文章编号: 1001- 2486(2005) 01- 0093- 05

利用交互依赖活动集对 C⁴ISR 系统进行重组*

修胜龙, 罗雪山
(国防科技大学 人文与管理学院, 湖南 长沙 410073)

摘 要: 一个复杂的 C⁴ISR 系统由若干子系统组成, 子系统之间的交互依赖关系应该尽量少。利用活动模型构造系统的活动邻接矩阵, 用图论中的路径矩阵来识别强连通子图, 从而得出交互依赖活动集。具有交互依赖关系的活动尽量安排在一个子系统内部。利用这种方法来对 C⁴ISR 系统进行重组。

关键词: 活动模型; 图论; 邻接矩阵; 路径矩阵
中图分类号: TP391. 7; E96 文献标识码: A

Using the Sets of Strongly Depended Activities to Recompose the C⁴ISR System

XIU Sheng-long, LUO Xue-shan
(College of Humanities and Management, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: A complex C⁴ISR system is composed of many subsystems. We use activity model to create the activity adjacency matrix, and path matrix (accessible matrix) of graph theory to recognize the strongly connected components, so as to get the sets of strongly depended activities. The activities in one set should be rearranged in one subsystem. We adopt this approach to recompose the C⁴ISR system.

Key words: Activity model; graph theory; adjacency matrix; path matrix

IDEF0 是活动模型建模语言的一种, 它是在结构化分析和设计(SADT) 方法的基础上发展起来的图形化建模方法[1]。由于 IDEF0 具有简洁性和综合性等诸多优点, 所以它在一些欧洲国家和美国被广泛使用, 工业界和军方都有许多应用实例[2~4]。

IDEF0 模型的建立通常依赖于建模人员的专业知识, 需要多次经过评审人员的反馈来进行修改。模型的建立过程通常是自顶向下分解的过程, 从而对系统的细节进行更加详细的描述。建模人员在构建模型的过程中, 对系统的了解也逐渐加深。可能会发现由于在建模初期子系统之间的界面划分不当, 所以子系统之间会出现过多的信息回路, 系统的整个信息流程就显得十分复杂。如果将具有相互依赖关系的活动安排到一个子系统内部, 子系统之间信息相互依赖就会减少, 系统的整个信息流程也会更加明晰。所以, 准确地识别出系统中具有交互依赖关系的活动, 对于进行 C⁴ISR 系统过程分析、过程重组以及过程仿真等具有重大意义。

在以往的研究中, 一般都采用专家评审的方法来进行系统优化, 这种优化方式依赖于专家个人的经验。A. Kusiak 对过程模型中的活动依赖进行了图形表示[5], 我们进一步提出了交互依赖活动集的概念, 利用图论来对活动模型进行分析, 据此对系统进行重组。这种方式自底向上对系统进行重构, 为 C⁴ISR 系统分析和设计优化提供了一种有效方法。

1 活动间的基本联系

活动(或任务) 间的基本联系可以归纳为以下三种方式:

(1) 并发型。两个或多个活动之间可同时独立进行, 活动间无信息交互, 相互依赖程度低。

* 收稿日期: 2004- 09- 10
基金项目: 国家部委基金资助项目
作者简介: 修胜龙(1976—), 男, 博士生。

(2) 单向依赖型。活动间有先后顺序的约束, 其间只存在单向依赖关系。一个活动必须在另一个活动完成之后才开始, 其动态特征表现在活动间的串行依赖关系。

(3) 交互依赖型。两个或多个活动间存在信息交互关系, 即活动 1 需要活动 2 的信息, 活动 2 需要活动 1 的信息, 有时, 这种信息交互需要反复多次。因而, 活动间相互依赖程度较高。

