Vol.2012-SE-176 No.13
Vol.2012-EMB-25 No.13
2012/5/22

情報システム構築時の アーキテクチャ自己評価支援手法の研究

足達健[†]　落水浩一郎[†]

アーキテクトが自身が作成したアーキテクチャを評価するためには評価のための観点が必要となる．既存の研究成果として実践的なチェックリストが提案されているものの，どのような場合にどのチェックをすべきか優先順位付けがされていないため活用が困難である．この問題に対してアーキテクチャの変更パターン毎に優先度の高い検証項目が何であるかを実際のプロジェクト実績から明らかにし，既存の研究成果として得られているチェックリストと関連付けすることによって，変更パターンごとに検証すべきチェック項目を選択することのできる選択基準表を導き出す．

Proposal on creating checklist for evaluating architecture based on architectural decision

KEN ADACHI[†]　KOICHIRO OCHIMIZU[†]

Existing research suggests check list for architect to evaluate the completeness of architecture they designed. It is difficult to use this list since it does not suggest any prioritization on check items. To address this problem, we propose an approach for clarifying what kind of evaluation is done due to the architecture modification pattern and relating those items with existing check list. The proposed approach enables architect to choose important item due to their architecture modification pattern.

1. はじめに

今日では情報システムが広く浸透して欠くことのできない重要インフラとなってきおり，情報システムの構築・保守等のプロジェクトではより高い品質を要求されるようになっている．こうした中でプロジェクトを成功に導くために，機能要件と非機能要件を満たす適切なアーキテクチャ設計を行うことが求められており，このような観点からアーキテクチャ設計・構築の方法論が提示されている[1].

これらの方法論では設計したアーキテクチャをステークホルダに提示する前に，見落としや検討もれ等が無いか自己評価を行うことが重要とされているが，具体的な手引きが不足していると感じている．例えば Eeles と Cripps はアーキテクチャ定義タスクのひとつとしてアーキテクチャ検証というタスクを定義しているが([1])，レビュー，ウォークスルー，インスペクションのような技法を使用するとしているものの，具体的にどのような観点でどのような方法で検証を行うかについては触れていない．

一方で Rozanski と Woods はアーキテクチャをビュー（アーキテクチャの静的構造）とパースペクティブ（品質特性） を用いて設計・記述する手法を提示した[2][3]．同時に各ビュー・パースペクティブを作成する際に，アーキテクトが考慮点の抜け漏れが無いことを自己評価するためのチェックリストを定義している．しかし数が膨大であり，どういったケースでどのチェックを優先して実施するべきか基準が明示されていないため活用することが困難である．

そこで本論文では，アーキテクトがアーキテクチャの特性に応じて検証すべき項目を Rozanski と Woods のチェックリストから効率よく選択し，設計したアーキテクチャを評価することを支援する手段を開発することを目指した．

この目標に対してアーキテクチャの変更のパターンと，影響を受けるビューとパースペクティブとの関係を明らかにした．具体的には実際のプロジェクトで利用されている「アーキテクチャ上の決定事項」(決定を要する事項，決定を必要とする理由・背景，決定事項，決定理由などを記したもの) から上記関係を抽出する手段を示した．そのようにして得られた分類を用いて，Rozanski と Woods によって定義されたチェック項目を選択する手段として基準表を作成した．

本論文で取り扱う変更のパターンとしては，機能的発展性(機能の追加・改善)，プラットフォーム発展性(バージョンアップなどのシステムプラットフォームの更新)，環境発展性(外部システム統合) の 3 つとし，それぞれについて上記手法を適用し具体的な選択基準表を作成した．これらの基準表を用いて実際の事例に対して有効性の確認を実施した結果，効率的にチェック項目を選択することができるという評価結果を得た．

2. 関連研究

本論文で扱う内容は ATAM（Architectural Tradeoff Analysis Method)）のような方法論を用いたソフトウェア・アーキテクチャ評価の研究領域と関連する．ATAM のような評価メソッドが，アーキテクチャがステークホルダによって定義された目標に対して適切かどうかを評価すること

[†] 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology

Vol.2012-SE-176 No.13
Vol.2012-EMB-25 No.13
2012/5/22

を目的としているのに対して，Rozanski と Woods の研究や本論文は，アーキテクトが自ら設計したアーキテクチャを検証し評価するプロセスを支援する．また本論文では基準表を作成するにあたって「アーキテクチャ上の決定」という成果物を利用する手段をとっている.「アーキテクチャ上の決定」はプロジェクトのライフサイクルを通じて実施されたアーキテクチャ設計上の意思決定を記録する成果物であり，決定事項を記録するためのテンプレートが様々な研究者から提案されている[4][5].

