ソフトウェア工学 110－1
（1996. 7. 25）

問題点と解決策に注目した要求会議分析における機能単位分析

蓬莱 尚幸†
(株) 富士通研究所 情報社会科学研究所

我々は、要求獲得方法論の研究の一環として、要求会議の記録(録音、録画)を元に次回会議や後段の開発に役立つ情報を得るためのUSPオフライン法を提案している。本稿では、USPオフライン法の一部である対象システムの機能単位を抽出するための分析について述べる。この分析の目的は、会議に現れる問題点と解決策に注目して、解決策をグループ化することで複数の機能単位を得ることである。その際、問題点に対する寄与度の違いにより解決策のグループおよびグループ間の関係を表す図表現を自動生成し分析者の機能単位分析作業を支援する。

Function Unit Analysis of Requirements Meeting from Viewpoint of Problem and Solution

Hisayuki Horai

Institute for Social Information Science,
FUJITSU LABORATORIES LIMITED

We are developing a requirements method using requirements meeting and have already proposed USP Offline Method to elicit various valuable information from an audio/video recording of requirements meeting. In this paper, we explain Function Unit Analysis in USP Offline Method in which we capture function units of the target system. In the method, analysts concentrate on problems and solutions which appear in a meeting and build up some function units by categorising the solutions. For helping analysts, our method provides automatic generation of groups of solutions, relations among them, and graphical notation of the groups and their relations.

†*horai@iias.flab.fujitsu.co.jp*

1 はじめに

顧客 (開発対象システムのユーザ) が満足するソフトウェアシステムを開発するためには、ソフトウェア開発プロセスの最上流に位置する要求段階 [1, 2] において顧客の要求を十分に獲得する必要がある。要求段階は顧客の持つ要求を獲得することが目的とされるが、必ずしも「そこにあるもの」を採取するというものではない。たとえば、要求段階初期に顧客組織内部で要求が固まっているとは限らず、むしろ要求段階を通して徐々に要求が明確化されてゆく場合もある。また、顧客の意思が統一しておらず、要求段階を通して要求が統一化されてゆく場合もある。このような顧客自身が要求を明確化し統一化してゆく過程を開発者は助言を与えながら観察し最終的な要求を獲得するのが要求段階である。このような考察をもとに、我々はユーザ指向要求獲得分析技術 [6] を研究してきた。

上記のような特徴を持つ要求段階では、顧客同士および顧客 - 開発者間のコミュニケーションが重要な問題となる [3]。意見陳述、提案、反論、質疑応答、合意形成、決定、確認など様々な活動を行うことができる会議は、インタビューや質問書などに比べて要求段階でのコミュニケーションとして適していると考えられる。そこで我々は要求会議を用いた要求獲得分析を支援するための方法論として USP 法を提案している。

会議の進行の支援については多く研究され実践されているが [4, 5]、USP 法では、会議の進行の支援だけでなく、要求会議の録音や録画をもとに次回の会議や開発プロセス後段に役立つような情報を抽出する分析を行うことにも焦点を当てている。我々は、要求会議の進め方の支援をオンライン法、要求会議のまとめ方の支援をオフライン法と呼んでいる。

本稿では、USP オフライン法の一部である機能単位分析について述べる。はじめに USP オフライン法の概略を説明し機能単位分析の役割を明確化する。次に、機能単位分析の方法論および支援ツールについて述べる。そこでは、機能単位分析の進化の過程を追って説明する。最後に、実験的な適用から得られた評価と今後の課題を簡単に述べる。

2 USP オフライン法

前述のように多くの内容と異なる作業を含みうる会議は要求段階に適したコミュニケーション方法である。しかしながら、通常の会議の後に残る直接のドキュメントとしては、1～2ページの議事録のみであるということが多い。このような議事録は、会議の持つ協力な能力に比べて、あまりにも内容が貧弱であると思われる。たとえば、決定事項が残るだけで否決された対案が記録されないことが多い。会議で多くの有意義な情報がもたらされても、会議の記録である議事録が貧弱であると、それらの情報は忘れさられ、次回以降の会議や開発プロセス後段に活かされない。このような問題点を含む従来の議事録を越える会議の記録を得るために、我々は会議の録音や録画をもとに会議の内容を分析する USP オフライン法を提案している。そこで、会議で現れた情報をもれなく集めることが USP オフライン法の一つの特徴になっている [USP オフライン法の網羅性]。

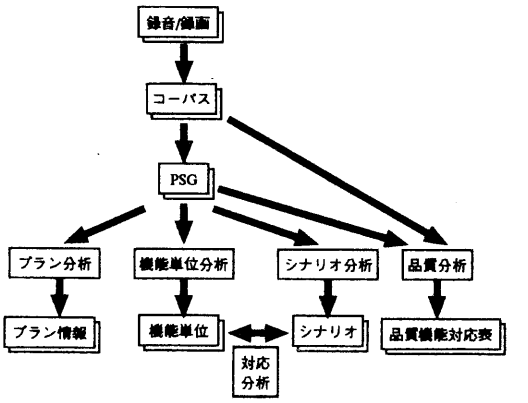

図 1: USP オフライン法 (抜粋)