这三种活动作用方式广泛存在于 C^4ISR 系统的活动模型中, 如图 1 所示(此处没有列出活动的名称, 而用活动标识符来标记)。活动 $A1$ 与其它活动之间就属于并发关系; 活动 $A21$ 和活动 $A22$ 之间属于单向依赖关系; 活动 $A3$ 和活动 $A4$ 之间是交互依赖关系。系统越复杂, 其交互依赖的活动也就越多。如果在子系统之间存在这种交互依赖关系, 子系统之间的界面就比较复杂。所以在系统重构的时候, 应该尽量将这些交互依赖程度较高的活动安排到一个子系统内部, 或者对其进行改造, 降低这种交互依赖程度。

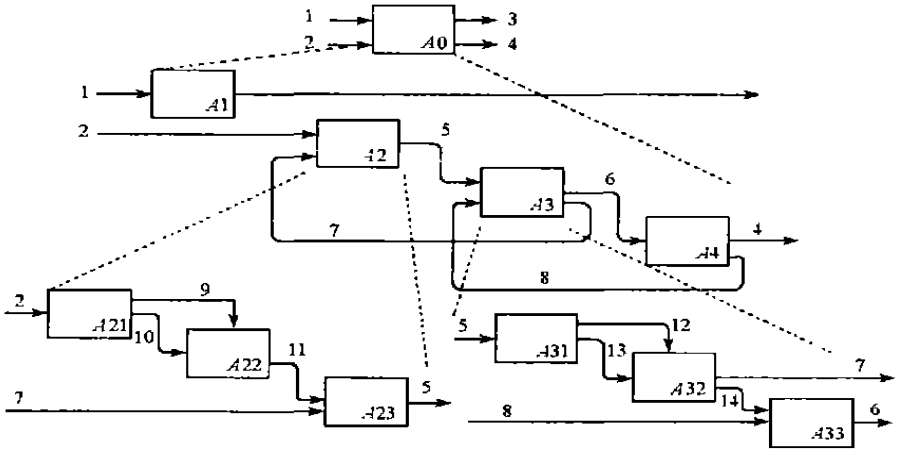

图 1 某 C^4ISR 系统的活动模型示例

Fig. 1 The example of activity model of a C^4ISR system

2 活动邻接矩阵

IDEF0 模型可以形式化描述成一个二元组[6], IDEF0= < A, L > , 其中, A 是一个类似于 BNF(巴科斯诺尔范式)产生式[7]的表达式, 表示 IDEF0 图的结构、层次和组成的活动方框, 可表示为: $A0::=A1|A2|A3\cdots|An$; $A1::=A11|A12\cdots$; $A11::=A111|A112\cdots$; …。通过回代, $A0$ 可表示为由结点活动方框组成的结构。如图 1 中的活动 $A0$ 通过回代可以表示为 $A0::=A1|A21|A22|A23|A31|A32|A33|A4$。$L$ 表示由结点活动方框组成的关系式 $A_{ij}: I \wedge C \wedge M \xrightarrow{} O$, 其中, I 为输入集合, C 为控制集合, M 为机制, $\wedge$ 为逻辑与, $\xrightarrow{}$ 表示生成。如在图 1 中 $A22$: $10 \wedge 9 \xrightarrow{} 11$。

假设活动模型中 $A0$ 的完全展开式中包含 n 个活动 Ai (i = 1, 2, …, n), 构造活动邻接矩阵 $\boldsymbol{D}$, 矩阵的行和列都与事务过程中的活动 A 相对应。

$$D_{ij}=\begin{cases} w, & \text{活动 } Ai \text{ 有 } w \text{ 个输出信息作为活动 } Aj \text{ 的输入信息} \\ 0, & \text{其它} \end{cases}$$

