ソフトウェア工学 130－14
（２００１．３．２３）

ソフトウェアアーキテクチャに関する考察
～AHPを活用したアーキテクチャ選択～

岸 知二

JAIST / NEC

アーキテクチャ設計においては、複数の評価基準に照らして、複数の代替案の中から適切なものを選択する必要があるが、異なった評価基準に関する評価を総合してアーキテクチャを選択することは困難な問題である。しかしながらアーキテクチャ設計の重要性を考えると、体系だった手法でその選択を行い、その判断の理由を明示化することは有用である。本稿では意思決定の手法のひとつであるAHP(Analytic Hierarchy Process = 階層的意思決定法)をアーキテクチャ選択に適用できるかどうか検討する。

On Software Architecture
- Architectural Selection based on AHP -

Tomoji Kishi

JAIST / NEC

In architectural design, we have to made architectural decision comparing architectural candidates based on multiple evaluation criteria. However, it is difficult to make decision based on different evaluation criteria. Considering the importance of architectural design, it is useful to adopt systematic method to make architectural design and make the reason of decision making explicit. In this paper, we have picked up one of the methods for decision-making, AHP (Analytic Hierarchy Process), and examine the adoptability of the method to the architectural selection.

1　はじめに

ソフトウェアアーキテクチャ（以下アーキテクチャ）[1]は、性能、信頼性、修正容易性といったソフトウェアの様々な特性に対して影響を持っており、その決定はソフトウェア開発において重要である。一般にソフトウェアアーキテクチャを決定するためには、開発されるソフトウェアあるいはそのファミリーに求められる要件を明らかにし、考えられるいくつかのアーキテクチャの候補を評価し、その要件の達成にとって適切と考えられるアーキテクチャを選択しなければならない。こうしたアーキテクチャの評価・選択における課題として、様々な異なった特性に基づいて総合的な判断を行わなければならないこと、評価・選択の時点においては判断に必要な情報が必ずしも十分にそろわないことなどが指摘できる。しかしながらアーキテクチャに関する何らかの判断を行わないと、さらに設計を進めることは困難である。

こうしたアーキテクチャの評価・選択は、最終的にはソフトウェアの性能と信頼性のどちらを重視するかとか、ソフトウェアの性能とソフトウェア開発容易性のどちらをとるかといった価値判断に大きく依存するが、すべての特性面ですぐれたアーキテクチャ候補が存在するといった自明な状況でない限り、どのアーキテクチャの候補を選択するかの判断は容易には行えない。またそのソフトウェア開発に関る他の開発者や、拡張や修正を行う開発者に対して、いったいどういう判断理由でそのアーキテクチャが選択されたかを示すことは重要である。

我々はこうしたアーキテクチャ選択に対して、意思決定の手法である AHP(Analytic Hierarchy Process = 階層的意思決定法)が適用できるかどうか、検討を行っている。AHP は、複数の評価基準に基づいて、代替案の中から特定の案を選択するための手法であり、評価基準間の重要性や、特定の評価基準に関する代替案間の優劣を、ペア比較することにより、最終的な案の選択を行おうとするものである。本稿では、実際に行われたアーキテクチャ選択を事例として、アーキテクチャ選択に対する AHP の適用方法について検討する。

2　アーキテクチャの選択

本章では、アーキテクチャ設計におけるアーキテクチャ選択の課題と現状の手法について述べる。

2.1 アーキテクチャ設計

ソフトウェアアーキテクチャはソフトウェア開発において捉えられる様々なソフトウェア構造を示しており、具体的にはモジュール構造、タスク構造、ファイル構造など、様々な構造を総称するものである。こうした具体的な構造はそれがソフトウェアやソフトウェア開発のどういう局面に関る構造であるかによって、経験的に、実行構造、開発構造、配置構造などといった分類がなされる。ソフトウェアの特性もこれと対応づけて、どういう局面における特性であるかというカテゴライズを行うことができ、それは対応するソフトウェア構造によって影響を受ける。例えば実行時のタスクの構造は、実行時の特性である性能や信頼性と関りを持ち、開発時のモジュール構造は開発時の特性である拡張性や再利用性と関りを持つ。このようにソフトウェアアーキテクチャの決定は、開発されるソフトウェアの特性を決定づける重要な要因のひとつとなる。

