ソリューション・アーキテクチャーのパターン化手法

林 孝太郎　柿本 達彦

Method for Pattern Solution Architecture Model

Kohtaroh Hayashi and Tatsuhiko Kakimoto

各プロジェクトでは属人的手法でアーキテクチャー設計を行っており，アーキテクチャー再利用がなされていると言い難い．また,主非機能要件への影響を考慮しないと最適なアーキテクチャー適用が難しい．本論文では問題領域に対するアーキテクチャーをパターン化し,非機能要件の観点でパターン化したアーキテクチャーのトレード・オフ分析を行う手法を提案する．この手法は，標準化したモデル表記にて実績のあるアーキテクチャーをパターン化し，マトリックス形式のフレームワークを用いてトレード・オフ分析結果を整理することで，最適なアーキテクチャー選択を可能とする．この手法により，さまざまな課題や要因に適用可能なアーキテクチャー意志決定を支援するアセット作成を可能とし，要件定義局面でのアーキテクチャー選択作業が効率化される．

In the real projects, the way of designing an architecture depends on individual's abilities and experiences, and it is difficult to reuse it. Considering the effects of NFR(Non-functional requirements) is critical to choose an appropriate architecture as a solution. In this paper, the authors propose a method that define architecture patterns for an issue and analyze trade-offs from standardized NFR viewpoints. In this method, we define architecture patterns from proven architectures with standardized notation, analyze them using the NFR-based trade-off analysis framework to support to make the right architectural decisions in the right way. This method makes it possible for us to create assets which support architecture decisions that can apply various problems and contexts and makes it easy for us to select architecture efficiently in the phase of requirement definition.

Words & Phrases：アーキテクチャー，非機能要件，オペレーショナル・モデル，トレード・オフ分析，アーキテクチャー意志決定
Architecture, Non-functional requirements, Operational model, Trade off analysis, Architecture decision

1．はじめに

ソフトウェア・エンジニアリング領域での活発な研究により，ソフトウェア主体のシステムを対象とした工学的アプローチによる設計・開発手法は急激に進化しており，Rational Unified Process などの開発手法も一般的に広く知られるようになっている．IBM は，サービス・ビジネスにおける世界規模での競争力強化を目指し，IBM Global Service Method（以下 GS-Method と略す）［1］と呼ばれるシステム開発手法をグローバルで展開している．GS-Method では，アーキテクチャー設計作業がシステム開発プロセスにおける重要な作業項目として定義され，UML 表記によるコンポーネント・モデリング［2］と IBM 独自の技術であるオペレーショナル・モデリング［3］（以下 OM と略す）がアーキテクチャー設計技法として規定されている．開発手法の適用により作業項目，技法等の標準化により成果物の再利用が推進され，プロジェクト現場における作業の効率化や品質確保の実現が可能となってきている．

では，アーキテクチャー自体の標準化は推進されているのであろうか．GS-Method では非機能要件（Non-Functional requirements, 以下 NFR と略す）の分類定義を行い，NFR をアーキテクチャー構成要素に割り当てると定義しているが，その手法は示されていない．プロジェクト現場では，与えられた制約のもとで，お客様要件を実現するシステムアーキテクチャー設計を属人的

提出日：2007 年 9 月 3 日　再提出日：2008 年 2 月 27 日

な手法で実施しており，アーキテクチャー自体の再利用が推進されているとは言い難い（**課題 1**）．アーキテクチャーには，品質向上の主目的となる NFR を持っており，副作用として他 NFR に好影響あるいは悪影響を与えるという特徴がある．この特徴を認識せずにアーキテクチャーの選択を行うと，副次的な NFR の品質劣化を招き，システム品質低下を招くリスクがある（**課題 2**）．［4］［5］［6］

本論文では，アーキテクチャーの再利用を促進するため，問題領域を定義し，機能要件を実現するアーキテクチャーをボトムアップで洗い出し，パターン化して OM にて文書化する手法について述べる．さらに，最適なアーキテクチャー選択を可能とするため，策定したアーキテクチャー・パターンに対して NFR の各項目を評価し，アーキテクチャー・パターン適用によるトレード・オフ分析結果をマトリックス形式化する手法について述べる．

