アーキテクチャ横断的要素に着目したトレーサビリティ確保によるアプリケーションライフサイクル高信頼性維持のためのアプローチ

羽部　高志[†]

クラウド時代のサービスアプリケーション開発においては，スモール・スタートによって迅速にサービスを開始し，頻繁に派生開発を継続して機能拡張していくことは重要なプラクティスとなっている．しかしながら，時間やコスト的な制約により，長期に渡るサービスライフサイクルにおいて，派生開発の高い信頼性を継続して維持することは非常に困難である．本論文は，従来からのソフトウェア機能に対するトレーサビリティに加え，アーキテクチャ実装時に生じる横断的要素に着目して，これら横断的関心事へのトレーサビリティをも確保することによって，インパクト分析の信頼性を高める手法について提案する．本手法が派生開発の生産性向上に貢献できれば幸いである．

An approach towards reliability through the application lifecycle by maintaining traceability of concerns that crosscut the architecture.

Takashi Habe[†]

In the development of cloud computing services, starting the service speedy from a small scale then expanding functionality of the service via derivational development is an important practice. It is very hard, however, to maintain derivational development with high reliability during the service lifecycle, under the constraint of a short period of time or a limited amount of cost.
This paper introduces the method to increase the reliability of impact analysis by tracing both traditional functional requirements and non-functional requirements that crosscut the architecture. This method should be improving productivity of the derivational development.

1. はじめに

近年のクラウドサービスの提供におけるビジネスモデルとして，出来る限り早くコアとなる機能のリリースを実施して，サービス利用者のニーズをリサーチしながら市場の求める機能拡張を実施していくスモール・スタートというアプローチが提唱されている．これは成功への確信をもつことが困難な新規サービスへの初期投資コストリスクと，コンペティター登場による機会損失を避け

【脚注】
† キヤノンソフトウェア株式会社 エンジニアリング事業本部 技術士（情報工学部門）
Canon Software INC. Professional Engineer (Information Engineering)

つつ素早く市場優位性を築くことを目的としており，特に大きな資本力を持たないものの，アイデアの新規性に富むベンチャー企業にとっては，親和性の高いビジネスモデルであると言えよう．

そして，当ビジネスモデルを成功させるための重要な成功要因である，短い周期での頻繁なサービス拡張を実現するためには，いわゆる派生開発の生産性向上に努めなければならない．しかし，初期のサービス立ち上げに引き続いて継続的に実施される派生開発の実行に当たっては，機能追加・修正に伴う様々なリスクが存在する．

[例 1]. 機能変更実施に当たり，プロジェクト開始時点では想定していなかったアーキテクチャに関わる変更が必要になることが実装フェーズに入ってから判明．大幅な修正と，影響を確認するための大規模なリグレッションテストが必要となってしまった．

⇒ 当初の見積を大幅に上回るコストと期間が必要になってしまう．

[例 2]. 機能追加を実施した後，検証フェーズになって初めて思わぬ性能劣化や，セキュリティ上の問題が発覚．アーキテクチャの修正のため，スケジュールの見直しが必要となる．

⇒ サービスリリース計画に致命的なダメージを与え，著しい期間損失を被ることになってしまう．

このような事態を避けるため，一般に派生開発の計画策定においては，予め追加・削除・修正される機能が，既存のシステムにおいてどのような影響を及ぼす可能性があるのかを検証するインパクト分析の実施が推奨されている．

そして，このインパクト分析を十分に機能させるためには，トレーサビリティの確保が必要不可欠であるとされている．特に障害の発生が直接人命に関わる可能性が高い医療機器用ソフトウェア開発および保守のプロセスに関する安全規格である JIS2304(IEC62304) や，自動車の電気・電子に関する機能安全規格である ISO26262 においては，要求からのトレーサビリティ確保を実現することが明確に開発プロセスに対して求められている．

表 1　要求 ID とユースケース ID 間の関係を表現するトレーサビリティマトリクスの例 .

