Vol.2012-SE-178 No.22
2012/11/2

ソフトウェア開発におけるトレーサビリティの複雑さの表現と単純化方法の提案

北村充晴[†1]　高木正則[†1]　山田敬三[†1]　佐々木淳[†1]

ソフトウェアへの要求とソフトウェア開発工程で作成される成果物との適合性を検証する上で、トレーサビリティは必要不可欠である．したがって，大量複雑なトレーサビリティを検証しやすい形に最適化することは重要である．本稿では，トレーサビリティの複雑さの測定および単純化する手法について提案する．さらに，実際の開発事例に対する実験結果を用いて，本提案による手法の有効性を示す．

A proposal on Representation and Simplification of Requirements Traceability links in a Software Development Project

MITSUHARU KITAMURA[†1]　MASANORI TAKAGI[†1]
KEIZO YAMADA[†1]　ZYUN SASAKI[†1]

Requirements traceability is requisite to verify the compatibility between software requirements and various work-products in development phases. For that purpose, it is important to optimize the verification so that complex traceability links can be simplified. In this paper, we propose the method that measures and simplifies the complexity of traceability. We also introduce effectiveness of the method based on the experimental data obtained in the real software development project.

1. はじめに

ソフトウェア開発において作成される各種成果物の関連性（以下，トレーサビリティ）を把握することは，要求管理，品質の検証，システムの全体理解を効率的に実施するうえで重要である．SLCP(Software Life Cycle Processes) においても，アクティビティ単位にトレーサビリティによる検証ポイントが示されている[1]．ソフトウェア開発における要件の抜けモレ等を発生させないためには，まず要件とのトレーサビリティを網羅的に特定し，それらの適合性を確実に検証する必要がある．そのための代表的手法の例として QFD(Quality Function Deployment)が挙げられる．要求から実装，製造プロセス，運用までの全体工程のトレーサビリティを可視化するマトリクス型手法であり，古くから製品開発分野を中心に広く利用されている[2]．この手法をソフトウェア開発に適用した多くの事例が発表されている[3][4]．さらに，要求との関連性を可視化することによって，要求対応モレ低減効果があるという実証研究成果や[5]，トレーサビリティから，エラーの少ない開発プロセスを導く研究も発表されている[6]．このように，品質向上のためには，従来のプロセス（作業内容）に着眼するだけでなく，トレーサビリティに着眼することが重要であることが認識されている．

その一方で，トレーサビリティの把握方法が不適切であるために，労力の割に思うような効果が得られない組織が多いという報告例もある[7]．例えば，CMMI レベル 3 の訓練された組織においても，マトリクス型で詳細に取得した要求事項と下位工程成果物との関連情報を有効利用できなかったという報告もされている[8]．主な原因として，ツール環境の未整備等もあげられるが，目的にあっていない成果物間関連情報の粒度や複雑さ，取得情報のモデルやガイダンスが組織内でバラバラであること，計画的でないことなどが挙げられている[9][10]．

トレーサビリティに関して，ツール化による可視化，抽出や自動取得の議論は進んでいるが[11]，大量，複雑なトレーサビリティそのものの改善手段が未だ未整備な状況にある．したがって，トレーサビリティを標準的な手法で可視化し，複雑な成果物の関連性を単純化するための手法が必要である．

本論文では，ソフトウェア開発における要求と各種成果物間の適合性検証を効率的に進めることを目的に，トレーサビリティの複雑さを指標化し，その関連を単純化する手法について提案する．また，一つのソフトウェア開発プロジェクト事例に適用してシミュレーションを行った結果，本手法を適用することによって単純化可能であることを示した．

2. 研究対象とするトレーサビリティの定義

トレーサビリティで扱う成果物間関連情報は，様々な観点に応じて多種多様な情報が存在する．その分類も標準的に確立された定義は存在しない[12]．本研究で対象とするトレーサビリティの種類について，他の関連研究成果を参

†1 岩手県立大学
Iwate Prefectural University

考にしながら定義する．

Pfleeger と Bohner による分類では，図 1 に示すように水平的トレーサビリティ関連と垂直的トレーサビリティ関連に分けられる．水平的トレーサビリティ関連とは，工程間の成果物間関連を意味しており，垂直的トレーサビリティ関連とは，工程内における成果物の部分的要素間の相互依存関係を意味している[13]．上位工程の成果物の要求を下位工程の成果物が満たしているか否かの検証が，いわゆる源流管理の要となる作業である。したがって，本研究では，この水平的トレーサビリティに焦点をあてて議論を進める．

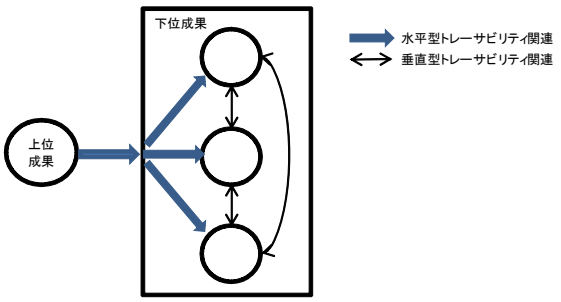

図 1　水平型トレーサビリティと垂直型トレーサビリティ

より詳細には以下の関係をもつ成果物間関連を対象範囲とする．

- 要求事項（機能要件や非機能要件含む）とそれを具現化するための機能および設計コンポーネントとの関係
- 機能と設計コンポーネントとの対応およびシステム，サブシステムといった設計コンポーネント間の階層構造関係
- それらの成果とテスト関連成果物との関係
- 上位規程や標準とそれに準拠する上記成果物との関係

