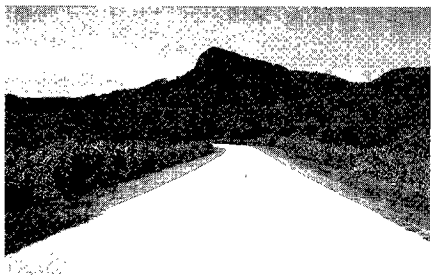
道しるべ：オブジェクト指向プログラミング言語の進化
― SmalltalkからJavaへ至る道程 ―



青山 幹雄
新潟工科大学 情報電子工学科

オブジェクト指向技術の進化

「オブジェクト指向」が一般に知られるようになったのは，Smalltalk-80の出現が契機である．それから，20年を経て，今や，「オブジェクト指向」は，プログラミング言語から分析・設計方法，ソフトウェアアーキテクチャ，コンポーネントなど，技術的広がりと奥行きを増した[1]．それは，オブジェクト指向がソフトウェアを構成する基礎的で基盤的な概念であり，技術である証左である．しかし，この概念はどのようにして育まれたのであろうか．ここでは，SmalltalkからJavaへ至るオブジェクト指向プログラミング言語の進化とその設計の背後にあるビジョンの道筋を辿ってみよう．

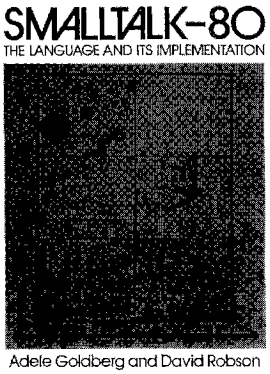


図-1　Blue Book

- 個人がマスターできる：システムが創造的な精神を支援できるためには，1人でシステムを完全に理解できるようになっていなければならない．
- 良い設計：システムは最小数の変更不要な部品から構成されているべきである．部品はできるだけ一般化され，すべての部品が一貫したフレームワークに組み込まれるようになっているべきである．
- 言語はコミュニケーションの枠組みを提供する．コミュニケーションはインタフェースとプログラミング（論理）の2つの側面がある．
- 言語のスコープ：コンピュータを利用するための言語は，人とコンピュータの間で，両者の内部モデル，外部メディア，両者の間のインタラクションの3つを扱えるように設計すべきである．

表-1　Smalltalkの設計原則（抜粋）

Smalltalkのビジョン：はじめにコミュニケーションありき

Alan KayやAdele GoldbergなどのSmalltalkの開発者たちは，Smalltalkを「コミュニケーションの言語」として開発した．本来は「雑談」や「おしゃべり」を意味するSmalltalkという名前にその意図が表れている．また，それは，Alan Kayの提唱したパーソナルコンピュータDynabook構想[9]を実現するものであった．

図-1に示すSmalltalkの原本Blue Book[4]の「まえがき」の最初の節は，"Smalltalk is a vision"という見出しで始まるSmalltalkの歴史である．そこでは，一般の人がコンピュータを使うことを目標にして言語のあるべき姿を問ったSmalltalk開発者のビジョンが強調されている．

実際，Smalltalkは1970年代初めから，子供やコンピュータの専門家でない人に使ってもらう観察実験を行い，理論化し，それに基づいて新たなシステムを構築し実証する，というサイクルを2～4年単位で繰り返して開発された．Smalltalk-80はこの5番目のシステムである[2]．

Smalltalk設計の中心となったDan Ingallsはいくつかの設計原則を挙げている（表-1）[6]．この中で，言語設計では，人間同士あるいは人間とコンピュータとの間で，図-2に示すBody間の明示的コミュニケーションとMind間の暗示的コミュニケーションの2つのレベルを視野に入れるべきであると主張した．明示的コミュニケーションはメッセージによって伝えられる情報であり，暗示的コミュニケーションは2者間の共通の論理やモデルである．コンピュータの場合，Bodyはユーザインタフェースである．したがって，コンピュータを使うための言語は，ユーザインタフェースのための言語とプログラミング言語の2面から設計すべきであると考えた．

プログラミングモデルとしては，従来の手続きとデータ構造に代わり「メッセージーアクティビティ」モ

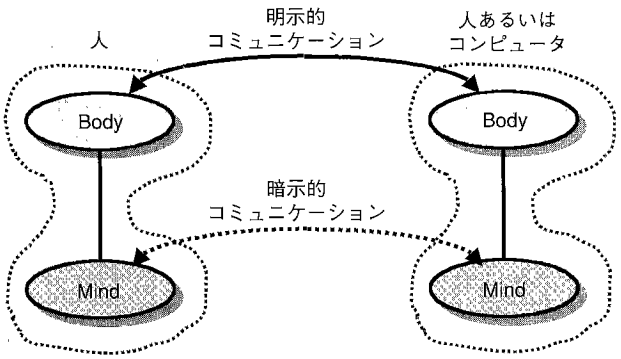


図-2　コミュニケーションと言語

デルを提案した[9]．手続きとデータ構造は，メッセージによって起動される，それ自体がコンピュータのような存在であるアクティビティという単一の概念で置き換えられるのである．アクティビティとは，オブジェクトにほかならない．したがって，プログラミングのためには，最小の5つの概念を理解すればよいとした．オブジェクト，メッセージ，クラス，インスタンス，メソッドである．