	REQ0110	REQ0121	REQ0110	REQ0110	Etc.…
UC001		✓			
UC002			✓	✓	
UC101		✓			
UC102	✓				
UC103			✓		
UC200					
Etc.…					

具体的なトレーサビリティ確保の手法としては，表 1 に例示するトレーサビリティ・マトリクス（TM）がよく知られており，また市販のトレーサビリティツールを利用してトレーサビリティの確保を図ることも，前述の医療や自動車等の高信頼性を要求されるソフトウェア開発においては，普及が進んでいる．

しかし，開発時には十分検討された非機能要求の実装については，局所化され易い機能要求の実装とは異なり，システムアーキテクチャや，ソフトウェアアーキテクチャが複数箇所に散りばめられて実装されるケースが少なくない．このため規定に則り，正しくアーキテクチャを記載した基本設計書や，実装に繋がる詳細設計書等のドキュメントを残していたとしても，それだけでは派生開発計画時にアーキテクチャに関するリスクの検出や，問題があった場合の代替案の検討を迅速に進めることは容易ではない．

本論文においては，一般にアーキテクチャの検証手法として知られる ATAM(Architecture Tradeoff Analysis Method)[Kazman 2000] を用いて，そのアウトプットである，アーキテクチャ検証のために作成した品質特性シナリオと，実装する設計ドキュメント，プログラムソース等へのトレーサビリティを確保することにより，従来の方法だけでは可視化することが困難であった非機能要求を実装するシステム横断的な関心事に対してもトレーサビリティを保証し，精度の高いインパクト分析を迅速に可能にするための一つの手法を提示する．

2.　既存のアプローチと解決すべき課題

2.1. 要求からのトレーサビリティ（V 字モデル /W 字モデル開発におけるトレーサビリティの確保）

一般的なソフトウェア開発のプロセスを V 字あるいは W 字モデルによって表現することは，上流工程と下流工程の関係と，各フェーズ間のアウトプットと対になるその検証プロセスを明確化出来ることから，これまで広く支持されてきた（図 1：トレーサビリティツール

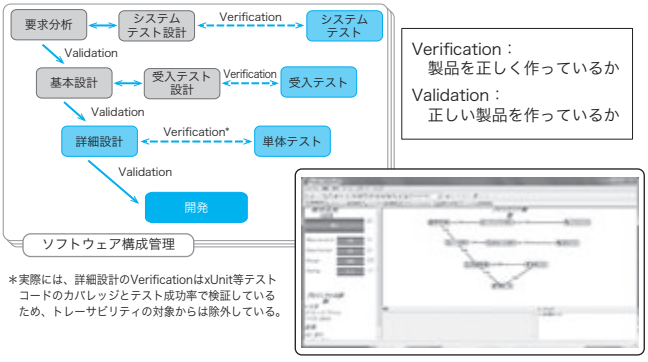

図 1　W 字開発モデルとトレーサビリティ

Reqtify でトレースした W 字モデルの例）．一般的に従来確保されてきたトレーサビリティは，設計した通りに正しく製品を作っているか (verification)，上位ドキュメントで定義した内容に基づき正しい製品を作っているか（validation）というＶ＆Ｖ検証をベースに設定することが多い．ある上流工程のアウトプットが，隣り合う下流工程で全て確実に実装されていることを検証できるようにするため，各ノード間のトレーサビリティを確保する．このノード間のトレーサビリティを積み重ねていくことで，要求からの一貫したトレーシングが可能になる．[IPA2013]

2.2. 従来のアプローチにおける課題

従来のトレーサビリティアプローチは，ウォーターフォール型の開発プロセスをベースに置き，Ｖ＆Ｖ検証を主たる目的に実施されてきた．また，従来型の自然言語をベースとした要求仕様と設計に対して，奥田等による UML(Unified Modeling Language) を用いたモデル駆動要求分析手法によって識別された要求仕様と設計仕様への適用（トレーサビリティ）が提案されている [Okuda2013].

一見，問題がないように見える従来からのトレーサビリティアプローチであるが，以下に述べる重要な課題が存在する．

すなわち，セキュリティや，スループットの確保等のいわゆる非機能要求として定義される分野の設計／実装に当たっては，インフラ基盤を含めたアーキテクチャ全体に渡る横断的な関心事として，その設計，実装がシステムアーキテクチャ，ソフトウェアアーキテクチャと階層を横断して散在するケースが多い．そしてこのようなケースにおいては，階層的に散在する実装が協調して動作することによって初めて，1 件の非機能要求を充足する仕組みになる．

このため，サービスの派生開発を実施する時点においては，機能要求における直列的なトレーサビリティを使用してのインパクト分析は実施可能であるものの，前述したようにアーキテクチャ上に散在する非機能要求へのインパクト分析を実施することが困難となる．

