解　説

5. 事　例

5.3 米国におけるプログラミング環境の開発†

斎　藤　信　男††

1. はじめに

プログラミング環境あるいはソフトウェア開発環境は，現在のソフトウェア工学でもっとも注目を集めている課題である．たとえば，ソフトウェア開発環境をもっとも一般的に構築するには，環境がどんなアーキテクチャをもつべきかということは，一つの興味のある問題である．ここでは，新しいソフトウェア開発環境のあり方を探る一つの方法として，米国におけるソフトウェア開発環境のプロジェクトをいくつか紹介し，その中から将来のあるべき方向を探ってみる．

プログラミング環境ないしはソフトウェア開発環境を実際に構築する方法は，さまざまなものが考えられるが，おおよそ次のように分類できる．

1) ツールインテグレーション方式 (tool integration approach)

もっとも簡単に考えると，プログラミング環境は，プログラミングツールあるいはソフトウェアツールの集合体として実現すればよく，これはいわゆるツールインテグレーション方式であり，もっとも一般的にみられるものである．

2) プロセスインテグレーション方式 (process integration approach)

ソフトウェア開発は，その過程であるソフトウェアプロセスが重要な要素である．これは，ソフトウェア開発環境をプロセスを基礎にして構築する方式である．

3) モデルベースト方式 (model based approach)

これは，ソフトウェア開発環境のモデルをあらかじめ想定し，環境を構築する方式である．

4) ヒューマンベースト方式 (human based approach)

ソフトウェア開発は，人間であるエンジニアの作業がもっとも重要な要素であると考え，人間の作業の解明やその支援を基礎として環境を構築するのが，この方式である．

このようなさまざまな方針で，ソフトウェア開発環境のプロジェクトが進められている．ツールインテグレーション方式としては，Gandalf[1]，Σ プロジェクト[2] などがある．プロセスインテグレーション方式として，プロセスプログラミングに基づいた Arcadia[3]，ISTAR[4] などがある．モデルベースト方式として，SDA[5] がある．また，ヒューマンベースト方式として，Leonardo[6]がある．

以下では，これらの方式の中で，比較的新しいと思われる三つのプロジェクト，すなわち，Arcadia，SDA および Leonardo についてその目標や現状などを紹介する．

2. プロセスインテグレーション方式—Arcadia

これは，L. Osterweil によって提唱されたプロセスプログラミング[7] に基づきソフトウェア開発環境を構築することを狙ったプロジェクトである．

プロセスプログラミングは，ソフトウェアプロセスをなんらかのプログラミング言語で精密に記述してしまい，それをソフトウェアとして利用しようというアイデアである．すなわち，プロセスプログラミングでは，図-1 に示すように，ソフトウェアの開発作業を表すプロセスアルゴリズムと，ソフトウェアプロセスで開発されるソフトウェアプロダクトを正確に記述する．プロセスプログラミングをオブジェクト指向的に捉えると，プロセスプログラムは一つのオブジェクトとして定義される．プロセスプログラミングでは，それに関わるエンジニアとして，プロセスプログラム自身を作成するソフトウェアエンジニアと，具体的なプロダクトを作成するソフトウェアプラクティショナの

† Programming Environment Project in the United States by Nobuo SAITO (Department of Mathematics, Keio University).

†† 慶應義塾大学理工学部数理科学科

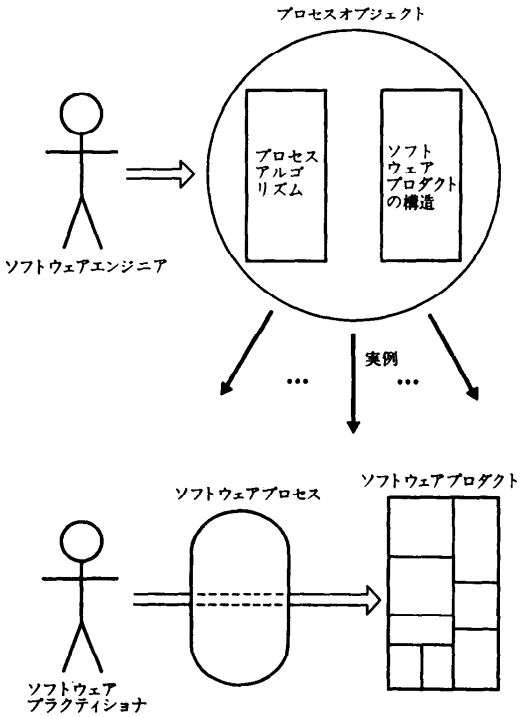

図-1 プロセスプログラミング

二つのクラスを考える．具体的なソフトウェアプロセスとして実行するために，ソフトウェアプラクティショナが自分用の実現例を一つ作り，プロセスプログラムのアルゴリズムに従って作業を行いながらその中のソフトウェアプロダクトの中身を入力し，最終的なプロダクトを作成していく．