この手法により，さまざまな課題や要因に適用可能なアーキテクチャー意思決定を支援する成果物作成を可能とし，作成した成果物の利用により，提案活動や要件定義局面におけるアーキテクチャー選択作業の効率化が図れる．

以下，2 章で提案の目的を整理し提案アプローチ全体の概要を紹介し，3 章以下で提案アプローチの 4 つのタスクである，①問題領域定義，②アーキテクチャー意思決定の文書化，③ソリューション・アーキテクチャー定義，④ NFR トレード・オフ分析，について順次紹介する．

2. 提案アプローチの全体像

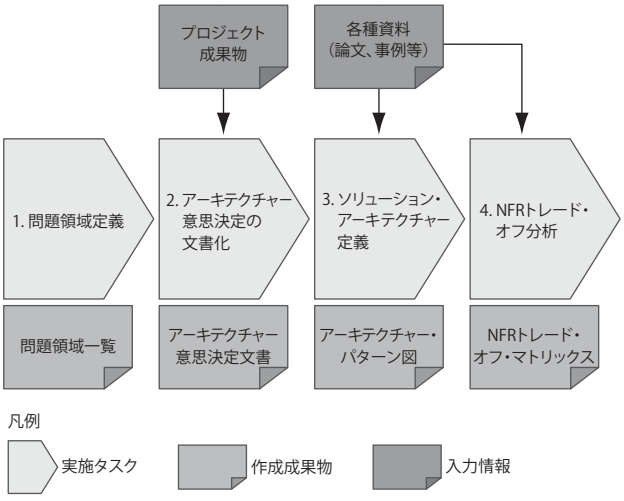


図 1. 提案手法におけるタスク関連図

本論文で述べる手法の全体的な流れを把握するため，この手法で定義した主要タスク，各タスクでの入力情報，各タスクでの作成成果物の関連を，**図 1** で示す．

提案する手法は，次の 2 点を目的としている．

（1）実際のプロジェクトにおいて対応すべき代表的な問題領域に対するソリューション・アーキテクチャー・パターンを定義し，NFR の観点で副次的影響を整理する．

（2）提案活動やプロジェクトの要件定義局面で実践可能な，アーキテクチャー選択手法を作成する．

目的の実現にかかる主要タスク作業として以下を行う．

（1）プロジェクトでの要件・課題の位置付けを明確化するため，問題領域を定義

（2）問題領域に対するプロジェクトでのアーキテクチャー意思決定の文書化

（3）（2）から課題（機能要件）とアーキテクチャー候補を抽出し，アーキテクチャーのパターン化および OM による文書化を実施

（4）NFR 分類の標準化，（3）で作成したアーキテクチャー・パターンに対するトレード・オフ分析および NFR トレード・オフ・マトリックスの作成

以下で各主要タスクの詳細について記述する．

3. 問題領域定義

プロジェクトで頻繁に発生すると考えられる要件・課題の位置付けを明確にするため，実際のプロジェクトにて対応すべき代表的な問題領域を定義する．対象とする問題領域の網羅性を確保するために，GS-Method を参照，カスタマイズして，**表 1** にある問題領域一覧表を定義した．

表 1. 問題領域一覧表

項番	問題領域名
1	プレゼンテーション・デリバリー・サービス
2	Web エンハンシング・サービス
3	Web アプリケーション・サービス
4	ビジネス・プロセス・オートメーション
5	インテグレート・エンタープライズ・サービス
6	認証・許可サービス
7	セキュリティー
8	システム・マネージメント・サービス
9	ネットワーキング・サービス
10	ビジネス継続性・災害対策サービス

4. アーキテクチャー意志決定の文書化

ソリューション・アーキテクチャーの意思決定には，対象とする課題（機能要件）に加えて，お客様の環境，要求事項，前提・制約および，それらの優先順位などさまざまな要因（コンテキスト）を考慮する必要がある．ソリューション・アーキテクチャーのパターン化にあたり，以下の二つのアプローチが考えられる．

