ソフトウェア工学 122－1
（1999．3．18）

変更容易なソフトウエア構成のための アーキテクチャの選択法

島村 敦司　徳田 雄洋

東京工業大学 大学院 計算工学専攻

適切なアーキテクチャを選択することが、作成及び変更の容易なソフトウエア構成において重要である。しかしながら、与えられた問題に対して、取り得る可能なアーキテクチャの中から最適なものを選択する方法論は、未だ確立していないのが現状である。本論文では、与えられた問題のそれぞれのアーキテクチャでの構成について、複雑さを公平に評価することで、適切と思われるアーキテクチャを選択する方法論を提案する。この方法論は、まず問題からアーキテクチャとは独立な名詞・動詞グラフを作り、次にこのグラフからそれぞれのアーキテクチャ用のグラフを構成し、簡易的な定量的尺度で比較する。これにより、異なるアーキテクチャでも、その構成を共通の要素からなるグラフで描くことが出来、アーキテクチャの選択を公平に行うことができる。

A Selection Method of Software Architectures for Constructing Easily Modifiable Software Systems.

Atsushi Shimamura　Takehiro Tokuda

Dept. of Computer Science, Tokyo Inst. of Tech.

Selecting an appropriate software architecture for a given problem is crucial to construct software systems whose initial construction cost and modification cost are both inexpensive. Unfortunately there exists no methodology for such selection yet. We propose a software architecture selection methodology using evaluation of the complexity of possible software architectures for the same problem. First we construct a noun-verb graph for a given problem. Then we transform the noun-verb graph into graphs representing possible software architectures. We measure the complexity of software architectures and choose the simplest one.

1　はじめに

大規模なソフトウエアを作成する際には、そのソフトウエアをいくつかのモジュールに分けて開発する。この時のモジュールの分割やその接続方式がソフトウエアアーキテクチャであるが、特にShaw と Garlan は “Software Architecuture”[17]の中で、同一の問題に対して複数のアーキテクチャによる解決法が存在すること、ソフトウエアアーキテクチャによって作成や変更の複雑さが変化することを指摘し、問題に対して適切なソフトウエアアーキテクチャの選択が重要であることを述べている。

そこでLung[13]らは、問題を解決するためのソフトウエアアーキテクチャの図を描き、そこから、モジュールやリンクの数を種類毎に数えることで、ソフトウエアアーキテクチャの複雑さ、変更に対する強さを評価した。

しかし、Lungらがこれらの数値と共に主観的な評価を加えているように、全く異なるソフトウエアアーキテクチャでは、これらの数値の直接の比較で複雑さ、変更に対する強さを計ることは困難である。それは、異なるソフトウエアアーキテクチャで描いた図では、表現されるものが全く異なるためである。よって、このLungらの方法は、定量的な評価を用いて異なるソフトウエアアーキテクチャから適切なものを選ぶ方法としては十分ではない。

本論文では、主要な4つのソフトウエアアーキテクチャにおいて、その構成のグラフを統一的に描き、その要素を数える事で、それぞれのモジュールを構成する際に、そのモジュールが他のモジュールに対して持たなければならない情報の量を定量的に計り、適切なソフトウエアアーキテクチャを選択する方法を提案する。

2 主要なソフトウエアアーキテクチャ

ソフトウエアアーキテクチャには様々な型が考えられるが、ここでは主要な4つのアーキテクチャ、すなわちメインプログラム・サブルーチン型、オブジェクト指向型、パイプ・フィルタ型、そして非明示的起動型を取り上げる [11, 7, 1, 8]。

- メインプログラム・サブルーチン型
  メインプログラム・サブルーチン型におけるモジュールは、全体のモジュールの起動の順序を制御するメインプログラムと、主要な動作毎に分けられたサブルーチンであり、それぞれのモジュールは相互の呼び出しにより起動される。また、データ構造は共有データとして扱われ、それぞれのモジュールが直接読み込み、書き込みを行う。

- オブジェクト指向型
  オブジェクト指向型においてモジュールは、データ構造とそれを直接操作する動作を単位として分けられる。それぞれのモジュールは相手のモジュールの中の動作（メソド）を起動することで全体の制御が行われる。またデータ構造はモジュールの内部に隠蔽されており、同じモジュール内からは直接参照できるが、他のモジュールからはデータを保持するモジュールのメソドを介して参照する。