ゆえに，従来からのトレーサビリティアプローチだけでは，1 章の例で上げたような事象の発生を抑制することはできない．上記事象の発生を抑制するためには，

① トレースすべきアーキテクチャ横断的な関心事を洗い出し，トレーサビリティ情報を設計ドキュメントあるいは，検証結果ドキュメントにタグ情報として埋め込むと共に，トレーサビリティ情報を可視化する必要がある．

② 機能要求 / 非機能要求とその実装に変更・追加・削除が加えられる場合，トレーサビリティ情報に基づいて，変更のインパクトを検証する必要がある．

③ 変更に伴いトレーサビリティ情報の変更・追加・削除が発生した場合，トレーサビリティ情報の保守作業を継続的に行う必要がある．

本論文は，主に上記①に述べたような課題，すなわちアーキテクチャ実装における横断要素に対するトレーサビリティの可視化について述べるものであり，今後の派生開発の進展により，②③の課題についても継続して検証する予定である．

3. 課題解決に向けてのアプローチ

3.1. 直列型のトレーサビリティ＋分散型のトレーサビリティアプローチ

我々は，従来からの直列型の機能要求のトレーサビリティに加え，分散型の非機能要求の設計／実装についてもトレーサビリティを確保することで，インパクト分析の信頼性を向上させる必要がある．

この実現に当たって有効なアプローチとして着目したのがアーキテクチャ検証手法の応用である．

アーキテクチャ検証のプロセスが実施される場合，ステークホルダが関心を持つ非機能要求，即ち品質特性についてどのように解決を図っているのか，その妥当性やリスクを検証することになる．

このため，アーキテクチャ検証結果と，それら横断的な関心事を実装する設計書等とのトレーサビリティを確保することができれば，派生開発実施時にリストアップされた変更項目として，当該設計箇所が選択された場合においても，何らかのアーキテクチャに関して分散する関心事の一片がそこに実装されていることを認識することが可能になる．

更には，資産としてカタログ化されているアーキテクチャ検証結果に照らして，修正に伴うリスクやセンシティビティの検証も可能となると考える．

3.2. アーキテクチャ検証の手法（ATAM）

本研究においては，幾つか知られているアーキテクチャ検証手法の中から，カーネギーメロン大学の R.Kazman，M.Klein，P.Clements によって開発された ATAM(Architecture Tradeoff Analysis Method) [Kazman 2000] を採用して検証することとした．

3.3. ATAM プロセスとアウトプット概要

ATAM の実施プロセスの詳細については，[Kazman 2000]，[IPA2005] 等を参照されたい．ここでは簡単な

プロセスの概略と，直接トレーサビリティに関連するアウトプット，実際の適用に当たっての工夫した点について説明する．

【ATAM によるアーキテクチャ評価プロセス】

[ステップ 1]：ATAM の提示
評価リーダーによる ATAM の説明．

[ステップ 2]：ビジネスドライバーの提示
プロジェクトオーナーによるシステムの概要説明．

[ステップ 3]：アーキテクチャの提示
アーキテクトによるアーキテクチャ説明．

[ステップ 4]：アーキテクチャ技法を特定する
採用されているアーキテクチャを明確化した後，主なアーキテクチャをカタログにまとめる．

[ステップ 5]：品質特性のユーティリティツリーを作成する
品質特性の目標を明確化，詳細化，優先順位付けを行うためにユーティリティツリーを作成する．

[ステップ 6]：アーキテクチャ手法を分析する
ステップ 5 で重要と判断されたシナリオについて，それを実現するアーキテクチャ手法を詳しく調査する．

[ステップ 7]：ブレインストーミングとシナリオの優先順位付け
ステップ 6 で詳細調査した重要なシナリオについてブレインストーミングを実施した後，メンバーの投票によってシナリオを優先順位付けする．

[ステップ 8]：アーキテクチャ手法を分析する
ステップ 7 で高い優先順位付けとなったシナリオを実行する．アーキテクトは，そのシナリオを実現するために，関連するアーキテクチャ上の決定がどのような貢献をしているのか説明する．

[ステップ 9]：結果の提示
ATAM によって集められた情報を要約し，利害関係者に対してプレゼンテーションを行い，プロセスを終結する．

3.3.1. ユーティリティツリー

[ステップ 5] で作成されるユーティリティツリーは，ステークホルダが関心を持つ品質特性をツリー構造で表現したモデルである．

ここで，[ステップ 2] で提示されたシステム概要だけをインプットとした場合，ステークホルダの意識に上っていない品質特性を洗い出せない可能性がある．このようなリスクを低減するためには，網羅性の高いテンプレートを使用することが望ましい．2013 年から実施している適用プロジェクトにおいては ISO9126（表 2）お

