ソフトウェア工学 115−16
（1997. 8. 1）

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の適用と改良

蓬莱 尚幸†
(株) 富士通研究所 情報社会科学研究部

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きた。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機能単位分析について、適用実験について報告する。具体的には、社内で行なわれた実際の会議の録画／録音に USP オフライン法を実験的に (実際の会議へのフィードバックをせずに) 適用した一連の実験から 3 例について、データの性質と分析作業を関連つけて述べる。

この適用実験は以下の 3 つの目的をもって行なわれた。

- 機能単位分析および機能単位分析で用意されている各種の支援の有効性の検証。
- 作業手順に関するノウハウの蓄積と得られたノウハウをもとにした標準的な分析手順の構築。
- 今後の機能単位分析およびその支援ツールの拡張や改良のための課題の把握。

本稿では、実験結果をもとに、これらの目的についても議論する。

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

Hisayuki Horai

Information Science Lab.,
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. Function Unit Analysis is a part of USP Offline Method, in which we capture function units of the target system. In this paper, we report experience that we applied it to several real meetings, verify effectiveness of Function Unit Analysis and its support facilities, obtain some knowhow concerning its procedure, establish a standard procedure of Function Analysis based on the obtained knowhow, and argue the future works including improvement of the analysis method and tool.

†*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獲得分析において顧客の要求を十分に獲得する必要がある [1, 2]。開発者にとって、要求獲得分析は「そこにある」要求を採取するという受動的な活動ではなく、顧客自身が要求を明確化し統一化してゆく過程において助言を与えながら最終的な要求を獲得するという能動的な活動である。このような視点にたって、我々はユーザ指向要求獲得分析技術である USP 法 [6] の研究を行なっている。

上記のような特徴を持つ要求獲得分析では、顧客同士および顧客 - 開発者間のコミュニケーションが非常に重要な問題となる [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、要求獲得分析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 USP 法ではコミュニケーション手段として要求会議を用いる。

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]、我々は、会議の進行の支援 (オンライン法) 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の支援 (オフライン法) も目的としている。

2 USP オフライン法

我々は、オフライン法の一つとして USP オフライン法を提案してきた。USP オフライン法 (図 1) は、会議で現れた情報をもれなく集めること [網羅性]、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きわだたせた分析結果を得ること [多面性]、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すること [トレーサビリティ] を主な特徴としている。

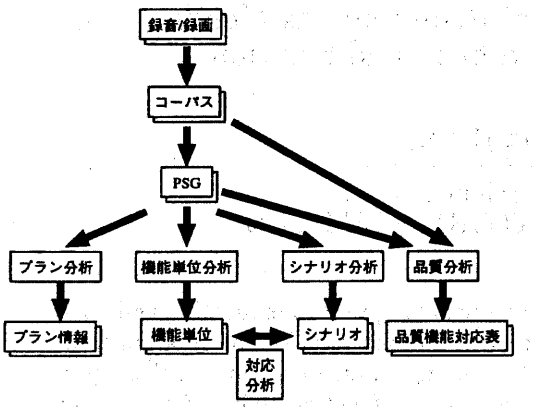

図 1: USP オフライン法 (抜粋)