从上面的关系式可以看出邻接矩阵 $\boldsymbol{D}$ 实际上就是活动—活动对应矩阵。主对角线元素标志着设计活动本身, 其它元素用以表示活动间联系的存在性, 这里的元素值所指的箭头都属于内部箭头[8]。活动模型本身的箭头元素分为四种类型, 分别为输入箭头、控制箭头、机制箭头和输出箭头。根据模型的构造语法, 输入箭头、控制箭头都表示输入活动的信息, 机制箭头表示活动的执行者。由于机制箭头的源(source)一般都是外部环境, 不属于内部箭头, 并不表示活动之间的关系, 所以在图 1 中忽略了该种类型的箭头。元素 $D_{ij}=w\neq 0$ 表示活动 Aj 需要来自活动 Ai 的信息, 元素 $D_{ij}=0$ ($i\neq j$) 表示活动 Ai 与活动 Aj 之间不存在直接信息联系。图 1 的活动邻接矩阵如下所示:

$$\boldsymbol{D}=\begin{array}{c} \\ A1 \\ A21 \\ A22 \\ A23 \\ A31 \\ A32 \\ A33 \\ A4 \end{array}\begin{array}{c} \begin{array}{cccccccc} A1 & A21 & A22 & A23 & A31 & A32 & A33 & A4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

对活动邻接矩阵进行分析, 可以加深对系统结构的认识。首先给出一些定义。

定义 **1** 对于活动邻接矩阵 $\boldsymbol{D}=(D_{ij})_{n\times n}$, 记 $S_n=\{1, 2, \cdots, n\}$, 对于任意的 $T\subseteq S_n$ 且 $|T|\geqslant 2$, 不妨设 $|T|=k$, 对于 T 中元素的任意一个排列, 不妨设为 $(m_1, m_2, \cdots, m_k)$, 如果 $D_{m1m2}\cdot D_{m2m3}\cdots D_{mkm1}\neq 0$, 则称 $(m_1, m_2, \cdots, m_k)$ 对应的活动集 $\{A_{m1}, A_{m2}, \cdots, A_{mk}\}$ 为交互依赖活动集。

定义 **2** 在活动邻接矩阵 $\boldsymbol{D}$ 中, 若某一行和列的元素全为零, 即 $\sum_{j=1}^{n}D_{xj}=0$ 且 $\sum_{i=1}^{n}D_{ix}=0$, 则其所对应的活动称为完全独立活动。

定义 **3** 在活动邻接矩阵 $\boldsymbol{D}$ 中, 若某一行(列)的元素全为零但该元素对应的列(行)不全为零, 即 $\sum_{j=1}^{n}D_{xj}=0$, 且 $\sum_{i=1}^{n}D_{ix}\neq 0$, $1\leqslant x\leqslant n$ ($\sum_{j=1}^{n}D_{ix}=0$, 且 $\sum_{j=1}^{n}D_{xj}\neq 0$, $1\leqslant x\leqslant n$), 则其所对应的活动 Ax 称为单向依赖活动。

完全独立活动与其它任意的活动之间都是并发关系, 如图 1 中的活动 $A1$。单向依赖活动或者不需要从其它活动得到任何信息, 或者不向其它活动提供任何信息, 如图 1 中的活动 $A21$。所以这两种活动不可能与其它任何活动构成交互依赖关系。

3 交互依赖活动集的求解原理

交互依赖活动集包括具有相互依赖关系的两个或多个活动, 表示了由活动间的信息联系所构成的信息回路。交互依赖活动识别的过程实际上是寻找所有信息回路的过程, 这种问题应用图论的知识来解决非常方便。

一个图(Graph)是一个序偶 $<V, E>$, 记为 $G=<V, E>$, 其中, $V=\{v1, v2, \cdots, vn\}$ 是有限的非空顶点集合, E 是有限边的集合, 边 e 与顶点对 (vi, vj) 相对应。

假定顶点已经有了从 $v1$ 到 vn 的次序, 则 n 阶方阵 $\boldsymbol{X}=(x_{ij})_{n\times n}(i, j=1, 2, \cdots, n)$ 为 G 的邻接矩阵, 其中

$$x_{ij}=\begin{cases}1, & \text{若以}(vi, vj)\in E \text{ 或者}(vj, vi)\in E \\ 0, & \text{其它}\end{cases}$$