表 2　ISO/IEC 9126 品質特性

品質特性	説明	品質副次特性
機能性 Functionality	機能とその特性に影響する	合目的性
		正確性
		相互運用性
		機密性
		標準適合性
信頼性 Reliability	定められた条件と期間においてソフトウェアがどの程度機能するかに影響する	成熟性
		障害許容性
		回復性
		標準適合性
使用性 Usability	ソフトウェアを利用するに当たり，使用者にとって必要とされる労力に影響する	理解性
		習得性
		運用性
		標準適合性
効率性 Efficiency	ソフトウェアの性能や，その実現に当たって必要とされるリソース量に影響する	時間効率性
		資源効率性
		標準適合性
保守性 Maintainability	何らかの変更を加えるに当たって必要となる労力に影響する	解析性
		変更性
		安定性
		試験性
		標準適合性
移植性 Portability	別の環境にソフトウェアを移植する可能性に影響する	環境適合性
		設置性
		共存性
		置換性
		標準適合性

表 3　ISO/IEC 9126-4　運用時品質特性

運用時における品質特性
有効性（effectiveness）
生産性（productivity）
安全性（safety）
満足性（satisfaction）

※なお，ISO/IEC9126 は，既に ISO/IEC25000:2005 に置換されているため，今後のプロジェクトにおいては，利用するフレームワークは変換する予定．

よび ISO9126-4（表 3）に定義された品質特性をフレームワークとして，適用プロジェクト非機能要求の洗い出しを行った．

完成したユーティリティツリーにおいて分解された最下層のノードは，当該システムにおける非機能要求と同義と捉え，後述する品質特性シナリオとそこに記載されたアーキテクチャの実装箇所とのトレーサビリティを確保していくこととする．

3.3.2. 品質特性シナリオ

品質特性シナリオは，[ステップ 6] で作成されるアーキテクチャ手法を分析した結果得られた，個々の品質特

SC01		異なるクラウドサービスにおいて SSO 可能			
品質特性		1. 機能性（fuctionality）	品質副次特性	1.3. 相互運用性（interoperability）	
環境					
刺激					
応答時間					
アーキテクチャ上の決定	ArchitectureID	Sensitivity	Trade-off	Risk	Non-risk
OpenAM による認証	SC0101		T01	R01	N01
	SC0102	S01	T02		
	SC0103	S02			
論理的根拠					
アーキテクチャ図					

図 2　適用プロジェクトにおける品質特性シナリオ

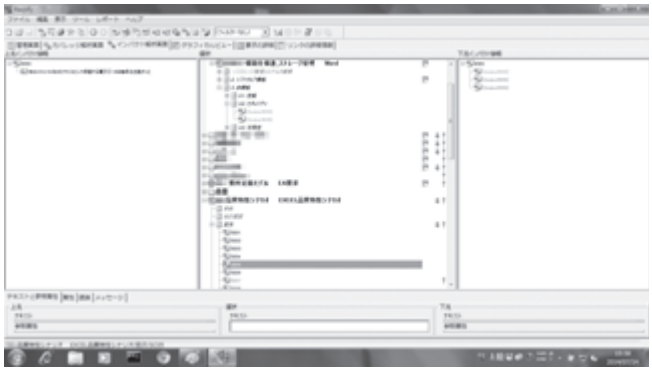

図 3　Reqtify によるインパクト分析

性実現に当たってのアーキテクチャのシナリオである（図 2）.

当該シナリオにおいて,「アーキテクチャ上の決定」としてシステム横断的な複数のアーキテクチャ上の決定が記述される．品質特性を実装するに当たって予めテンプレートを作成して使用するのだが，この時同一ファイル内の別シートに，当該シナリオ実施に当たって分析，識別したリスク，センシティビティ，トレードオフ等の情報を記載するようにしておく．1 つの品質特性シナリオ当たり，少なくとも 1 つ以上の「アーキテクチャ上の決定」があることから，決定毎にこれらリスク，センシティビティ，トレードオフが詳細化される．

即ち，この品質特性シナリオこそが，システム階層横断的な関心事を取りまとめて説明する重要な成果物となる．

3.4. 品質特性シナリオとのトレーサビリティ

品質特性シナリオによって明らかになった，システム横断的な関心事に対して，当該アーキテクチャを実装するプログラム設計ドキュメント，システム基盤設計書等とのトレーサビリティを確保する．但し，品質特性シナリオを作成するのは，全ての非機能要求に対してではなく，ATAM のプロセスの中で重要とされたシナリオに対してのみであることに注意されたい．コスト，期間とのバランスを取り，重要な品質特性についてのみ詳細化の対象とする．