2.1 コーパスと PSG

次に、本稿の主題である機能単位分析 [7] までの USP オフライン分析の流れについて簡単に説明する。

```
├× 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

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 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、決定未決状況やコーパスへのトレーサビリティなどの属性を与えるが図 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。PSG の木構造とジャンプで表される要求項目間の有向関係をアークと呼ぶ。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。

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に機能単位分析の流れを示す。機能単位分析の結果である機能単位は解決策の集合である。すなわち、機能単位分析は、解決策のグループ化を行ない、名称 (機能単位名) をつけることである。解決策のグループ化の支援として、機能単位の叩き台となるような解決策のグループ (解決策ネット) を PSG から自動生成する [解決策ネット生成]。分析者はこの叩き台をもとに機能単位を作成する。この作業は、分析者が PSG 内の要求項目を読みながら行なうことになるが、その際、解決策ネットを用いた注目点の移動による支援が用意されている [注目点支援]。さらに、機能単位分析では、作成した機能単位の間の関

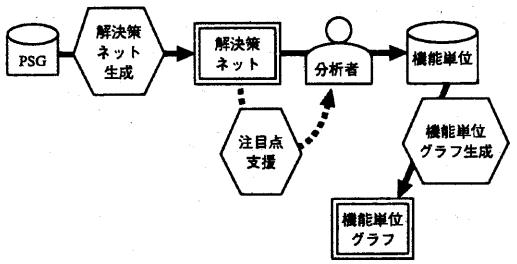

図 3: 機能単位分析

係を解決策ネットのアークから自動的に抽出し図示する[機能単位グラフ生成]。

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(解決策ノード)および解決策ノード間の関係を自動的に生成することは有効であると考えられる。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)。

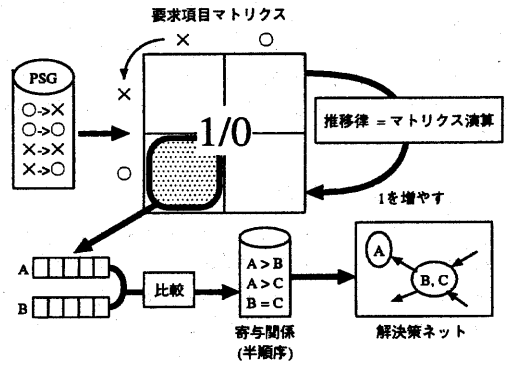

図 4: 解決策ネット生成

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。

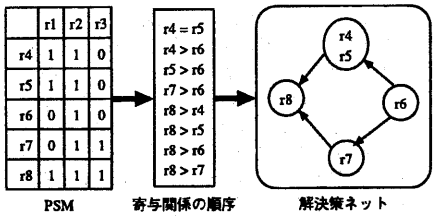

図 5: 解決策ネット生成の例

- 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。
- 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})、同じ解決策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。

このように、解決策ネットのアーク(解決策アーク)は、解決策ノードに含まれる解決策の寄与関係の量の順序関係を表している。そこで、解決策アークで表される解決策ノード間の関係は、解決策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と考えられる。

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る。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と考えられる。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、次に分析者がどの解決策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と考えられる。なぜなら、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からである。

解決策ネット上で近い解決策ノードに含まれる解決策は遠い解決策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と予想できる。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解決策ノードと解決策アークで直接結ばれた解決策ノード(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と考えられる[近隣ノード戦略]。

解決策ネット内で最も上位にある解決策ノードには、最も多くの問題点に寄与する解決策が含まれている。このような解決策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と予想できる。そこで、まだどの機能単位にも含まれていない解決策ノードのうちで最上位の解決策ノード(最上ノード)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と考えられる[最上ノード戦略]。