（1）意思決定文書の形で，機能要件とコンテキストを収集・パターン化し，コンテキスト・パターンとアーキテクチャー・パターンを関連付ける．お客様から提示されたコンテキストと，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ー・パターンを意思決定する．
（2）機能要件を実現するアーキテクチャー・パターンを定義して，NFR 間のトレード･オフ分析を行う．そして，分析結果とコンテキストの Fit&Gap を行うことで，最適なパターンを意思決定する．

どちらのアプローチを選択すべきか判断することを目的の一つとして，3 章で定義した問題領域に対して，プロジェクトで経験したアーキテクチャー意思決定の文書化を行った．文書化項目として，アーキテクチャー意志決定における問題記述（機能要件），動機（NFR），仮定や前提･制約,検討した選択肢（アーキテクチャー・パターン候補），選択した理由，妥当性（トレード・オフ分析），派生要件（リスクと軽減策）などを定義する．これにより，意思決定のアカウンタビリティーを明確化し，トレーサビリティーを確保でき，再利用性向上が図れる．文書化フォーム作成には GS-Method 内に記述されているサンプルを参照して，意志決定記述形式と内容を標準化して記述内容の統一化を行う．

前述（1）のコンテキストのパターン化を行うためには，アーキテクチャー意思決定に影響を与える要素数と可変性により指数関数的にパターンが増加してしまう．例えば，「データ同期」という課題（機能要件）を考慮したとき，

アーキテクチャー意思決定に影響を与える代表的な要因としては，以下のような事項が挙げられる．

- データ配置場所（同一インスタンスか否か）
- データ配置場所（同一ロケーションか否か）
- DBMS の種類（同一種類か否か）
- 同期対象データの鮮度
- 更新データ量
- オリジナル・データの更新頻度
- 同期処理におけるデータ加工の必要性

これらの項目（7 項目）に対して、2 値の可変性しかないと仮定したときも，最大で 27 通りの組み合わせを考慮する必要がある．また，それぞれの組み合わせに対応した意思決定文書の作成が求められる．これは，限られた時間の中でのタスクでの作業の実現性だけでなく，アセット利用の容易性の観点でも非現実的であると判断して，機能要件を実現するアーキテクチャーのパターン化のアプローチ（2）を選択し，課題に対するリファレンス・アーキテクチャー・カタログ作成を目指した．

5. ソリューション・アーキテクチャー定義

1 章で提示した課題 1 の解決策として，アーキテクチャーの再利用性を推進するため，4 章で作成した課題（機能要件）を解決するアーキテクチャーを洗い出してパターン化し，リファレンス・アーキテクチャーとして活用できるようにカタログ作成する手法について記述する．以下でその手法ならびに工夫した点を記述する．

5.1 設計対象物の標準化

設計対象物の粒度が標準化されていないと，設計対象粒度に差異が生じ成果物再利用の阻害要因となる．よってモデリング手法を利用するにあたり，モデル化対象物の粒度を標準化することは，最初の段階で実施すべき重要な項目である．特に，モデル図上の主要構成要素である，ノード，コンポーネント群の粒度，名称，ID 等の標準化は非常に重要である．特にコンポーネント群などの構成要素に ID を付与し後工程に引き継ぐことで，設計と実装・テスト作業間のトレーサビリティーを確保することが可能である．

5.2 モデル表記方法の標準化

成果物表記内容の統一性を図るため，OM を使用する際の表記方法は GS-Method や［7］を参照し，以下の設計対象の記述内容と想定される構成要素を事前に定義して，成果物の標準化を行う．

- ロケーション／ゾーン／境界
- ノード
- 実行コンポーネント（プロセス）／データ・コンポーネント（データ・ストア）／プレゼンテーション・コンポーネント
- コネクション
- コンポーネント群のノードへの配置記述