図 1のように、USP オフライン法は複数の分析の集合体である。これは、会議に現れる情報の多面性を互いに関係をつけながらも個別に際だたせてもれなく分析するためである [USP オフライン法の多面性]。たとえば、機能単位分析は議題である対象システムの機能に関する静的な分析 (どのような機能が話されたか) であり、シナリオ分析は議題である対象システムの機能の動的な側面に関する分析であり、品質分析は文字通り会議で現れた機能の品質に関する分析であり、プラン分析は開発プロセスに関する情報の分析である。また、シナリオ分析と機能単位分析の両結果の関係など各分析間の関係も分析する。

USP オフライン法は多段階かつ並行に行なわれる複数の分析の集合体であるが、各分析では、すべての分析結果についてどの入力情報から得られたかを明らかにし分析結果の属性として蓄えている [トレーサビリティ]。トレーサビリティは、要求の起源を明らかにするだけでなく、各分析結果間の関係を明らかにするためにも有用である。たとえば、機能単位とシナリオの対応関係について、双方から PSG へのトレーサビリティをたどることで、PSG 内での関係を利用して議論できる。

上記のように、網羅性と多面的分析とトレーサビリティが USP オフライン法の特徴であり、USP オフライン法の詳細な分析結果は次回の会議および開発プロセス後段に対して多くの有意義な情報となりうる。

2.1 コーパスと PSG

次に、本稿の主題である機能単位分析までの USP オフライン分析の流れについて述べる。

USP オフライン分析では、まず、録音 / 録画を元にコーパスを作成する。コーパスは発言や行動をテキスト化しコンピュータから容易にアクセスできるようにしたものであり、USP オフライン分析におけるトレーサビリティの鎖のアンカーポイントとなる。

PSG(Problem Solution Graph) は、会議の内容を「問題点と解決策」の観点から整理したものである。図

```
├× 書記等の電話取次時、行先が不明で困る                r100
| ├× ホワイトボードに予定を記入してくれない            r101
| ├○ 留守番電話設定                                   r102
| | └× 一,二人しかしていない                          r103
| ├○ 各研究員に携帯電話を持たせる                     r104
| | ├× 携帯電話は置き忘れる                            r105
| | └→ r035
| └○ 在席管理システムを開発する                       r106
|   ├× 入力を怠ると情報が抜ける                       r107
|   | ├○ センサで出入りを自動認識                     r108
|   | | └× 現実的でない                                r109
|   | ├○ login で出社と判別                           r110
```

図 2: PSG の例

2に示すように、PSG のノードは項目種類と項目内容と項目番号から構成され、要求項目と呼ばれる (他に要求項目にはコーパスへのトレーサビリティなどの属性を与えられるが図 2では表示されていない)。項目種類は、問題点を「×」で表し解決策を「○」で表す。項目内容は分析者がコーパスを要約して作成した文章である。PSG は有向グラフであるが、「→」(ジャンプ) という特殊ノードを用いることで木構造で表現する。すなわち、アークの始点と終点は木構造の親と子、および、ジャンプノードの親と「→」の後の項目番号で表される。解決策 A から解決策 B へのアークは B が A の部分解決策であることを意味し、解決策 A から問題点 B へのアークは B が A の問題点であることを意味し、問題点 A から解決策 B へのアークは B が A の解決策であることを意味し、問題点 A から問題点 B へのアークは B が A の部分問題点となっていることを意味する。アークにもノードと同様に属性を付けることができる。

3 機能単位分析

会議の内容を正確に掴むためには、会議で話されている対象システムの機能を知る必要がある。網羅的に集めた PSG 内の解決策は、粒度にばらつきがあり、システム全体を捉えるために用いるには数が多過ぎる場合が多いので、そのままシステムの機能として扱うには問題がある。そこで、細かい解決策をグループ化して「機能単位」を抽出することは次回の会議にとっても開発プロセス後段にとっても有効である。