以下では，これらの関係を，上位成果の要件を継承するという意味で，“継承関係”と定義して呼称する．

トレーサビリティの複雑さを示す方法として，サイクロマティック数が Pfleeger と Bohner より提案されている[13]．しかし，サイクロマティック数はもともと構造化プログラムを分析対象としており[14]，この工程間の継承関係とは前提としている要件が適合しない．

このため，本研究では，継承関係の複雑さを表す方法として，図 2 に示す基本モデル（2 部グラフ）から考えることとした．以降では，この継承関係の構造特性および指標化の方法について論述し，関係の単純化の方法について，事例を適用しながら考察する．

3. 継承関係モデル

3.1 継承関係のモデル化

3.1.1 2 部グラフ表現

数字で表した頂点（ノード）が各成果物であり，上位部集合が上位工程内成果物集合を示し，下位部集合が後続工程内成果物集合を示している．辺（リンク）は成果物間の継承関係を示している．

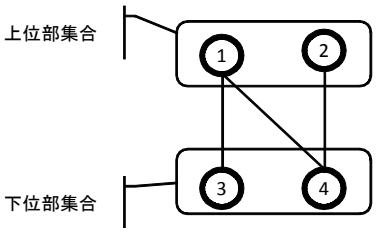

図 2　継承関係の 2 部グラフ

対象となる検証作業領域内において，このような連結グラフにある継承関係の単位を“継承関係検証単位”と定義する．以降議論を進める上で，グラフ表現の記号を以下に定義する．

U(V,E) :継承関係検証単位
V = {v | v1,v2,v3…} :U を構成する頂点の集合
E = {e | e1,e2,e3…} : U を構成する辺の集合
F ⊂ V :上位部集合
B ⊂ V: 下位部集合
E(v) :頂点 v に隣接する辺の集合
N(v) :頂点 v に隣接する頂点の集合
d(v) = |E(v)| :頂点 v に隣接する辺の数（次数）

前述したように，相互依存関係のトレーサビリティは対象としないため，部集合内の辺（リンク）は発生しない．2 部グラフとなる．

$$(v \in F \rightarrow N(v) \in B) \vee (v \in B \rightarrow N(v) \in F) \qquad \text{数式 1}$$

3.1.2 継承関係検証原単位

継承関係検証単位（U）を構成する検証の最小単位を“継承関係原単位”（以下，単に“原単位”と呼ぶ）として以下のように定義する．3 つのパターンに分けられる（図 3）．

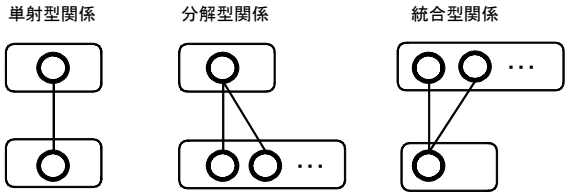

図 3　継承関係検証原単位（原単位）

- 単射型関係:
  $E(v) \quad (|F| = 1 \wedge |B| = 1 \quad)$ 数式 2
  上位工程の要件と下位工程の仕様の適合性を 1：1 に検証する．この時，d(v)=1 である．
- 分解型関係:
  $E(v) \quad (v \in F, d(v) > 1)$ 数式 3
  要件に，複数の実現仕様が矛盾や冗長なく連携して適合しているか検証する．

- 統合型関係:
  E(v)　(v∈B, d(v)>1)　　数式 4
  複数の要件を網羅し，冗長なく共通的な実現仕様であるか検証する．

分解型関係と統合型関係は，d(v)>1 であるため，単射型関係に比べて複数要素固有の検証特性が必要になる．また，分解型関係と統合型関係とは，複数要素が上位部集合に属すか下位部集合に属すかで検証特性も異なってくる．検証作業は，基本的にこの 3 つの作業特性の複合形態である．この時の辺 e∈E(v)を“原単位リンク”と定義する．

継承関係検証単位は，数式 2,3,4 の定義より，一つ以上の原単位で構成されることが分かる．例えば，図 2 を一つの連結グラフとした場合，図 4 に示すように分解型と統合型の原単位を識別できる．（厳密には辺の集合．）