林と柿本はアーキテクチャの設計作業の効率化を図るため，機能要件を実現するアーキテクチャをボトムアップで洗い出してパターン化を行い，顧客要件に応じてアーキテクチャ・パターンから最適なものを選択する手法を提言している[6]．この際，アーキテクチャのパターン化を実施するに当たって，実際のプロジェクトのアーキテクチャ上の決定を利用している．また Zimmerman は SOA を適用してアプリケーション設計するケースにおいて，頻出する設計課題やそれに対する解決策を集め，Service-Oriented Architecture Decision Modeling Framework という名称の設計ガイドを作成した．この際に，頻出する設計課題を抽出するにあたって, SOA を適用したシステムの構築プロジェクトのアーキテクチャ上の決定を利用している[5],[7]. 図 1 に Eeles らや Rozanski らのアーキテクチャ定義プロセス([1],[2],[3]) を元に筆者にて加筆し本論文の範囲を提示する．

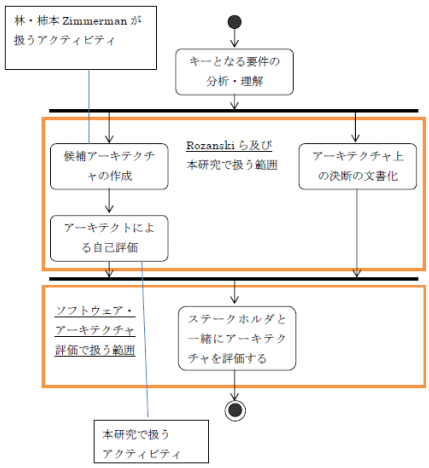

図 1　アーキテクチャ作成プロセスにおける本論文の範囲

アーキテクチャ作成プロセスはキーとなるシステム要件を把握・理解するところから始まる．次いでこれらの要件を充足するための候補アーキテクチャを作成する．この段階で作成された候補アーキテクチャは評価および更なる改良の土台としての役割を持つ．またこの過程でアーキテクトが下した意思決定は「アーキテクチャ上の決定」として記録される．次にアーキテクトは作成した候補アーキテクチャを評価し，モデルが実際に機能することや要求を満たすこと，隠れた問題が無いことを検証する．アーキテクトの評価が終わったアーキテクチャはステークホルダとの公式なアーキテクチャ評価プロセスに進む．

図 1 が示す通り ATAM のようなソフトウェア･アーキテクチャ評価が扱うのはステークホルダとの評価アクティビティであり，本論文が扱うのはその前のアクティビティであるアーキテクトによる自己評価である．また林と柿本らによるアーキテクチャ・パターンは候補アーキテクチャ作成時に事前に洗い出されたパターンより選択を行うことで，作成作業を効率的に行うことを目的としたものである．Zimmerman の設計上の課題抽出も候補アーキテクチャ作成時に何を設計する必要があるかを示すものである．それに対し本論文ではアーキテクトによる自己評価を支援することを目指す．

3. 課題点の絞込み

3.1 Rozanski と Woods のアプローチ

本節では Rozanski と Woods のアプローチの中心となるビューとビューポイント，そしてパースペクティブについて説明する．

3.1.1 ビューとビューポイント

ビューやビューポイントの概念は，IEEE1471-2000 で定義されたアーキテクチャ記述の標準で定義されているものである．ビュー，ビューポイントは以下のように定義されている（[8]．日本語訳は[2] による）．

ビュー：関連する一組の関心事の視点からシステム全体を表現したもの

ビューポイント：ビューを作成し，使用するための約束事の詳細仕様である．ビューの目的や利用者を明確にし，それぞれのビューを作成するためのパターンもしくはテンプレートや，それらを作成し分析するためのテクニックを定義する．図 2 は要素間の概念を示したものである．

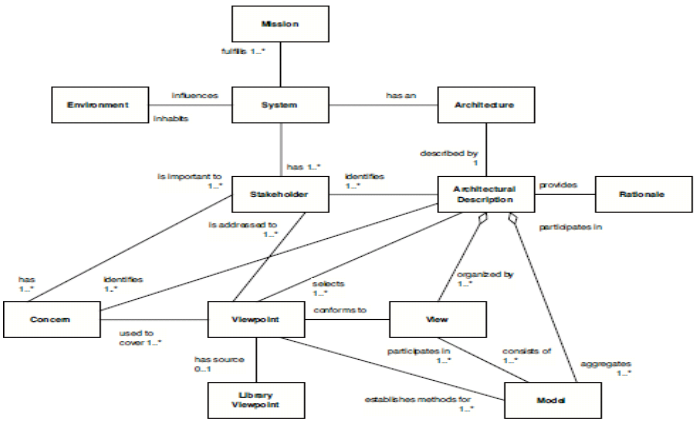

図 2　アーキテクチャ記述の概念モデル[8]より引用

当該概念図によると，アーキテクチャ記述は 1 つ以上のビューによって構成され，アーキテクチャを記述するためには複数のビューを用いる．ビューはビューポイントに従う. ビューポイントはビューを作成するための手段であり，言語や表記法を定義するため，ビューを作成するためにはビューポイントにしたがって作成する必要がある．

Vol.2012-SE-176 No.13
Vol.2012-EMB-25 No.13
2012/5/22

IEEE1471-2000 では具体的なビューとビューポイントについては定義していないが，様々な研究者によりビューポイントが提案されている. 表 1 は Rozanski と Woods が提案しているビューポイントカタログである.