一般にアーキテクチャ設計は、開発するソフトウェア、あるいはそのファミリーに対して定義される要件に基づき、考えられる複数の代替案の中から、最も適切と考えられる代替案を選択する作業と捉えることができる。要件は上述した特性と対応づけて捉えることができ、例えば実行時特性としての性能や信頼性、開発時特性としての拡張性や再利用性などに対する様々な要件が定義されうる。また、性能よりも信頼性が重視されるとか、拡張性が性能よりも重視されるというように、それぞれの特性に対する要件は重要度が異なることも一般的である。アーキテクチャ設計においては、こうした様々な要件を考慮しつつ、考えられるアーキテ

クチャ設計上の選択肢がそれぞれの要件に対してどのような特性を持つかを検討し、それらの中から選択を行わなければならない。こうした選択においては、選択肢の特性に対する知識を活用したり、あるいは実証的にその特性を確認するなどの手法を用いたりすることができるが、最終的には重要度に対する価値判断に基づいて判断を行う意思決定の問題と捉えることができる。

2.2 課題

こうしたアーキテクチャの選択においては、以下のような課題を指摘することができる。

- **複数の要件が存在する**：アーキテクチャ選択時には、一般に複数の特性に対する要件を考慮しなければならない。例えばある選択肢は性能にすぐれ、他の選択肢はコストにすぐれるといった場合に、どちらの選択肢を選ぶべきかは自明ではない。仮に性能を重視する方針であるとしても、性能面では僅差であり、コスト面では大きな差がある場合など、容易に意思決定できない状況が多々存在しうる。
- **選択時に十分な情報がない**：アーキテクチャ設計は、ソフトウェア開発の比較的早い段階で行われるため、その時点では選択を行うために必要な情報が十分にないことが通常である。例えば稼働環境の詳細など要求事項そのものがそろっていない場合もあるし、最終的な特性などある程度実証的に確認しなければ分からない場合もある。しかしながらアーキテクチャは開発の方向づけを行うものであるから、情報が不十分であっても開発の何らかのタイミングでその判断を行わなければならない。
- **体系だった意思決定手法が必要である**：アーキテクチャ選択は何らかの価値観に基づいて行われるものであり、またその時点で十分な情報がないことが通常であるから、その時点で結果を保証できる結論を導くことは不可能である。しかしながら設計者は、手持ちの情報に基づいてできるだけ適切と考えられる判断を行うことが求められるし、その判断理由を開発の関係者に説明し、共有することが望まれる。判断の根拠が明示化されていれば、新しい情報が手に入った時点でその判断の妥当性を再考することも容易となる。

2.3 既存の手法

アーキテクチャの選択を行う際の体系だった手法の提案は少ないが、アーキテクチャを評価するための以下の技術を選択に際して活用することができる。

スコアリング：アーキテクチャをいくつかの評価基準に照らして評価し、それによって得点づけする手法である。得点は評価基準における相対的な優劣に基づく場合もあるし、評価基準に対応する測度の絶対的な数値に基づく場合も考えられる。この手法は複数の評価尺度に照らして評価を行うことができるが、異なった評価尺度の間で得点をどのように比較すべきか等、運用上難しい問題を含んでいる。本稿で扱う AHP も、このスコアリングの一種と見なすことができる。

モデルの解析やシミュレーション：アーキテクチャをモデル化し、そのモデル上で、対応するアーキテクチャの特性を検討する手法である。例えば性能モデルに照らして、性能を評価するなどがそれに対応する。またモデルをシミュレーションすることにより、評価を行うことも行われる。こうした手法はモデルが妥当であれば、実用性の高い結果が得られるが、一般にモデルは特定の特性を捕らえるために作られるため、複数の特性を総合的に扱ったり、複数の特性間の重みづけを行ったりする目的には必ずしも適していない。

プロトタイピング：実際にアーキテクチャに沿ってソフトウェアを作成し評価を行う実証的な手法であり、適切な範囲のプロトタイプを作成することができれば、様々な特性の検討に使うことができる。また実際に動かすなどして確認するため、実態に近い評価ができることが期待される。しかしながら仮に部分的なプロトタイプであっても、その作成と評価には多くの

コストがかかるため、様々な代替案をすべてこの手法で評価することは現実的には不可能である。

またアーキテクチャ選択を行うための手法としては、シナリオに基づくスコアリングを行うSAAM[3]や、品質特性を類推するモデルを活用するATAM[4]などがある。このうちATAMはアーキテクチャとその特性を理解するための手法であり、特に複数の特性とその間のトレードオフを検討することなどに特徴を持つ。