図 3に USP オフライン法における機能単位分析 (USP 機能単位分析) の流れを示す。基本的に、分析者が PSG

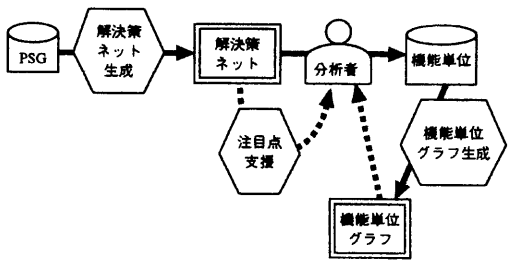

図 3: USP 機能単位分析

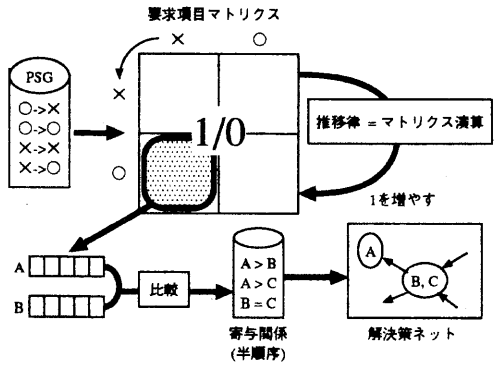

図 4: 解決策ネット生成

内の要求項目を読みながら解決策をまとめ名前を付けることで機能単位を作成するが、USP 機能単位分析では有効な支援を提供する。すなわち、機能単位の叩き台となるような解決策のグループを PSG から自動生成し [解決策ネット生成]、分析者のさらなるグループ化作業のためにどの解決策ネットのノードに注目するとよいかの示唆を与える [注目点支援]。すなわち、分析者は自動生成された解決策ネットを見て注目点支援を受けながら解決策ネット内のノード (解決策のグループ) をさらに大きなグループにまとめて機能単位を作成する。さらに、USP 機能単位分析では、作成した機能単位の間の関係を明らかにするために、解決策ネットのアークから自動的に機能単位間の関係を生成する [機能単位グラフ生成]。次に上記の三つの支援について詳細を述べる。

3.1 解決策ネット生成

網羅的に集めた PSG 内の解決策は非常に多いので、分析者が解決策をグループ化して機能単位を作成する際にすべての解決策を一度に扱うのはほとんど不可能である。そこで、分析者の作業の負荷を軽減するために有用な解決策のグループおよびグループ間の関係を自動的に生成することは有効である。USP 機能単位分析では、解決策が寄与する問題点 (寄与関係) の違いによりグループ化およびグループ間関係付けを行なう (図 4)。

PSG 内の問題点から解決策へのアークは寄与関係を直接的に表したものである。しかしながら、PSG が問題点と解決策の木構造になっていることから、PSG のアークには以下のような特殊な推移律が存在する。

- 解決策 O2 が問題点 X1 の問題点であり問題点 X3 が解決策 O2 の問題点であり解決策 O4 が問題点 X3 の解決策ならば、解決策 O4 が問題点 X1 の解決策でもある。
- 解決策 O2 が問題点 X1 の解決策であり解決策 O3 が解決策 O2 の部分解決策ならば、解決策 O3 が問題点 X1 の解決策でもある。
- 解決策 O2 が解決策 O1 の部分解決策であり問題点 X3 が解決策 O2 の問題点ならば、問題点 X3 が解決策 O1 の問題点でもある。

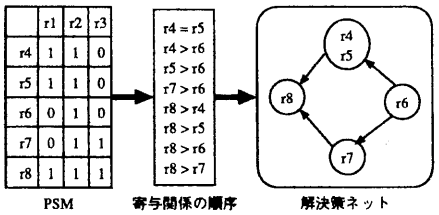

図 5: 解決策ネット生成の例

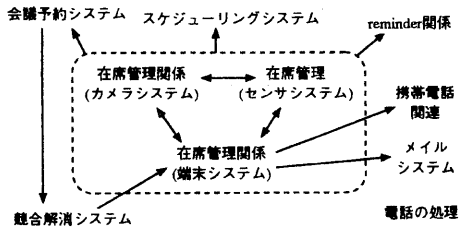

図 6: 機能単位グラフの例

- 問題点 X2 が解決策 O1 の部分解決策であり問題点 X3 が問題点 X2 の部分問題点ならば、問題点 X3 が解決策 O1 の問題点でもある。

この推移律をほどき、すべての寄与関係を洗い出すために、PSG のアークを要求項目マトリクスに表現し、各推移律により値を変更するマトリクス演算を用意し、これらのマトリクス演算の合成演算の閉包をとる。要求項目マトリクス M は要求項目を縦横に並べた正方マトリクスで、横に並ぶ要求項目 X から縦に並ぶ要求項目 Y への PSG のアークが表す寄与関係が存在するかしないかを $M(X,Y)$ の値が 1 であるか 0 であるかで表す。

最終的に、$M(\{問題点\},\{解決策\})$ が解決策の問題点への寄与関係を表す (この部分マトリクスを PSM と呼ぶ)。次に、2 つの解決策 A と B に関する PSM の行について値が 1 がある箇所に対応する問題点の集合を P_A と P_B とする。P_A が P_B と等しければ、A と B はちょうど同じだけの問題点に寄与する。P_A が P_B を真に含めば、A は B より多くの問題点に寄与する。すなわち、P_A と P_B の包含関係は A と B の寄与関係の違いを表しており、P の包含関係により解決策間に寄与関係の半順序を付けることができる。寄与関係が等しい解決策をグループにまとめ、寄与関係の半順序をグループ間の非循環有向グラフに表したものが解決策ネットである。たとえば、3 個の問題点と 5 個の解決策から構成される PSM から解決策ネットを生成する例を図5に示した。r4 と r5 が寄与する問題点は等しいので (ともに {r1,r3})、同じノードにまとめられる。r4 が寄与する問題点は r6 が寄与する問題点 ({r2}) を含むので、寄与関係の順序関係 (r4>r6) が存在する。r4 が寄与する問題点と r7 が寄与する問題点 ({r2,r3}) には包含関係がないので、r4 と r7 の間には順序関係はない。