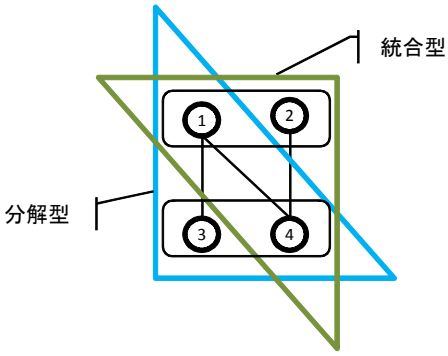

図 4　継承関係検証単位と原単位の構成

3.1.3 原単位複数構成時の継承関係検証単位の特性

数式 2～4 の原単位の特性より，同種の原単位のみで複数構成は成立しない．また，分解型と統合型の間でのみ常にリンクを共有する．以下が成立する．

∀v[(d(v) > 1, v∈V) → ∃s (d(s) > 1, s∈N(v))]　数式 5

この辺 e｛v,s｝を“原単位連結リンク”と定義する．例えば，図 4 における①－④のリンクに相当する．

3.1.4 継承関係検証領域

上述したグラフ理論的な分割の前に，継承関係作業領域をできる限り独立的に捉えられる観点で分離する．その分離した単位を“継承関係検証領域”と定義する．ここでいう観点とは，ISO/IEC/IEEE42010[15]で定義される Architecture Viewpoints に相当する．「システムに対するある特定の関心を形式化（分離）する観点であり，全体システム仕様（構造）定義上の上位概念」となっている．Nuseibeh, Kramer, Finkelstein は，ViewPoints とはシステムに関する（設計開発）対象を関連性の低い単位に分離し、独立した管理を可能にするものとしている[16]．ある観点で関心を分離し，他の観点とはできるだけ独立的に，よりシンプルに検討を進めるために実務上必要な概念である．プロジェクト立上時に、工程や方法論選定と同様に検討すべき事項である．

以上の定義により，与えられた検証作業範囲は，図 5 に示す階層として捉えることができる．

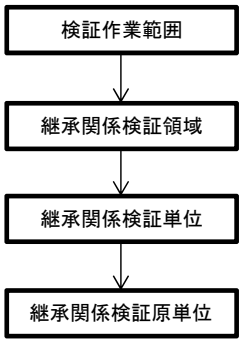

図 5　継承関係の検証単位

3.2 複雑さの定義

3.2.1 継承関係検証単位の複雑さ

原単位リンク数は，原単位が持つ継承関係の継承関係検証単位内総量であり，原単位連結リンク数は原単位間の関連（リンクが共有されること）の継承関係検証単位内総量である．原単位はある程度独立的に検証できる単位であるが構成する原単位リンク数が増えれば原単位の複雑さは増す．また，原単位連結リンクは，数式 5 の定義より，一つのリンク上で分解型の要素，統合型の要素，双方の検証をみたす必要があり，単一要素のリンクに比べ検証の難易度が高くなる．また，原単位間の関連する量が増えれば，検証の際に考慮する関係範囲が増大し，複雑さが増すと考えられる．

(1) 原単位リンク数

継承関係検証単位内に構成される原単位リンクの総数(Cp)であり，数式2～4の定義より，|F| = 1 Λ |B| = 1 の場合，

Cp = 1　　数式 6

それ以外の場合は，

Cp = Σd(v)　(v∈V, d(v)>1)　　数式 7

(2) 原単位連結リンク数

原単位連結リンク数（Cc）は，数式 5 の定義より，1 つのリンク（物理リンク）に原単位リンクが 2 つ重なるので，以下の式で算出できる．

Cc = Cp – |E|　　数式 8

今後，|E| のことを原単位リンク数と混同しないよう，敢えて“物理リンク数”と称する．

継承関係検証単位全体の複雑さは，Cp と Cc で決まる．図 6 に示すように，Ⅱゾーンに位置づけられる関係は他に比べて原単位のリンクおよび原単位間の関連も双方ともに多く複雑である．Ⅰゾーンは，原単位のリンクが比較的に多く複雑であるが，原単位間の関連は少ない．対照的にⅣ

ゾーンは，原単位の複雑さは小さいが，原単位間関連の複雑さは高いといえる．それぞれの特性を考慮しながらⅢゾーンに向けて単純化を行う必要がある．

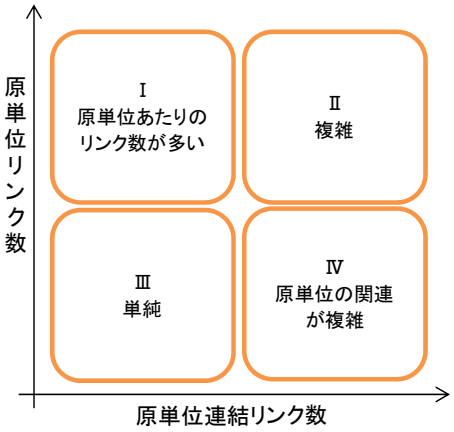

図 6　複雑さの定義

3.2.2 継承関係検証領域の総合的な複雑さ

(1)　平均原単位リンク数

継承関係検証領域内の 1 原単位あたりの原単位リンク数を表す．領域全体について，縦軸方向の増減効果を示す．

原単位リンク総数/原単位総数　　数式 9

(2)　平均原単位連結リンク数

継承関係検証領域内の 1 検証単位あたりの原単位連結リンク数を表す．領域全体について，横軸方向の増減効果を示す．

原単位連結リンク総数/継承関係検証単位総数　数式 10

4.　単純化の方法

単純化は，基本的に原単位リンクの削減（図 6 の Y 軸の値を下げる）や原単位連結リンクの削減（X 軸の値を下げる）によって達成できる．特に原単位連結リンク削除の場合は，リンク削除効果だけでなく，さらに継承関係検証単位から原単位を分離できる可能性があるため，その効果は比較的に大きい．まずは，原単位連結リンク削除から着眼したほうが効率的である．また，検証作業範囲でみた際に，原単位リンク総数は増加するが原単位連結リンク総数が低減され，継承関係検証単位では単純化する方法もある．