3 AHPのアーキテクチャ選択への適用と試行

本章ではアーキテクチャ選択へ、意思決定の手法のひとつであるAHPを適用した試行結果を述べる。

3.1 AHP

AHP[5][6]は意思決定を行う際の手法のひとつである。AHPにおける意思決定とは、いくつかの代替案の中からある目標に適したものを、いくつかの評価基準に照らして選択する問題として捉えられる。自明な状況をのぞいて、どの代替案が適しているかを判断することは一般に困難であるので、AHPでは評価基準の重みづけと、個々の評価基準に対する選択肢の重みづけを行い、それに基づいて最も適した代替案を判断する。すなわち、まず目標に対して個々の評価基準がどの程度の重みで関るかを決定し、次に個々の評価基準毎に各代替案がどの程度の優劣を持つかを示す重みを決定し、それらを総合して、目標に対してどの代替案が適しているかを決定する。

AHPの特徴は、こうした重みづけを、一対比較によって行う点である。例えば目標に対して各評価基準がどの程度の重みで関るかをいきなり決めることが困難であるため、評価基準のすべてのペアに対して、どちらが目標に対してより重要性を持つかを比較する。こうした一対比較に基づいて全体の重みを計算する手法がとられる。

3.2 試行

AHPのアーキテクチャ設計への適用性を検討するために、過去に行われたアーキテクチャ選択の事例[2]を題材にして、AHP適用の試行を行った。

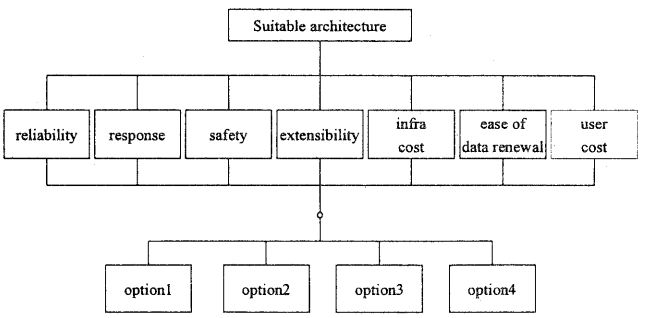


図1：目標、評価基準、代替案

この事例では、ある機能を実現するために考えられる複数のアーキテクチャの代替案を、複数の評価基準に基づいて評価し、選択している。これをAHPで通常用いられる階層図として表現すると図1のようになる。ここでは4つの代替案と、7つの評価基準とが示されている。

まずこの 7 つの評価基準が最終的な目標に対して、どれだけ重要性を持つかを評価する。通常のスコアリングではこの重要性を重みとして与えるが、AHP ではいきなり重みを検討する代わりに、評価基準を一対比較して相対的な重要性を与える。表 1 はこの一対比較の結果を示すもので、各欄の数字は、その行に対応する評価基準が、その列に対応する評価基準にどれだけ重要かを、相対的に示している。ここでは同じであれば 1、より重要であれば 3、さらに重要であれば 5 というように、1 から 9 までの数字で示す。逆に重要性が低ければ 1/3,から 1/9 までの逆数が用いられる。例えば信頼性は応答性とは同様の重要度(1)であり、安全性よりはより重要(3)であり、逆にユーザコストよりは重要度が低い(1/3)。このようにすべてのペアに対して一対比較を行った結果が表である。こうした一対比較ができると、それぞれの評価基準の重みを求めることができる。表の右側に示す重みはその計算結果である。

表 1 ：評価基準の一対比較と重み

	reliability	response	safety	extensibity	infra cost	of data rer	user cost	weight
reliability	1	1	3	1	1	1/3	1/3	0.101
response		1	3	1	1	1/3	1/3	0.101
safety			1	1/3	1/3	1/5	1/5	0.041
extensibility				1	1	1/3	1/3	0.101
infra cost					1	1/3	1/3	0.101
ease of data renewal						1	1	0.279

次に 7 つの評価基準毎に、各代替案のどれがすぐれているか、その優劣を重みとして表現する。ここでもいきなり重みを検討するのではなく、一対比較を行うことでその重みを計算することができる。表 2 は信頼性に対する一対比較の結果であり、例えば代替案 1 は、代替案 2 より相当に信頼性が低く(1/7)、代替案 3, 4 と比べてもかなり低い(1/5)と判断されている。評価基準と同様に、これから信頼性に関する各代替案の優劣が重みとして計算される。ここでは信頼性に関しては代替案 2 が最も適していることを示している。他の 6 つの評価基準に対しても同様の一対比較と計算を行う。