解決策ノードに含まれる解決策の数は解決策ノードごとに異なる。ある解決策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で発言されたことを意味する。このような解決策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と予想できる。そこで、まだどの機能単位にも含まれていない解決策ノードのうちで最も多くの解決策を含む解決策ノード(最大ノード)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と考えられる[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。すなわち、分析者は任意のタイミングで(好きなときに)任意の(好きな)戦略をとることができる。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と考えられるので、戦略や候補の選択のためのノウハウが必要となる。そこで、このノウハウの獲得を適用実験の目的の一つに設定した。

3.3 機能単位グラフ生成

作成した機能単位を羅列するのではなく、機能単位間に関係を付けて図示することは、会議全体を把握するためにより有効であると考えられる。そこで、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6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。

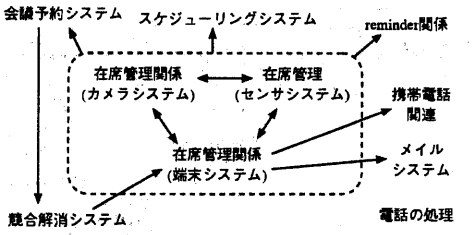

図 6: 機能単位グラフの例

4 適用実験の目的

社内で行なわれている実際の会議を録画または録音し、USP オフライン法を適用する実験を行なった。この適用実験の目的には、機能単位分析における支援に関するものとして、以下の3項目が含まれている。他に、機能単位分析以外の分析方法の有効性や USP オフライン法全体の有効性の検証も目的に含まれているが、本稿では機能単位分析における支援に関するものについてのみ議論する。

4.1 目的1: 支援の有効性

機能単位分析について、以下のような観点から支援の有効性の検証することが適用実験の一つの目的である。

- 解決策ネット生成で自動生成される解決策ノードは機能単位を作成するときの叩き台として有効か? [解決策ノードの有効性]
- 解決策ネット生成で自動生成される解決策アークは機能単位を作成するために有効に利用できるか? [解決策アークの有効性]
- 解決策ネット内の注目点として解決策ノードを示唆し、その解決策ノードを核として機能単位を成長させる手法は有効か? [注目点支援の有効性]
- 近隣ノード戦略は機能単位を成長させるために有効か? [近隣ノード戦略の有効性]
- 最上ノード戦略は機能単位の核になる解決策ノードの示唆として有効か? [最上ノード戦略の有効性]
- 最大ノード戦略は機能単位の核になる解決策ノードの示唆として有効か? [最大ノード戦略の有効性]

4.2 目的 2: ノウハウの獲得

解決策ネットを用いながら、解決策をまとめて、機能単位を作成する際の手順的なノウハウを獲得することが適用実験の一つの目的である。特に、注目点支援の利用のタイミングに関するノウハウを獲得する。これは会議の内容、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と考えられるので、複数の適用実験を行ない、最終的には標準的な分析手順としてまとめたい。

4.3 目的 3: 分析方法の改良

様々な種類の会議に適用することで、分析方法の問題点を洗い出し、今後の分析方法の改良に対する課題をみつけることも適用実験の目的の一つである。また、分析ツールである *USP/CASE* を利用／評価し、次バージョンにおいて改良すべき点をみつけることも分析方法の改良の一部として実験の目的とする。

5 適用実験の概要

本稿では 3 適用実験についてのみ報告する。本章では、適用実験について、会議自体の概略、機能単位分析の進行状況について述べる。表 1に各会議の統計データを示す。

表 1: 会議の統計データ

	実験 1	実験 2	実験 3
発話数	847	575	436
要求項目数 (解決策数)	342 (102)	437 (244)	379 (197)
解決策ノード数	45	70	102
機能単位数	11	19	17

5.1 [実験 1] 提案検討会議

顧客サイトに常駐しているＳＥと社内コンサルテーションを行なうＳＥによる顧客への次期提案に関する会議である。話題は、提案システム自身だけでなく顧客へのアプローチ方法についても議論されたので、提案システムの機能単位以外に顧客への将来アクション計画も得られた。

5.1.1 解決策ネットの特徴

解決策ネット生成により得られた解決策ノードに含まれる解決策の数 (表 2) には以下のような特徴がある。

- 非常に大きな解決策ノードが 2 個存在する。
- 1 個の解決策からなる解決策ノード (単一ノード) が半分以上存在する。

他の解決策ノードと連結していない解決策ノード (孤立ノード) を除くと、解決策ネットは 5 つの連結していないサブネット (net_{1-5}) に分かれた。その内、4 個のサブネット (net_{1-4}) の構成要素である解決策ノードには解決策が 1 個ないしは 2 個のみ含まれた (大きな解決策ノードはなかった)。また、あまり多くの解決策アークは生成されず、孤立ノードは 26 個 (57.8%) と多かった。