- パイプ・フィルタ型
  パイプ・フィルタ型においてモジュールは、入力と出力を持つフィルタとして構成される。それぞれのフィルタは入力のデータを読み込みそれを内部で計算した結果を出力する。このフィルタを相互にパイプで接続することにより全体の構成を行い、データフロー計算が実行される。

- 非明示的起動型
  非明示的起動型ではモジュールはメインプログラム・サブルーチン型と同様に主要な動作ごとに分けられ、またデータ構造は共有データとしてそれぞれのモジュールが直接に読み込み、書き込みを行う。しかし、メインプログラム・サブルーチン型と違い、このアーキテクチャにおける動作の制御はイベントによって行われる。すなわち、それぞれのモジュールは特定のイベントが発生した時に起動し、逆に他のモジュールの動作を必要とする時にはイベントを発生する。

3 アーキテクチャの選択

3.1 名詞・動詞グラフ

与えられた問題を異なったソフトウエアアーキテクチャで構成したものを比較するために、名詞・動詞グラフを定義する。この名詞・動詞グラフは問題の構造を表すもので、ソフトウエアアーキテクチャとは独立に表現されるものである。

名詞・動詞グラフは無向グラフであり、次のように作成する

- 問題の記述からデータを表す名詞と、動作を表す動詞を取り出す
- 関連のある名詞と動詞をリンクで結ぶ

POSシステム問題について、この名詞・動詞グラフを描くと、図1のようになる。POSシステム問題は、商店でのレジでの買物の計算を模したも

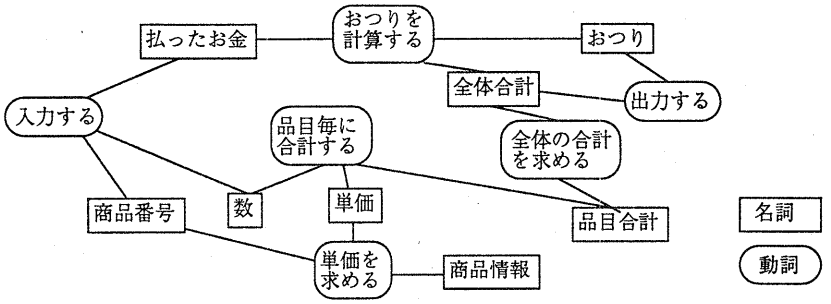

図 1: 名詞・動詞グラフ

ので、客の買った商品の番号や数から、合計金額などを随時計算し、表示するものである。

3.2 名詞・動詞グラフのソフトウエアアーキテクチャへの構成

問題をあるソフトウエアアーキテクチャで実現するということは、ここで選んだ名詞、動詞をそれぞれのソフトウエアアーキテクチャの形式のモジュールに分割し、ソフトウエアアーキテクチャによる制御の機構を与えることである。よって、この名詞・動詞グラフを分割しソフトウエアアーキテクチャによる制御の線を加えることで、それぞれのアーキテクチャによる問題の構造を表すグラフを作ることが出来る。

メインプログラム・サブルーチン型、オブジェクト指向型、非明示的起動型、パイプ・フィルタ型について名詞や動詞を分割し、4種類のソフトウエアアーキテクチャの名詞・動詞グラフを描く方法を示す。また、このソフトウエアアーキテクチャへの分割は、それぞれのソフトウエアアーキテクチャで公平に行われるように、各アーキテクチャ毎に既に存在する個別の方法論は使わずに行う。

- メインプログラム・サブルーチン型
  - 必要ならば全体の制御を表す動詞を置く
  - 関連の強い動詞をまとめて1つのモジュールとする。
  - 名詞を共有データ、モジュールのローカル変数、呼び出しの引数、返り値として配置する
  - 動詞の呼び出し関係を調べ、リンクを作る

POSシステムの名詞・動詞グラフをメインプログラム・サブルーチン型に構成すると図2となる。

- オブジェクト指向型
  - 必要ならば全体の制御を表す動詞を置く
  - 動詞とその動詞が直接定義、参照する名詞をモジュールとして配置する
  - 必要な動詞の呼び出しを調べリンクを作る