表 2 : 信頼性に関する代替案の一対比較

	option1	option2	option3	option4	weight
option1	1	1/7	1/5	1/5	0.052
option2		1	3	3	0.528
option3			1	1	0.21
option4				1	0.21

評価基準の重みと、評価基準毎の代替案の重みが求まれば、それから 7 つの評価基準を相当した重みを求めることができる。以下は最終的な値であり、ここでは代替案 4 がすぐれていることが示されている。

表 3 ：最終的な重み

	weight
option 1	0.18
option 2	0.3
option 3	0.205
opton 4	0.319

4 議論

本章では試行結果に対する議論を行う。

4.1 AHP のメリット

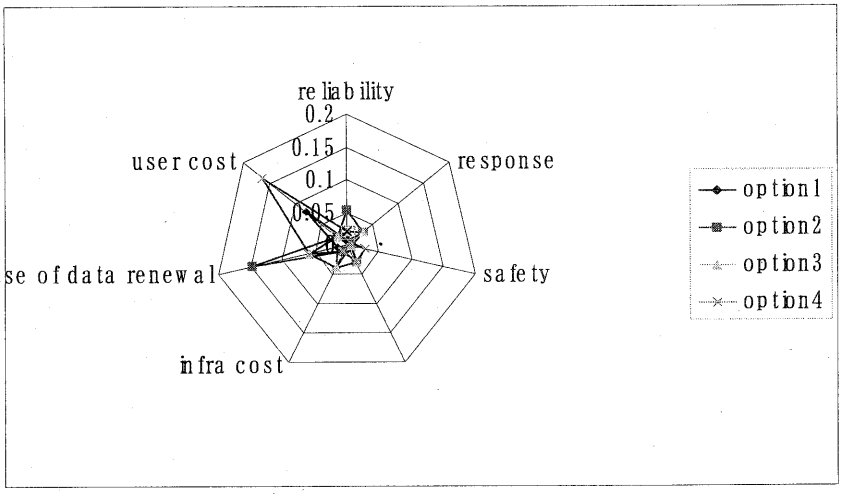
AHP は、あくまで意思決定のための手法であり、そこで求められた結果の正しさを保証するものではない。AHP のメリットは、今回の例題のように複数の評価基準が存在し、どの代替案が妥当であるかをいきなり判断することが困難な場合に、それをより判断が容易な小さな判断単位（一対比較の単位）に分割し、それを積み上げることで最終的な判断へと導けることにある。また個々の一対比較の結果に若干の矛盾があったとしても、全体として一定の整合性があれば全体の判断結果に影響しないため、個々の判断の間違いがある程度許容される点もメリットといえる。

4.2 結果の分析

AHP では一対比較を行うことで最終的な結果（代替案の重み）が導かれる。従ってその最終的な結果が、判断者自身として納得のいく判断結果であることを納得できることが必要である。このことは、開発に関る他の人に判断結果を説明する上でも、重要なことである。今回の例題においては、それを確認するために、評価基準のポートフォリオを整理することで行った。

図２は個々の代替案が、どの評価基準に対してすぐれているかをレーダーチャートで示したものである。これから、重要度の高いユーザコストとデータ更新の容易性というふたつの評価基準において、最終的に選ばれた代替案 4 はユーザコスト面では勝れているものの、データ更新の容易性では代替案 2 の方が勝れていることなどが理解できる。

図２：結果に対する各評価基準の影響度



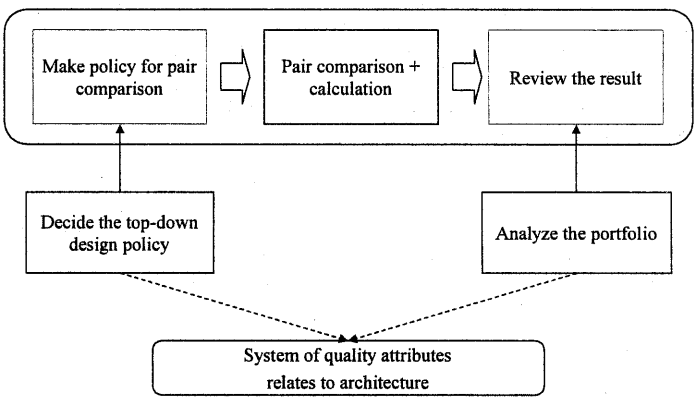
同様に、この評価基準を、その評価基準に関る特性がソフトウェアのどの局面に関る特性であるかという観点からカテゴライズしたものである。ここではこれらの評価基準を、実行時特性に関るもの（信頼性、応答性、安全性）、開発時特性に関るもの（拡張性、インフラコスト）、これから、代替案 2 は実行特性に、代替案 4 は運用特性にすぐれている等がわかる。また、この評価基準が、どのステークホルダーに関る価値に関係するものであるかを検討した。ここではユーザと開発者という二つのステークホルダーを検討し、例えば拡張性インフラコス