5.3 アーキテクチャーのパターン化

OM にて課題（機能要件）に対するアーキテクチャーをパターン化するにあたり，以下の作業項目を定義する．

- ソリューション・アーキテクチャー定義
- アーキテクチャー・パターン図作成
- ウォークスルー実施と詳細記述

以下に実際に各作業で作成した成果物を利用して，詳細内容と工夫した点について記述する．

（1）ソリューション・アーキテクチャー定義

4 章で作成したアーキテクチャー意志決定文書より，課題（機能要件）とアーキテクチャー候補を抽出し，パターン化するアーキテクチャーの定義を明記する．**表 2** は作成文書の一例であり，提示するアーキテクチャー・パターンの特徴を記述している．

表 2. アーキテクチャー・パターン特徴

項目	内容記述
アーキテクチャー目的	Web システム環境におけるノード配置決定のためのアーキテクチャー・パターン
動機	アプリケーション・ノード及び DB ノードシステム基盤実現で，可用性を評価対象としてアーキテクチャーを決定する．
前提事項	• アプリケーション呼び出しは http を想定． ： • 通常稼働時可用性，システム保守時 1 ノード停止を考慮し，アプリケーション・ノードおよび DB ノードは 3 ノード構成とする．
アーキテクチャー・パターン	• アプリケーション , DB ノード同一ノード配置 • アプリケーション , DB 非同一ノード

（2）アーキテクチャー・パターン図作成

アーキテクチャー・パターン図作成にあたり，5.2 項でモデル表記方法の標準化を行って作成した成果物を参照して，モデルの標準化を行えるようにしている．

アーキテクチャー・パターンをモデル化する第 1 ステップとして,アーキテクチャーの目的および前提事項をもとに，アーキテクチャー・パターンに必要なノード配置，ノード配置後アーキテクチャーの機能要件を満たすコンポーネント群をノードに配置する．

第 2 ステップとして，テクニカル・コンポーネントのノードへの配置を行う．機能要件を実現するために必須のコンポーネント群をアプリケーション・コンポーネント群と，要求された NFR を実現するために追加された構成要素をテクニカル・コンポーネント群と定義する．モデル図において両者の違いを明記するために，テクニカル・コンポーネントにはコンポーネント名に「T」の接頭語を付与する．

表 2 で示すように，例とするアーキテクチャー・パターンで問題を解決する主要な NFR は可用性である．可用性を向上するためにテクニカルな判断により追加した構成要素は，E_ApplServer や E_DBMS の自動リカバリー機能要素（TE_RRS,TE_ARM）である．**図 2** は，アプリケーション・ノードや DB ノードにテクニカル・コンポーネント群を追加したアーキテクチャー・パターン図を示している．

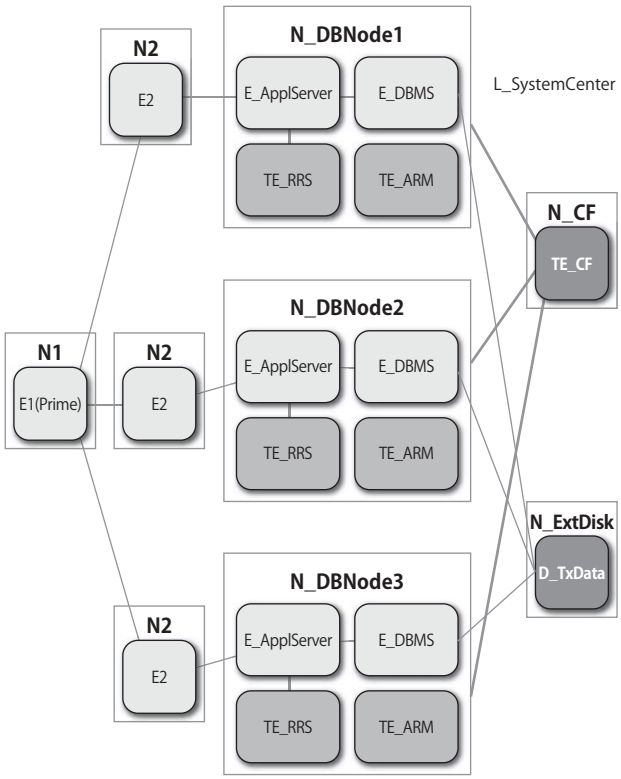


図 2. アーキテクチャー・パターン図

（3）ウォークスルー実施と詳細記述

アーキテクチャー・パターン図のノードとコンポーネント配置の充足性･妥当性，コンポーネント間連携の妥当性，モデルの実現性可能性を検証するため，アーキテクチャー・パターン図上でウォークスルーを実施し，その詳細記述を文書化して OM の検証を行う．

