ソフトウェア工学 47－5
（1986. 5. 28）

抽象型構成子概念に基づくソフトウェア部品体系を用いたソフトウェア構築方式

小林正和
（株）日立製作所システム開発研究所

既存のソフトウェアを再利用することは，ソフトウェアの生産性及び品質向上の有効な手段の一つである。その際，機能と構造とが明確に定義されているソフトウェアを利用することが，特に重要である。さらに，ソフトウェアが十分に抽象化されていることも重要である。ソフトウェアの形式的定義と，抽象化のためには，数学的概念を用いることが有効である。本論文では，数学的体系を抽象データ型と見なし，数学的構造を抽象型構成子と見なすソフトウェアの体系化技法を論ずる。さらに，本論文では，数学的体系である抽象データ型と，数学的構造である抽象型構成子とに基づくソフトウェア部品体系を用いて，ソフトウェアを構築する方式についても論ずる。

"Software Construction Technique founded on Abstruct Type Constructor" (in Japanese)

by Masakazu KOBAYASHI
(Systems Development Laboratory, HITACHI, Ltd.
1099 OHZENJI ASAO-KU KAWASAKI-SHI, 215, Japan)

To reuse existing software is one of the promissing solution for improvement of productivity and quality in software production. In such cases, it is important that the function and structure of existing software should be well defined and well represented in general form. Mathematical concepts are very useful for formal definition and generalization of software components. This paper presents a technique for software systematization which considers MATHEMATICAL SYSTEM to be Abstruct Data Type and MATHEMATICAL STRUCTURE to be Abstruct Type Constructor. It also describes software construction technique making use of software components founded on the concept of Abstruct Data Type and Abstruct Type Constructor.

1. はじめに

ソフトウェアの再利用についての期待が高まっている。再利用という言葉に関して，少くとも，次の4つの概念を区別する必要がある。

(1) 既存のソフトウェアの全体又は一部を，他のソフトウェアに転用活用すること（既存ソフトウェアが想定した利用対象，環境とは異なるコンテクストで，既存ソフトウェアを複製利用すること），

(2) 既存ソフトウェアをくり返し使用すること（既存ソフトウェアが想定した利用対象，環境と同じコンテクストで，既存ソフトウェアを使用すること，例えば言語コンパイラやエディタをくり返し利用することなど），

(3) 既存ソフトウェアの機能を保って，構造を一般には優れたものにかえること（go to 文をふんだんに使用した プログラムを構造化プログラムに変換することなど），

(4) 既存ソフトウェアの機能を保って，実現手段をかえること（例えば，COBOL で記述された プログラムを PL/1 の プログラムにコンバージョンすることなど）。

ここでは，(1)を再利用，(2)を使用，(3)を再生，(4)を変換と呼んで区別する。本論文では，(1)の再利用の問題を扱う。

既存のソフトウェアの再利用に当っては，次の2つの要件がとくに重要である。

(1) ソフトウェアの機能・構造が明確に定義されていること
——定義が不明確なソフトウェアは再利用できないし，よしんば利用したとしても，それを更に改良したり，機能の拡張をはかったり，さらに他へ再利用することは，難しくなる。

(2) ソフトウェアが十分に抽象化されていること
——抽象化されていないソフトウェアは再利用の範囲が狭くなる

(1)，(2)の問題の解決には，数学でとられてきたアプローチを採用することが有効である。

本論文では，数学の構造にならって，既存ソフトウェアを体系化する方式を提案する。

2. ソフトウェアの体系化

数学では，数学的体系と，数学的構造とを，次のように定義している（赤摂也著 現代数学概論 槇書房 1976 pp88-93）

<変数 $X_1, X_2, \cdots, X_m$ の上の構成図式>の定義．

変数 $X_1, X_2, \cdots, X_m$ の上の構成図式とは，変数 $X_1, X_2, \cdots, X_m$ を直積演算記号 $\times$，巾演算記号 $\wp$ によって，任意に組み合わせた，次のようなBNF記法で表わされる表現 <構成図式> のことである。