これらの単純化作業は，設計に変更を伴う方法であるため，設計段階において精緻化作業の一環として実施すべきである．

(1)　ノード間の直交性を高める

各部内におけるノード間の包含，重複，依存関係を回避することによって，余分なリンクを削減する．

例えば，図 7 左に示すような要求と実現機能の関係だった場合に，①の要求事項が②の要求事項を包含する，もしくは①と②が目的と手段の関係にあった場合，④は①の手段として冗長な関連を持つ．①と②の要求事項の直交性を保てる粒度や範囲を調整することによって，①－④のリンクは削除できる．それによって，継承関係検証単位 A は，単純化された B と C に分割される．

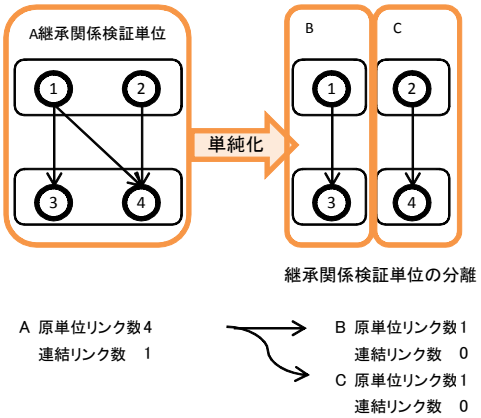

図 7　要求事項改善による単純化の例

(2)　ノードを抽象化する

それぞれのノード，リンクを抽象化し，段階的に細分化する．

(2.1)図 8 に示すように，ある程度集約した中間的な単位に分割してから詳細化するケースである．これは，全体をある程度まとまった単位で全体を概観する場合に有効である．一つの継承関係検証単位における原単位リンク数が少なくなるため，検証が容易になる．ただし，次の工程まで含めた全体の原単位リンク数は増加する．

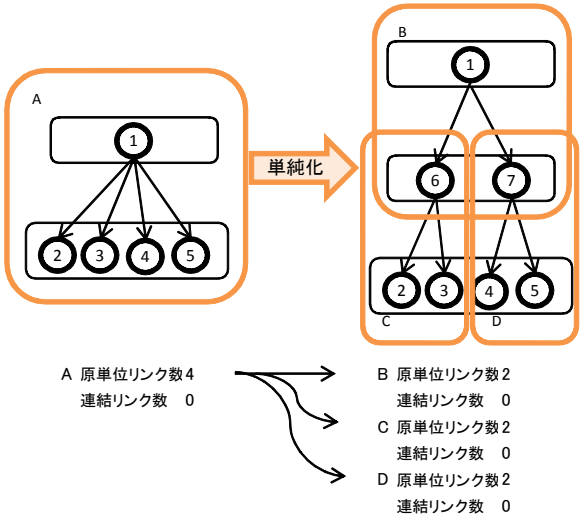

図 8　抽象化による単純化の例(2.1)

(2.2)図 9 で示すパターンは，バラバラの要件それぞれに機能を対応させるのではなく，要件の本質，パターンを理解し，それを実現機能に展開する場合である．問題→原因→解決策といった思考パターン等の成果物対応ケースに合致する．また，創造的思考プロセ

スの基本パターンは，特定問題→汎用問題→汎用解決策→特定解決策であるという意見にも合致する[17].

このパターンは，詳細から詳細に対応させた時にできる原単位連結リンクを削除する方法である．

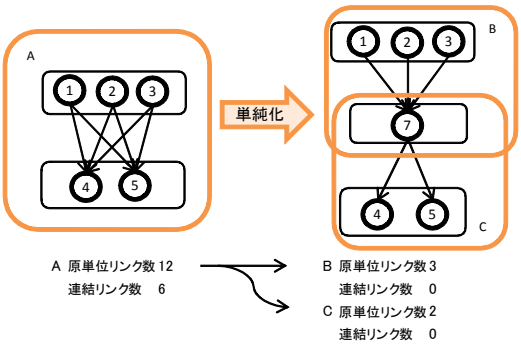

図 9 抽象化による単純化の例(2.2)

(3) ノードを分離する

ある独立した観点（要求）によって下位のノードを一旦分離して，その後に統合する．観点による分離の重要性は前述したが，それに従った改善パターンである．

図 10 に示すように，①と②の要件は，最終的には③，④に継承されるべきであるが，例えば機能要件，非機能要件といったある程度は独立的に検討できる⑤⑥と⑦⑧の要件に分離して検討する．最終的にまとまった内容を③，④で継承する．そうすることによって，原単位連結の複雑さを解消する．

また，実装局面でよくあるケースだが，③と④のコンポーネントで，①，②の複数要件を同時に満たす設計をする場合に，まず一旦，①，②それぞれの要件ごとに，③，④に実装する場合の要件⑤～⑧にブレークダウンする．そのあとに，⑤，⑦を③へ，⑥，⑧の要件を④へ統合する検討を行う，という段階を踏んだ方が設計も検証も分かりやすい．

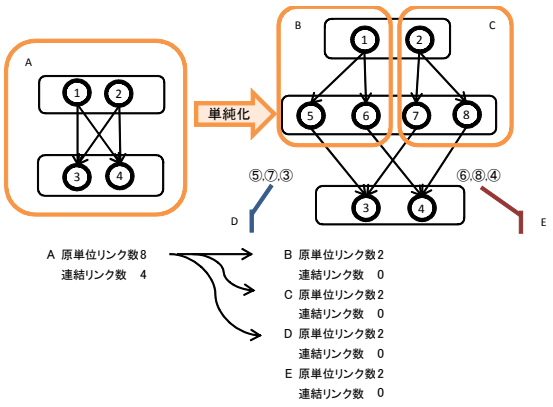

図 10 分離による単純化の例

(4) d(v)>1 のノードを集約化する

d(v)>1 のノードを集約化し，d(v)=1 のノード増やし，原単位を削減する．（数式 3,4 より原単位は d(v)＞1 のため）

図 11 の A 例は，①，②に必要な共通の機能が③，④に内在しており，そのために①，②からの関連があり，B 例は，⑤にその共通機能を切り出すことによって，③，④に対する①，②双方からのリンクを回避できるケースを表している．C 例は，②の独自要求部分を①に吸収してもよい場合である．