表 1　ビューポイントカタログ [2]より引用

ビューポイント	定義
機能的	システムの機能要素，その責務，インタフェースおよび主な相互作業を記述する.
情報	アーキテクチャが情報を格納，操作，管理および配布する方法を記述する.
並行性	システムの並行性構造を記述し，機能要素を並行性単位へマップして，並行して実行可能なシステム部分と，並行実行を調整および制御する方法を識別する.
開発	アーキテクチャがソフトウェア開発プロセスに対して課す制約を記述する.
配置	実行時環境でのシステムの依存性を把握することを含め，システムが配置される環境を記述する.
運用	本番環境での実行時におけるシステムの運用，管理，サポート方法を記述する.

Rozanski と Woods は IEEE1471-2000 のビューポイントの標準構造を拡張し，ビューポイントについて，以下の構造を提示している.

- ビューポイントの定義
- ビューポイントが対応する最重要関心事項
- ビューに興味を持つ可能性が最も高いステークホルダ
- ビューを説明するために構築する最重要モデルと，使用する表記法
- 注意すべき問題および落とし穴と，それを軽減するためのリスク削減テクニック
- 妥当性と完全性および正確性を保証するために使用することのできる，ビューポイント開発時と，それを見直すときに考慮すべき事項のチェックリスト

3.1.2 パースペクティブ

前記のビューポイントを利用することでアーキテクチャ記述を作成することができる一方で，ビューポイントの観点ではステークホルダにとって重要な品質属性（パフォーマンス，セキュリティ，可用性等）については考慮されていないという問題がある．そこで Rozanski と Woods は新たにビューポイントに対する補足的な概念としてパースペクティブを提唱している．Rozanski と Woods による定義を以下に示す.

パースペクティブ：
アーキテクチャパースペクティブとは，多数あるシステムのアーキテクチャビュー全体にわたって熟慮を必要とする，関連した品質特性の特定セットを，システムが提示すると保証するために用いられるアクティビティや戦術および指針の集まりである.

パースペクティブはビューポイントに非常によく似た概念であるが，ビューポイントがシステムの構造に着目したものであるのに対し，パースペクティブは重要な品質属性に特化したものである．ビューポイントを使ってビューを作成することによりアーキテクチャを記述することができるのに対して，パースペクティブは単体ではアーキテクチャの一部を記述することはできない．パースペクティブは既に作成したビューをシステムにとって重要な品質属性を達成できるようにアーキテクトをガイドし修正できるようにするものである．Rozanski と Woods は具体的なパースペクティブとして表 2 の内容を提唱している.

表 2　パースペクティブカタログ [2]より引用

パースペクティブ	望まれる品質
セキュリティ	誰がどのような操作をどのリソースに対して実行することができるかを確実に制御・監視・監査して，セキュリティメカニズムの障害を検出・回復するシステムの能力
パフォーマンスとスケーラビリティ	指定されたパフォーマンスプロファイルの範囲で予測したとおりに稼動して，増大する処理量をさばくシステムの能力
可用性とレジリエンス	必要な時に応じて全体または部分的に運用可能であり，システムの可用性に影響の恐れがある障害を効果的に処理するシステムの能力
使用性	システムと対話する人々が効果的に仕事をすることができる容易さ
アクセシビリティ	障がいのある人々によって使用されるシステムの能力
配置場所	構成要素の絶対位置とその要素間の距離によってもたらされる問題を克服するシステムの能力
規則	国内法と国際法，準法律的規則，会社方針，その他の規則や基準を遵守するシステムの能力

3.2 Rozanski と Woods のアプローチの課題点

ここでは Rozanski と Woods のアプローチについての課題点を特定する.

筆者が実際に実務で Rozanski と Woods のチェックリストを活用しようとした際に発見した最大の課題点は，ビュー・パースペクティブの選択基準が提示されていないことであった．この点に関して Rozanski と Woods は,「ビューポイントすべてがアーキテクチャに当てはまるとは限らず」, ビューポイントの選択には「アーキテクチャの性質と，ステークホルダのスキルと経験，それに費やせる時間とその他の制約を理解することが最初に取り掛からなければならない仕事」であると述べている[2]．またパースペクティブについても「アーキテクトが適用すべきパースペクティブのセットを選択するのは個別のシステムのニーズに完全に依存するため，アーキテクトのスキルと判断の問題である」と課題があることを認めおり[3]，パースペクティブの選択の基準は,「ステークホルダのニーズと，そのニーズに対するいろいろな品質特性の相対的な重要度および自分自身の経験と判断である」と述べるにとどまっている[2]．6 つのビューポイントと 7 つのパースペクティブそれぞれにおよそ 10 から 20 個のチェックリストが用意されているため，モデルを全部作成して各々を検証するには相当の時

間を要する．また汎用的なチェックリストになっているため，アーキテクチャ変更のパターンによっては該当しないものが多い．

4. ビュー・パースペクティブ選定基準の検討

4.1 選定の基準

Rozanski と Woods の作成したチェックリストを活用してアーキテクチャ妥当性検証を効果的に実施するためには，各ビュー・パースペクティブの選択基準を作成する必要がある．前述した Rozanski と Woods の記載を元に選択基準の要素となる候補を挙げると以下のようになる．