- <直積式> ::= <直積項> <直積演算記号> <直積項>．
- <直積因子> ::= <変数名> | "(" <直積式> ")"．
- <巾式> ::= <巾演算記号> "(" <変数名> ")" | <巾演算記号> "(" <直積式> ")" | <巾演算記号> "(" <巾式> ")"．
- <直積項> ::= <直積因子> | <巾式>．
- <構成図式> ::= <直積式> | <巾式>．
- <直積演算記号> ::= "×"．
- <巾演算記号> ::= "℘"．

例えば，X_1, X_2 が変数のとき

$\wp(\wp(X_1) \times X_2) \times \wp(\wp(\wp(X_1))) \times X_1$

などは，構成図式である。

変数 $X_1, X_2, \cdots, X_m$ の上の構成図式を，$T(X_1, X_2, \cdots, X_m)$ で表わす．

＜集合 $A_1, A_2, \cdots, A_m$ から構成図式 $T(X_1, X_2, \cdots, X_m)$ によって構成された集合 A＞　の定義

A および $A_1, A_2, \cdots, A_m$ を集合，$T(X_1, X_2, \cdots, X_m)$ を変数 $X_1, X_2, \cdots, X_m$ の上の構成図式とする．

このとき，A が，この構成図式にふくまれる $X_1, X_2, \cdots, X_m$ にそれぞれ $A_1, A_2, \cdots, A_m$ を代入して得られる集合に等しいならば，すなわち，

$A = T(A_1, A_2, \cdots, A_m)$ であるならば，A は，集合 $A_1, A_2, \cdots, A_m$ から構成図式 $T(X_1, X_2, \cdots, X_m)$ によって構成された集合であるという．

＜型 $(T_1, T_2, \cdots, T_n)$ を持つ次数 m の数学的体系＞，＜数学的体系における基礎集合＞，＜数学的体系における基礎概念＞　等の定義

$A_1, A_2, \cdots, A_m$ を集合；

$T_1, T_2, \cdots, T_n$ を変数 $X_1, X_2, \cdots, X_m$ の上の構成図式；

$M_1, M_2, \cdots, M_n$ をそれぞれ，集合 $A_1, A_2, \cdots, A_m$ から構成図式 $T_1, T_2, \cdots, T_n$ によって構成された集合とする．

このとき，集合 $A_1, A_2, \cdots, A_m$；集合 $M_1, M_2, \cdots, M_n$ の要素 $\alpha_1, \alpha_2, \cdots, \alpha_n$；

すなわち，

$$\alpha_i \in M_i = T_i(A_1, A_2, \cdots, A_m) \quad (i = 1, 2, \cdots, n)$$

および，構成図式 $T_1, T_2, \cdots, T_n$ の組

$$(A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n; T_1, T_2, \cdots, T_n)$$

のことを，型 $(T_1, T_2, \cdots, T_n)$ をもつ次数 m の数学的体系という．

そして，

$A_1, A_2, \cdots, A_m$ をその基礎集合，

$\alpha_1, \alpha_2, \cdots, \alpha_n$ をその基礎概念

と呼ぶ．

今後，数学的体系をドイツ大文字 $\mathfrak{O}, \mathfrak{A}, \mathfrak{L}, \cdots$ で表わし，数学的体系の型および，次数を，それぞれ，

$T(\mathfrak{O})$，$d(\mathfrak{O})$ と書くことにする．

＜次数 m の数学的構造＞，＜数学的構造における型＞，＜数学的構造における公理＞　等の定義

$X_1, X_2, \cdots, X_m; \xi_1, \xi_2, \cdots, \xi_n$ を変数，

$T_i(X_1, X_2, \cdots, X_m)$ $(i = 1, 2, \cdots, n)$ を，変数 $X_1, X_2, \cdots, X_m$ の上の構成図式とする，