POSシステムの名詞・動詞グラフをオブジェクト指向型に構成すると図3となる。

- パイプ・フィルタ型
  - 動詞一つ、もしくは関連の強い動詞をまとめて1つのモジュールとする。このモジュールがフィルタとなる
  - フィルタを計算の順に並べその間をパイプとする
  - 名詞はパイプに配置する。この時一つの名詞ごとに前後のフィルタと接続するリンクを作り、その上へ名詞を配置する

POSシステムの名詞・動詞グラフをパイプ・フィルタ型に構成すると図4となる。

- 非明示的起動型

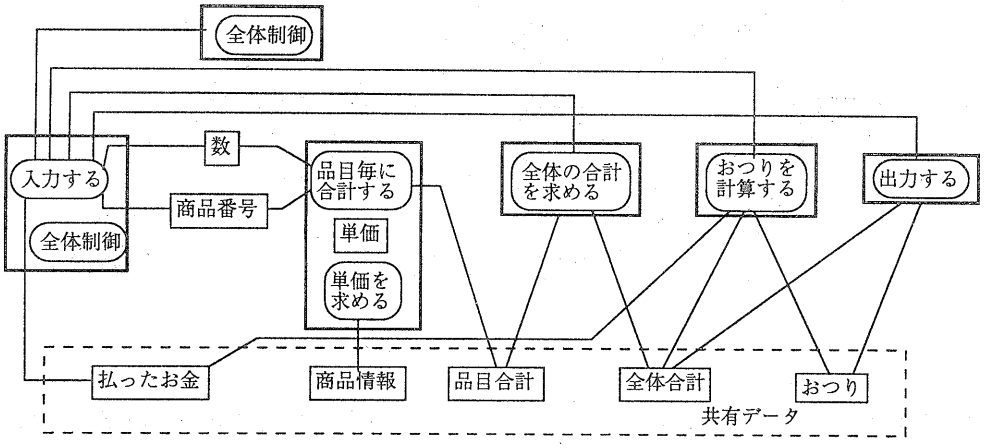

図 2: メインプログラム・サブルーチン型に構成した名詞・動詞グラフ

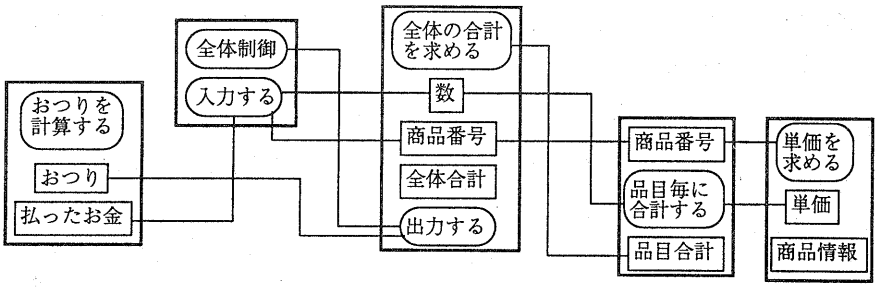

図 3: オブジェクト指向型に構成した名詞・動詞グラフ

- 必要ならば全体の制御を表す動詞を置く
- 関連の強い動詞をまとめて 1 つのモジュールとする。
- イベントを表す名詞を置く
- イベントを発生させる動詞、イベントを受けとる動詞と、上のイベントを表す名詞をリンクでつなぐ
- 名詞は共有データ、あるいはモジュールのローカル変数として配置する

POS システムの名詞・動詞グラフを非明示的起動型に構成すると図 5 となる。

このように、問題から得られた名詞・動詞グラフをそれぞれのソフトウエアアーキテクチャに構成すると、元のグラフにあったリンクは、1 つのモジュールの中のみのリンクと、モジュール間にまたがるリンクに別れる。図 2 から図 5 までは、このうちのモジュールにまたがるリンクのみを描いた。これは、このモジュールにまたがるリンクが、モジュール間での通信に必要な知識を表すためである。

3.3 名詞・動詞グラフの定量的評価

名詞・動詞グラフでは、ソフトウエアアーキテクチャとは無関係なアルゴリズムの複雑さは 1 つの動詞として括られ、複雑なデータ構造を必要とする部分も 1 つの名詞として括られている。よって、名詞・動詞グラフの接続関係は問題の各部分の複雑さを除いた、全体の構成の複雑さを表し、ソフトウエアアーキテクチャの型へ分割した名詞・動詞グラフは、そのソフトウエアアーキテクチャでの構成の複雑さを表している。

