ソフトウェア工学 115−11
（1997. 8. 1）

ミニチュートリアル
ソフトウェアアーキテクチャの今日的意義

岸 知二、野田夏子

NECマイコンソフト開発環境研究所

ソフトウェアアーキテクチャについて、その定義や意義を概観した上で、どうして今ソフトウェアアーキテクチャが注目されているのか、その意義を考える。

Mini Tutorial
Why Software Architecture is being paid attention now?

Tomoji Kishi, Natsuko Noda

Microcomputer Software Development Laboratory
NEC Corporation

In this paper, we will show the definition and the significance of software architecture, and argue why software architecture is widely noticed now.

1 はじめに

近年ソフトウェアアーキテクチャ（以下アーキテクチャ）という用語がよく使われている。アーキテクチャが大切だと言われて特に反論する人はいないかもしれないが、どうして今アーキテクチャなのだろうか？今アーキテクチャを議論することに何か特別の意味があるのだろうか？

本ミニチュートリアルではアーキテクチャに関するそうした疑問を踏まえ、アーキテクチャに関わる状況について簡単に解説する。なお現時点ではアーキテクチャに関して安定した体系化がなされているわけではなく、多様な目的、視点、アプローチで研究が進められている。本稿はそれらを網羅的に調査し整理するものではない。客観性に留意はしているが、この分野に対する一つの視点だと理解して頂きたい。

2 アーキテクチャとは何か？

現時点ではアーキテクチャについて誰もが認める共通の定義は存在せず、いくつかの定義が提案されているが、それがソフトウェアの静的・動的な構造に関わるものである点ではおおむね共通している。ここではひとつの例としてBuschmannの定義[2]を紹介する。

「ソフトウェアアーキテクチャとは、ソフトウェアのサブシステムやコンポーネント、及びそれらの関係の記述である。サブシステムやコンポーネントはソフトウェアシステムの機能的あるいは非機能的な特性を示すために様々なビューから記述されることが多い。システムのソフトウェアアーキテクチャは成果物である。これはソフトウェア設計行為の結果である。」

しかしながらこうした定義は広範な内容を含んでおり、概念の解像度が粗い。そこでアーキテクチャをいくつかに分類する試みもなされている。Soni[9]は開発に用いられているアーキテクチャが、概念アーキテクチャ、モジュール相互接続アーキテクチャ、実行アーキテクチャ、コードアーキテクチャに分けられることを指摘している。またKruchten[6]はアーキテクチャを利用する観点から、論理ビュー、プロセスビュー、物理ビュー、開発ビューに分類することを提案している。

なお繰り返し使われる、もしくは繰り返し発見されるアーキテクチャ中の構造をアーキテクチャパターンと呼ぶ。アーキテクチャパターンは、設計上の何らかの問題に対して適用されるアーキテクチャ中の構造である。

3 アーキテクチャは有用か？

アーキテクチャを議論することにどのような実用的な価値があるのだろうか。本章ではアーキテクチャがなぜ必要かを述べる。

- システム設計：大きなシステムをいきなり設計することはできない。システムの中に適切な構造を定義することはシステム設計の基本である。ここには抽象化、カプセル化、強度と結合度等の技術が関係している。一旦システムが作られるとその基本構造、すなわちアーキテクチャを変更することは困難となるのでアーキテクチャ構築はシステム設計にとって重要な意味を持つ。

- 拡張性・カスタマイズ性：拡張性やカスタマイズ容易性はシステムの重要な特性であり、また開発や維持のコストに大きな影響を及ぼすが、これらの特性もアーキテクチャによって大きく影響される。逆にアーキテクチャに反したりアーキテクチャに対して無考慮でいることにより、アーキテクチャが風化したりふらついたりすることも指摘されている[7]。

- 再利用：再利用はソフトウェア開発における重要な課題である。再利用には多くの技術的困難が伴うが、ソフトウェアが肥大化し開発期間が短くなっている現実のソフトウェア開発において、再利用を避けて通ることはほとんど不可能な状況となっている。こうした再利用資産は、それを使う潜在的なアプリケーションのアーキテクチャを想定して作られるし、再利用資産が想定するアーキテク

チャとの不整合が再利用を困難にすることも指摘されている[4]。

- 非機能的特性：分析・設計技法の進展により、対象世界の構造を捉え、拡張性や再利用性に優れた論理構造を見いだすための技法は着実に進歩しているが、性能，信頼性，安全性といった非機能的特性を達成するための技法は必ずしも整備されていない。しかしながらアーキテクチャがそうした特性に強く関連していることが認知されてきており、そうした観点からのアーキテクチャへのアプローチも進められている[1][5]。

- リファレンスモデル：具体的なシステムの設計や構築とはやや局面が異なるが、アーキテクチャはリファレンスモデルとしての役割も果たす。例えば OMA などは、概念レベルのアーキテクチャではあるが、リファレンスモデルとして広く使われている。このようにアーキテクチャは、概念レベルであれ実装レベルであれ、共通概念の形成に貢献している。

4 今なぜアーキテクチャなのか？

前章で指摘したようにアーキテクチャはソフトウェア開発において重要な役割を果たしていることが認識できる。しかしながらそうした問題は昔から重要だったわけであり、今特にアーキテクチャを取りざたする理由があるのだろうか。システム設計の問題ひとつを取り上げても、それはソフトウェア工学の歴史そのものであり、古くは複合設計におけるモジュール強度や結合度の議論、モジュール相互接続言語(MIL)、から最近の技法やパターンの議論に至るまで、常にアーキテクチャは重要な問題で有り続けてきたと考えられる。