具体的なトレーサビリティ確保の方法としては，機能 ID 等と同様に個々の品質特性シナリオに対して品質シナリオ ID を振り出し，更に品質シナリオを実現するための個々のアーキテクチャに対してアーキテクチャ ID を採番する．横断的関心事を実装するプログラム設計ドキュメント，システム基盤設計書等に該当する品質シナリオのアーキテクチャ ID をタグ情報として埋め込むものとする．

3.5. トレーサビリティツール

表 1 に例示した TM による可視化では，主に関連する 2 つのノード間の関係が可視化されるに過ぎない．市販のトレーサビリティツールを使用した場合，そのツールの機能および仕様にロックインされてしまうリスクが存在するものの, 表ベースで管理される TM を用いてトレーサビリティを管理するよりは一般に高い生産性とツールの機能によりインパクト分析の精度向上を期待することができる．今回の適用プロジェクトにおいては, ダッソーシステムズ（株）が提供するトレーサビリティツール「Reqtify」を採用し，各種ドキュメントに埋め込まれたタグ情報のトレーサビリティを可視化することとした．

3.6. トレーサビリティツールによる横断的関心事に関するインパクト分析

Reqtify のインパクト分析機能は，中央ペインの着目するノードを選択すると，左ペインに関連する上位のノード，右ペインに下位のノードを一目で表示するというものである（図 3）.

ここであるノードは，3.4 に述べた品質特性シナリオの実装箇所であり，予めシナリオのタグ情報が埋め込まれているものとする．Reqtify から当該ノードを選択すると，上位階層に関連する品質シナリオが左ペインに表示される．更に上位の当該品質シナリオを選択すると，選択したノードは中央ペインに，下位階層に横断的に関連する全ての実装箇所へのリンク情報は右ペインに表示されることになる．

図 4 における機能 11 を修正するケースを例に説明する．機能要求に関するトレースを考えた場合，クラス A, B の修正だけを考慮することになる. しかしここではアーキテクチャ ID11 に関してトレースが確保されているため，まずアーキテクチャ ID11 のセンシティビティ，トレードオフ，リスク等についての影響を検討することになる. 続いて, 上位階層の品質特性シナリオ 1 へのトレーサビリティを確認した後，当該シナリオの下位階層へのトレーサビリティに注目する．すると横断的な関心事としてアーキテクチャ ID12 とこれを実装するクラス D へのトレーサビリティが確認できる．機能 11 の修正に当

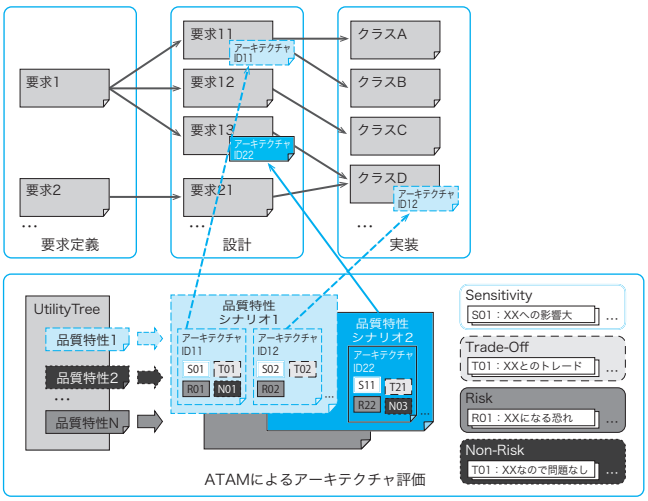

図 4 アーキテクチャへのインパクト分析

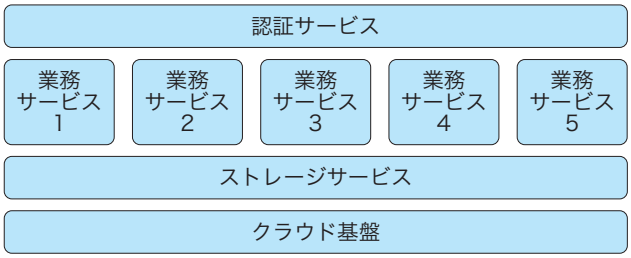

図 5　適用プロジェクトのサービス階層

たっては，これら横断的な関心事に関しても品質特性シナリオに沿った検証を実施することによって，インパクトについて高い精度で分析することが可能となる．

4.　適用と評価

4.1. 適用

これまで述べてきた ATAM 検証結果をシステム横断的な関心事へのトレーサビリティとして利用する手法について，以下のような実プロジェクトにおいて適用を実施して評価を行った．