そこで、この名詞・動詞グラフを評価すること

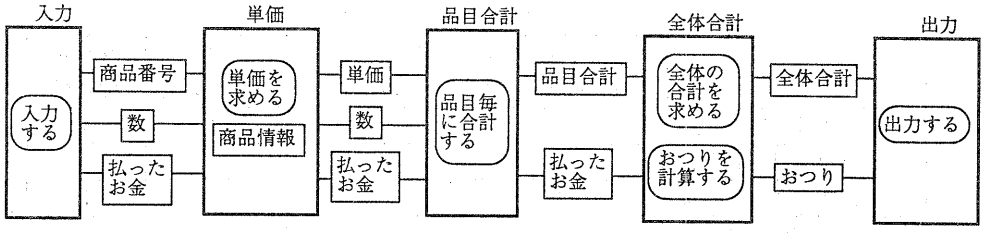

図 4: パイプ・フィルタ型に構成した名詞・動詞グラフ

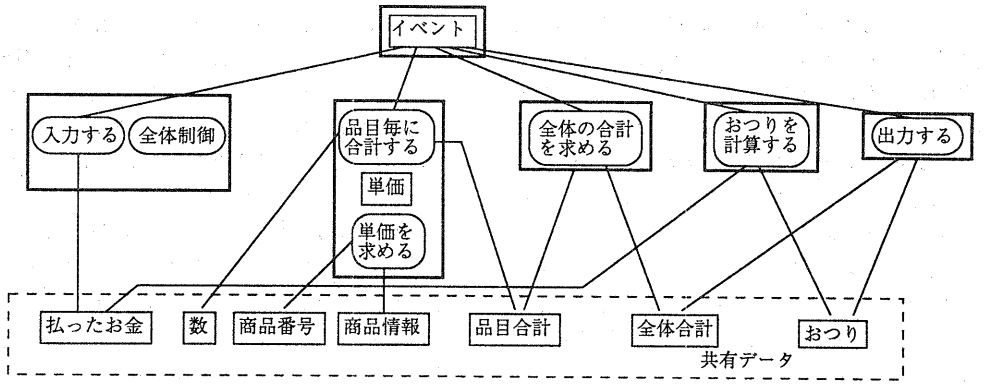

図 5: 非明示的起動型に構成した名詞・動詞グラフ

で、あるソフトウエアアーキテクチャによる問題の解法の複雑さを、特にソフトウエアの構成や変更を行う時の複雑さや必要な手間で評価する。

評価に用いる尺度として、次の 4 つを用いる。このうち、1、2、4 については始めの状態での数値と、問題に変更がある場合にはその変更による名詞・動詞グラフの変化分を足す。3 は問題の変更がある時に、変更する必要のあるモジュールの数を、問題の変更毎に加える。

1. モジュールの外に現れている名詞や動詞の間のリンクの数と制御のためのリンク（合わせて名詞・動詞リンクとする）の数。
2. 1 つのモジュールに含まれる動詞の数
3. 変更を要するモジュールの数
4. モジュール間のつながりの数（2 つのモジュールに含まれる名詞や動詞の間に名詞・動詞リンクがある場合には合わせて 1 つと数える）

1 はモジュール同士がどの程度強く結びついているかを示している。つまり、この 1 の数字が大きいということは、他のモジュールとの結びつきが強く、そのモジュールを構成する時や変更する時に、他のモジュールについての知識を多く知らなければならないということである。2 は 1 つのモジュールの複雑さを示す。3 と 4 はモジュール同士の構成の複雑さを調べるためのものである。

これらの尺度において、比較する各ソフトウエアアーキテクチャのグラフは、全て 1 つの名詞・動詞グラフから作られている。すなわち問題を表す共通の名詞、動詞とその間のリンクから出来ているので、ソフトウエアアーキテクチャの間の数字の比較が意味のあるものとなっている。

アーキテクチャの選択のための評価法の目標は、その構成において、あるモジュールが他のモジュールについて知らなければならない知識の量ではかることとする。よって、アーキテクチャの選択は、1 によって行なう。すなわち、1 の数値の小さくなっているアーキテクチャを、その問題に適したアーキテクチャであるとして選択する。ただし、全てを 1 つのモジュールに集めた時などの極端な場合を除く為、2、3、4 の尺度も補助的に用いる。

