化而扩张或收缩, 这种扩张或收缩反映了系统运行在 SA 特定条件域下 SA 形态的变化, 这种形态的变化可用 \mathbf{X} 原子过滤及其序列来描述.

由此可见, 对 SA 动态演化波及效应分析是基于两个层面的: 第一, 通过不动点转移所构成的不动点转移链上的波动分析; 第二, 在特定的不动点支撑下的 SA 静态演化过程的波及效应分析. 不动点转移链上的波及过程实质上可由一系列的 \mathbf{X} 原子过滤来量化描述, 而特定不动点支撑下的 SA 静态演化波及效应的分析和量化界定可参阅文献[11]中的方法.

5 结束语

软件演化技术是软件需求和维护中的关键技术^[15, 16], 而 SA 作为软件的支撑骨架, 为软件开发提供了统一的规约环境. 所以, 对软件的维护和变更首先表现在对 SA 的维护和变更之上. 近几年来, 对软件变化的研究呈现不断的上升趋势, 但专门针对 SA 演化的研究, 特别是模型支撑下的定量研究相对比较贫乏. 本文基于 SA 的构件——连接件动态语义模型, 建立了 SA 动态语义网 G_{SA} , 凭借动态语义网中的浸润理论和局部稳定(不动点)特征, 分析了 SA 动态语义网的浸润过程, 并给出了 SA 不动点转移的概念和 \mathbf{X} 过滤以及 \mathbf{X} 原子过滤的概念. 最后指出, SA 动态演化过程可用一个矩阵(\mathbf{X} 原子过滤)序列来刻画和描述. 这一切为基于矩阵变换的 SA 动态演化的进一步研究和计算机自动量化处理奠定了基础.

进一步的研究工作包括: 在 G_{SA} 上浸润过程收敛的判定; 与 SA 动态演化对应的有实际意义的 \mathbf{X} 原子过滤序列的确定; 基于 \mathbf{X} 过滤和 \mathbf{X} 原子过滤序列的 SA 演化过程的评估; 计算机辅助 SA 演化工具的研制和应用等.

致谢 非常感谢杨英清院士提供的博士后研究条件. 同时也感谢何新贵院士在浸润理论文献方面提供的支持.