解決策ネット内のノードは、それに含まれる解決策の寄与関係に関する順序関係に基づいたアークで結ばれるので、ノードをグループ化して機能単位を作成するために有用である。

3.2 注目点支援

解決策ネットにより解決策は寄与関係によってグループ化され構造化された。分析者が解決策をまとめて機能単位を作成するときに、この解決策ネットは有用である。しかしながら、解決策ネットが巨大なときは、機能単位生成 (解決策のグループ化) の作業を解決策のどこから手を付けてよいか判断しにくい場合がある。このような場合、解決策ネットを見ながら機能単位を生成する作業中に、次に分析者が解決策ネット内のどのノードに注目したらよいかの示唆を与える支援が有効となる。このような示唆を与えることは、機能単位生成の戦略を分析者に示すことに他ならない。

解決策ネット上で近いノードに含まれる解決策は遠いノードに含まれる解決策より寄与関係が近く同じ機能単位に入る可能性が高い。すなわち、解決策ネットをトラバースしながら解決策をグループ化し機能単位を成長させてゆく手法が有効となる。そこで、既に分析者が作成した機能単位に含まれる解決策を含むノードの近隣ノードを次の注目点の候補として提供することは有効な支援となる [近隣ノード戦略]。

解決策ネット内で最も上位にあるノードには、最も多くの問題点に寄与する解決策が含まれている。このようなノードは機能単位を生成する際に重要な位置を占めるときが多い。そこで、まだどの機能単位にも含まれていないノードのうちで最上位のノードを次の注目点の候補として提供することは有効な支援となる [最上ノード戦略]。

解決策ネットのノードに含まれる解決策の数はノードごとに異なる。あるノードに含まれる解決策の数が多いということは、ある問題点の集合に寄与する解決策が多く会議に現れたことを意味する。このようなノードも機能単位を生成する際に重要な位置を占めるときが多い。そこで、まだどの機能単位にも含まれていないノードのうちで最も多くの解決策を含むノードを次の注目点の候補として提供することは有効な支援となる [最大ノード戦略]。

上記の三つの戦略は、機能単位を作成する際に、いかなるタイミングでも利用できる。また、どの戦略をとっても、一般的には注目点の候補は複数存在する。最良の戦略と最良の候補は、会議の内容、解決策ネットの形状的特徴、機能単位作成作業の進行具合などにより異なる。そこで、現状では、注目点支援のタイミングは分析者に任せ、注目点を提供したあとの作業も強制するのではなく分析者に判断にゆだねることになっている。

3.3 機能単位グラフ生成

作成された機能単位は次の会議や開発プロセス後段に重要な情報として渡される。機能単位ごとに項目をたて簡単な説明文を付けた文書 (USP 機能単位分析に基づく議事録) は、大きさとして通常の議事録程度になる。前述のように通常の議事録には問題点も多いが、会議全体

を把握するためには妥当な量だと思われる。USP機能単位分析により会議全体を対象システムの機能の観点から把握するために適した量の情報単位が得られたことになる。通常の議事録とは異なり、USP機能単位分析に基づく議事録は網羅的なPSGの解決策へのトレーサビリティを保持しながら作成されているので、利用者が適宜詳細な要求を参照することができる点が優れている。

上記のUSP機能単位分析に基づく議事録のように単に機能単位を並べるだけではなく、機能単位間に関係を付けて図示することが会議全体を把握するためにより有効である場合も多い。そこで、USP機能単位分析では、得られた機能単位の間の関係を解決策ネットのアークを利用して生成する。いま、機能単位Fに含まれる解決策が解決策ネットのノードNに含まれ、機能単位Gに含まれる解決策が解決策ネットのノードMに含まれているとする。もしNからMへのアークが存在すれば、機能単位Fから機能単位Gに関係を付ける。これをすべての機能単位のすべての解決策について行なった結果を図示したものが機能単位グラフである。図6に示すように、機能単位グラフは、解決策ネットの持つ非循環性を持たない。図6では、相互に関係ある機能単位間のアークを両矢印にしたり、複数の機能単位のグループ化(図5では点線で表現、グループの要素すべてに共通のアークはグループのアークとする)を用いることで、アークの数を減らし、見やすさの向上をはかっている。

4 汎用な機能単位分析

前述のようにPSGは会議の内容を「問題点と解決策」の観点から整理したものであり、問題点と解決策に分類される要求項目をノードとする有向グラフである。しかしながら、会議には、問題点と解決策の観点からは重要であるにも関わらず、問題点にも解決策にも分類できないようなものが現れることがある。たとえば、問題点の根拠である事実とか、解決策に関する一般的な情報などである。このようなものを取り入れるためには、PSGは拡張してゆく必要がある。今後、PSGの研究が進み様々なノードが導入される可能性もある。USP機能単位分析のように他の分析結果を利用する分析は、入力となる他の分析結果の形式の変更に耐えられるように汎用性を持つ必要がある。すなわち、USP機能単位分析は将来的なPSGの形式の変更にある程度は耐えられるようになっていなければならない。