ビューポイント：

- アーキテクチャの性質
- ステークホルダのスキルと経験
- 時間の制約

パースペクティブ：

- ステークホルダのニーズとニーズに対する品質の相対的な重要度
- 自分自身の経験と判断

これらの項目のうち本論文ではアーキテクチャの性質を取り上げる．第 1 の理由はアーキテクチャの性質により優先すべき検証項目が大きく異なるからである．例えば機能の追加・改善のケースであれば変更の難易度と既存アーキテクチャへの影響の評価が重要となるのに対して，システムのバージョンアップ等のプロジェクトでは新システムで達成すべき品質属性の要求の把握や新システムへのアプリケーションやデータの移行が重要な検証項目となる．このように検証項目はアーキテクチャの性質に大きく依存して変わると考える．Rozanski と Woods も「プロジェクトタイプそれぞれに，異なる優先順位があり，それが，検討する必要があるビューポイントとパースペクティブに反映される」としている[2]． 第 2 の理由は利便性である．当該基準表はアーキテクトが自分の担当するシステムの特性を把握し，その特性に対応するビュー・パースペクティブを選択するという利用の仕方を目指す．前記の基準要素のうちアーキテクチャの特質以外は，個々のシステムやプロジェクトに依存して様々なバリエーションが存在しえるため，基準表が複雑になり利用することが難しくなる．

次に「アーキテクチャの性質」をより具体的に定義する．Rozanski と Woods は「発展性パースペクティブ」の中で，一般的に発生し得る変更のタイプを以下のように分類している[2].

表 3 変更のパターン [2]より引用

変更のパターン	パターン記述
機能的発展性	システムが提供する機能への変更を指す．単純な欠陥の修正から，サブシステム全体の追加または置換までさまざまなものがある．
プラットフォーム発展性	プラットフォームの移行，プラットフォームの拡張（新プラットフォームへの製品の移植等)
環境発展性	他の外部システムとの統合を必要とするケース．即時のシステムからの情報取り出しや他システムへの処理のためのデータ提供などを含む．

本論文ではこの 3 つの変更のタイプを，既存システム拡張型プロジェクトにおける変更のパターンとし，このパターンごとに影響を受けるビュー・パースペクティブと Rozanski と Woods のチェックリストを選択できる基準を作成する．ここまでの議論より表 4 の形式の選択基準表を作成することを目指す．

表 4 ビュー・パースペクティブの選択基準表

変更のパターン	影響を受けるビュー・パースペクティブ	Rozanski と Woods のチェック項目

当該選択基準表とビュー・パースペクティブとの関係を，Rozanski と Woods の主要概念間関係図[2]に追記して図 3 に示す．

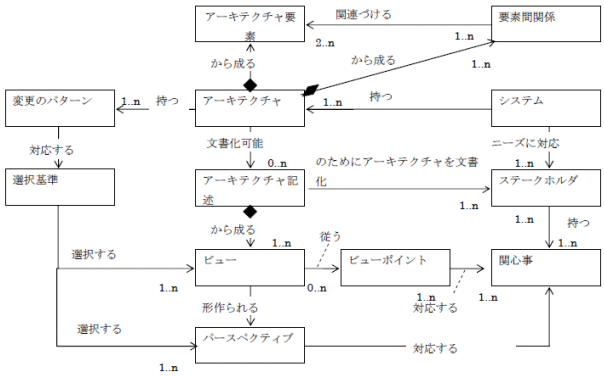

図 3 選択基準表の位置づけ [2]より引用して加筆

- アーキテクチャは 1 つ以上の変更のパターンを持ち，この型に応じて選択基準表を選択する
- 選択基準表より対応するビューとパースペクティブを選択する

4.2 作成手法

当該基準表を作成するためには，実際の既存システム拡張型プロジェクトの事例を調査し，以下の項目を収集する必要がある．

- どのような検証が実施されたか
- どのビュー・パースペクティブと関連性が高いか

その方法として，ここではアーキテクチャ上の決定事項を使用するアプローチを採用した．本節では最初にアーキテクチャ上の決定について説明し，次いでこれを用いた研究例を取り上げる．最後にアーキテクチャ上の決定を利用して基準表を作成するアプローチを，例をあげて説明する．

4.2.1 アーキテクチャ上の決定

アーキテクチャ上の決定はプロジェクトのライフサイクルを通じてアーキテクトが実施した意思決定を記録する

成果物である．Zimmerman はアーキテクチャ上の決定を以下のように定義している[5]．Architectural decisions capture key design issues and the rationale behind chosen solutions. They are conscious design decisions concerning a software intensive system as a whole or one or more of its core components and connectors in any given view. The outcome of architectural decisions influences the system's nonfunctional characteristics including its software quality attributes.