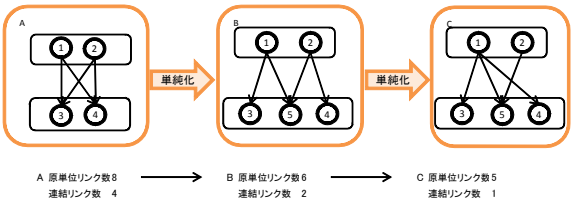

図 11 集約化による単純化の例

5. 実験

ある小売店の本社における店舗からの商品の発注依頼を受けて問屋に発注するシステムのプロジェクトについて，要求事項から機能設計，システム設計部分における成果物に対して本手法を適用し，効果を検証する．

5.1 継承関係の表現方法

継承関係は図 12 に示す通り QFD 手法で表現する．今まで 2 部グラフで表現してきたノードが，行と列になる．たとえば，I ゾーンのマトリクスであれば，列のタイトル「要件」が上位部集合であり，行のタイトル「機能」が下位部集合となり，各配下の要素がノードである．リンクに対応するマトリクス上のセルに“○”を記述する．指標については，継承関係検証単位毎に以下を集計する．

・物理リンク数(|E|) : “○”出現数をカウントする．
・上位部ノードの d(v) : 列の“○”出現数をカウントする．
・下位部ノードの d(v) : 行の“○”出現数をカウントする．

以上より，原単位リンク数（数式 6,7）および原単位連結リンク数（数式 8）は計算できる．

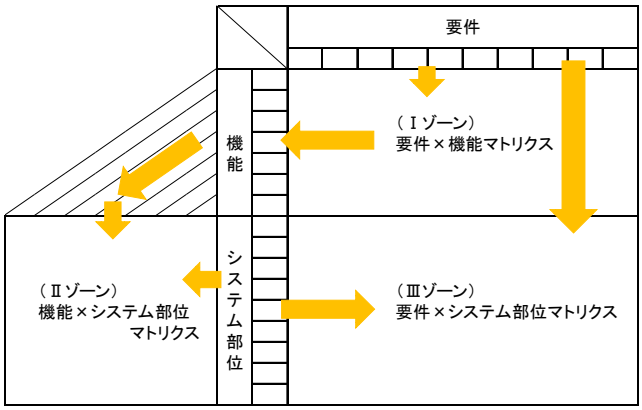

図 12 QFD 手法による検証の流れ

(1) 適用手順

検証作業対象範囲は，要件と機能の継承関係検証領域（Ⅰゾーン），機能とシステム部位（最上位のシステムコンポーネント）の継承関係検証領域（Ⅱゾーン）および要求とシステム部位との継承関係検証領域（Ⅲゾーン）とで構成される．これらの継承関係について単純化を行った．

単純化の手順は，図 13 に示す通りである．今回は，最初のタスクで検証作業範囲を機能要件と非機能要件に分割する．基本的には，前述した継承関係単純化の方法を適用したものである．

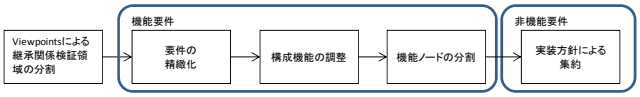

図 13　単純化の適用手順

5.2 適用結果

(1) Viewpoints による継承関係検証領域への分割

今回の開発において，機能要件と非機能要件は比較的に独立して検証し易いと考え，Viewpoints として分割の軸に活用する．結果として図 12 から図 14 のような検証領域に分割される．その結果の複雑さの測定値が表 1 の通りである．ⅠゾーンおよびⅡゾーンが観点による分離によって，単純化されたことを表している（継承関係単純化方法における「ノードを分離する」の事例）．前半の単純化タスクは機能要件領域のためⅠ-1 ゾーン，ⅡゾーンおよびⅢ-1 ゾーンを対象とし，最後のタスクで非機能要件に関わるⅠ-2 ゾーン，ⅡゾーンおよびⅢ-2 ゾーンを対象とする．

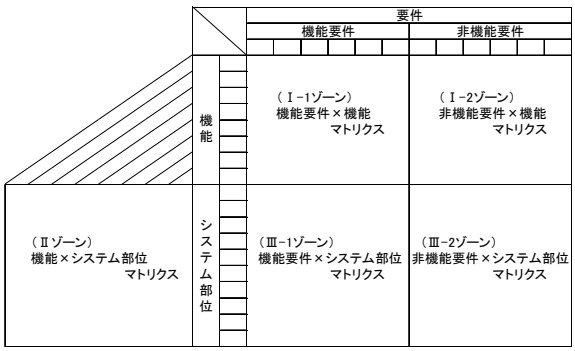

図 14　Viewpoints による継承関係検証領域の分割結果

表 1　Viewpoints 分割前と分割後

検証領域	分割前			分割後		
	検証単位数	原単位リンク数	原単位連結リンク数	検証単位数	原単位リンク数	原単位連結リンク数
Ⅰ-1ゾーン	1	115	53	1	22	9
Ⅰ-2ゾーン				1	92	43
Ⅱゾーン	14	34	2	14	34	2
Ⅲ-1ゾーン	1	222	109	1	57	27
Ⅲ-2ゾーン				1	163	80