参 考 文 献

- 1 Liu Yu, Zhang Shi-Kun, Wang Li-Fu, Yang Fu-Qing. Component-based software frameworks and role extension form. *Journal of Software*, 2003, 14(8): 1364~1370(in Chinese)
(刘瑜, 张世琨, 王立福, 杨芙清. 基于构件的软件框架与角色扩展形态研究. 软件学报, 2003, 14(8): 1364~1370)
- 2 Bass L., Clements P. C., Kazman R.. *Software Architecture in Practice*. Aoston, MA: Addison-Wesley, 1998
- 3 Mei Hong, Chen Feng, Feng Yao-Dong *et al.*. ABC: An architecture based, component oriented approach to software development. *Journal of Software*, 2003, 14(4): 721~732(in Chinese)
(梅宏, 陈峰, 冯耀东, 杨杰. ABC: 基于体系结构、面向构件的软件开发方法. 软件学报, 2003, 14(4): 721~732)
- 4 Zhang Shi-Kun, Wang Li-Fu, Yang Fu-Qing. Software architecture style based tier message bus. *Chinese Science, Series E*, 2002, 32(3): 393~400(in Chinese)
(张世琨, 王立福, 杨芙清. 基于层次消息总线的软件体系结构风格. 中国科学(E辑), 2002, 32(3): 393~400)
- 5 Sun Chang-Ai, Jin Mao-Zhong, Liu Chao. Overviews on software architecture research. *Journal of Software*, 2002, 13(7): 1228~1237(in Chinese)
(孙昌爱, 金茂忠, 刘超. 软件体系结构研究综述. 软件学报, 2002, 13(7): 1228~1237)
- 6 Clements P. C.. Formal methods in describing architecture. In: *Proceedings of Workshop Formal Methods and Architecture Conference*, 1995. <http://www.ieee.org>
- 7 Kogut P., Clements P. C.. Feature analysis of architecture description languages. In: *Proceedings of Software Technology Conference*, 1995. <http://www.ieee.org>
- 8 Medvidovic N., Richard N. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 2000, 26(1): 70~93
- 9 Allen R., Garlan D.. A formal basis for architectural connection. *ACM Transactions on Software Engineering and Methodology*, 1997, 6(3): 213~249
- 10 Luo Hua-Jun, Tang Zhi-Song, Zheng Jian-Dan. An visual ADL: XYZ/ADL. *Journal of Software*, 2000, 11(8): 1024~1029(in Chinese)
(骆华俊, 唐雅松, 郑建丹. 可视化体系结构描述语言 XYZ/ADL. 软件学报, 2000, 11(8): 1024~1029)
- 11 Wang Ying-Hui, Zhang Shi-Kun, Liu Yu, *et al.*. Ripple-effect analysis of software architecture evolution based reachability matrix. *Journal of Software*, 2004, 15(8): 1107~1115(in Chinese)
(王映辉, 张世琨, 刘瑜, 王立福. 基于可达矩阵的软件体系结构演化波及效应分析. 软件学报, 2004, 15(8): 1107~1115)
- 12 Gao Yuan, Wang Guo-Ren, Zhao Hui-Qun. An Abstract model of software architecture. *Chinese Journal of Computers*, 2002, 25(7): 730~736(in Chinese)
(高原, 王国仁, 赵会群. 软件体系结构的抽象模型. 计算机学报, 2002, 25(7): 730~736)
- 13 Wei Hui, Zhu Ping, He Xing-Gui. Object-oriented design for dynamic semantic network and the soaking on it. *System Engineering and Electronic Technology*, 1998, 20(3): 56~60(in Chinese)
(危辉, 朱平, 何新贵. 动态语义网及其上浸润的面向对象设计. 系统工程与电子技术, 1998, 20(3): 56~60)
- 14 Li Pan-Lin, Li Li-Shuang, Li Yang *et al.*. *Discrete Mathematics*. Beijing: Higher Education Press, 2001(in Chinese)
(李盘林, 李丽双, 李洋, 王春立. 离散数学. 北京: 高等教育出版社, 2001)
- 15 Magee J., Kramer J.. Dynamic structure in software architectures. In: *Proceedings of the ACM SIGSOFT'96: the 4th Symposium, Foundations of Software Engineering (FSE4)*. New York: ACM Press, 1996, 3~14
- 16 Kung D., Gao J., Hsia P., *et al.*. Change impact identification in object oriented software maintenance. In: *Proceedings of the Conference on Software Maintenance*, 1994, 202~211. <http://www.ieee.org>



WANG Ying-Hui, born in 1967, Ph. D., associate professor. His research interests include software architecture and software evolution technology.

Background

This research is a part of work of the projects which is supported by the National High Technology Research and Development Program (863 Program) under grant No. 2001AA113171; National Basic Research Program of China (973 Program) under grant No. 2002CB312006 and the National Postdoctoral Research Foundation of China(2004035025).

All above projects are involved in software evolution. Authors have done many works in the area of software engineering, such as object-oriented technique, software architec-

LIU Yu, born in 1971, Ph. D., associate professor. His research interests include GIS principle, GIS software and component technique.

WANG Li-Fu, born in 1945, Ph. D., professor. His research interests include software engineering and software architecture.

ture, software framework, CBSD (Component-Based Software Development), and so on. But it's difficult to control software changes/evolution in many software development activities. SA (Software architecture) acts as blueprint of software, so authors probe into software evolution through SA. This paper focuses on quantity description of SA, and this research gives an approach to support related content of software evolution of all above projects, and establishes foundation of the next research work, especially for software evolution.