このとき，変数 $X_1, X_2, \cdots, X_m$ と，命題

[1] $\xi_1 \in T_1(X_1, X_2, \cdots, X_m)$，

[2] $\xi_2 \in T_2(X_1, X_2, \cdots, X_m)$，

$\cdots$

[n] $\xi_n \in T_n(X_1, X_2, \cdots, X_m)$，

と，

変数 $X_1, X_2, \cdots, X_m; \xi_1, \xi_2, \cdots, \xi_n$ 以外の変数をふくまないいくつかの命題，

(1) $P_1(X_1, X_2, \cdots, X_m; \xi_1, \xi_2, \cdots, \xi_n)$，

(2) $P_2(X_1, X_2, \cdots, X_m; \xi_1, \xi_2, \cdots, \xi_n)$，

$\cdots$

(r) $P_r(X_1, X_2, \cdots, X_m); \xi_1, \xi_2, \cdots, \xi_n)$，

との組

$(X_1, X_2, \cdots, X_m; [1], [2], \cdots, [n]; (1), (2), \cdots, (r))$

のことを，次数 m の数学的構造という．組 $(T_1, T_2, \cdots, T_n)$ は，その型，組 $((1), (2), \cdots, (r))$ はその公理系，各命題 (i) は，その公理といわれる．$(i = 1, 2, \cdots, r)$．

今後，数学的構造 Σ の型を，$T(\Sigma)$，次数を $d(\Sigma)$ と書くことにする．

<数学的体系 $\mathcal{O}\ell$ が，数学的構造 Σ のモデルである> の定義

数学的体系

$\mathcal{O}\ell=(A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n; T_1, T_2, \cdots, T_n)$ と，

数学的構造

$\Sigma=(X_1, X_2, \cdots, X_s; [1], [2], \cdots, [t]; (1), (2), \cdots, (u))$ とが，

与えられたとする．

このとき，次の3つの条件が，みたされるならば，

数学的体系 $\mathcal{O}\ell$ は，数学的構造 Σ のモデルであると言う．

(i) $T(\mathcal{O}\ell)=T(\Sigma)$ (したがって $n=t$)

(ii) $d(\mathcal{O}\ell)=d(\Sigma)$ (したがって $m=s$)

(iii) $A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n$ は Σ の公理 (1), (2), …, (u) をすべてみたす．

すなわち，それらの公理に含まれる，変数 $X_1, X_2, \cdots, X_m; \xi_1, \xi_2, \cdots, \xi_n$ に，それぞれ，$A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n$ を代入して得られる命題，

$P_1(A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n)$,

$P_2(A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n)$,

$\cdots$

$P_u(A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n)$,

は，すべて正しい．

(以上は，市川惇信著による前掲書 pp88－93 よりの引用である)

以上のように厳密に定義された

数学的体系

$\mathcal{O}\ell=(A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n; T_1, T_2, \cdots, T_n)$ を

抽象データ型と見なし，

数学的構造

$\Sigma=(X_1, X_2, \cdots, X_m; [1], [2], \cdots, [u]; (1), (2), \cdots, (u))$ を

抽象型構成子と見なす．

すなわち，

数学的構造 Σ の公理に含まれる変数 $X_1, X_2, \cdots, X_m; \xi_1, \xi_2, \cdots, \xi_n$ に数学的体系 $\mathcal{O}\ell$ の基礎集合 $A_1, A_2, \cdots, A_m$ および，基礎概念 $\alpha_1, \alpha_2, \cdots, \alpha_n$ を，代入し，命題

$P_1(A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n)$,

$P_2(A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n)$,

$\cdots$

$P_u(A_1, A_2, \cdots, A_m; \alpha_1, \alpha_2, \cdots, \alpha_n)$,

が，すべて正しいことを確かめることができれば，数学的体系 $\mathcal{O}\ell$ は，数学的構造 Σ のモデルとなる．この意味で，数学的構造 Σ は，parameterized concepts になっている．

ソフトウェアのコンポーネントを，上記に定義した，抽象データ型と，抽象型構成子とに分けて，体系化することによって，厳密に定義され，しかも，普遍的に利用できるソフトウェア・コンポーネントが得られる．