要求定義フェーズにおいて，従来からの直列的なトレーサビリティ要素の起点となるソフトウェア機能要求として約 500 件の実装すべき機能を抽出し，合わせて ATAM によるアーキテクチャ評価の結果，ステークホルダ間によって特に重要と認識された品質特性が 24 件となったため，同じく 24 本の品質特性シナリオを作成し，アーキテクチャの検証を行った（表 4）．

また，本プロジェクトは異なる 7 つのサービスから構成され，内フロントサービスとしての認証サービス，バックエンドとしてのストレージサービスを 5 つの業務系サービスが共有して使用する構成となっている（図 5）．

ここで重要とされた品質特性の内，横断的な関心事の一例として，機能性の中の機密性の確保がある（表 5）．

表 5 の機密性確保の例においては，認証によるアク

表 4　適用プロジェクトの諸元値

適用プロジェクト	クラウドサービスの新規構築プロジェクト
開発期間	15 ヶ月
開発工数	約 170 人月
実装サービス数	7 件
ソフトウェア機能要求数	約 500 件
ATAM 実施により重要と認識された品質特性数	24 件（品質シナリオ数と同値）
トレーサビリティ設計工数	5 人日
トレーサビリティ進捗監視工数	1.5 人時 / 週

表 5　機密性の確保における例

品質シナリオ No.SC02

アーキテクチャ上の決定	Sensitivity	Trade-Off	Risk	Non-Risk
OSS XX による認証機能実装	S03		R02	
パスワードのハッシュ化保存		T03		N02
外部ストレージアクセスキーの暗号化	S04	T04	R03	N03
クライアント送信データの暗号化	S05	T05		
HDD の暗号化	S06	T06	R03	

Sensitivity

No.	説明	備考
S03	認証機能の修正に当たっては OSS XX の I/F，パラメータ設定に影響がおよぶ可能性がある	
S04	秘密鍵を変更する場合，暗号化済データの再暗号化処理が必要になる	
…	…	

Trade-Off

No.	説明	備考
T03	ハッシュは非可逆性のためシステム管理者もパスワードは不明となる	
T04	セキュリティは強固になるが鍵管理の仕組みが必要となる	
…	…	

Risk

No.	説明	備考
R02	OSS XX にセキュリティホールが検出された場合，ライセンス上，自社による修正が困難	
R03	秘密鍵が危殆化した場合，テーブル全体のデータの安全性が落ちてしまう	
…	…	

Non-Risk

No.	説明	備考
N02	ハッシュ値のみを保存するため，漏洩した場合でもオリジナルのパスワードを特定されることはない	
…	…	

セス権管理だけではなく，ある業務サービスにおいては，クライアントへサービスするデータの暗号化を実施して機密性の確保に貢献している．また，クラウド基盤の実装として，ハードディスクの暗号化を分担する等して，システム全体の機密性を担保している．このようにセキュリティについては，何処か一カ所にセキュリティホールが埋め込まれているだけでシステム全体が危殆化する恐れがある．これは，システム横断的な関心事として，リスク，センシティビティ，トレードオフについて十分な検証が必要になる一例である．

4.2. 評価

図 6 は Reqtify によってプロジェクト仕掛かり中のトレーサビリティ実装の進捗状況を表示した画面をスナップショットしたものである．

図 6 上は，Reqtify のフォルダ機能を使用して，最上位の「要求モデル」（図 6 上のノード名では「要求定義モデル」）と，「要求モデル」内に定義したユーティリティツリーから導出された「品質特性シナリオ」に対して，図 5 に提示した 7 つのサービスとクラウド基盤に関するノードをサービス毎に取りまとめた 8 つのフォルダとのトレーサビリティを可視化したものである．

ここでは，「要求モデル」と下位のフォルダ間の関連線によってソフトウェア機能要求に対するトレーサビリティを（図 6 上の黄色の関連線※），また「品質特性シナリオ」と各サービスのフォルダ間の関係において，横断的な関心事についてのトレーサビリティを追跡（図 6 上の緑色の関連線※）するようにしている．