USP機能単位分析においてPSGの形式の変更が影響をする点は、項目の種類の変更による要求項目マトリクスの縦横の要求項目の種類の変更とPSG内の推移律をほどくためのマトリクス演算の変更である。これらの変更に耐えられるように、PSGのノードの種類とPSG内の推移律の定義を与え、これらの定義をもとに要求項目マトリクスを生成し推移律をほどくための演算を行うようにUSP機能単位分析を拡張する。図7はPSGの定義の例である。type行でPSGのノードの種類を列挙して定義し、推移律ごとにrule行を記述する。図7では、「○」と「×」と「□」を定義し、3.1節で述べた4個の推移律と「□」に関する新しい規則を記述している。rule行は':'の前に条件を記述し、後に追加する関係を記述する。たとえば、図7の4番目のrule行は「もし○から×に関係がありその×から他の×に関係があれば、最初の○から最後の×へ関係を追加する」と読み、図7の5番目のrule行は「もし○か×か□から□に関係がありその□から他の○か×か□に関係があれば、最初の○か×か□から最後の○か×か□へ関係を追加する」と読む。なお、○や×に付いている数字は':'の前後でノードの対応を取るためのものである。

```
type ○, ×, □ ;
rule ×1 -> ○2 -> ×3 -> ○4 : 1 -> 4 ;
rule ×1 -> ○2 -> ○3 : 1 -> 3 ;
rule ○1 -> ○2 -> ×3 : 1 -> 3 ;
rule ○1 -> ×2 -> ×3 : 1 -> 3 ;
rule ○×□1 -> □2 -> ○×□3 : 1 -> 3 ;
```

図7: PSGの定義の例

5 柔軟な機能単位分析

前節ではPSGのノードの種類の変更に関するUSP機能単位分析の拡張を行ったが、本節では要求項目マトリクスの値を寄与関係を表す0/1の2値から寄与度を表す0から1までの実数値に変更して、解決策ネットの柔軟な生成を実現する。

寄与関係から寄与度への変更に伴い、PSGから要求項目マトリクスを生成するときの計算式、PSGの推移律をほどくためのマトリクス演算、解決策間の寄与度に関する半順序を得るための方法を新たに考慮する必要がある。PSGのアークから要求項目マトリクスの初期値の決定については、そのアークの属性値(確信度や関連度など)やアークの両端のノードの属性(確信度や重要度など)を用いて計算する式を与える。また、PSGの推移律をほどく規則を拡張して関連する寄与度から新たな寄与度を求める式を与える。さらに、閾値を指定して、要求項目マトリクスを2値化してからこれまでと同様の方法で解決策間の半順序を求める。

PSGの定義を図8のように拡張して、計算式や値を分析者が指定できるようにする。param行は計算式で用いる定数の定義であり、thresh行は要求項目マトリクスの2値化のための閾値の定義である。new行ではPSGから要求項目マトリクスの初期値の決定のための式を定義する。図8ではノードの種類に関係なくPSGのアークはそのアークの属性certの値とすることが定義されている。rule行では、関連する要求項目マトリクスの値(図8ではV1, V2など)から新しい値を計算する式を追加した(where部)。図8では加重平均と現在の値との大きい方を新たな値としている。

計算式やパラメータを変更することで、1個のPSGから様々な解決策ネットの生成が可能になる。そこで、分析者は解決策ネットを適宜調節して機能単位生成作業を行うことができる。

分析者が解決策ネットを調節するためには、生成された解決策ネットの評価を行わなければならない。良い解決策ネットとは、それを使って作業がしやすいもの、すなわち、機能単位を作成しやすいものである。しかるに、機能単位が作成しやすいかどうかは、実際に機能単位の作成を始めなければ分からない。そこで、解決策

```
param oo = 1, ox = 0.5, xo = 0.5, xx = 1 ;
thresh 0.2 ;
type ○, × ;
new ○× 1 -> ○× 2 : *.cert ;
rule × 1 -V1-> ○ 2 -V2-> × 3 -V3-> ○ 4
       : 1 -V4-> 4
  where V4 =
     max(V4, (V1*xo+V2*ox+V3*xo)/(2*xo+ox)) ;
rule × 1 -V1-> ○ 2 -V2-> ○ 3 : 1 -V3-> 3
  where V3 = max(V3, (V1*xo+V2*oo)/(xo+oo)) ;
rule ○ 1 -V1-> ○ 2 -V2-> × 3 : 1 -V3-> 3
  where V3 = max(V3, (V1*oo+V2*ox)/(oo+ox)) ;
rule ○ 1 -V1-> × 2 -V3-> × 3 : 1 -V3-> 3 ;
  where V3 = max(V3, (V1*ox+V2*xx)/(ox+xx)) ;
```

図 8: PSG の定義の例 (2)

ネットの柔軟な生成を行う拡張により、USP 機能単位分析の作業の流れは変化し、解決策ネットと機能単位の双方を変更しながら解決策ネットの生成と機能単位の作成を漸進的に行うことになる。