表 2: 実験 1 の解決策ノードの特性

解決策数	1	2	3	4	5	6	12	13
ノード数	26	10	2	2	1	2	1	1

5.1.2 分析作業手順

以下の手順により分析を行なったが、分析者は正しい手順であったと評価している。

1. 最大ノード戦略を行なうことで、解決策を 3 個以上含む解決策ノード (9 個) と解決策を 2 個含む解決策ノードの一部 (5 個) に異なる名称を与えて機能単位にできた。
2. net_5 について、近隣ノード戦略により機能単位の拡大を行なった結果、すべての解決策ノードがいずれかの機能単位に含まれた。
3. net_{1-4} については、最上ノード戦略を用いて機能単位を作ろうとしたが、実際は最上ノードだけでなく、上位から数個の解決策ノードをまとめて機能単位を作った。さらに、近隣ノード戦略を用いることで、net_{1-4} は各々 1 個ずつの機能単位となった。
4. 残った孤立＆単一ノードは、いずれかの機能単位に入れることができた。
5. 機能単位のいくつかを一つにまとめる作業を行ない、最終的な機能単位を得た。このまとめの段階では、近隣ノード戦略が使える可能性があったが、実際は機能単位に付けた名前やそれに含まれる解決策の内容から判断してまとめた。

5.2 [実験 2] 方法論検討会議

ソフトウェア開発の上流工程の方法論について検討の会議である。ある参加者がある方法について自作の資料を用意し、それを題材に他の方法も含めて議論された。開発方法およびその部分的なプロセスが機能単位として抽出された。

5.2.1 解決策ネットの特徴

解決策ネット内に孤立ノードは存在しなかった。また、解決策ネット生成により得られた解決策ノードに含まれる解決策の数 (表 3) には以下のような特徴がある。

- 非常に大きな解決策ノードが 2 個存在する ([実験 1] より顕著)。
- 中規模の解決策ノードが多く存在する ([実験 1] より多い)。
- 単一ノードは約半分である ([実験 1] より割合が小さい)。

表 3: 実験 2 の解決策ノードの特性

解決策数	1	2	3	4	5	7	8	9	10
ノード数	36	14	2	6	4	1	2	1	2
	25	53							
	1	1							

5.2.2 分析作業手順

以下の手順により分析を行なったが、分析者は正しい手順であったと評価している。

1. 最大ノード戦略を行なうことで、解決策を 3 個以上含む解決策ノードから機能単位を作成できた。
2. 解決策を 2 個以下しか含まない解決策ノードからは、開発方法またはその部分機能が明記されているものを機能単位とした。
3. 残りの解決策を 2 個以下しか含まない解決策ノードについては、近隣ノード戦略を変形して機能単位に入れていった (変形近隣ノード戦略)。すなわち、近隣ノード戦略では既に作成した機能単位とそれに含まれる解決策ノードに注目するが、ここでは機能単位に入っていない解決策ノードとその近隣ノードが含まれる機能単位に注目した。

5.3 [実験 3] 障害対策会議

ある製品の保守体制および障害対策について、その製品に関連する製造部門、ＳＥ、ＣＥなどが議論した会議である。技術的な話題より体制に関する話題が多く、「体制」や「体制変更方法」が機能単位として得られた。

5.3.1 解決策ネットの特徴

解決策ネット生成により得られた解決策ノードに含まれる解決策の数 (表 3) には以下のような特徴がある。

- 非常に大きな解決策ノードが 2 個存在する ([実験 1] と同様)。
- 単一ノードが非常に多い ([実験 1,2] より顕著)。
- 解決策を 2 個含む解決策ノードも非常に多い。

表 4: 実験 3 の解決策ノードの特性

解決策数	1	2	3	4	6	12	16
ノード数	61	24	7	6	2	1	1

また、あまり多くの解決策アークが生成されず、孤立ノードは 52 個と多く、解決策ネットは多くの小さいサブネットに分かれた。

5.3.2 分析作業手順

以下の手順により分析を行なったが、分析者は正しい手順であったと評価している。しかしながら、解決策ネットには小さい解決策ノードがばらばらに存在しているので、[実験 1, 2] に比べて、機能単位の作成が難しかったことも指摘している。

1. 最大ノード戦略を行なうことで、解決策を 3 個以上含む解決策ノードから機能単位を作成できた。
2. 解決策を 2 個以下しか含まない解決策ノードについては、変形近隣ノード戦略を用いたが、多くの解決策ノードが機能単位にならずに残ってしまった。
3. 残った解決策ノードについては、1 つずつ内容をみながら、既存の機能単位に入れるか、または、新たに機能単位にしていった。この際、既存の機能単位を一覧するために、機能単位グラフが有用であった。

6 考察

上記の実験結果をもとに、機能単位分析に関して以下のような考察を行なった。

6.1 支援の有効性

本節では、4.1節で述べた適用実験の目的であった有効性について考察し、その後に他の観点から有効性について論じる。

解決策ノードの有効性

3 個以上の解決策を含む解決策ノードは、それに名前を付けて機能単位にできたことから、機能単位を作成するために有用であることが検証された。ただし、分析作業をとおして、作成した個々の機能単位が小さ過ぎると判断して、複数個を適切な機能単位としてまとめ直す場合があることも判明した。

