

一种需求变更影响的评估算法

杨鹤标, 张继敏, 朱玉全

(江苏大学计算机科学与通信工程学院, 镇江 212013)

摘 要:通过对需求依赖关系的分析与归纳, 给出了需求依赖关系的依赖形式; 采用回溯法来搜索和界定需求变更的影响范围, 给出了变更影响的量化方法; 从需求依赖关系的视角, 设计了一个可以量化评估需求变更影响的算法。通过一个“样例学习”的例子, 展现了该评估算法的稳定性。

关键词:需求变更; 依赖关系; 评估算法

Evaluation Algorithm for Impact of Requirements Change

YANG Hebiao, ZHANG Jimin, ZHU Yuquan

(School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang 212013)

【Abstract】 The relations of the dependencies are found and formalized. The scope of impact raised by requirements change is confirmed by backtracing algorithm, and the impact is quantitatively described by using impact factor. Then the algorithm is presented to evaluate the impact of this quantitative description. An example of case study shows the feasibility of this evaluation algorithm.

【Key words】 Requirement change; Dependency; Evaluation algorithm

在软件生命周期中发生需求变更是无法回避的^[1]。需求变更对软件项目进度控制、成本核算、开发周期等方面有着巨大的影响。需求变更影响评估是指: 确定需求变更影响范围及分析变更带来潜在的影响^[2]。现有需求变更影响评估的主要方法是借助需求跟踪链或矩阵来确定需求变更给设计、编码等软件产品元素带来的影响^[3~5]。由于现有的评估方法极少地考虑依赖因素所带来的影响, 给评估结果带来了不确定性。因而在评估过程中, 如何搜索和界定依赖关系对评估的影响范围变得尤为重要。

针对上述问题, 本文对需求间依赖关系进行了分析与归纳, 给出了对需求依赖关系的依赖形式; 采用回溯法搜索依赖关系矩阵, 确定需求变更的影响范围, 并对影响程度进行了量化处理; 最后设计了评估变更影响的算法; 并在研究实例上进行了验证, 对评估结果的可信度进行了分析。

1 需求依赖关系

在一个软件系统中, 需求依赖关系是普遍存在的, 并且存在着多种的依赖关系^[6~8]。由于需求之间的依赖关系的存在, 使得某需求发生变更后, 势必对其相关的需求产生影响。因此, 合理的确定需求变更影响范围是评估结果有效性的关键所在。

1.1 需求依赖关系定义

定义 1 如果一个需求 R_s 变更时, 要求另一需求 R_t 也要发生相应的变化, 则称为需求 R_t 依赖于 R_s , 记为 $\text{dep}(R_s, R_t)$ 。如果需求 R_t 不依赖于需求 R_s , 则记为 $\neg \text{dep}(R_s, R_t)$, 其中 R_s 称为依赖源需求, R_t 称为依赖目标需求。

1.2 需求依赖关系类型及性质

需求依赖关系形式多样, 关系复杂, 综合起来可将需求依赖关系归纳为 4 类: 子集依赖关系, 完全依赖关系, 父集依赖关系, 回路依赖关系。

定义 2 如果 $\forall R_a \forall R_b(\text{dep}(R_a, R_b))$, 仅当 $\exists R_c((R_c \subseteq$

$R_a) \wedge \neg \text{dep}(R_a - R_c, R_b))$, 则称 R_b 对 R_a 子集依赖, 记为 $\text{childdep}(R_a, R_b)$; 否则 R_b 对 R_a 非子集依赖, 记为 $\neg \text{childdep}(R_a, R_b)$ 。

定义 3 如果 $\forall R_a \forall R_b(\text{dep}(R_a, R_b) \wedge \neg \text{childdep}(R_a, R_b))$, 则称 R_b 对 R_a 完全依赖, 记为 $\text{fulldep}(R_a, R_b)$, 需求 R_a 的变更就会对 R_b 产生影响。

定义 4 如果 $\forall R_a \forall R_b \exists R_c (\text{dep}(R_a, R_b) \wedge (R_c \subseteq R_b) \wedge \neg \text{dep}(R_a, R_b - R_c))$, 那么称 R_b 对 R_a 父集依赖, 记为 $\text{fardep}(R_a, R_b)$ 。在需求 R_a 与需求 R_b 的依赖关系中, 仅 $R_c \subseteq R_b$ 受到 R_a 的变更影响。

子集依赖和父集依赖的存在, 可以认为是需求建模粒度过大而产生的。这种非原子型的需求模型可能掩盖了部分需求间的精确依赖关系, 同时也可能导致依赖关系对变更影响描述的不精确性和不同步性。

需求依赖关系具有两个性质:

(1) 依赖关系传递性

如果 $\forall R_a \forall R_b \forall R_c(\text{dep}(R_a, R_b) \wedge \neg \text{childdep}(R_a, R_b) \wedge \text{dep}(R_b, R_c) \wedge \neg \text{childdep}(R_b, R_c))$, 则称 R_c 对 R_a 传递依赖。

(2) 不可逆性

如果 $\forall R_a \forall R_b(\text{dep}(R_a, R_b) \wedge \neg \text{dep}(R_b, R_a))$, 则需求 R_b 发生变化不会对需求 R_a 产生影响。

定义 5 如果 $\forall R_1 \forall R_2 \dots \forall R_n((\text{dep}(R_1, R_2) \wedge \neg \text{childdep}(R_1, R_2) \wedge \dots \wedge \text{dep}(R_{i-1}, R_i) \wedge \neg \text{childdep}(R_{i-1}, R_i) \wedge \text{dep}(R_i, R_1) \wedge \neg \text{childdep}(R_i, R_1))(i \geq 2))$, 则称 R_1, R_2, \dots, R_n 之

作者简介: 杨鹤标(1960 -), 男, 副教授, 主研方向: 数据库, 软件体系结构, 信息系统和软件工程; 张继敏, 硕士生; 朱玉全, 博士、副教授

收稿日期: 2006-07-24

E-mail: yhbjj@ujs.edu.cn

间的关系为回路依赖关系，记为 $\text{loopdep}(R_1, R_2, \dots, R_n)$ 。

回路依赖关系是结构上的一种强耦合关系，它降低了模型对变化的适应能力和可复用的程度。

2 确定需求变更依赖集

本文采用回溯算法来求解需求间的依赖关系。

2.1 解空间构成

算法中问题的解空间用有向图来表示。有向图 $G=(N_G, E_G)$ ，其中 $N_G=\{R_i | i=1..n\}$ 组成， R_i 为需求， $E_G \subseteq N_G \times N_G$ ，对于任意 $\langle u, v \rangle \in E_G$ ，则表示 $\text{dep}(u, v)$ ，且称 v 为弧尾，称 u 为弧头。以 v 为头的弧的数目称为 v 的入度；以顶点 u 为尾的弧的数目称为 u 的出度。用邻接矩阵作为有向图 G 的存储结构，邻接矩阵定义为

$$A[i, j] = \begin{cases} 1, & \text{若 } \langle R_i, R_j \rangle \in E \\ 0, & \text{若 } \langle R_i, R_j \rangle \notin E \end{cases}$$

2.2 解空间结构

将 n 个需求按 $1, 2, \dots, n$ 进行编号，解空间结构可表示为一棵高为 $n+1$ 的完全 n 叉树。解空间第 i 层($i \leq n$)中每个结点都有 n 个儿子，每个儿子对应于用 N_G 中可能存在的依赖关系之一。 $n+1$ 层为叶结点。图1示例了一棵三元需求的解空间结构的排列树。

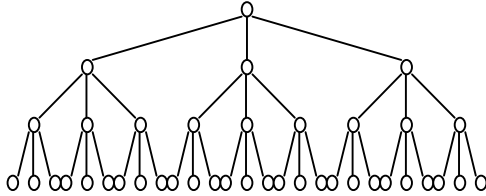


图1 $n=3$ 时的解空间排列树

其中2到 $n+1$ 层的结点为需求结点。

2.3 算法描述

算法采用深度优先搜索法遍历解空间进行求解。在算法递归Backtrack中，当 $i>n$ 时，算法搜索到叶结点，得到一组依赖关系集。假设搜索的初始需求结点为 i_0 。

当 $i \leq n$ 时，当前扩展结点是解空间内部结点，该结点有 n 个儿子，在此检查其可行性，以深度优先的方式对可行子树搜索，或剪去不可行子树。

相互关系的依赖确定，每搜索到一个新的结点 i 时，检测图 G 是否存在 i_0 到 i 和顶点 i 到 i_0 的边，如果存在，则结点 i_0 和 i 间存在相互依赖关系。

当 $i>3$ 时，检测图 G 是否存在从顶点 $i-1$ 到 i 和顶点 i 到 i_0 的边，如果存在，找到一条回路。搜索过程中用 $x[]$ 存放当前解。递归算法如下：

```
Backtrack (i) {
    if (i>n) 输出依赖关系集及依赖关系路径
    else {
        if(A[x[1],x[i]]=1&&A[x[i],x[1]]=1)
            //输出回路依赖关系及依赖关系路径;
        if (i>=3)
            if(A[x[i-1],x[i]]=1&&A[x[i],x[1]]=1)
                //输出回路依赖关系及依赖关系路径;
        else {
            for(j:=1;j<=n;j++) {
                x[i]:=j;    //选择下一搜索顶点;
                if(A[x[i-1],x[j]]=1)
                    backtrack(i+1);
            }
        }
    }
}
```

```
else
    //输出依赖关系集及依赖关系路径;
}}}
```