如果把活动模型节点树中叶节点所对应的活动作为有向图 G 中的顶点, 将活动之间的联系作为 G 中的有向边, 所形成的有向图如图 2 所示。其中图的顶点 $v1$、$v2$、$v3$、$v4$、$v5$、$v6$、$v7$、$v8$ 分别与原活动模型中的活动 $A1$、$A21$、$A22$、$A23$、$A31$、$A32$、$A33$、$A4$ 相对应。在前面提到过, 表示活动之间联系的箭头为内部箭头, 而与外界环境联系的箭头(如图 1 中的箭头 1、2、3、4, 它们的源(source) 或汇(end) 点不属于图的顶点集合)不属于内部箭头。G 的邻接矩阵 $\boldsymbol{X}$ 与活动邻接矩阵 $\boldsymbol{D}$ 的元素之间存在一一映射。

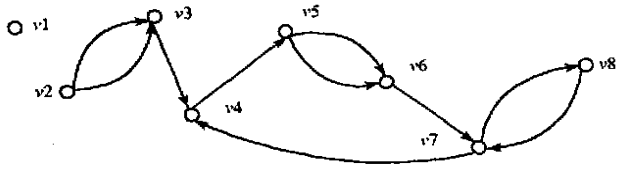

图 2 原活动模型所对应的图

Fig. 2 The corresponding graph of the activity model

定义 **4** 设 u、v 为有向图 $G=<V, E>$ 中的两个顶点, 若存在从顶点 u 到顶点 v 的通路, 则称顶点

u 到顶点 v 是可达的。若 G 中任两顶点都是相互可达的, 则称 G 是强连通图。

根据上述定义可知, 找出活动模型相互依赖活动集合的过程, 就是找出相对应的有向图中的强连通子图的过程。

定义 **5** 设 $G=<V, E>$ 是个有向图, 其中, $V=\{v1, v2, \cdots, vn\}$, 并假定顶点已经有了从 $v1$ 到 vn 的次序, 定义 n 阶方阵 $\boldsymbol{P}=(p_{ij})n\times n(i, j = 1, 2, \cdots, n)$, 其中

$$p_{ij}=\begin{cases}1, & vi \text{ 到 } vj \text{ 至少存在一条非零长度的通路}\\ 0, & \text{否则}\end{cases}$$

称矩阵 $\boldsymbol{P}$ 为图 G 的可达性矩阵, 也称之为路径矩阵。

从路径矩阵的定义可知, 如果从结点 vi 到 vj 是可达的, 则有 $p_{ij}=1$; 如果从结点 vj 到 vi 是可达的, 则有 $p_{ji}=1$。因此, 当且仅当 $p_{ij}\bullet p_{ji}=1$, 结点 vi 和 vj 是相互可达的。路径矩阵的求解过程可通过下面两个定理来进行。

定理 **1** 设 $G=<V, E>$ 是个有向图, 其中, $V=\{v1, v2, \cdots, vn\}$, $\boldsymbol{X}$ 是 G 的邻接矩阵, 矩阵 $\boldsymbol{Y}=\boldsymbol{X}^m(m = 1, 2, 3, \cdots)$ 中 (i, j) 点的值是从顶点 vi 到 vj 长度为 m 的路径数目。

定理 **2** 设 $G=<V, E>$ 是个有向图, 有 n 个顶点, 其邻接矩阵为 $\boldsymbol{X}$, 若有矩阵

$$Q=\boldsymbol{X}+\boldsymbol{X}^2+\boldsymbol{X}^3+\cdots+\boldsymbol{X}^n$$

并令

$$p_{ij}=\begin{cases}1, & \text{若 } q_{ij}\neq 0\\ 0, & \text{否则}\end{cases}$$

则 $\boldsymbol{P}$ 就是路径矩阵。

这两个定理的证明可以参见文献[9]。

下面介绍如何利用图的路径矩阵求出图的所有强连通分支。

假设 $\boldsymbol{P}=(p_{ij})_{n\times n}$ 是图的路径矩阵, $\boldsymbol{P}^{\mathrm{T}}$ 是 $\boldsymbol{P}$ 的转置矩阵, 定义矩阵运算 $\boldsymbol{P}\bullet\boldsymbol{P}^T$ 如下:

$$\boldsymbol{P}\bullet\boldsymbol{P}^{\mathrm{T}}=\begin{bmatrix}p_{11} & p_{12} & \cdots & p_{1n}\\ p_{21} & p_{22} & \cdots & p_{2n}\\ \cdots & \cdots & & \cdots\\ p_{n1} & p_{n2} & \cdots & p_{nn}\end{bmatrix}\bullet\begin{bmatrix}p_{11} & p_{21} & \cdots & p_{n1}\\ p_{12} & p_{22} & \cdots & p_{n2}\\ \cdots & \cdots & & \cdots\\ p_{1n} & p_{2n} & \cdots & p_{nn}\end{bmatrix}=\begin{bmatrix}p_{11}^2 & p_{12}\bullet p_{21} & \cdots & p_{1n}\bullet p_{n1}\\ p_{21}\bullet p_{12} & p_{22}^2 & \cdots & p_{2n}\bullet p_{n2}\\ \cdots & \cdots & & \cdots\\ p_{n1}\bullet p_{1n} & p_{n2}\bullet p_{2n} & \cdots & p_{nn}^2\end{bmatrix}$$

这样, 若矩阵 $\boldsymbol{P}\bullet\boldsymbol{P}^{\mathrm{T}}$ 的第 i 行的非零元素在第 $j1, j2, \cdots, jk$ 列, 则顶点 $vi, vj1, vj2, \cdots, vjk$ 在同一个强连通分支中, 即 $\{vi, vj1, vj2, \cdots, vjk\}$ 导出的子图是 G 的一个强连通分支。

最后, 由于图的强连通分支与活动模型中的交互依赖活动集相对应, 所以根据图中顶点与活动结点的对应关系, 可以得出原活动模型的交互依赖活动集。在对原来的 C^{4}ISR 系统进行重组的时候, 应该尽量将这些活动组合到一个子系统中。

4 路径矩阵的简化运算

通过上述的定义和定理, 可以看出求解交互依赖活动集的基本过程。但是求解路径矩阵的算法相当费时间, 因为对每个元素而言, 需要进行 n^2 次乘法, 系统中有 n^2 个元素, 为得到 $\boldsymbol{P}$ 矩阵, 就必须进行 n^4 次乘法。

对于该算法可以进行一些改进。首先是简化邻接矩阵。由于要得到的结果是关于交互依赖的, 而完全独立活动和单向依赖活动不可能与其它活动之间产生交互依赖关系, 所以可以先将这些活动从活动邻接矩阵中去掉。如果某一行(列)元素均为 0, 就将该元素所对应的行和列都从邻接矩阵中去掉。不断重复这个过程, 直至矩阵中不存在某一行(列)的元素全为 0 的情况。这样就减少了图中顶点的数目。如图 1 的活动邻接矩阵经过简化后结果如下:

$$\boldsymbol{D}=\begin{array}{c} \\ A23 \\ A31 \\ A32 \\ A33 \\ A4 \end{array}\begin{array}{c}\begin{array}{ccccc} A23 & A31 & A32 & A33 & A4 \end{array}\\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}\end{array}$$

由于只是寻找相互依赖的活动, 而对于活动之间到底有几条路径并不关心, 所以还可以用布尔矩阵来简化运算[9, 10]。将邻接矩阵定义成一个布尔矩阵:

$$D_{ij}=\begin{cases}1, & \text{活动 } Ai \text{ 至少有 1 个输出信息作为活动 } Aj \text{ 的输入信息}\\ 0, & \text{其它}\end{cases}$$