Smalltalkは画期的な言語であった．1970年代後半のコンピュータ利用環境はパンチカード全盛であり，文字ベースのCRTさえまれであったことを考えれば，このビジョンは先進的である．私自身，Starワークステーションの上で初めてSmalltalkを動かした時の感動は今でも記憶に新しい．しかし，Smalltalkは画期的であるがゆえに，また，現実とのギャップも大きかった．そのビジョンを実現するためには，開発環境を含む世界全体を作り出さざるを得なかったといえよう．これは，むしろ，現実との障壁となった．

新たな言語ではなく，広く利用されている言語を発展・進化させてオブジェクト指向へ進む道をつけようとする発想が出てきても不思議ではない．また，当時のハードウェア性能ではインタープリタによる実行方式では実用的な性能が得られなかった．この結果，当時広く利用されつつあったC言語をオブジェクト指向へ発展させたC++やObjective-Cなどの言語が生まれた．

C++の誕生と台頭：オブジェクト指向と実践との対峙

C++はオブジェクト指向概念とCの効率のよさを両立できるシステムプログラミング言語"C with Classes"としてベル研究所のBjarne Stroustrupによって1979年に開発が始まった[2),11)]．ベル研究所に入所する前，Stroustrupは，ケンブリッジ大学で分散処理システムのシミュレータをオブジェクト指向プログラミング言語のルーツであるSIMULAによって記述した．この経験は，彼がオブジェクト指向に目を向ける契機となった．SIMULAの背後

(1) 設計目標
1) プログラミングを楽しくすること
2) 次の目的の汎用プログラム言語であること
　a) より良いC言語（better C）
　b) データ抽象化の支援
　c) オブジェクト指向プログラミングの支援
(2) 設計原則
i) 現実の問題に則して言語を発展させる
ii) 無益な完璧を追求しない
iii) 今，役に立つ
iv) すべての機能が現実的に実装できる
v) 進化の道筋を持つ
vi) 言語であって，完全なシステムではない
vii) 分かりやすい支援を提供する
viii) 人に強要しない

表-2　C++の設計目標と設計原則

にあるクラスなどのオブジェクト指向の概念がStroustrupに大きな影響を与えた．また，モジュール化の機構として，抽象データ型とそれに基づくカプセル化，型チェックの有効性，柔軟性に強くひかれた．しかし，SIMULAそのものは性能などの点で実用にはならなかった．さらに，シミュレータの次の版は，Cの元となったBCPLを使って書き直そうとしたが，これは難行に終わった．このような開発における苦難ともいえる経験が，C++の開発を促した．

Stroustrupはケンブリッジ大学でPh.D.を取得し，ベル研究所に入所した．そこで，Cをその開発者から学ぶとともに，外部には1985年に本として公開されたC++[10)]の開発に着手した．

彼は，ケンブリッジ時代の経験から，プログラミング言語のあるべき姿として次の3つの目標を掲げた．彼は，C++の第2版で，これを満たしたとしている．

1) プログラム構成の支援を通して設計を支援するよいツールである
2) 実行効率がよい
3) OSとの独立性が高く，ポータビリティが高い

これを実現するため，表-2の設計目標と設計原則を設定した．

C++の開発は，画期的な言語の開発よりは，現実的目標の達成に主眼が置かれていることが分かる．彼は，「C++の設計は問題解決のためであって技術探求のためではない」[11)]と述べている．

Cとの親和性の高さに加え，実用性の高さによって，C++は，Cにオブジェクト指向概念を導入した多くの言語の中で主流となり，現在まで広く利用されている．しかし，Cとの親和性のよさは，逆に，旧来のプログラミングモデルを引きずるという負の遺産も継承することになった．

C++からJavaへ：はじめにネットワークありき

C++からちょうど10年後の1995年にJavaが誕生した．Javaはネットワークコンピューティングのビジョンを実現するための言語といえる[8]．Sunのビジョナリストであるbill joyや言語開発の中心であったJames Gosling，あるいはGuy Steeleといった Javaの開発者たちのビジョンと技術の結晶である．インターネットではネットワークを介して異なるプラットフォーム上で実行可能となるクロスプラットフォーム性が本質的な要求である．その意味で，インターネットが生んだプログラミング言語といえる．

オブジェクト指向プログラミング言語の進化の流れから見れば，C++がCから継承したメモリ管理やポインタなどの欠点を除き，SmalltalkとC++の長所を併せ持つ言語といえる．しかし，もし，Javaの設計目標が単に「ベターC++」であったなら，これほど人々を魅了しなかったであろう．ネットワークコンピューティングのビジョンがJavaの開発にブレークスルーをもたらした．クロスプラットフォームを実現するために，JavaはSmalltalkと同様の仮想マシンのアプローチを採っている．しかし，型付けのないSmalltalkと異なり，プログラムのすべての実行において型状態（type state）と呼ぶ型付けを要求し，実行経路にかかわらず，プログラムの同一実行点では型状態が一致することを要求している[5]．この要求はプログラム実行に関するさまざまな特性を静的にチェックできるなどの，多くの利点をもたらした．C++がSmalltalkに対し強い型付けを導入したように，JavaはC++に対し実行状態の型付けを導入した．また，C++における，親クラス（基底クラス）の変更によってそのサブクラスをすべて再コンパイルし直さなければならないという，脆弱な基底クラス問題（Fragile Base Class Problem）も軽減された．