3 需求变更影响评估算法

3.1 影响因素

上述算法给出了需求变更的影响范围全集。为度量需求之间的依赖及影响的强度，采用如下的公式量化两个需求间的影响因子：

$$FI(R_s, R_t) = \text{workload}[R_t] / \text{workload}[R_s] \cdot \text{dep}(R_s, R_t) \wedge \text{change}(R_s) \quad (1)$$

需求 R_s 发生变更后，式(1)中的 $\text{source}[R_s]$ 表示在 R_s 上追加的工作量，而 R_t 受到 R_s 变更的影响， $\text{workload}[R_t]$ 为在 R_t 上增加的工作量。

在 $FI(R_s, R_t)$ 量化过程中，由于涉及众多的因素，如开发者的个体能力、工程计划的合理性、功能结构的复杂度规模等都可能影响到 $FI(R_s, R_t)$ 量化值的准确性，在 $FI(R_s, R_t)$ 生成过程中应在对经验数据的统计分析结果的基础上，可依据具体情况进行调整。

3.2 影响程度的度量

根据上述影响因素的量化方法，当变更发生时对整个系统的影响程度可用所有由变更而发生的工作量增量来描述。

设 W_s 为依赖源 R_s 工作增量， R_t 依赖于 R_s ，则变更后工作增量 DI 计算公式如下：

$$DI = W_s \cdot FI(R_s, R_t) \quad (2)$$

图2为一需求依赖关系图，设 $FI(R_1, R_2)=0.5$ ， $FI(R_2, R_3)=1.4$ ， $FI(R_2, R_4)=0.8$ 。

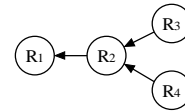


图2 需求依赖关系

假定 R_1 变更的 $W_1=4$ ，则利用式(2)可得： R_2 应增加的工作量为 $W_1 \cdot FI(R_1, R_2)=4 \cdot 0.5=2$ ； R_3 应增加的工作量为 $W_2 \cdot FI(R_2, R_3)=2 \cdot 1.4=2.8$ ； R_4 应增加的工作量为 $W_2 \cdot FI(R_2, R_4)=2 \cdot 0.8=1.6$ 。这样，变更后的工作增量 $DI=2.8+1.6+2=6.4$ 。

3.3 评估算法

采用第2节的算法，获得需求依赖关系初始状态下的影响全集用 C_a 来表示。当系统部分需求发生变更时，评估需求的变更所引起的蔓延现象，对系统所产生的影响的评估的算法步骤如下：

(1)从全集 C_a 中抽取需要进行评估的需求变更影响集。其方法：抽取以需求变更源为树根的影响集；

(2)消去变更影响集中的冗余依赖关系。其方法：用层叠法消去影响集中重复的结点和路径分枝，得到需求变更所产生影响的最小依赖子集；

(3)采用式(1)，进行影响的量化处理；

(4)采用式(2)，进行评估计算。

评估的算法设计如下：

采用邻接矩阵存储变更影响集，初始时 $a[i, j]=0$ ； $i, j=1, 2, \dots, n$ ；假设：需求变更源集合 $SETcs = \{R_i | R_i \subseteq N\}$ 。需求 R_s 变更影响的顶点集合 $\text{ImpactSet}(R_s) = \{R_i | R_i \subseteq C_a\}$ 。初始时需求的工作量集 $\text{InitSet} = \{\langle R_i, W_i \rangle | R_i \in SETcs\}$ ，其中， W_i 为工作量。变更后需求的工作量集 $\text{EvaSet} = \{\langle R_i, W_i \rangle | R_i \in \text{ImpactSet}\}$ ，其中， W_i 为工作量。

算法 1 EliminateRedundance 用于消去影响集中冗余依赖关系并进行量化处理。

```
EliminateRedundance (a,ImpactSet) {
```

```
for ( each <Rs,Rt> ImpactSet ) {
```

```
  a[Rs,Rt]= FI(Rs,Rt); }
```

算法 2 Evaluation 完成评估计算

```
Evaluation() {
```

```
  for ( each Ri SETcs ) {
```

```
    ImpactSet=Ca(Ri); //获取需求Ri变更的影响集
```

```
    EliminateRedundance (a,ImpactSet); //消去冗余依赖
```

```
  }
```

```
  for ( each Ri SETcs){
```

```
    从 IniSet 集中获取 Ri 对应的 Wi;
```

```
    Rs=Ri; Wsi:=Wi;
```

```
    EvaSet= EvaSet {<Rs,Wsi>; //加入评估集
```

```
      for (  $\forall$  Rt ImpactSet(Ri)&&<Rs,Rt> E(ImpactSet(Ri)) {
```

```
        WtI:= Wsi*a[Rs,Rt];
```

```
        EvaSet= EvaSet {<Rt,Wtr>; //加入评估集
```

```
        Rs:=Rt; WsI:=WtI;
```

```
      } }
```

```
//比较 EvaSet 与 IniSet 得到影响评估结果}
```