POS システムをそれぞれのソフトウエアアーキ

表 1: POS システムの評価

	メインプログラム・サブルーチン型	オブジェクト指向型	パイプ・フィルタ型	非明示的起動型
名詞・動詞リンクの数	27	15	23	28
モジュール内動詞最大数	3	4	3	3
変更モジュール数	8	6	9	7
モジュール間リンクの数	10	5	4	10

テクチャに構成した名詞・動詞グラフについて評価すると表1のようになる。これは、POS システム問題の始めの状態での評価と、レシート出力の追加、入力した商品をキャンセルする機能の追加の2つの問題の変更を行った時の合計である。

この表によると、4つのソフトウエアアーキテクチャの中で、オブジェクト指向型において名詞・動詞リンクの数が特に小さく、また他の評価尺度の値も同定度、または小さくなっている。よって、POS システム問題ではオブジェクト指向型による構成を選択する。

また、同様な比較を一般加入電話の通話料金を計算する電話料金問題、電話料金を銀行から引き落とすシステムを模した電話システム問題に適用し評価を行うと、電話料金計算問題ではパイプ・フィルタ型を、電話料金システムではオブジェクト指向型を選択するという結果となる。

4 実際のプログラムとの比較

名詞・動詞グラフからの評価と実際のプログラムからの評価の比較を行った。比較に用いた例題は POS システムと、電話料金計算問題、電話料金システムの3つである。実験は、これら3つの問題について始めの問題定義の状態から、POS システムに対して2つ、電話料金計算に対して5つ、電話料金システムに対して3つの問題の変更を順に加えるものである。

実際のプログラムは C および C++で作成し、プログラム中のそれぞれのモジュールにおいて、他のモジュールを呼び出す関数の数や、他のモジュールと通信するための変数の数を計測する。この結果が表2である。

これを名詞・動詞グラフから、名詞・動詞リンクを数えた表3と比較すると、両者はアーキテクチャによっては大きく異なっていることが分かる。

これは、実際のプログラムにおいての、モジュール間の知識である関数や変数の数は、名詞・動詞グラフのリンクの数とは同じではないことをしめしている。

しかし実際のプログラムと、名詞・動詞グラフにおいて、モジュールの外部について必要な知識は意味的にはほぼ同じであった。よって、この2つの表の数字の差は、1つの名詞や動詞を扱うためのそれぞれのソフトウエアアーキテクチャと、プログラミング言語の複雑さを表してる。

5 まとめ

与えられた問題を主要な4つのソフトウエアアーキテクチャで構成した時の複雑さを評価する方法を提案した。この方法は、ソフトウエアアーキテクチャとは独立な同一のグラフから変形して描いた名詞・動詞グラフを定量的に評価する方法である。

この名詞・動詞グラフではモジュールの外について必要な知識をはかることが出来る。この時、それぞれのアーキテクチャのグラフは一つの名詞・動詞グラフからの変形で作成しているため構成要素が共通であり、定量的な評価が意味のあるものとなっている。

また、実際に例題のプログラムを作成し、名詞・動詞グラフからの評価と比較した。その結果、1つのモジュールを作成や変更する際の、そのモジュールの外部とのつながりの複雑さは、名詞・動詞グラフでほぼ表現することが出来ていることが分かった。この、実際のプログラムにおける関数や変数の数と名詞・動詞グラフの評価は、ソフトウエアアーキテクチャによっては異なった数値となった

表 2: 実際のプログラムの評価 (関数、変数)

	メインプログラム・サブルーチン型	オブジェクト指向型	パイプ・フィルタ型	非明示的起動型
POS システム	32	39	43	46
電話料金計算	38	112	45	112
電話料金システム	33	54	48	52

表 3: 名詞・動詞グラフの評価 (モジュール間のリンク)

	メインプログラム・サブルーチン型	オブジェクト指向型	パイプ・フィルタ型	非明示的起動型
POS システム	27	15	23	28
電話料金計算	38	41	24	47
電話料金システム	27	14	23	29