動的関連図はシーケンス・ダイアグラム図で記載するように定義されている［8］が，作業工数削減と OM の検証という目的を達成するため，**図 3** で示すコンポーネント群の静的関連図上に時系列の番号をつけた処理フローを記述して，ウォークスルーを実施する．また，通常時および障害（ノード障害）時の処理の違いを明確にするため，独立したモデル図としてウォークスルーを文書化する．

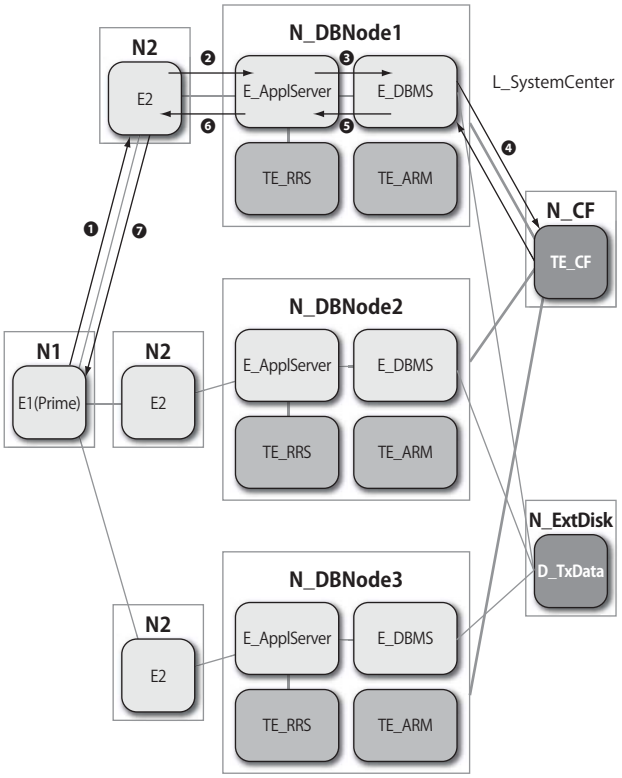


図 3. 通常時処理シーケンス図

6. NFR トレード・オフ分析

1 章で提示した課題 2 への対応策として，アーキテクチャーの NFR を評価する手法を提案する.

提案手法では，定義したアーキテクチャー・パターンの評価を行うにあたり，GS-Method で定義される NFR 分類を統一フレームワークとして採用し，アーキテクチャー・パターン毎に，NFR 分類の観点で，パターン適用により想定される影響をマトリックス形式で記述する手法を導入する. マトリックスの各項目（マトリックスの交点）には，パターン適用による副次的影響，制約事項，およびリスクを定義しており，この内容と実際のお客様の前提・制約事項を比較することで，最適なソリューションの選択を可能とする分析フレームワークとしている.

6.1 NFR 分類の標準化

トレード・オフ分析を行う上で，NFR は分析の評価基準項目に位置付けられる. よって各 NFR 項目内容を定義付けすることは，トレード・オフ分析の精度を向上する点で重要であり，前述したように GS-Method をリファレンスとして，トレード・オフ分析における分析観点を標準化する. NFR 分類項目名だけでは解釈が属人的になるため，各分類における分析観点も標準化し文書化する. **表 3** で NFR 分類項目と分析観点の文書化の抜粋を示す.