一般的に、ある情報 A の変更作業とそれをもとに他の情報 B を作成する作業を漸進的に行う場合、情報 A が変更されたときの既に作成した (部分的な) 情報 B の有効利用が問題となる。USP 機能単位分析において、解決策ネットを変更したときに、既に作成していた機能単位を利用し、その後の作業を円滑に続行する方法を考慮しなければならない。なぜならば、機能単位の作成は解決策ネットのノードをまとめてゆく作業であり、解決策ネットの変更により新しいノードには異なる機能単位に分類した解決策が混在したり機能単位に既に含まれる解決策とまだ含まれない解決策が混在する場合が起こりうるからである。USP 機能単位分析では、このような混在が起こったときの対処方法を用意している (後述)。

6 PSG の変更への対処

前節では解決策ネットの変更に対する機能単位の作成の対処について言及したが、PSG の変更に対する USP 機能単位分析の対処がより大きな問題として存在する。前者は後者の特殊な場合として取り扱うことができるので、本節ではまず PSG の変更への対処方法を述べ、最後に解決策ネットの変更への対処方法を述べる。

PSG が変更されても、解決策ネットは自動的に生成できる。問題となるのは分析者が作成した (または作成途中の) 機能単位である。機能単位に関する PSG の変更への対処は、以下の 9 個の場合に分けられる。

6.1 解決策が変更されない場合

PSG の変更のうちでもっとも扱いやすいものは、解決策の増減がなく項目内容や項目番号の変更もない場合、すなわち、問題点などの他の要求項目やアークの増減／変更のみである場合である。このような変更に対しては以下のような対応手段を用意する。

1. 解決策ネットのどのノードにどの解決策が属しているかを利用して、ノードの新旧対応関係を洗い出す。具体的には、旧ノード X に含まれる解決策が 1 つでも新ノード Y に含まれるときのみ、X と Y の間に新旧対応関係があるとする。
2. 新旧対応関係をもとにノードを以下の 5 つに分類する。
   - OK：同じ旧機能単位に含まれる旧ノードの解決策のみから構成されるノード。
   - NG：複数の旧機能単位に含まれる旧ノードの解決策が混在するノード。
   - NC：どの旧機能単位にも含まれない旧ノードの解決策のみから構成されるノード。
   - OK'：同じ旧機能単位に含まれる旧ノードの解決策およびどの旧機能単位にも含まれない旧ノードの解決策のみから構成されるノード。
   - NG'：複数の旧機能単位に含まれる旧ノードの解決策およびどの旧機能単位にも含まれない旧ノードの解決策が混在するノード。
3. 分類毎に以下のような支援を行なう。
   - OK ノードは新機能単位に入れる。
   - NG ノードは新機能単位には入れずに、NG ノードがどの旧機能単位 (複数) に含まれるかを分析者に示す。分析者の講ずべき処置は、ノードを分割するのか、示された機能単位のどれかに入れるか、そのノード用に新たな機能単位を作るかである。新たな機能単位を作ったときは、関連する機能単位に含まれる他のノードを新たな機能単位に移すかどうかを考慮する必要がある。
   - NC ノードについては何もしない。将来的に、分析者が通常の機能単位分析によって処理する。
   - OK' ノードは新機能単位に入れる。ただし、OK' ノードであることおよびそれに含まれるどの解決策が「新参物」かを分析者に明示する。分析者の講ずべき処置は、新参物も同じ新機能単位に入れてよいかを判断し、入れるべきではないと判断した場合は NG ノードと同様の処置を行なう。
   - NG' ノードについては NG ノードに準ずる。

上記の対処方法は、以下に述べるより扱い難い変更への対処の基礎となる。

6.2 解決策の項目番号が変更される場合

解決策については増減がなく項目内容の変更もないが項目番号が変更されている場合 (問題点などの他の要求項目やアークの増減／変更は許す) に対して以下のような対処手段を用意する。

- 旧版のデータの項目番号を新たな項目番号に置き換えたものを用意し、それと新版のデータについて前述の「解決策が変更されない場合」と同様の処理を行なう。
- 変更された項目番号について対応表を分析者が参照できる機能を提供する。

6.3 解決策が増加する場合

旧 PSG の解決策がすべて新 PSG に存在し、さらに新しい解決策が加わっている場合 (問題点などの他の要求項目やアークの増減／変更は許す) に対しては、旧 PSG の解決策の項目番号が変更されているかどうかによって前述の「解決策が変更されない場合」または「解決策の項目番号が変更される場合」と同様の処理を行なう。また、増えた解決策を参照する機能を用意する。解決策に増減がない場合と比べると、新たに解決策が加わったこ