Ⅰ-1 ゾーン（継承関係検証領域）の継承関係は，図 15 の通りである．u18 が継承関係継承単位である．

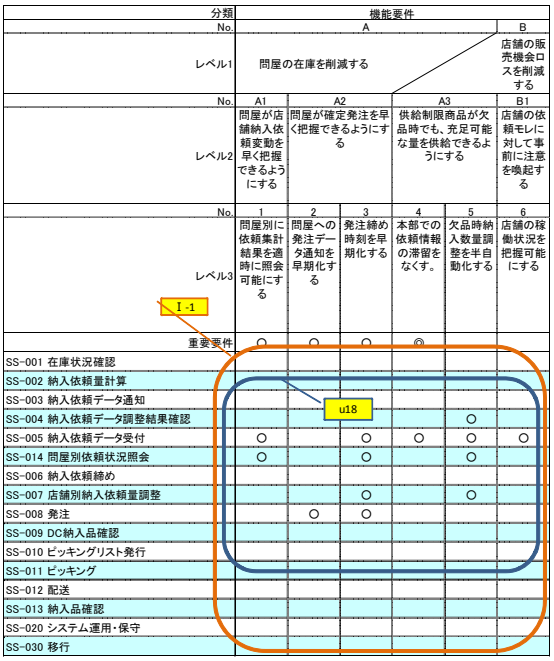

図 15　Viewpoints による分割の結果

(2) 要件の精緻化

継承関係単純化方法における「ノード間の直交性を高める方法」の事例であるが，要件を目的手段展開図に表した結果，発注締め時間の早期化と他の要件に依存関係があり，より直交性のある要件に変更する．図 15(u18)は図 16(u22,u23)のように単純化される．

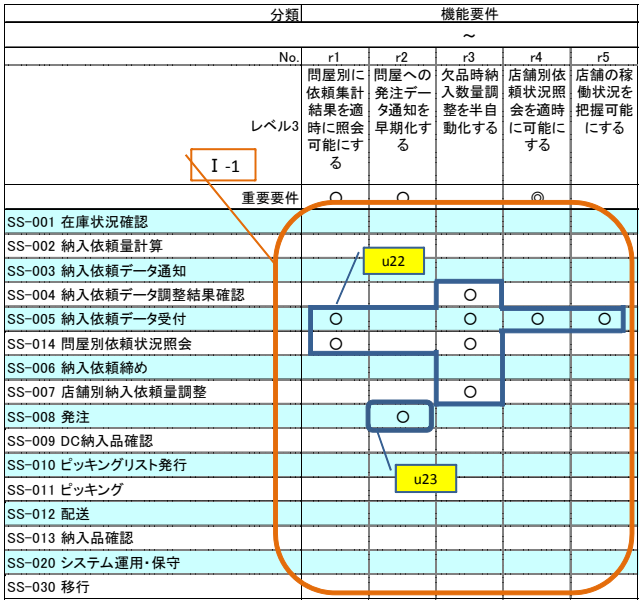

図 16　要件の精緻化結果

(3) 構成機能の調整

継承関係単純化方法における「d(v)>1 のノードを集約化する方法」の事例として，機能に内在するサブ機能を他の機能に構成変更することによって，原単位リンクを削減する方法を説明する．

図 16 の行 ss-005 中に，列 r1 と列 r3 でのみ使用するサブ機能があり，さらに列 r3 はそのサブ機能のみを使用している．そのサブ機能を行 ss-014 に移せば，列 r3 と行 ss-005 のリンクは不要になる．図 16(u22)は図 17(u27)のように単純化される．d(v)=1 のノード増加までは至っていない事例だが，検証の簡素化にもつながる事例である．

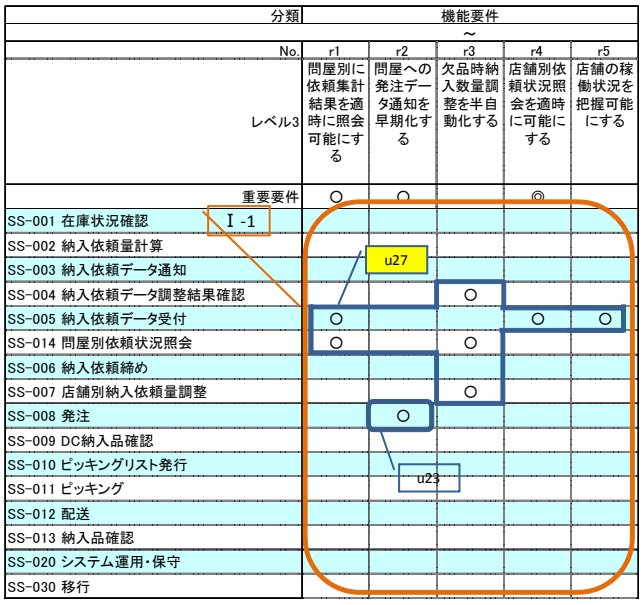

図 17　構成機能の調整結果

(4) 機能の分離

継承関係単純化方法における「ノードを分離する方法」の事例として，機能を分離する方法を説明する．図 17 の行 SS-014 問屋別依頼状況照会であるが，列 r1（通常運用時）に対応した場合と列 r3（欠品発生時）で対応した場合では，同様の処理にみえるが，対象データの条件，業務上の見るポイントなどは異なってくる．要件を精緻に把握するためにも，通常運用時の問屋別状況照会機能と欠品発生時の調整用問屋別状況照会機能（行 SS-014’）を分けて仕様検討することにすることで，図 18 に示すように，継承関係検証単位も分割され(u31,u32)，検証も簡素化する．