基準表を作成するにあたって，アーキテクチャ上の決定を利用する第 1 の理由は，それがアーキテクチャを評価した結果を反映した成果物であるためである．候補アーキテクチャを作成する際に実施した意思決定が記録され，次いでアーキテクチャのオプションを探る際に評価が実施される（図 1 参照）．その結果で既存のアーキテクチャ上の決定が見直され，また新たな決定事項が付加される．そのため当該成果物を参照することで，どのような検証が実施されたかを確認することができると考えた．

第 2 の理由としては，アーキテクチャ上の決定を参照することで最も重要な検証項目に集中することができると考えられるためである．アーキテクチャ上の決定は通常当該アーキテクチャを形作った主要な判断に限定される（[2],[5]）．したがってこのアーキテクチャ上の決定事項から取り出した検証項目は，対応するアーキテクチャのパターンにとって最も重要で優先度の高いものである可能性が高く，効率的に情報を収集することができると考えた．

4.2.2 アーキテクチャ上の決定を利用したアプローチ例

林と柿本はアーキテクチャの設計作業の効率化を図るため，機能要件を実現するアーキテクチャをボトムアップで洗い出してパターン化を行い，顧客要件に応じてアーキテクチャ・パターンから最適なものを選択する手法を提言している．ここで林と柿本が取り上げているアーキテクチャ・パターンとは主に配置ビューにおける配置パターンを指す(例：Web システム環境において配置モデルでのノード配置を行う際に，アプリケーション実行ノードと DB ノードを同一ノードとするパターンと別にするパターン)．アーキテクチャを選択し意思決定を下す際には対象とする課題(機能要件) に加えて，顧客の環境，要求事項，前提・制約および，それらの優先順位など様々な要因を考慮する必要がある．林と柿本はこれらの要因を「コンテキスト」と呼び，パターン化したコンテキストとアーキテクチャのパターンを関連づけることを試みた．これによりアーキテクチャ設計時に顧客ら提示されるコンテキストと，事前に定義したコンテキスト･パターンの Fit&Gap を行うことで最適なアーキテクチャ･パターンを選択することを目指した．林と柿本はこれらのコンテキストとアーキテクチャをパターン化するにあたって，経験したプロジェクトで実施したアーキテクチャ上の意思決定を文書化し，コンテキスト・パターン候補とアーキテクチャ・パターン候補の抽出を実施している．

一方 Zimmerman は複数のシステムで登場するアーキテクチャ上の決定事項をもとに，アーキテクチャ設計時に活用することのできる設計ガイドを作成することが可能とし，この考え方に基づいて Service-Oriented Architecture Decision Modeling Framework という設計ガイドを作成している[5]．このガイドは Guidance model と Decision model から成る．Guidance model は特定のアーキテクチャスタイルを特定のアプリケーション領域に適用する際に，決定を必要とする事項(Issue) を含んでいる．アーキテクトは Guidance model の内容を自分のプロジェクトに合うようにテーラリングを行いながら実際の設計を行う．このようにして設計時に下したアーキテクチャ上の決定を記録したものが Decision model である．Zimmerman はこの Guidance Model を作成するために，独自のアーキテクチャ上の決定テンプレートを定義し，それを利用して実際のプロジェクトのアーキテクチャ上の決定を調査している[7].

以上の研究成果を参考にして，アーキテクチャ上の決定事項を利用してプロジェクト実例を収集するアプローチを取ることとした．

4.2.3 アーキテクチャ上の決定を利用した基準表作成方法

ここではアーキテクチャ上の決定より基準表を作成する手法について述べる．アプローチとしては以下の 2 つが考えられる．

1. アーキテクチャ上の決定テンプレートに検証項目を記述する属性を追加し収集する
   「検証項目」という属性をテンプレートに追加し，当該アーキテクチャ上の決定を実施する際にどのような検証が実施されたかプロジェクト参画者に記載させる．4.2.2 の Zimmerman のアプローチが該当する．
2. 既存のアーキテクチャ上の決定テンプレートの情報と Rozanski と Woods のチェック項目との関連づけを行う
   完了したプロジェクトのアーキテクチャ上の決定より情報を抽出し，Rozanski と Woods のチェック項目と関連づけを行う．4.2.2 の林と柿本のアプローチが該当する．

1 は検証項目を特定して収集ができるため，実際の検証項目を正確に集めることができる一方，新たに詳細な調査が必要となるため作業期間が必要となる．2 は既存の入手可能な情報より作業を行うため短期間で実施可能であるが，アーキテクチャ上の決定に記載されている項目と，Rozanski と Woods のチェック項目それぞれを分析し，記載内容に基づき関連有無を判断する必要がある．本論文はアーキテクチャ変更パターンより Rozanski と Woods のチェック項目を効率的に選択できることを示すことを優先して実施するため，2 のアプローチを取ることとした．

Vol.2012-SE-176 No.13
Vol.2012-EMB-25 No.13
2012/5/22

筆者が参画した実際のプロジェクトで経験したアーキテクチャ上の決定を利用し，そこから基準表を導く．アーキテクチャ上の決定事項は筆者が業務で使用している表 5 のテンプレートを用いた．

表 5　アーキテクチャ上の決定記載テンプレート

項目	記述内容
タイトル	当該決定事項の表題
問題記述(決定を要する事項)	解決する必要のあるアーキテクチャ上の課題を記述する
決定を必要とする理由・背景・動機	なぜその決定を行う必要があるかを記載する
前提・制約	決定を行う上での前提･制約を記載する
重要度	他のアーキテクチャ上の決定事項に比較した重要度を記載する
決定状況	
選択肢	検討した選択肢を記載する
決定事項	決定内容を記載する
決定理由	なぜその選択肢を選んだか､理由を記載する