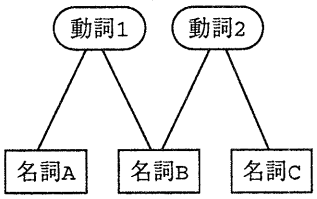

図 6: 問題からの名詞・動詞グラフ

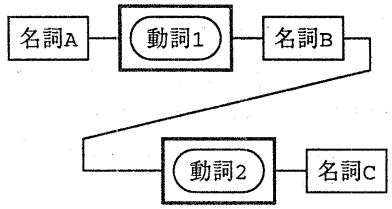

図 7: パイプ・フィルタ型への構成

が、これは実際のプログラムからの評価が複雑さではなく手間を表しているためである。

今後の課題としては、名詞・動詞グラフのトポロジーと適切なソフトウエアアーキテクチャとの関係、名詞・動詞グラフの描き方の精密化、評価法の精密化があげられる。

例えば、名詞・動詞グラフのトポロジーと適切なソフトウエアアーキテクチャとの関係については、以下のようなことが考えられる。図 6 の名詞・動詞グラフのように、名詞と動詞を交互に一筆でたどることが出来るようなグラフで、またこの時の名詞と動詞の順序が、計算の意味的な順序と一致するような場合には、この名詞・動詞グラフは図 7 のように、パイプ・フィルタ型に容易に構成することが出来る。

このように、ある種のトポロジーを持った名詞・動詞グラフは、特定の型のソフトウエアアーキテクチャに有利となっている。

参考文献

[1] S.R.Bourne: *The Unix System*, Addison-Wesley, 1982.

[2] H.Dhama: *Quantitative Models of Cohesion and Coupling in Software*, J.Systems & Software, 29, (1995),65-74.

[3] D.Garlan, M.Shaw: *An Intoroduction to Software Architecture*, Advances in Sw. Eng. and Knowledge., vol.1, 1993.

[4] D.Garlan, D.Perry: *Introduction to the Special Issue on Software Architecture*, IEEE Transactions on Software Eng., April 1995.

[5] D.Garlan, A.Robert, J.Ockerbloom: Architectural Mismatch or, Why it's hard to build systems out of existing parts, ICSE-17, April 1995.

[6] D.Garlan, A.Robert, J.Ockerbloom: Exploiting Style in Architectural Design Environments, SIGSOFT '94, December 1994.

[7] A.Goldberg: *Smalltalk-80:The Language*, Addison-Wesley, 1983.

[8] F.Hayes-Roth: *A blackboard architecture for control*, Artificial Intelligence 26:251-321, 1985.

[9] M.Hitz, B.Montazeri, *Measuring Coupling in Object-Oriented Systems*, Object-Currents, vol.1, issue 4, April 1996.

[10] M.A.Jackson: *System Development*, Prentice Hall, New Jersey; 1983.

[11] D.E.Knuth: *Fundamental Algorithms (3rd Ed)*, Addison-Wesley, 1997.

[12] C.Larman:*APPLYING UML AND PATTERNS - An Introduction to Object-Oriented Analysis and Design*, Prentice Hall,1998.

[13] C.-H.Lung, K.Kalaichelvan: *An approach to Quantitative Software Architecture Sensitivity Analysis*, SEKE'98, pp.185-192.

[14] D.L.Parnas: *On the criteria to be used in decomposing systems into modules*, CACM 15(12):1053-1058, December 1972.

[15] D.E.Perry: *Foundations for the Study of Software Architecture*, Software Engineering Notes, vol 17, no. 4, pp. 40-52, October 1992.

[16] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy, W.Lorensen: *Object-Oriented Modeling and Design*, Prentice Hall, New Jersey; 1991

[17] M.Shaw, D.Garlan:*Software architecture: perspectives on emerging discipline*, Prentice-Hall,1996.

[18] A.S.Tanenbaum: *Modern Operating Systems*, Prentice-Hall, 1992.

[19] A.S.Tanenbaum: *Computer Networks Third Edition*, Prentice-Hall, 1996.

[20] E.Yourdon, P.Coad:*Object-Oriented Analysis*, Englewood Cliffs, N.J.: Prentice-Hall, 1991.

[21] E.Yourdon, P.Coad:*Object-Oriented Design*, Englewood Cliffs, N.J.: Prentice-Hall, 1991.