表 3. NFR 分類項目表

NFR 分類	分析観点
パフォーマンス・キャパシティー	レスポンス，スループット，キャパシティーを分析.
可用性	システム機能等、使いたい時使えるという性質
運用管理	システム運用管理容易性
:	:
信頼性	機能が正しく動くこと，故障が起こりにくい性質
:	:
派生要件	・分析にて劣化する NFR を向上するため必要な追加機能 ・課せられる制約事項

6.2 NFR トレード・オフ・マトリックスの作成

5 章で作成したソリューション・アーキテクチャー定義に対し，標準化した NFR 分類項目とその定義内容に基づいて各 NFR 項目に対する評価を行い，他 NFR に対する副次的な影響，効果および派生要件を定義し，**図 4** に示すような NFR トレード・オフ・マトリックスを作成する.

作成するマトリックスへの記述項目と内容，トレード・オフ分析結果の記述に対し工夫した点を，以下に示す.

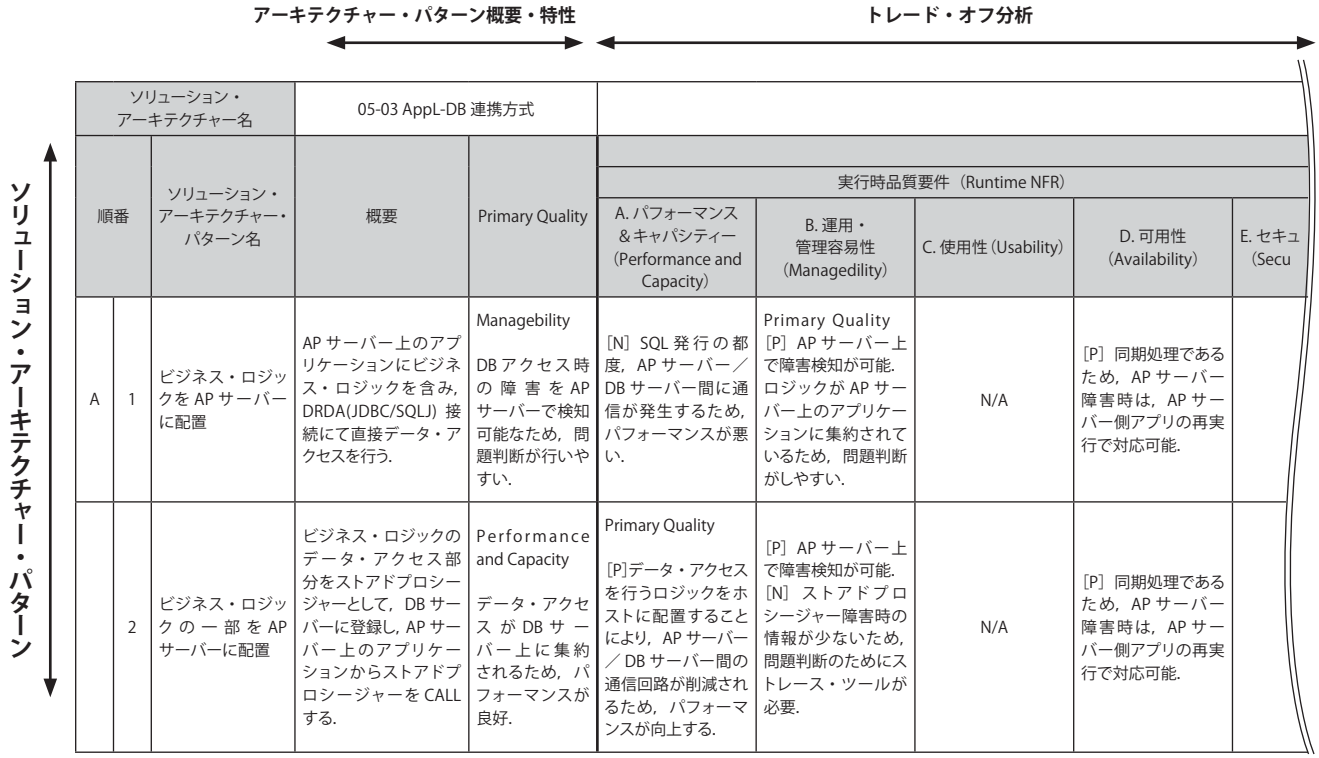
(1) アーキテクチャー・パターンに関する記述

トレード･オフ分析の対象とするアーキテクチャー･パターンの概要や特性を示すため，以下を記述する.