Smalltalkの再生：Squeak

Javaの公開と同じ頃，Smalltalk開発者のDan IngallsらはSmalltalkにより記述したSmalltalkの処理系Squeakの開発に着手した[7]．この仮想マシンは，Cへのトランスレータを備え，実行速度の高速化を実現している．

1999年5月にLos Angelsで開催されたソフトウェア工学国際会議でAlan KayがSqueakのデモを交えて，パーソナルコンピュータの過去と現在を展望した．半年後の11月にDenverで開催されたOOPSLA '99で，Dan Ingallsが，同じSqueak上で一段と洗練されたデモを交えて，Smalltalkの過去と現在を語った．今，2人は共に，Disneyの Imagineering Laboratoryに所属している．そこには20年以上にわたり，時と組織を超えて，1つの技術を追求する姿がある．

オブジェクト指向プログラミング言語のビジョンと実現

オブジェクト指向プログラミングの適用はSmalltalkに始まり，C++で本格化し，現在はJavaへの移行段階にある．Smalltalkの背後にあるビジョンはパーソナルコンピュータであり，コミュニケーションである．C++は実践体験が土台となった．Javaはネットワークコンピューティングのビジョンを実現するものである．この3つのプログラミング言語は，いずれも，1人ないし少数の人たちが中核的な設計を行っていることに気付く．彼らは，コンピュータのあり方やプログラミング言語のあるべき姿へのビジョンを描き，問題の本質を問い，その結果，これらの言語を開発したことに留意すべきである．

一方，Smalltalkは優れたビジョンを持ちながらも広く普及するには至らなかった．Richard Gabrielはプログラミング言語が広く受け入れられるための条件として次の4つを挙げている[3]．

1）言語が技術ではなく社会プロセスによって受け入れられ，進化できる
2）言語が必要とするコンピュータ資源が過大でない
3）実行モデルが単純である
4）利用者に高度な数学的素養を要求しない

C++やJavaは，既存の言語の欠点を克服し，進化させるアプローチを採ることにより，実世界との障壁を下げ，広く受け入れられた．プログラム言語の普及も他の諸技術と同様，社会的なプロセスに従う．優れた技術はビジョンによって生まれるが，広く受け入れられるためには，ビジネスや社会への理解も必要である．それは，人の成長に似ている．

SmalltalkからJavaへ至る20年にわたるオブジェクト指向プログラミング言語の進化は，技術開発のあり方についての示唆に富んでいる．現代の，新技術の情報洪水の中で，時には，技術の進化の背後にある思想ともいうべき考えの軌跡を辿ってみてはどうだろうか．問題の本質へ立ち戻り，新たな発見の手がかりとなるかもしれない．

参考文献

1) 青山幹雄: オブジェクト指向開発技術の進化, 情報処理, Vol.39, No.6, pp.588-591 (June 1998).
2) Bergin, T.J. and Gibson, R.G. (eds.): History of Programming Languages-II, ACM Press (1996).
3) Gabriel, R.P.: Patterns of Software, Oxford University Press (1996).
4) Goldberg, A. and Robinson, D.: Smalltalk-80: The Language and Its Implementation, Addison-Wesley (1983).[相磯秀夫(監訳), Smalltalk-80, オーム社 (1987)].
5) Gosling, J.: Java Intermediate Bytecodes, Proc. ACM SIGPLAN Workshop on Intermediate Representations (IR '95), pp.111-118 (Jan. 1995).
6) Ingalls, D.: Design Principles Behind Smalltalk, BYTE, Vol.6, No.8, pp.286-298 (Aug. 1981).あるいは, Peterson, G. (ed.): Object-Oriented Computing, Vol.1, IEEE CS Press, pp.70-74 (1987).
<http://users.ipa.net/~dwighth/smalltalk/ byte_aug81/design_principles_behind_smalltalk.html>
7) Ingalls, D. et al.: Back to the Future-The Story of Squeak, Proc. ACM OOPSLA '97, pp.318-326 (1997).
<http://st.cs.uiuc.edu/Smalltalk/Squeak/docs/OOPSLA.Squeak.html>.
8) Joy, W.: The Joy of Computing, <http://java.sun.com/features/1999/07/bill.joy.html>
9) Kay, A.C.: アラン・ケイ, アスキー (1992).
10) Stroustrup, B.: The C++ Programming Language, 1st ed., Addison-Wesley (1986).
11) Stroustrup, B.: The Design and Evolution of C++, Addison-Wesley (1994).

（平成11年12月14日受付）