次に当該アーキテクチャ上の決定の各項目と Rozanski と Woods のチェック項目の記載内容間でどのような関係が成り立ち得るかを検討する．Rozanski と Woods のチェック項目は多岐に渡っているが以下の 4 種類に分類することができると考える．

1. 技術的リスクのチェック

   それぞれのビューポイント･パースペクティブに特化した，見逃されがちな問題点や技術リスクを挙げ，それらを避けるための対策を取っているかを確認するものである．アーキテクチャ上の決定との対応は以下の 2 つの場合がありえると考えた．

- 当該技術的リスクへの対応自体をアーキテクチャ上の決定として行う
- アーキテクチャ上の決定を行う場合に当該技術的リスクへの対応を理由に決定を行う

   前者の場合は「問題記述」自体に，後者の場合は「決定事項」の「決定理由」と対応すると考えた．

2. 要求の把握・定義もれが無いかのチェック

   パースペクティブが対応する品質属性について，具体的に定義をしておく要求事項が何かをチェックする．これは「決定を必要とする理由・背景・動機」と対応すると考えた．

3. ビュー間で整合性がとれているかのチェック

   ビュー同士の整合性を確保するためのチェックリストである．あるビューが変更となった場合に依存関係にあるビューに対しての変更を行う決定を実施することになると考えられるため，「問題記述」と関連付けを行う．

4. 品質を充足するために考慮すべきアーキテクチャ戦略

   パースペクティブが対応する品質属性を達成するために一般的にとり得る対策をリストアップしたものであり，これらの選択肢を考慮したかチェックすることができる．アーキテクチャ上の決定を実施する際の「選択肢」の内容と対応すると考えた．

この関係に基づいて実際に関連づけを実施する手順を図 4 に示す．

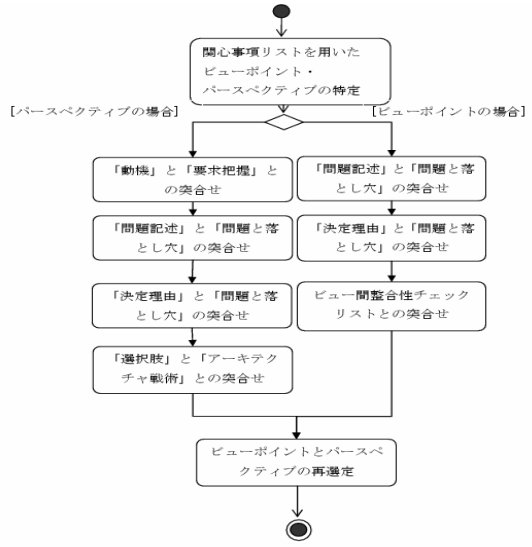

図 4　アーキテクチャ上の決定とビューポイント・パースペクティブ、チェックリスト間関連づけ作業プロセス

ビューポイント・パースペクティブの特定は，アーキテクチャ上の判断がどのような関心事項を扱っているか Rozanski と Woods が定義した関心事項リストと照らし合わせることで実施する．

次にビューポイント・パースペクティブを選定し，チェック項目とアーキテクチャ上の決定の内容との突合せを行う．ただし関心事項リストの記述が少ないビューポイント・パースペクティブもあるため，最初に選択しなかったビューポイント・パースペクティブについてもチェック項目を確認し，ビューポイント・パースペクティブの再選定を行う．

上記の関連付けアプローチで基準表を作成できるか例を取り上げて検証する．対象となるプロジェクトはインターネットを利用した電子申請システムの HW，SW の一括更新プロジェクトである．変更のタイプはプラットフォーム発展性に該当する．新システムのアーキテクチャ設計時に実施された意思決定の中から，データ移行の方法の選択に関連するアーキテクチャ上の決定例を表 6 に示す．

表 6　アーキテクチャ上の決定例

項目	記述内容
タイトル	データ移行方法の選定
問題記述(決定を要する事項)	現行システムから新システムへのデータ移行の方法を決定する．
決定を必要とする理由・背景・動機	移行対象のデータ DBMS 上のデータであり容量は 1TB である．切替のために停止可能な時間は 24 時間である．

前提・制約	分割移行は実施しない
選択肢	a)DBMS の Export/Import 機能を利用した移行方式 b)レプリケーションによる差分適用を利用した移行方式
決定事項	a) を選択
決定理由	現行環境での Export/Import 作業時の実績時間及び今回の想定移行対象容量より要件を充足できると判断.