とにより、NCノードやOK'ノードやNG'ノードが増加したり、NCノードやOK'ノードやNG'ノードの中の新参物が増加する。

6.4 解決策が減少する場合

新PSGでは存在しない解決策が旧PSGには存在し、さらに新しい解決策が加わっていない場合(問題点などの他の要求項目やアークの増減／変更は許す)に対しては、旧PSGの解決策の項目番号が変更されているかどうかによって前述の「解決策が変更されない場合」または「解決策の項目番号が変更される場合」と同様の処理を行なう。解決策に増減がない場合と比べると、解決策が減ったことにより、要求項目を含まない機能単位が現れる可能性がある。そのような機能単位は新機能単位から削除する。

6.5 解決策が単純に増減する場合

前述の「解決策が増加する場合」と「解決策が減少する場合」が複合した場合である(「単純に」とした理由は次項を参照)。これらの場合と同様に、旧PSGの解決策の項目番号が変更されているかどうかによって前述の「解決策が変更されない場合」または「解決策の項目番号が変更される場合」と同様の処理を行なう。

6.6 解決策の項目内容が変更される場合

項目内容の変更は解決策の増減とみなすことができる。これ以降の項で述べる変更も解決策の増減とみなすことができる。前項では、「単純に」という語句により、これらの場合を除いた。解決策の項目内容が変更される場合については、以下のような支援を行なう。

- 基本的には「解決策が単純に増減する場合」と同様の処理を行なう。
- 増えた解決策と減った解決策を参照する機能に加えて、項目内容を変更した新旧の解決策のうちで対応する(同一視できる)ものを1対1に指定する機能を追加する。この機能で対応を与えられた解決策については、「解決策の項目番号が変更される場合」に準ずる対処を行なう。

6.7 1個の解決策が複数に分割される場合

前述の「解決策の項目内容が変更される場合」では新旧の解決策が1対1に対応付けられるが、本項では1個の旧解決策Xが複数の新解決策Ysに分割される場合について述べる。この場合、以下のように対処する。

1. 新解決策Ysを旧解決策に加える。
   - 旧解決策Xが旧機能単位Fに属している場合、新解決策Ysすべてもその旧機能単位Fに属しているとする。
   - 旧解決策Xがどの旧機能単位にも属していない場合、新解決策Ysすべてもどの旧機能単位にも属していないとする。
2. 旧解決策Xを旧解決策から取り除く。
3. 「解決策が変更されない場合」と同様の対処を行なう。

6.8 複数の解決策が1個に統合される場合

本項では複数の旧解決策Xsが1個の新解決策Yに統合される場合について述べる。それらの旧解決策Xsが入っている旧機能単位Fsの違いにより、以下のような異なる対処を行なう。

- 旧機能単位Fsがない場合：新解決策Yをどの旧機能単位にも属していない解決策として旧解決策に加え、旧解決策Xsを取り除き、「解決策が変更されない場合」と同様の対処を行なう。
- 旧機能単位Fsが1個だけの場合：新解決策Yを旧機能単位Fsに属している解決策として旧解決策に加え、旧解決策Xsを取り除き、「解決策が変更されない場合」と同様の対処を行なう。さらに、新解決策Yを分析者に提示する機能を用意し、新解決策Yが新機能単位Fsに含まれない場合は、分析者の責任でそれを新機能単位Fsから除いてもらう。
- 旧機能単位Fsが複数個の場合：新解決策Yを旧機能単位Fsすべてに属している解決策として旧解決策に加え、旧解決策Xsを取り除き、「解決策が変更されない場合」と同様の対処を行なう。ここで、新解決策Yは旧機能単位Fsすべてに属しているため、必然的に、新解決策Yが入る新ノードはNGノード(またはNG'ノード)になる。

6.9 複数の解決策が複数に統合し分割される場合

前述の「解決策の項目内容が変更される場合」では新旧の解決策が1対1に対応付けられるが、本項ではn対mに対応付けられる場合について述べる。このように複数の旧解決策Xsが複数の新解決策Ysに分割される場合には、新解決策Ysすべてを旧機能単位Fsすべてに属している解決策として旧解決策に加え、旧解決策Xsを取り除き、「解決策が変更されない場合」と同様の対処を行なう。ここで、各々の新解決策Ysは旧機能単位Fsすべてに属しているため、必然的に、新解決策Ysが1個でも入る新ノードはNGノード(またはNG'ノード)になる。

6.10 解決策が増減する場合

前述の「解決策が単純に増減する場合」「解決策の項目内容が変更される場合」「1個の解決策が複数に分割される場合」「複数の解決策が1個に統合される場合」「複数の解決策が複数に統合し分割される場合」は自動的には判断できない。そこで、支援ツールでは、減った(旧PSGのみに現れる)解決策と増えた(新PSGのみに現れる)解決策を表示し分析者がお互いを対応付ける機能を用意し、その対応の付け方が、ない／1対1／1対n／n対1／n対mの違いによりどの場合かを判断し適宜支援する。

6.11 解決策ネットの変更への対処

5節で言及した解決策ネットの変更は、PSGの変更における解決策が変更されない場合とみなすことができる。そこで上記のPSGの変更への対処の第1の場合と同様の対処を行なう。