このようにして定義された，ソフトウェア体系は，図のようになっている．

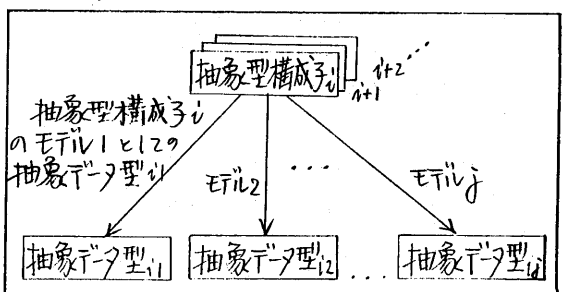

図　抽象型構成子と抽象データ型とを用いたソフトウェア体系

なお，数学的体系における基礎概念として，算法または，作用に着目するとき，その数学的体系を，代数系という[23]．代数系をモデルとしたソフトウェアの抽象化については，稲垣，坂部両氏による優れたサーベイ（文献4）がある．

3．ソフトウェアの再利用方式

前章で述べたように，ソフトウェアのコンポーネントが体系化されているとして，この体系を用いて，ソフトウェアを構築する方式の概要を以下に示す。

新しく開発するソフトウェアの各対象を，次の①～②によってモデル化したあとで，次に示す③により，既存ソフトウェアの体系を利用して，ソフトウェアを構築する。

①集合概念を用いて対象をモデル化する．

ある性質（述語）$C(a)$を持つ対象aをまとめて集合

$A=\{a \mid C(a)\}$ を定める．

今，このようにモデル化された集合を，$A_1, A_2, \cdots, A_m$とする．

②集合としてモデル化した対象間の関係（対象間に成立する条件）をモデル化する

m変数の条件（関係）

$\alpha(x_1, x_2, \cdots, x_m)$の各変数の変域が，集合$A_1, A_2, \cdots, A_m$，これらの直積$A_1 \times A_2 \times \cdots \times A_m$の要素$(a_1, a_2, \cdots, a_m)$で，$\alpha(a_1, a_2, \cdots, a_m)$が成立するものの全体からなる集合を，

$$\{(a_1, a_2, \cdots, a_m) \mid \alpha(a_1, a_2, \cdots, a_m)\} \in 2^{(A_1 \times A_2 \times \cdots \times A_m)}$$

と書く（文献1 P39）。

$\alpha(a_1, a_2, \cdots, a_m)$と，

$\{(a_1, a_2, \cdots, a_m) \mid \alpha(a_1, a_2, \cdots, a_m)\}$

とは，同一視することができるから$\alpha(a_1, a_2, \cdots, a_m)$は，

$M=2^{(A_1 \times A_2 \times \cdots \times A_m)}$の要素とみなすことができる。

今このようにしてモデル化した関係を$\alpha_1, \alpha_2, \cdots, \alpha_n$とする。これらを要素として含む冪集合を$M_1, M_2, \cdots, M_n$とする．

M_iを集合$A_1, A_2, \cdots, A_m$から，変数$X_1, X_2, \cdots, X_m$の上の構成図式

$T_i(X_1, X_2, \cdots, X_m)=2^{(X_1 \times X_2 \times \cdots \times X_m)}$

によって構成された集合とする

$(i=1, 2, \cdots, n)$。

以上により，新しく開発するソフトウェアをモデル化することによって，抽象データ型（数学的体系）のが，定義できる。このモデル化は，一種の対象指向モデル化法とも，ERモデル化法の拡張とも考えられる。なお，上記のモデル化は，先きに報告した方式（文献10）の改良となっている。改良の主な点は，先きの報告では，上記②に写像を用いたが，ここではさらに広い関係概念を用いている点である。