ト、データ更新の容易 SEI は、開発者に対する価値に関するものとして検討した。このように評価結果のポートフォリオを検討することは、その結果がどのような要因によって導かれたかを自身で理解する際に有用であると考えられる。

4.3 適用のフレームワーク

以上を踏まえ、AHP をアーキテクチャ選択に適用する際のフレームワークとして、以下を検討した。意思決定の手法として、手法として体系だっていること、導かれた結果が自分自身として納得のいく結果であること、またそれを他の人に対して説明できること、といったことを考えると、以下のように適用することが有用であると考えている(図 3)

図３：アーキテクチャ選択への AHP 適用のプロセス



- **設計方針を評価基準に照らして説明すること**：一対比較は比較的容易ではあるが、その判断の方針に一貫性がなければ納得性が低い。そういう意味から、評価基準に対する一対比較のポリシーをある程度トップダウンに認識することが重要と考える。すなわち個々の判断が、設計方針そのものと整合することを確認できることが望ましい。例えば設計方針が「使い勝手の重視」であれば、実行時特性に関る評価基準が、開発時特性に関る評価基準より重視されるし、「開発期間の短縮」であれば、開発時特性に関る評価基準が重視されるであろう。
- **得られた結果の定性的な傾向を理解し納得すること**：最終的に計算される代替案の重みは、自己の一対比較の帰結ではあるが、それが自身の判断と背反するものでないかどうかを、定性的な傾向として確認することは重要であると考える。そのためには、上述したポートフォリオの確認などが有効であると思われる。この際に、アーキテクチャの特性が、実行時、開発時といったどの局面に関るものであるかという視点や、その特性がどのステークホルダーに関るものであるかという視点などを活用することが考えられる。
- **分析結果等に基づき必要に応じて評価のプロセスを繰り返すこと**：一対比較を行い最終的な重みを計算すればそれで終わりという一方向のプロセスである必要はなく、それを繰り返しながら妥当と考えられる意思決定へと導く道具である。そういう観点か

ら、ポートフォリオの分析や、感度分析行いながら、自身にも他人にも妥当性が納得しやすい結論を検討するプロセスが重要であろう。

5　おわりに

以上、アーキテクチャ選択にAHPを試行し、その適用に関する考察を行った。アーキテクチャ選択の問題は、判断に必要な情報が必ずしもそろった時点で行われるわけではなく、また、異なった評価基準に対する重みづけは価値観に依存するものであるため、その判断はあくまで恣意的なものであるとの主張もあり得る。しかしながら、重要性の高い開発を行っている際には、その時点での情報に基づいて最も妥当と考えられる選択を行うように検討することが重要であるとともに、その判断理由を関る人に説明可能であることが、開発を進める上でも大切である。また判断理由がトレース可能な状態で明示化されることにより、新たな情報が得られたときにその判断結果への影響を検討することも可能となる。従って設計における意思決定の体系化は、実際のアーキテクチャ設計においても必要性の高いものであると考える。今回はAHPという手法を、実際に行われたアーキテクチャ選択の実例に照らして、試行した。また適切な適用方法について考察を加えた。意思決定の手法は他にも存在するため、アーキテクチャ設計においては、どのような意思決定が適しているか検討するとともに、適用方法の妥当性について確認をする必要があろう。

謝辞

本研究に対して大変有益なご指導を頂きました北陸先端科学技術大学院大学の落水浩一郎教授に深謝いたします。

参考文献

[1] Bass, L., et.al.: Software Architecture in Practice, Addison-Wesley, 1998.
[2] 自動車走向電子技術協会: ITS 車載システムアーキテクチャの研究, 1998.
[3]Kazman, R., et.al.:SAAM: A Method for Analyzing the Properrties of Software Architectures, Proceedings of the 16th ICSE, 1994.
[4]Kazman, R., et.al.:The Architecture Tradeoff Analysis Method, SEI, 1998.
[5]木下: 孫氏の兵法の数学モデル, 講談社, 1998.
[6]刀麻: ゲーム感覚意思決定法, 日科技連, 1986.