このアーキテクチャ上の決定と，Rozanski と Woods の定義したビューポイント・パースペクティブとの関連づけを行う．関心事項リストと照らし合わせると，情報ビューの関心事項である「情報の構造と内容」「データボリューム」と合致するため情報ビューが抽出される．次に運用ビューの関心事項のひとつである「データの移行」を扱っていることから運用ビューが取り上げられる．

次にアーキテクチャ上の決定の決定理由を確認すると，移行時間内に収まることを考慮した決定であることがわかる．Rozanski と Woods のチェックリストを確認すると，運用ビューポイントの「問題と落とし穴」に「不十分な移行期間」という項目があり，ここではデータ移行が求められる移行期間よりも長くかかるリスクを指摘している．したがってこのチェック項目との関連づけが可能となる．

さらにこの例では情報ビューと運用ビューの 2 つが関連するビューとして取り上げられている.「情報ビューと運用ビューの整合性」チェックリストを確認すると,「移行が必要な場合，運用ビューは，データ移行の方法を明確にしているか」という項目があり，当該項目との関連づけが可能であることがわかる．

以上のことからこの例では表 7 のような選択基準を導き出すことができる．

表 7　選択基準表例

変更のパターン	影響を受けるビュー・パースペクティブ	Rozanski と Woods のチェック項目
プラットフォーム発展性	情報ビュー 運用ビュー	データ移行方法を明確にしているか。 データ移行に使用できる時間内に収まっているか

本節で記述した方法で実際のプロジェクトの 12 のアーキテクチャ上の決定より 3 つの変更タイプ全てについて基準表を作成した．付録 A.1 に一部を例示する．

5. 事例研究

ここでは実際に筆者が参加している既存システム拡張型プロジェクトを例にとり，ビュー・パースペクティブ選択基準の有効性を検証する．以下を抽出すること目的として実施する．

- 当該基準表を用いることによってどのような効果が得られるか
- 当該基準表の課題/問題点

対象システムは国内各都道府県にある支店の営業成績を収集して本部にて分析し，営業オペレーションの変革につなげることを目指すシステムである．第 1 フェーズでごく限られた支店から，顧客に関するデータのみの収集がすでに実施されていた．次フェーズでは収集するデータの範囲を拡大すると共に，残りの支店に対する接続を実施することがプロジェクトスコープとなっていた．第 1 フェーズで構築されたシステムの拡張型プロジェクトとなる．システムプラットフォームの増強も想定されていたことから，機能的発展性，システムプラットフォーム発展性，環境発展性全ての特性に合致するプロジェクトであった．

自チームには第 1 フェーズの構築プロジェクトに参画したメンバはいない中で，作業をどのように進めればよいか方針を立てる必要があった．筆者は計画局面の途中から参加し，立案されていた作業計画をレビュー・検討する立場にあった．当初はアーキテクチャ特性としては,「機能的発展性」を中心にとらえられていたため，追加する機能要件の検討や画面設計を中心としたアクティビティが予定されていた．しかし当該選定基準を適用してみたところ収集するデータ範囲の拡大と接続支店の増加が「環境的発展性」にあてはまると考えた．基準表（付録 A.1）にしたがって Rozanski と Woods のチェックリストを評価したところ情報ビューの「データ品質評価」を行うことが計画されていないことを発見した．

次に当該作業を実施することを計画に加えるようにプロジェクトメンバーに提言したが，この作業の経験が無かったこと及び当該作業に大きな工数が必要となることが予想されため必要性を認められず，すぐには計画がされなかった．そこで当該基準表を導く根拠となったアーキテクチャ上の決定事例を提示した．実際にデータ品質評価を実施したプロジェクトがあること，その結果より重要なアーキテクチャ上の決定が実施されたことを示すことで，当該作業の重要性・必要性を共有することができた．この結果当該作業はプロジェクト計画に加えられることとなり，いくつかのエンティティについてデータの完全性に問題があることを発見することができた．

この問題はここで発見されていなければそのまま見過ごされていた可能性があり，プロジェクトの実装フェーズで発見された場合手戻りとなる可能性が高いため，問題を識別できたことの効果は大きいと考えられる．また当該項目を効率的に選択することができたことも重要な点である．基準表が無い状態で Rozanski と Woods のチェックリストを使用するためには，機能的ビューから始まる各ビュー・パースペクティブの記述を最初から読み込み，各チェック項目が自システムにあてはまるのかどうかを順番に確かめていく必要がある．これには時間がかかると共に，場合によっては優先度の高い項目を見過ごしてしまう可能性がある．更に当該システムの第 1 フェーズの構築に携わっていないメンバでも当該タスクの必要性を理解し，実施するこ

とができたという効果も挙げられる．初期のシステム構築に携わっていたり，保守を長期に担当していたりするメンバは，自らの経験からデータ品質調査のような項目の重要性を指摘することができる．しかし既存システム拡張型プロジェクトの場合は，自らが構築に参画していないシステムを取り扱う場合もあるため，個人の経験・スキルに依存せずに検証ができることは重要である．また単に Rozanski と Woods のチェック項目を書籍上で参照するだけではその必要性を認識することが難しく，アーキテクチャ上の決定を合わせて参照することで，重要性を理解することができるようになった．適用した結果得られた効果をまとめると以下のようになる．

- 検証項目の抜け・漏れを防ぎ，見過ごされていた可能性のある問題点を早い段階で検知することができる．
- チェック項目を効率的に選択することができる
  当該基準表が無い状態では，200 ページを超える Rozanski と Woods の書籍を参照する必要がある．時間がかかるだけでなく，優先度が高い重要な項目を見逃してしまう可能性がある．
- 当該システムに携わっていない者でも基準表を用いることで，重要な検証項目を発見することができる．
- 同種のプロジェクトを経験していないメンバでも，当該基準表を用いることで重要な検証項目を発見することができる．