実際の会議について解決策ネット生成を適用すると、多くの単一ノードが生成される傾向がある。単一ノードは、(a) 既存の機能単位に入れるか、(b) 単独で新しい機能単位とするか、(c) 複数個まとめて新しい機能単位とすることになる。適用実験では、もっとも難しいと思われる (c) は稀有であり、(b) より (a) 方が優先された。また、(b) は (a) より困難であることが分析者から報告されている。

よって、適用実験からは、分析作業の初期段階で新しい機能単位をみつけるためには、3 個以上の解決策ノードは有効であるが、単一ノードは有効ではないと結論付けられる。解決策を 2 個含む解決策ノードは、両者の中間にあたる評価が得られた。

解決策アークの有効性

解決策アークは解決策ノード間の類似性を表している。解決策を 2 個以下しか含まず単独で機能単位とできない解決策ノードを近隣ノード戦略や変形近隣ノード戦略を用いて既存の機能単位に入れることができるという意味で、解決策アークは解決策ノードをまとめるために有効であることが適用実験により検証された。

注目点支援の有効性

本稿で述べた3適用実験すべてにおいて、分析者は、解決策をまとめて機能単位を作成する際に、解決策ネットを用い、機能単位になりそうな解決策ノードを探すことから開始して、機能単位分析に成功した。この事実より、解決策ノードを注目することで機能単位を作成すること、および、そのために注目点を示唆することの有効性が検証できた。

近隣ノード戦略の有効性

解決策を2個以下しか含まず単独で機能単位にできない解決策ノードを既存の機能単位に入れる作業においてある程度の有効性が適用実験により認められた。

しかしながら、既存の機能単位を拡大する近隣ノード戦略よりどの機能単位にも含まれていない解決策ノードを既存の機能単位に入れてゆく変形近隣ノード戦略の方がはるかに有効であることが分かった。

また、適用実験について考察を加えると、近隣ノード戦略だけでは変形近隣ノード戦略と同等の結果を得るためのは困難である(時間がかかる)が、逆に、変形近隣ノード戦略だけで近隣ノード戦略と同等の結果を得ることができると予想させる。

最上ノード戦略の有効性

[実験1]により、小量の小さい解決策ノードから構成される独立したサブネットについては、上位の解決策ノードを利用した機能単位の作成が有効であることが検証された。

しかしながら、最上ノードからだけでは機能単位を作成することは困難で、上位の数個のノードを同時に考慮する必要がある。この事実は、最上ノード戦略では近隣ノード戦略を補助的に使うべきであることを示唆している。

さらに、最上ノード戦略が[実験1]ではサブネットに対してのみ利用されたことと[実験2, 3]ではまったく利用されなかったことから、最大ノード戦略は最上ノード戦略より優位にあり、最上ノード戦略は最大ノード戦略が使えない部分に限り有効な場合があると結論できる。

最大ノード戦略の有効性

すべての適用実験をとおして、最大ノード戦略は非常に有効であることが分かった。すなわち、分析の初期段階において機能単位を作成する際に最も有効な手がかりを与え、このように得られた機能単位は分析の中間段階において他の小さい解決策ノードを取り込む機能単位となり、複数の機能単位をまとめる必要性がある場合でも最終的な機能単位として残ることが多い。

機能単位グラフの有効性(1)

本項と次項では、4.1節では目的として設定していなかったが、適用実験をとおして確認された機能単位グラフの有効性について報告する。

[実験3]により、多数の小さい解決策ノードが機能単位とならずに残っているときに、それらをいずれかの既存の機能単位に入れるか、新たに機能単位にするかを判断するために、機能単位グラフを用いて機能単位を一覧することが有効であることが分かった。

機能単位グラフの有効性(2)

前項までは解決策ネットを利用した機能単位の作成に関する有効性について議論してきたが、本節では機能単位分析の分析結果である機能単位グラフの有効性について述べる。分析結果の有効性は、分析結果を会議参加者に提示し、主観的に評価してもらった。

機能単位グラフは、作成した機能単位の表示と機能単位間の関係の表示を意図したものである。機能単位の表示については、適用実験により得られた機能単位が妥当なものであると会議参加者の評価が得られたので、有効性が検証されたと考える。

しかしながら、機能単位グラフに表される機能単位間の関係は、個々のアークが何を表しているかを一瞥して理解することが困難であるという問題点が会議参加者から指摘された。この指摘を受けて分析者に確認したところ、分析者からは各アークの意味について納得のゆく説明が得られた。よって、解決策アークを集約した機能単位間のアークには意味があるが、現時点での機能単位グラフはその意味を表現するためには不適切であると結論できる。

6.2 作業手順に関するノウハウ

一連の適用実験をとおして、機能単位分析における細かい作業手順に関するノウハウが獲得できた。この獲得されたノウハウをもとに設計した標準的な機能単位分析の作業手順を以下に示す。

1. 最大ノード戦略を繰り返し用いて、解決策を3個以上含む解決策ノードすべてに(大きい順に)名前をつけて機能単位とする。
2. 解決策を2個以下しか含まない解決策ノードのみで構成される解決策ネットの独立したサブネットについて、最上ノード戦略を基本にして、近隣ノード戦略を組み合わせることで、上位の数個の解決ノードを集めて機能単位とする。