运用布尔运算(逻辑乘∧, 逻辑加∨)来计算路径矩阵。

$$X^{(r-1)}\wedge X=X^{(r)}\quad (r=2,3,\cdots)$$

$$\boldsymbol{P}=X^{(1)}\vee X^{(2)}\vee\cdots\vee X^{(n)}=\bigvee_{k=1}^{n}X^{(k)}$$

通过这两项化简, 最后计算出的 $\boldsymbol{P}$ 矩阵如下:

$$\boldsymbol{P}=\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\0&0&0&1&1\\0&0&0&1&1\end{bmatrix},\quad \boldsymbol{P}\cdot\boldsymbol{P}^{\mathrm{T}}=\begin{bmatrix}1&1&1&0&0\\1&1&1&0&0\\1&1&1&0&0\\0&0&0&1&1\\0&0&0&1&1\end{bmatrix}$$

路径矩阵还可以用 Warshall 算法来求解[9], 该算法可以直接从邻接矩阵得到路径矩阵, 最多只需要进行 n^3 次逻辑加运算。

从矩阵 $\boldsymbol{P}\cdot\boldsymbol{P}^{\mathrm{T}}$ 中可以看出, 简化后的邻接矩阵中包含两个强连通子图, 其顶点集合分别为{$v1$, $v2$, $v3$}和{$v4$, $v5$}, 则原 C⁴ISR 系统的活动模型中包含两个交互依赖活动集合, 分别为{$A23$, $A31$, $A32$}和{$A33$, $A4$}。在调整活动模型的时候, 应该尽量将每个交互依赖活动集合安排到一个页面中去。图 1 所示的活动模型经过调整后的结果如图 3 所示。注意, 为了清楚地说明调整的对应关系, 没有修改该图中的活动标号。

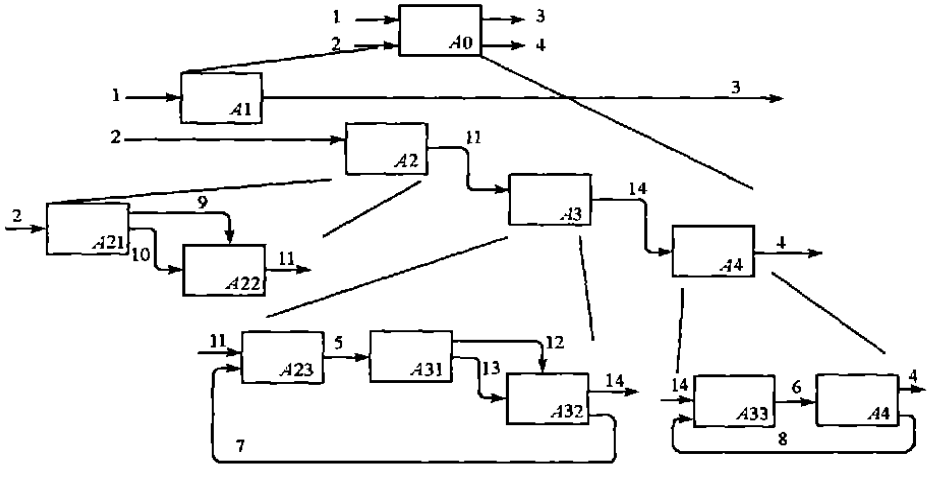

图 3 调整后的 C⁴ISR 系统活动模型

Fig. 3 The activity model of C4ISR system created after adjustment

将图 1 和图 3 所示的活动模型进行对比可以发现, 经过调整后, 活动 $A1$、$A2$、$A3$、$A4$ 所表示的子系统之间的交互依赖关系变成了单向依赖关系。调整后的系统结构与原系统是同构的, 但却更加容易进行测试。