- 実装例の記述
  特定の課題（機能要件）における NFR を実現するために，特定の製品機能を前提とするケースがある. また，アーキテクチャーのシステム設計への転換を支援するために，実装例を記述する.
- 評価するアーキテクチャー・パターンの前提・制約事項
  パターン適用における代表的な前提・制約を定義することで，パターン選択における迅速な意思決定を支援する.
- アーキテクチャー・パターン選択により向上する主 NFR 項目

(2) トレード・オフ分析結果の記述

各 NFR 項目に対し，該当アーキテクチャー・パターンを選択することにより，良くなる事項および劣化する事項を記述する. またトレード・オフ分析の結果，他の NFR とのトレード・オフにより品質劣化する NFR がある場合，



ソリューション・アーキテクチャー名			05-03 AppL-DB 連携方式						
順番		ソリューション・アーキテクチャー・パターン名	概要	Primary Quality	実行時品質要件（Runtime NFR）				
					A. パフォーマンス＆キャパシティー（Performance and Capacity）	B. 運用・管理容易性（Managedility）	C. 使用性（Usability）	D. 可用性（Availability）	E. セキュ（Secu
A	1	ビジネス・ロジックを AP サーバーに配置	AP サーバー上のアプリケーションにビジネス・ロジックを含み，DRDA(JDBC/SQLJ) 接続にて直接データ・アクセスを行う.	Managebility DB アクセス時の障害を AP サーバーで検知可能なため，問題判断が行いやすい.	［N］SQL 発行の都度，AP サーバー／DB サーバー間に通信が発生するため，パフォーマンスが悪い.	Primary Quality ［P］AP サーバー上で障害検知が可能. ロジックが AP サーバー上のアプリケーションに集約されているため，問題判断がしやすい.	N/A	［P］同期処理であるため，AP サーバー障害時は，AP サーバー側アプリの再実行で対応可能.	
	2	ビジネス・ロジックの一部を AP サーバーに配置	ビジネス・ロジックのデータ・アクセス部分をストアドプロシージャーとして，DB サーバーに登録し，AP サーバー上のアプリケーションからストアドプロシージャーを CALL する.	Performance and Capacity データ・アクセスが DB サーバー上に集約されるため，パフォーマンスが良好.	Primary Quality [P]データ・アクセスを行うロジックをホストに配置することにより，AP サーバー／DB サーバー間の通信回路が削減されるため，パフォーマンスが向上する.	［P］AP サーバー上で障害検知が可能. ［N］ストアドプロシージャー障害時の情報が少ないため，問題判断のためにストレース・ツールが必要.	N/A	［P］同期処理であるため，AP サーバー障害時は，AP サーバー側アプリの再実行で対応可能.	

図 4. NFR トレード・オフ・マトリックス例

品質劣化を抑止するための対策（リスク軽減策）の一例を記述する.

トレード・オフ分析結果の具体例として，図 4 にて NFR トレード・オフ・マトリックス作成の一例を示す. マトリックスの列名は，アーキテクチャー・パターン概要・特性および NFR トレード・オフ分類項目を示し，行名はソリューション・アーキテクチャー・パターンを示している.

分析結果は，アーキテクチャー・パターン選択の上で考慮しなければいけない事項を示すものであり，ここでの考慮事項を考慮しないと，全体の品質劣化を招くリスクがある. 分析結果を，お客様のコンテキストとの Fit&Gap を行うことで，パターン選択の意思決定を支援することを目的としている.

7. おわりに

本論文ではプロジェクトでの代表的な問題領域に対するソリューション･アーキテクチャーをパターン化し,トレード・オフ分析による制約事項・リスクを整理することで，アーキテクチャー意思決定を支援するためのパターン化手法を提示した. 今回提示した手法の特徴は以下の通りである.