一方，図 6 下は，図 6 上のあるフォルダの中身 (サービス毎のトレーサビリティ対象ノード) を別途表示させたものである．ここでは，最上位の「要求定義モデル」から導出した「ソフトウェア機能一覧」と「ユースケース記述書」が定義されている．「ソフトウェア機能一覧」からは機能 ID 毎に詳細機能を記述した「機能仕様書」と機能を実装した「ソースコード」が，また，「ユースケース記述書」からは，これに記載したシナリオに沿って機能の検証を行う「受入テスト設計書」に対してトレーサビリティを定義している．また，「テストケース」は，シナリオの実施手順毎に関連する機能 ID に対しても関連付けすることにより機能検証の網羅性も確保している．更に，「品質特性シナリオ」に記載されたアーキテクチャの一部について，「機能仕様書」に記載された該当箇所とのトレーサビリティを確保している．

ここで Reqtify におけるノード間の関連線の色について説明する．上位のノードに定義した ID に対して下位のノードでカバーされている割合によって色が変わる仕様になっており，カバー率が 70% 以下である場合には赤色，70 ～ 90% の場合には黄色，90% 以上になると緑色で表示される．この機能により，現在上位ノードに定義された要素が下位ノードで設計あるいは実装されたかというトレーサビリティの進捗が分かる仕組みとなっている※．

図 6 上のようにトレーサビリティ定義を行うことにより，プロジェクトマネージャは，従来からの直列系のトレーサビリティについても，今回の研究テーマとした非機能要求のような横断的な関心事についても，同時に現在のトレーサビリティの実装状況について視認性の高い形式で可視化を実現することが出来る．

一方，プロジェクトメンバは，自身が担当するサービスのトレーサビリティ確保に責任を持つ．従って，図 6 下の自身の担当するサービスについて，アーキテクチャ実装を含めたトレーサビリティの進捗についてのみ認識していればよい．

トレーサビリティを詳細に追おう

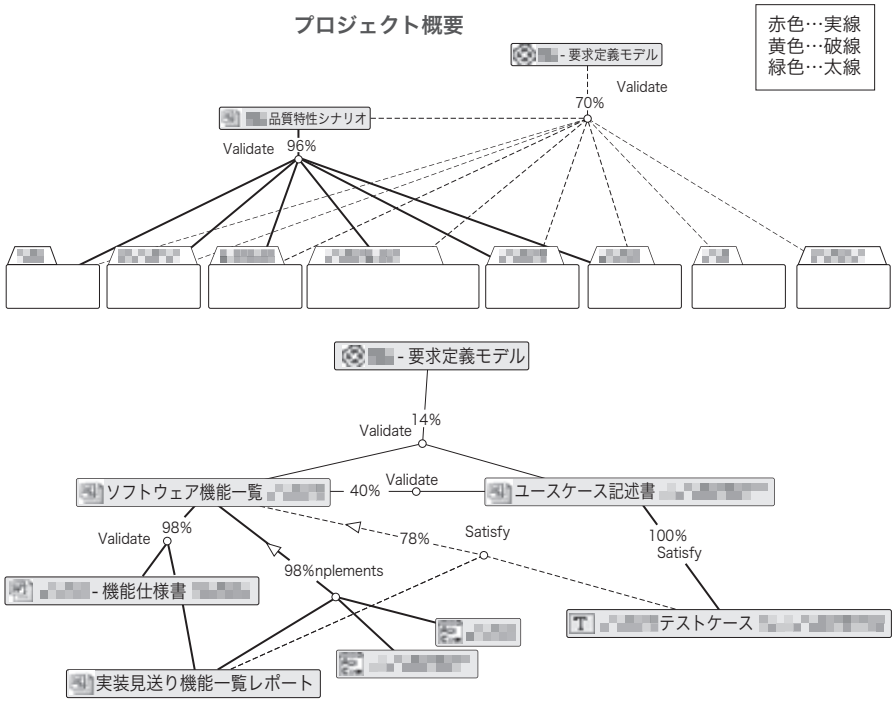

図 6 （上）：要求モデルから見た全体トレーサビリティの進捗と
（下）：個別サービスレベルのトレーサビリティ進捗

【脚注】

※ Reqtify 上では本来、関連線の色によって数値を示すが、本誌掲載にあたっては線種によりそれを示した

とすると，タグ情報が膨大となるため，PM/ プロジェクトメンバといった立ち位置に応じた粒度でトレーサビリティの進捗を可視化することが重要となると考える．

また，実装が既に完了した機能に対して変更要求が実施されることになったケースにおいて，3.6 に述べた手順の適用を試みたところ，インパクト分析に必要な関連ノードを抽出するコストは高々 1-2 人時であることも合わせて確認することができた．