4 研究实例

本节通过一个研究实例来验证前节所提出的评估算法。实例中给出了一组需求及其之间的依赖关系，如图 3 所示。

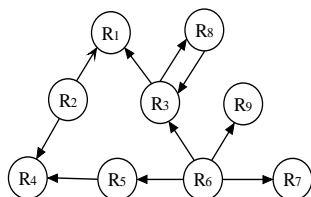


图 3 需求及依赖集实例

假设初始状态下，图 3 每一需求的工作量如表 1 所示。表 2 为需求依赖所产生的影响因子。

表 1 需求初态时工作量

需求号	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆	R ₇	R ₈	R ₉
工作量	5	4	3	5	7	3	7	8	4

表 2 需求依赖所产生的影响因子

Rs	R ₁	R ₁	R ₃	R ₃	R ₄	R ₄	R ₅	R ₇	R ₈	R ₉
Rt	R ₂	R ₃	R ₆	R ₈	R ₂	R ₅	R ₆	R ₆	R ₃	R ₆
FI(Rs, Rt)	0.8	0.6	1.0	2.67	0.8	1.4	0.43	0.43	0.38	0.75

图 4 给出了需求变更源 R1、R4 的最小依赖集，虚线表示部分为环路依赖。

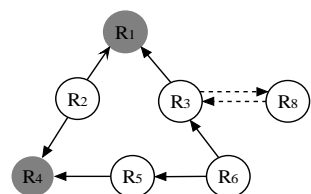


图 4 最小依赖集

假设 R₁、R₄ 的工作增量分别为 4 和 2，则通过上述评估

算法的计算，可得评估数据如表 3 所示。评估结果为算法计算结果，统计结果为开发过程中测算结果。

表 3 评估结果

需求号	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆	R ₈
评估结果	4.0	4.8	2.4	2.0	2.8	1.2	6.41
统计结果	3.5	4.0	2.9	2.0	3.3	1.1	7.10

通过对上表的分析，可得结论：

(1)最小依赖子集与需求变更的影响范围一致。

(2)即便受影响因子的精确性以及评估参数缺失影响，通过表 3 中统计与评估结果的对比，可以看出，需求变更的影响与工作增量的趋势是一致的。

5 结语

本文从需求依赖关系的角度讨论了需求变更对系统的影响程度，给出了一个基于需求依赖的变更评估算法，在一个实际工程开发阶段中验证了这一算法的稳定性。然而，鉴于需求依赖关系自身存在的多层面多阶段的复杂性，要精确描述需求间的所有依赖关系以及对各种不同的依赖给出一个相对合理的度量体系不是一蹴而就的，它是一个需要不断迭代，不断完善的过程。对于需求间依赖关系定性和定量的分析，依赖关系量度体系的更合理更精确的数学描述，以及需求及需求依赖关系在软件开发过程不同阶段的进化过程中进一步产生的新的依赖关系等，这一系列更复杂的问题还有待在后续的工作中作更深入的研究。

参考文献

- Boehm B W. Software Engineering Economics[M]. New Jersey: Prentice-Hall, Inc., Englewood Cliffs, 1981.
- Ramzan S, Ikram N. Decision in Requirement Change Management Information and Communication Technologies[C]. Proc. of the 1st Inter- national Conference on ICIT, 2005-08: 27-28.
- Arnold R S, Bohner S A. Impact Analysis-towards a Framework for Comparison[C]. Proceedings of the International Conference on Software Maintenance, 1993: 292-301.
- Strens M R, Sugden R C. Change Analysis: A Step Towards Meeting the Challenge of Changing Requirements[C]. Proceedings of the IEEE Symposium and Workshop on Engineering of Computer Based Systems, 1996.
- Wen Lian, Dromey R G. From Requirements Change to Design Change: A Formal Path Software Engineering and Formal Methods [C]. Proceedings of the 2nd International Conference on SEFM, 2004: 104-113.
- Carlshamre P, Sandahl K. An Industrial Survey of Requirements Interdependencies in Software Product Release Planning[C]. Proceedings of the 5th IEEE International Symposium on Requirements Engineering, 2001: 84-91.
- Giesen J, Volker A. Requirements Interdependencies and Stakeholders Perferences[C]. Proceedings of IEEE Joint International Conference on Requirements Engineering, 2002-09: 206-209.
- Ramesh B, Jarke M. Toward Reference Models for Requirements Traceability[J]. IEEE Transactions on Software Engineering, 2001, 27(1): 58-93.