一方基準表を適用した結果以下の課題が明らかになった．

- 検証項目の必要性や実施要否はプロジェクトの判断に依存する
  基準表で検証項目の存在を認識することができても，その検証作業自体を実施することや，検証した結果対策を立てるかどうかは依然として個々のプロジェクトの判断に依存する．
- チェックリストの網羅性
  Rozanski と Woods のチェックリストでも十分に網羅性が確保されているわけではない．利用者の方で自組織の経験等に基づいてチェック項目をカスタマイズ(追加・変更等)する必要がある．

6. おわりに

本論文では Rozanski と Woods によって定義されたチェック項目を有効活用して，アーキテクチャの評価を支援する手段を開発することを目指した．既存システム拡張型プロジェクトを対象に,「アーキテクチャ上の決定事項」から変更のパターンと，影響を受けるビューとパースペクティブとの関係を明らかにする手法を提示した．その結果システムの変更のパターンごとにチェックリストの選択基準表を作成し，当該基準表を用いることで，アーキテクチャの評価作業を支援できることを示した．

今回は筆者自身が参画したプロジェクトを対象にしたため，アーキテクチャ上の決定事項を整理し，それがどのビュー・パースペクティブに関連するかを比較的容易に識別することができた．しかしより多くの事例を収集するためには自分自身が関係していないシステムの事例を確認する必要があり，選択基準表を作成するためには，よりシステム的に対応できる手法が必要となる．

また当該手法を利用した場合でも，検証作業自体や検証の結果識別した対策自体を適用するかどうかは，依然としてプロジェクトやアーキテクトの判断に依存する点は残る．しかし本論文で取り上げた方法により，少しでもアーキテクチャ設計作業から属人性を排し，構築するシステムの品質を向上させることにつながることができればと考える．

謝辞 本稿作成にご協力頂いた皆様に，謹んで感謝の意を表する．

参考文献

1) イールズ，ピーター，クリップス，ピーター著，榊原彰監修，「システムアーキテクチャ構築の実践手法」，ISBN: 978-4798121642，翔泳社, 2010.
2) ロザンスキ，ニック，ウッズ，イオイン，榊原彰監修，「システムアーキテクチャ構築の原理」，ISBN: 978-4-7981-1642-6，翔泳社, 2008.
3) Woods E, Rozanski N, ‘ Using Architectural Perspectives’, Software Architecture,2005. WICSA 2005. 5th Working IEEE/IFIP Conference on , vol., no., pp.25-35,2005..
4) Tyree J, Akerman A, ‘Architecture Decisions: Demystifying Architecture’, Software, IEEE, Vol. 22, Issue. 2, pp.19-27,2005
5) Zimmerman O., ’Architectural Decisions as Reusable Assets’, Software, IEEE, Vol.28, Issue. 1, pp. 64-69, 2011.
6) 林孝太郎，柿本達彦(2007)，『ソリューション・アーキテクチャのパターン化手法』，「ProVISION」，No57，pp.82-88，2007. http://www-06.ibm.com/ibm/jp/provision/no57/ibm paper3.html, 2012/1/4 アクセス
7) Zimmermann O, et al., ’Managing architectural decision models with dependency relations, integrity constraints, and production rules, Journal of Systems and Software, Volume 82, Issue 8, pp.1249-1267 ,2009.
8) Software Engineering Standards Committee of the IEEE Computer Society, IEEE, Recommended Practice for Architectural Description of Software-Intensive Systems, 2000.

付録

付録 A.1 基準表例

変更のパターン	影響を受けるビュー/パースペクティブ	項番	該当するRozanskiらのチェック項目	
環境的発展性	情報ビュー	1-1	17.3.1	データ構造と主要なデータのアトリビュートからおよびそのドメインからなる，高レベルのモデルを開発し，システム全部(内部および外部)に対して妥当性を確認する． 外部エンティティをモデルに含めることを忘れてはならない(他社とデータ交換する場合など)．
		1-2	17.3.2	データ品質の評価は行われているか．粗悪なデータに対処する戦略が作成されているか．
		1-3	17.3.4	全エンティティのキーを識別し，そのキーがアーキテクチャ全体で矛盾しないことを確認したか． エンティティが異なるキーを持って複数のシステムや場所に分散している場合，それらのキーの間にマッピングが定義されているか．データ項目が作成されたときに，そのマッピングを維持管理するプロセスはあるか．
		1-4	17.3.5	データの待ち時間(鮮度)の要求が明白に識別され，それを実現するためのメカニズムが整っているか．
	情報ビュー->開発ビュー整合性	3-1	22.8	開発環境とテストデータプラットフォームのサイジングは，情報ビューで作成されたデータ量を反映しているか． 情報ビューが外部のデータコンポーネントを定義している場合，開発ビューはそれを考慮に入れているか(スタブ環境や現実的なテストデータの作成など)
	情報ビュー->配置ビュー整合性	4-1	22.9	配置ビューは，情報ビューで指定されたキャパシティ要件を充足するのに十分なストレージを含んでいるか．