しかしながら今あえてアーキテクチャが取り上げられる理由としては、以下が考えられる。

- 環境の変化と新しいシステムの台頭：ネットワーク環境の急速な広がり、マルチメディア技術の進展、ダウンサイジングなど、システムを取り巻く環境は急激に動いており、その中で新しいシステムが次々と作られるダイナミックな状況になっている。その時点のニーズやシーズにとって最も適切なアーキテクチャを早く新たに構築することが求められているが、旧来のアーキテクチャの蓄積や考え方の延長線上では捕らえがたくなってきている。

- 再利用資産の巨大化：フレームワークやコンポーネントウェアなど、巨大な、あるいは疎粒度の再利用資産が作られ、使われるようになってきている。単機能のライブラリとは異なり、こうした大きな再利用資産はアーキテクチャへの明示的な意識がなければ構築することも利用することもできなくなってきている。

- オープン化の進展：特定のメーカーの技術のみに閉じた形でシステムを構築することはあり得なくなっており、複数のベンダの提供する技術を組み合わせてシステムを構築しなければならなくなっている。こうした状況の中で複数のベンダやユーザ間でアーキテクチャを共有することが必要となってきている。あるいは自らの製品を業界に広く受け入れてもらうためには、製品の持つアーキテクチャを公開し、そのアーキテクチャに沿ったソフトウェア開発をしてくれる仲間作りをすることが必須となってきている。

- モデリング技術の進化・浸透：構造化技法やオブジェクト技法等が進化し、実開発への適用も進んでいる。それによりソフトウェア開発のためのモデリング技術が進化するとともに、適用技術も蓄積されている。従来は素朴な形でしか取り扱えなかったアーキテクチャを、こうしたモデリングの技術を使ってより適確な形で取り扱うことができるようになっている。

- 個から構造への動き：デザインパターン[3]の浸透に端的に示されるように、ソフトウェアの構造の重要性が強く認識されるようになってきている。それらは

アーキテクチャと強い関連を持っており、アーキテクチャ研究を刺激している。そうした中でアーキテクチャスタイルやアーキテクチャパターンの蓄積も進められてきている。

5 これからどうなるのか？

パターン等のアイデアはあるものの、アーキテクチャに関しては多くの研究課題がある。現在アーキテクチャに関して以下のような研究が進められている[8]。

- 記述方法・記述言語：アーキテクチャの記述に適した記法や言語は、開発者間のコミュニケーションにとっても、あるいはアーキテクチャの分析等にとっても重要である。
- カタログ化：実際のソフトウェア開発の中で発見された様々なアーキテクチャや、それを用いる基準や根拠ををカタログ化することがなされている。
- 形式的な基礎：アーキテクチャの分析等を行う際の背景として、記述されたアーキテクチャに対して形式的な意味を定義するなどの研究も進められている。
- 構築手法：アーキテクチャの構築をし、あるいはそれによってソフトウェアの開発を行う手法やガイドラインの確立が求められ、なされている。
- 支援環境：アーキテクチャの分析、構築、あるいはそれに基づく実装を支援する環境についての研究も行われている。

こうした研究の個々の進展や成果も重要だが、前章で指摘したような新しい時代において、アーキテクチャそのものがどうなっていくのかを考えることも重要である。

一昔前には個々のアプリケーションに閉じたアーキテクチャのみを考えればよかったが、現在は技術のオープン化やネットワーク技術の進展により多くの人が共有すべきアーキテクチャが存在し、その上に個々のアプリケーションのアーキテクチャが構築される形態になっている。一方アーキテクチャというものがソフトウェアを重たく固くしているという見方もありうる。今後ソフトウェアはどういうアーキテクチャになっていくのか、あるいはアーキテクチャそのものの考え方がどう変化していくのか、興味の持たれる点であろう。

6 おわりに

以上ごく簡単ではあるが、アーキテクチャに関する状況をかいつまんで説明した。

参考文献

[1] Buhr, R.J., et.al.: *Use CASE Maps for Object-Oriented Systems*, Prentice-Hall, (1996).

[2] Buschmann, F., et.al.: *Pattern-Oriented Software Architecture - A System of Patterns*, Wiley, (1996).

[3] Gamma, E., et.al.: *Design Patterns - Elements of Reusable Object-Oriented Software*, Addison-Wesley, (1995).

[4] Garlann, D., et.al.: *Architectural Mismatch: Why Reuse Is So Hard*, IEEE Software, Nov. 1995, (1995).

[5] 岸, 他: ソフトウェアアーキテクチャに基づく安全性ソフトウェアの開発, ソフトウェア工学研究会, Vol.96, No.112, (1996).

[6] Kruchten, P.B., et.al: *The 4+1 View Model of Architecture*, IEEE Software, Nov. 1995, (1995).

[7] Perry,D.E., et.al: *Foundation for the Study of Software Architecture*, SOFTWARE ENGINEERINGNOTES, vol.17, no.4, (1992).

[8] Shaw, M., et.al.: *Software Architecture, Perspectives on an Emerging Discipline*, Prentice-Hall, (1996).

[9] Soni, D., et.al.: *Software Architecture in Industrial Application*, Prod. of 17th ICSE, (1995).