7 支援ツール

現在、我々はUSPオフライン法のための支援ツールである *USP-CASE* を開発している。ここでは、*USP-CASE* に組み込まれているUSP機能単位分析のためのいくつかの支援機能に限定し、その概略を示す。それらの支援機能は、各種自動生成と機能単位生成支援に大別される。

7.1 自動生成

PSGの定義を利用してPSGから要求項目マトリクスを生成し要求マトリクス上でのPSGの推移律を展開し、さらに閾値を利用して解決策間の寄与度に関する半順序を求め、解決策ネットを生成する作業は完全に自動化できる。さらに、*USP-CASE* では、解決策ネットが非循環有向グラフであることを利用して解決策ネットを自動配置し描画する。また、PSGの定義を変更したり、Tcl/Tk[7] のscaleのようなグラフィカルな入力方法によりパラメータや閾値を簡単に変更できるようにする。

作成した(または、作成途中の)機能単位間の関係を解決策ネットのアークから求め機能単位グラフを生成する作業も完全に自動化できる。機能単位グラフについても自動配置を行なうが、機能単位グラフは解決策ネットと異なり一般的なグラフ構造であるので自動配置には限界があると思われる。そこで、*USP-CASE* では自動配置された機能単位グラフを分析者が適宜配置変更できる機能を付加する。

7.2 機能単位生成支援機能

解決策マップと機能単位グラフを図で表示し、解決策マップと解決策に関する「ノード - 項目番号 - 項目内容」の関係を表で表示し、「機能単位 - ノード」の関係を表として分析者に作成させる。分析者は、複数の任意の解決マップや機能単位グラフのノードおよび解決策番号に注目点を設定できる。これらのノードや番号の間の包含関係に基づいて、注目点は連動する。たとえば、ある解決マップのノードに注目点を設定すると、そのノードに含まれる解決策の番号やそのノードが含まれる機能単位グラフのノードにも注目点を自動的に設定する[注目点連動]。*USP-CASE* では、この注目点連動は機能単位生成支援の中に閉じず、PSGへのトレーサビリティを利用して対応するPSGの部分を表示したり、さらに、PSGからコーバスへのトレーサビリティを利用して対応するコーバスを表示することができる。また、注目点が設定されると、その注目点が画面内に表現できるように表示を変更する[適正画面表示]。

また、3.2節で述べた注目点支援を行なうためのコマンドを戦略ごとに提供している。これらのコマンドを実行すると、コマンドに関連付けられた戦略を用いて複数の注目点(候補)を設定する。この注目点(候補)の設定においても、上記の注目点連動および適正画面表示を行なう。

さらに、6節で述べたPSGや解決策ネットの変更への対処を行なうための支援機能も提供している。

8 分析の実施

実験的な会議および現場の会議を利用して数回の分析を行った。たとえば、図6のような機能単位グラフを得た会議は、参加者6人と司会者1人が2時間32分間にわたり自由形式で行なった。全発話数は1860個であり、PSGの項目数は275項目(問題点が124個、解決策151個)であり、解決策ネットのノードが67個でアークが70本であり、最終的に作成された機能単位は10個であった。これらの分析経験から、USP機能単位分析は実際規模の会議に対して適用可能であることが実証できたと考える。また、これらの分析経験から、コーバスやPSGの作成にかかる時間を短縮することが今後の課題となった。*USP-CASE* では、これらの時間がかかる作業の効率の改善も大きな目的である。ただし、機能単位分析についてはこのような分析時間の問題はほとんどなかった。

9 おわりに

本稿では、会議を用いた要求獲得分析技法であるUSPオフライン法における機能単位分析について述べた。そこでは、会議に現れる解決策を問題点に対する寄与関係や寄与度によりグループ化することが機能単位を発見するうえで有効な支援になることがわかった。

今後の課題としては、最良のパラメータや戦略の選択などの機能単位分析に関するノウハウの蓄積、*USP-CASE* の開発、USPオフライン法の適用と評価、USPオンライン法の提供などが考えられる。

謝辞

本稿で報告した研究成果を得るために助言をいただいた当研究所ユーザ指向システムプランニングプロジェクトの研究員各位(USP法の研究に従事)に感謝します。

参考文献

[1] *procs. of IEEE International Symposium on Requirements Engineering*, 1993.

[2] A. Finkelstein, "Requirements Engineering: a review and research agenda", *procs. of Asia-Pacific Software Engineering Conference*, 1994.

[3] C. Potts, K. Takahashi, J. Smith, K. Ota, "An Evaluation of Inquiry-Based Requirements Analysis for an Internet Service", *procs. of IEEE International Symposium on Requirements Engineering*, 1995.

[4] 海谷治彦, 佐伯元司, "ソフトウェア仕様作成会議支援ツールの設計", 電子情報通信学会, **1993-07**, 1993.

[5] "全社システム計画技法 FUJITSU EPGII/C-NAP 解説書", 1990.

[6] 片山、蓬莱、渡部、土井、園部, "ユーザ指向ソフトウェア開発のための要求分析法の実践について", *procs. of WINGS*, 1996.

[7] 宮田重明, 芳賀敏彦, "Tcl/Tk プログラミング入門", オーム社, 1995.