3. いずれの機能単位にも属していない残りの解決策ノードについて、ひとつずつ変形近隣ノード戦略を試す。
   - 妥当な機能単位がみつかった場合：解決策ノードをその機能単位に取り込む。
   - 妥当な機能単位がみつからなかった場合：機能単位グラフを参照し、妥当な機能単位を大域的に探す。
     - 妥当な機能単位がみつかった場合：解決策ノードをその機能単位に取り込む。
     - 妥当な機能単位がみつからなかった場合：その解決策ノードを新たな機能単位にする。

また、作成した機能単位を複数まとめて大きな機能単位とした方がよい場合があるので、上記の作業手順全体をとおして、機能単位同士をグループ化してより大きな機能単位を作成する作業を適宜行なう。

適用実験の結果として、上記の手順において、注目点支援の戦略および機能単位グラフ参照によって、妥当な機能単位がみつからず、小さい解決策ノードを新たな機能単位にすることは極めて稀であると予想できる。

6.3 分析方法の改良

本節では、適用実験をとおして得られた分析方法の改良案について報告する。

変形近隣ノード戦略

前述のとおり、近隣ノード戦略より有効な注目点支援として変形近隣ノード戦略が適用実験から獲得できた。

現状の *USP/CASE* は注目点支援として各戦略により注目点を自動的に求め分析者に提示する機能を持っているが、次期バージョンでは変形近隣ノード戦略についても同等の機能を追加すべきであろう。

多段階グループ化

前述のとおり、機能単位分析では、作成した機能単位同士をグループ化してより大きな機能単位を作成する作業を行なう。これは、解決策のグループ化が、「解決策－機能単位」という一段階ではなく、「解決策－小さい機能単位－……－大きな機能単位」という多段階で行なえることを意味している。

最終的に得られる大きな機能単位だけでなく、グループ化の途中段階で現れた小さい機能単位を分析結果として加えることが有効であると考えられる。特に、多くの解決策／解決策ノードを含む機能単位は、機能単位を提示する際に、この小さい機能単位名によって項目立てすることが有効であろう。そこで、機能単位分析における機能単位のグループ化を多段階化することは重要な改良となると考えられる。

現状の *USP/CASE* では解決策の１段階のグループ化しか行なえないが、次期バージョンでは多段階のグループ化および各段階の名前付けの機能、機能単位グラフの階層表示機能などを追加すべきであろう。

機能単位アークの名前付け

前述のように、現時点での機能単位グラフにはアークの意味を表現する方法がないという問題点があることが適用実験をとおして判明した。

USP/CASE の次期バージョンでは機能単位アークに名前が付けられるようにし、さらに、すべての機能単位アークについて名前（アークの意味）をつけることを機能単位分析の一部に組み込むべきであろう。

6.4 会議の性質と機能単位分析

本節では、4章で述べた適用実験の目的以外について考察する。ここまでは機能単位分析の支援についてPSGから自動合成される解決策ネットの性質と関連して考察してきたが、ここではより直接的に会議の性質との関係について述べる。

前述のように最大ノード戦略は非常に有効な戦略であるが、会議中に同じ問題点について多くの解決策の対案が発言されたときやある解決策を実現するための部分的解決策が深く議論されたときに大きな解決策ノードは生成される。一方、最大ノード戦略に基づく機能単位分析は、より多くを話した話題(解決策)から機能単位としてまとめる傾向にある。

独立した様々な話題についてあまり脈絡なく議論した(「散慢な」)会議では、解決策ネットに小さい独立したサブネットが多く現れる。孤立＆単独ノードはこのようなサブネットの顕著なものとみることができる。非常に散慢な会議においては、様々な戦略や機能単位グラフの参照によっても解決策を入れる機能単位がみつからず小さい解決策ノードを新たな機能単位にしなければならないことが多く、非常に機能単位分析がやりにくい。

また、機能単位分析は、当初、ソフトウェア要求会議で発言される解決策をもとにソフトウェアの機能単位を抽出することを目的としていたが、一連の適用実験により、要求会議よりも一般的な「問題解決を目的とした会議」に適用して会議内容の整理に役立つことが実証できた。

7 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の実際の会議への適用実験について報告した。適用実験をとおして、基本的には、機能単位分析の有用性、機能単位分析における解決策ネットを利用した支援の有用性を検証でき、機能単位分析の作業手順に関するノウハウおよびそれをもとに構築した標準的な機能単位分析の作業手順を得ることができたと考えられる。しかしながら、いくつかの問題点も明らかになった。今後、実用化に向けて、これらの問題点を克服すべく分析方法やツールの改良を行なってゆく予定である。

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。また、適用実験の場を与えていただいた富士通社内のかたがたにも感謝いたします。

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] "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", 第110回ソフトウェア工学研究会, 1996.