5. 考察と今後の課題

5.1. 考察

トレーサビリティを開発当初から実装することにより，次期派生開発を待たなくとも，設計凍結後の変更管理の運用段階より，リスク評価の点で非常に大きな効果を発揮することが期待できると考える．

但し，開発当初から設計ドキュメントやプログラムコードに対してタグの埋め込みを実施する場合，設計の修正・変更や，プログラムの障害修正・リファクタリングの実施に伴い，トレーサビリティ情報の修正が必須となる．

今回採用した手法，ツールについては，目視によって修正に誤りがないかどうか担保するため，特に修正漏れによって古いトレーサビリティ情報が残存してしまうリスクについて十分考慮する必要がある．

また，今回提示の手法は，既存の開発資産に対する適用を実施する場合，特に初期プロジェクトにおいて，アーキテクチャ評価を実施していないプロジェクトへ後追いで適用する場合において特に注意を要する．すなわち，正しいアーキテクチャ評価が可能なインプット情報を収集する事が出来るかどうかに強く依存することに留意されたい．

このため，既存開発資産へのアーキテクチャ再評価に当たっては，ドキュメント類のインプットだけではなく，初期開発に携わったアーキテクトの参加が必須となると考える．

5.2. 今後の課題

今回のような新規適用プロジェクトについては，考察に述べた設計変更やプログラム修正時のトレーサビリティ保守が大きな課題と認識している．

今後は，差分認識に優れた構成管理ツールとの組み合わせによって，トレーサビリティのメンテナンス漏れを検出できる仕組みを検討していきたい．

また，既存の開発資産に対する適用を実施する場合，トレーサビリティの整備に必要なコストについて見積もる必要がある．

独立行政法人情報処理推進機構，トレーサビリティ確保におけるソフト開発データからの効果検証 [IPA2013] においてトレーサビリティ対策工数の試算とその検証が試みられている．但し，トレーサビリティ対策工数については，開発規模だけではなく費用対効果の観点や，プロジェクトに求められる品質レベルの観点等から，必要となるトレーサビリティの粒度は異なって然るべきである．

6. おわりに

6.1. まとめ

今回は新規クラウドサービス開発プロジェクトにおいて，アーキテクチャ横断的な関心事についてもトレーサビリティを確保することでソフトウェア信頼性の向上に対するアプローチを試みた結果について報告させていただいた．

まずは今後の派生開発に備えた初期開発の一環として，アーキテクチャ横断的な関心事に対してもトレーサビリティの対象にすることで，変更管理における効果を確認することができた．

引き続き，今後の派生開発においても当手法を継続適用することで，トレーサビリティ確保によるアーキテクチャ横断関心事へのインパクト分析が，アプリケーションライフサイクル全般の信頼性確保に対して貢献できることを検証していきたいと考える所存である．

6.2. 謝辞

今回の試みに協力していただいた適用プロジェクトの開発メンバーに対して，まずお礼を申し上げます．

また，本プロジェクトで採用した Reqtify の使用に当たって数々の助言をいただき，また検証にもご協力いただいた弊社 澤木 孝さん，舟田 学さん，論文主旨の英訳に協力していただいた弊社 和良品 文之丞さん，元上司である塚田 恒博さんに感謝の意を表します．

最後に，当論文執筆に当たり監修いただくだけでなく，様々な示唆に富むアドバイスをいただいた同じく弊社の上原 敏幸さんに対して特にお礼の言葉を捧げます．

【参考文献】

[Kazman2000] R.Kazman, M.Klein, P.Clements, ATAM: Method for Architecture Evaluation, 2000.

[IPA2013] 独立行政法人情報処理推進機構，トレーサビリティ確保におけるソフト開発データからの効果検証，pp.3-4, pp7-11, pp30-56, 2013

[IPA2005] 独立行政法人情報処理推進機構，参照アーキテクチャ調査報告（2005 年度），II-ii IT アーキテクチャ評価技法 pp.4-12, 2005

[Okuda2013] 奥田 博隆，小形 真平，松浦 佐江子，要求仕様と設計の機能要件のトレーサビリティを保持する為の Web アプリケーション設計手法の評価，情報処理学会第 75 回全国大会，pp1_441-1_442,2013