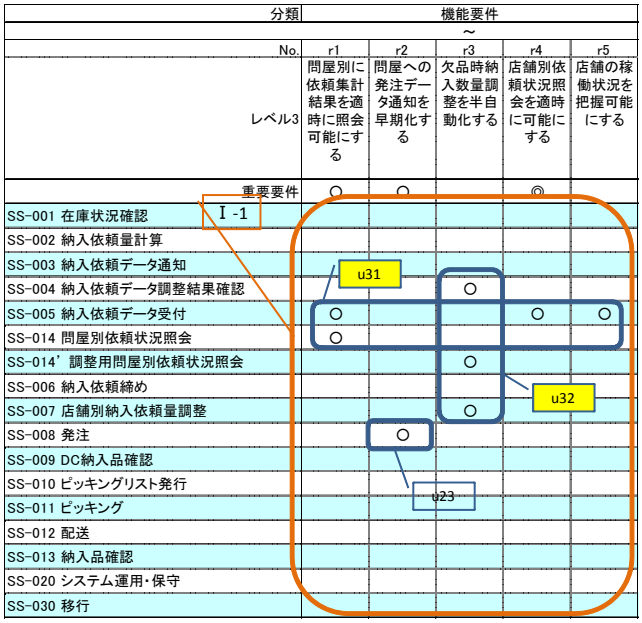

図 18　機能の分離結果

(5) 実装方針による集約化

継承関係単純化方法における「ノードを抽象化する方法」の事例として，システム全体に関連する信頼性や運用要件に関わる要件については，実装方針として集約するノード（機能）を追加し，それぞれの機能全体へのリンクは削除する．またシステム部位にソフトウェア方針，基盤構成といった，これもコンポーネント全体に関わる方式としてノードを追加することによって，全体に関わる非機能要件，実装方式について対応するコンポーネントを集約化する．

事例は割愛するが，非機能要件についても，機能要件同様に要件の精緻化～機能ノードの分割等のタスクを実施すべきである．

6. 考察

以上述べてきた改善手順によって、Ⅰ-1 のゾーンについて継承関係検証単位が単純化される推移を図 19 に示す．図 19 のように，本稿で示した単純化施策を実施することにより，原単位リンク数および原単位連結リンク数ともに数値が下がり，複雑さが低減することが確認できた．

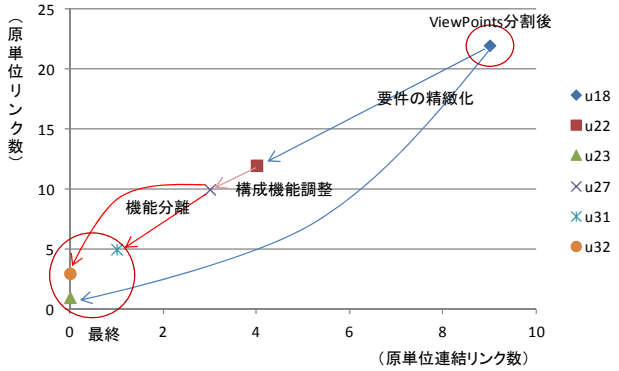

図 19　継承関係検証単位の複雑さ推移（Ⅰ-1 ゾーン）

継承関係検証領域（Ⅰ-1 ゾーン）の総合的な複雑さの推移を図 20 に示す．図 20 より，平均原単位リンク数は平均原単位連結リンク数に比べて相対的に下げ幅は大きくないことがわかる．これは，今回の事例では，Ⅰ-1 ゾーンへの各施策が原単位関連の切り離し効果が大きいことを表している．

表 2 は，今回の実験における各施策における複雑さの総括表である．表 2 に示されたように，単純化施策前の原単位連結リンク数／物理リンク数の比率がかなり高く，ほとんどのリンクが原単位連結リンクであったことがわかる．また，実装方針による集約では非機能系(Ⅰ-2，Ⅲ-2)の領域で改善効果があったが，一方でⅡゾーンについては，個別機能と方針との継承関係の複雑さを増加させる結果となった．