- アーキテクチャーの再利用を促進するため，問題領域を定義し，機能要件を実現するアーキテクチャーをボトムアップで洗い出し，パターン化して OM にて文書化した .
- プロジェクトで実施した意思決定を文書化して，アーキテクチャー・パターン候補の抽出に活用した.
- 対象とする問題領域として業務機能だけではなく，システム・ライフサイクルを考慮した網羅的な領域を対象とした.
- アーキテクチャーのトレード・オフ分析を行うための統一フレームワークを定義し，異なる問題領域に属するアーキテクチャーの分析を統一基準で行うことを可能とした.
- アーキテクチャー・パターン適用によるトレード・オフ分析結果をマトリックス化し，パターン適用による副次的影響やリスクを定義し，お客様とのコンテキストを比較し，最適なソリューション選択を可能とする分析フレームワークとした .

本論文で提示した手法を適用して，タスクチームにて，20 程度のソリューション・アーキテクチャー・パターン定義と NFR トレード・オフ・マトリックスを成果物として作成した. また，成果物を提案活動やアーキテクチャー設計作業にて活用し，意志決定におけるリファレンスアーキテ

クチャーとして効果があることを確認した.

今後の検討課題点として，ソリューション・アーキテクチャー・パターンの拡張，アーキテクチャー・パターン選択時における閾値を定義し，品質属性の観点から意思決定を評価する手法の追求を行いたいと考えている.

謝辞

社内タスク活動を通じて多大な助言と御指導をいただいた大嶽隆児氏，山本久好氏にあらためて深謝いたします.

参考文献

［1］内藤裕史:“IBM Global Services Method概要,”ProVISION, No43, pp.38-41（2004）.
［2］菊間裕二: “コンポーネント･モデリング,” ITアーキテクトのためのシステム設計完全ガイド2008, 日経BP社（2007-9）
［3］長井浩:“実行基盤的側面のモデル化（オペレーショナル･モデリング）,” ITアーキテクトのためのシステム設計完全ガイド2008, 日経BP社（2007-9）
［4］山下眞澄: “情報システムの設計思想 第2回要件と制約,” 日経ITプロフェッショナル, pp.106-111,（2004-4）
［5］Eelke Folmer, Jan Bosch, A Pattern Framework for Software Quality Assessment and Tradeoff Analysis tradeoffs, International Journal of Software Engineering and Knowledge Engineering, Volume 17, Issue 4 pp.515-538 June 2007
［6］Rick Kazman:ATAM: Method for Architecture valuation , CMU/SEI-2000-TR-004
［7］Patterns for e-business: http://www-128.ibm.com/developerworks/patterns/
［8］匠目指せ, ITアーキテクト講座 第3回ITアーキテクチャの設計技法: https://www-304.ibm.com/jct03004c/easyaccess/jpgsind/contenttemplate/!!/xmlid=126085

日本アイ・ビー・エム株式会社
ソフトウェア事業ソフトウェアテクニカル・セールス & サービス
System zSW 主任 IT アーキテクト

林 孝太郎 Kotaro Hayashi

［プロフィール］
1989 年，日本 IBM 入社．以来，システム・エンジニアとして複数プロジェクトに参画し，メインフレーム系のシステム基盤のソリューションのグランドデザインを担当.
現在は，ソフトウェア IT アーキテクトとして，システム z のソフトウェアのセールス活動に従事している.
e20894@jp.ibm.com

日本アイ・ビー・エム株式会社
ICP アドバイザリー IT アーキテクト

柿本 達彦 Tatsuhiko Kakimoto

［プロフィール］
1993 年に日本 IBM 入社．UNIX/ メインフレームのネットワーク・スペシャリストとして，さまざまな業種のプロジェクトを経験した後，レガシー刷新，大規模 Web システム構築等のプロジェクトにおいてリード IT アーキテクトを担当．現在、第一線のシステムズ・エンジニアとしてお客様支援活動に従事している.
kakky@jp.ibm.com