(下转第 110 页)

$$P_{0,n}^{k,l}(t) = \int_0^{\infty} P_{0,n}^{k,l}(t,u)\,\mathrm{d}u, \quad n \geqslant 1,\ k,l \geqslant 0$$

$$P_{m,n}^{k,l}(t) = \int_0^{\infty}\int_0^{\infty} P_{m,n}^{k,l}(t,u,v)\,\mathrm{d}u\mathrm{d}v, \quad m \geqslant 1, \quad n \geqslant m, \quad k,l \geqslant 0$$

其中 $P_{0,n}^{k,l}(t,u)$、$P_{m,n}^{k,l}(t,u,v)$ 由(2)、(3)和(4)式确定。

由定理1可导出

$$P\{L(t) = m\} = \sum_{\substack{n\in N\\ n\geqslant m}} \sum_{k,l\in N} P_{m,n}^{k,l}(t), \qquad m \in N$$

参 考 文 献:

[1] Cox D R. The Analysis of Non-Markovion Stochastic Processes by the Inclusion of Suplementary Variables[J]. Proc. Camb. Phil. Soc., 1955, 51: 433- 441.

[2] Davis M H A. Markov Models and Optimization[M]. Chapman & Hall, 1993.

[3] 基赫曼 N N, 斯科罗霍德 A B. 随机过程论[M]. 北京: 科学出版社, 1986.

[4] 刘国欣. 等, 逐段决定马尔可夫骨架过程[M]. 长沙: 湖南科技出版社, 2000.

[5] Alfa A S, Rao T S. Supplementary Variable Technique Models[J]. J. Probability in the Engineering and Informational Sciences, 2000, 14: 203- 218.

(上接第 97 页)

5 结 论

C^4ISR 系统的结构一般比较复杂。如何对系统的结构进行调整, 使得子系统之间的界面更加简洁, 对 C^4ISR 系统分析、仿真等具有十分重要的意义。本文从减少子系统之间的交互依赖关系的角度出发, 分析活动之间的关系, 构造活动模型的活动邻接矩阵, 定义了交互依赖活动集的概念。结合一个例子, 说明了用图论的路径矩阵来求解交互依赖活动集的原理及路径矩阵简化算法。最后根据交互依赖活动集对原系统进行了重组, 取得了比较好的效果。

当然, 对于系统的重组原则, 还要包括减少子系统之间单向依赖的数量, 以及降低子系统内部复杂度等问题, 我们将进一步进行研究。

参 考 文 献:

[1] 陈禹六. IDEF 建模分析和设计方法[M]. 北京: 清华大学出版社, 1999.

[2] Shapiro Robert M, Pinci V O, Mameli R. Modeling a NORAD Command Post Using SADT and Colored Petri Nets[A]. Proceedings of the IDEF Users Group[C], Washington DC, May 1990.

[3] Wisnosky, Dennis E, Allen W. Batteau. IDEF in Principle and Practice[J]. Gateway, 1990, 5: 8- 11.

[4] 罗雪山, 朱德成, 沈雪石. IDEF0 方法在军事综合电子信息系统分析设计中的应用[J]. 长沙: 国防科技大学学报, 2001, 23(3): 88- 92.

[5] Kusiak A. Engineering Design: Products, Process, and System[M]. London: Academic, 1999.

[6] 王君英, 段广洪. 基于 IDEF0 的 CIMS 底层控制 Petri 网模型的自动生成方法[J]. 自动化学报, 1997, 23(5): 400- 403.

[7] NAUR, Peter. Revised Report on the Algorithmic Language ALGOL 60. [J]. Communications of the ACM, 1960, 3(5): 299- 314.

[8] NTIS No. FIPSPUB183/HDM. Integration Definition for Function Modeling (IDEF0)[S]. 1993, 12.

[9] 范逢曦. 图论方法及应用[M]. 太原: 山西科学教育出版社, 1987.

[10] 唐敦兵, 李东波, 张世琪. 一种并行设计过程中耦合活动识别算法的研究[J]. 系统工程与电子技术, 1999, 21(12): 26- 28.