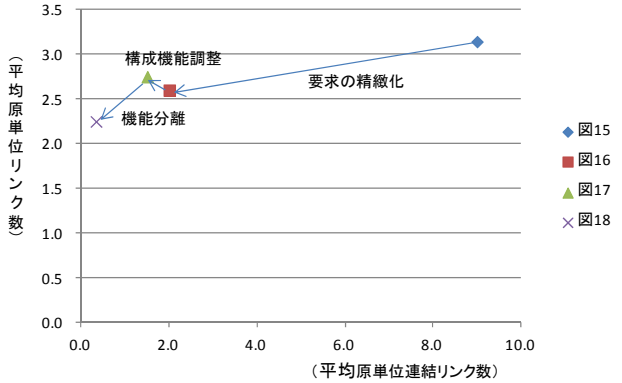

図 20　Ⅰ-1 の総合的複雑さ推移

表 2 複雑さの総括

改善策	検証領域	検証単位数	原単位数	物理リンク数	原単位リンク数	原単位連結リンク数	連結リンク/物理リンク	平均原単位リンク	平均原単位連結リンク
初期	Ⅰゾーン	1	25	62	115	53	0.85	4.6	53.0
	Ⅱゾーン	14	16	32	34	2	0.06	2.1	0.1
	Ⅲゾーン	1	43	113	222	109	0.96	5.2	109.0
Viewpoints	Ⅰ-1ゾーン	1	7	13	22	9	0.69	3.1	9.0
	Ⅰ-2ゾーン	1	22	49	92	43	0.88	4.2	43.0
	Ⅱゾーン	14	16	32	34	2	0.06	2.1	0.1
	Ⅲ-1ゾーン	1	17	30	57	27	0.90	3.4	27.0
	Ⅲ-2ゾーン	1	39	83	163	80	0.96	4.2	80.0
要件精緻化	Ⅰ-1ゾーン	2	5	9	13	4	0.44	2.6	2.0
	Ⅲ-1ゾーン	3	9	18	26	8	0.44	2.9	2.7
構成機能調整	Ⅰ-1ゾーン	2	4	8	11	3	0.38	2.8	1.5
	Ⅱゾーン	15	15	31	31	0	0.00	2.1	0.0
ノード分割	Ⅰ-1ゾーン	3	4	8	9	1	0.13	2.3	0.3
	Ⅱゾーン	16	16	33	33	0	0.00	2.1	0.0
	Ⅲ-1ゾーン	4	7	18	22	4	0.22	3.1	1.0
実装方式	Ⅰ-2ゾーン	1	21	37	73	36	0.97	3.5	36.0
	Ⅱゾーン	15	17	44	47	3	0.07	2.8	0.2
	Ⅲ-2ゾーン	1	15	29	52	23	0.79	3.5	23.0

7. まとめ

本稿ではソフトウェア開発におけるトレーサビリティの複雑さの表現方法を提案し，単純化する方法についての理論を示した．さらに，実際の開発事例を用いて，本提案によるトレーサビリティ単純化の施策を示し，実際に単純化されることを確認した．その結果，本提案は検証作業の段階的な改善において実用的な方法であるといえる．

本稿では，一つの開発事例に対する改善を示したが，今後は他の事例についても同様の実験を試み，開発事例の特性と改善方法の効果についても検討する予定である．

参考文献

1) 情報処理推進機構 SEC 編: 共通フレーム 2007 第 2 版, Ohmsha(2007)
2) 赤尾洋二,水野滋: 品質機能展開, 日科技連(1990)
3) 大森晃: ソフトウェア品質管理への品質展開アプローチ, 情報処理学会論文誌, Vol.31, No.10, pp1474-1485(1990)
4) 小松由香里,吉原信也,石橋慶一,秋山義博, 他:組込みソフトウェア開発上流工程における要求トレーサビリティ・モデルを用いた非正常系分析支援,プロジェクトマネージメント学会 2006 年度春季研究発表大会予稿集 ,pp.119-124(2006)
5) 大森晃,辻田祐純: ソフトウェア「要求機能実現管理表」による機能欠落低減効果の統計的検証, 品質, Vol.36, No.4, pp.126-135(2006)
6) 中條武志: 情報の流れに着眼した設計開発プロセスの標準化, 品質, Vol.35, No.2, pp.142-149(2005)
7) Jane Cleland-Huang: Traceability Research: Taking the Next Steps, TEFSE'11, pp1-2 (2011)
8) Jane Cleland-Huang, Jane Huffman Hayes, J.M.Domel: Model-Based Traceability, TEFSE'09, pp6-10(2009)
9) Balasubramaniam Ramesh: Toward Reference Models for Requirements Traceability, IEEE Transaction on Software Engineering, Vol.27, No.1, pp58-93(2001)
10) Jane Cleland-Huang: Toward Improved Traceability of Non-Functional Requirements, TEFSE 2005, pp14-19(2005)
11) 小谷正行,落水浩一郎: UML 記述の変更波及解析に利用可能な依存関係の自動抽出, 情報処理学会論文誌,Vol.49 No.7, pp2265-2291(2008)
12) Andrian Marcus,Xinrong Xie,Denys Poshyvanyk: When and How to Visualize Traceability Links?, TEFSE 2005, pp.56-61(2005)
13) Shari Lawrence Pfleeger and Shawn A.Bohner: A Framework for Software Maintenance Metrics, in Proceedings of Conference on Software Maintenance,pp.320-327(1990)
14) Thomas J.McCABE: A Complexity Measure, IEEE Transaction on Software Engineering, Vol.SE-2, No.4, pp.308-320(1976)
15) International Standard ISO/IEC/42010:2011(E) System and software engineering-Architecture description, First edition 2011-12-01, (2011)
16) B. Nuseibeh,J.Kramer and A.Finkelstein: A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification, IEEE Transaction on Software Engineering, Vol.20, No.10, pp.760-773(1994)
17) 長谷川浩志,園田有希,塚本美喜,佐藤勇介: 創造的工学設計支援システムの構築, 日本機械学会論文集(C 編),75 巻,759 号 (2009-11), pp.244-253 (2009)