③次に，上記①,②で定まった数学的体系（Ω）と型，次数が一致する数学的構造（Σ）を，ソフトウェア体系から探し出す。

Σの公理が，(1), (2), …, (u)であるとして，上記①,②で定まった集合$A_1, A_2, \cdots, A_m$と関係$\alpha_1, \alpha_2, \cdots, \alpha_n$が，(1), (2), …, (u)のすべてをみたすかどうか調べる．(1), (2), …, (u)がすべてみたされるΣが，既存ソフトウェアの体系の中に見出せれば，ソフトウェア構築は完了する。もし，このような，数学的構造を，見出すことができなければ，利用できる既存のソフトウェアが存在しない。この場合は，新しい抽象型構成子を作成することになる。

なお，上記③で，公理は，項と，論理式とによって形式的かつ記号的に記述できる。公理の記述には，Prolog言語や，代数的記述言語などを使用することによって，上記③は，かなりの程度計算機化することができる。しかし，公理がみたされるかどうかの判断など，対話は必要である。

4. まとめ

本論文では，ソフトウェアを再利用するためには，既存のソフトウェアを厳密に定義しておく必要があることの重要さを述べた。この視点から，数学的体系を抽象データ型，数学的構造を抽象型構成子とみなして，ソフトウェア・コンポーネントを体系化する方式を述べた。

また，ソフトウェア再利用時には，新しく作成するソフトウェアの対象を集合と，関係概念によってモデル化し，既存の抽象型構成子（数学的構造）を探索する方式を述べた。

ソフトウェアの再利用方式の現状を見るに，既存ソフトウェアのコンポーネント化の基礎技術が確立されていないため，苦労して，コンポーネント化をはかっても，再利用が思わしく行かないことが多いようである。基礎技術の探求とともに，再利用のもととなるソフトウェア・コンポーネントの作成には，ソフトウェアを公理化できる抽象能力を持つ人材と，抽象型構成子，抽象データ型を，実際の計算機で効率的な実現を行なうことのできる技術を持つ人材との協同が必要であろう。自由放任の形で作成されたソフトウェアを再利用することは，短期にみれば，ソフトウェアの生産性が向上するようにみえても，ソフトウェアのライフサイクルを考える場合には，ソフトウェアの問題を一層悪化させてしまう。ソフトウェアの体系化の基礎技術開発がますます重要である。

5. 参考文献

1) 赤摂也：現代数学概論，岩波書房，数学講座の第一巻（1976）

2) 岩波 数学辞典 第3版 121 構造 pp325-327 （1985）

3) J.A. Goguen, J.W. Thatcher and E.G. Wagner: An Initial Algebra Approach to the Specification, Correctness, and Implementation of Algebraic Data Types, Current Trends in Programming Methodology IV: Data Structuring (R. Yeh ed.), Prentice-Hall, pp80-144 (1978).

4) 稲垣康善，坂部俊樹：抽象データタイプの代数的仕様記述法の基礎 (1)～(4)：情報処理学会誌 Vol 25 No1 (1984.1) pp47-53, Vol 25 No5 (1984.5) pp491-501, Vol 25 No7 (1984.7) pp708-716, Vol 25 No9 (1984.9) pp971-986,

5) J.A. Goguen: Parameterized Programming Trans. Software Engineering IEEE-10 No5 Sept 1984, pp528-543

6) K. Futatsugi, J.A. Goguen J.P. Jouannaud, and J. Meseguer: Principles of OBJ2, Proc. 1985 Symp. Principles of Programming Languages, ACM Vol 12 1985 pp52-66

7) J.A. Goguen, J. Meseguer: Equality, Types, Modules and Generics for Logic Programming Proc. the 2nd Int. Conf. on Logic Programming, at Uppsala Univ., (1984).

8) J.A. Goguen: Reusing and Interconnecting Software Components, IEEE Computer Vol 19 No2 1986-2 pp16-28,

9) 萩谷昌己：構成的型理論における一般化について，日本ソフトウェア科学会第2回大会論文集 1985.11.20～22 pp189-192

10) 小林正和：データに着目した非手続型仕様記述から手続型のプログラム構造を生成する一手法，情報処理学会第28回（昭和59年前期）全国大会 1K-6