プロセスプログラムを記述するプロセスプログラミング言語に対しては，豊富な情報構造の記述，豊富な制御構造の記述，並行処理や例外処理の記述，オブジェクト指向型プログラミングの記述など，いくつかの要求がある．本来，そのような要求に応じたプロセスプログラミング言語を設計すべきであるが，もちろん，既存のプログラミング言語を利用することも可能である．

Arcadia プロジェクトは，プロセスプログラミングを提唱した L. Osterweil を中心にして，コロラド大学，マサチューセッツ大学，カリフォルニア大学アーバイン分校，スタンフォード大学，TRW などいくつかの大学や研究所が参加している開発環境プロジェクトである．

Arcadia では，プロセスプログラミング言語として，Appl/A (Ada Process Programming Language with ASPEN) を設計した．これは，Ada のサブセットである．Ada は，もともとパッケージ機能があり，ある程度のオブジェクト指向プログラミングが可能である．さらに，並行処理や例外処理も記述できる．3)によれば，Appl/A は，オブジェクト間の関係を，tuple によって自由に定義することができる．この言語は，かなりプロセスプログラミング言語としてよく働いたけれども，たとえば型の階層の扱い，動的なプログラミングの欠如などが欠点となっている．また，プロセスプログラミングのある場面では，手続き的な記述よりもむしろ宣言的記述，規則ベースのプログラミングを使ったほうがよいこともあった．

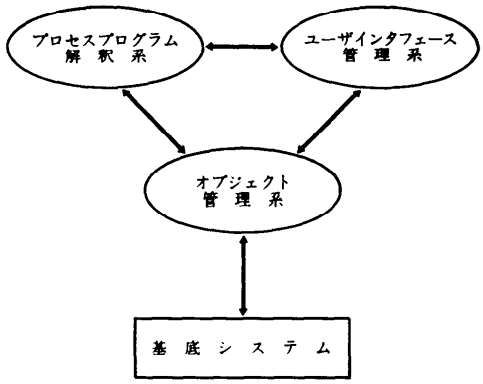

図-2 Arcadia における環境アーキテクチャ

Arcadia における環境のアーキテクチャは，図-2 に示すように基底システムの上に，プロセスプログラム解釈系，オブジェクト管理系およびユーザインタフェース管理系の三つのシステムが構築されている．オブジェクト管理系では，型をオブジェクトに導入することを考えているが，まだ具体化はされていない．このような方向に沿っているオブジェクト管理系として，Cactis[8]，PGraphite[9] などのプロトタイプを構築している．また，ユーザインタフェース管理系のプロトタイプとして，Chiron[10] を開発している．そこでは，抽象データ型の扱い，一般的なディスプレイの支援，入力の機能の提供などが行われている．

3. モデルベースト方式—SDA

SDA (Software Designer's Associates)[11] プロジェクトは，ソフトウェア設計を含む上流工程を支援するソフトウェア開発環境の構築を目指すプロジェクトである．これは，SPC (Software Productivity Consor-

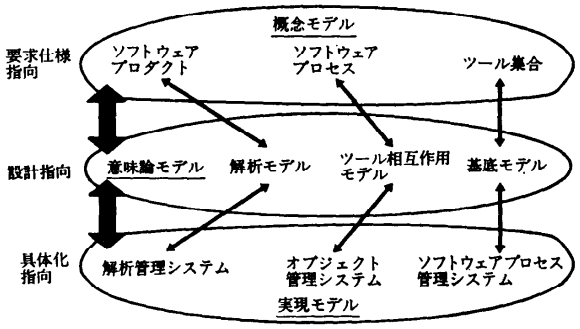

図-3 SDA における概念モデル，意味論モデル，実現モデルの関係

thium) の W. Riddle ら米国の研究者，大阪大学，東工大，静岡大学，慶応大学など日本の研究者，それに日本の企業の参加のもとに行われている日米，産学共同プロジェクトである．

SDA は，Riddle の提唱したソフトウェア開発環境のモデル(図-3)を基礎として出発している．これは，概念モデル，意味モデル，および実現モデルの三段階の階層からなる．一方，概念モデルでは，ソフトウェア開発環境で考慮すべき要素，すなわちソフトウェアプロダクト，ソフトウェアプロセス，およびソフトウェアツールの集合を考える．その詳細を定義するために，意味モデルでは，解析モデル，基底モデル，およびツール相互作用モデルを考える．さらに，実現モデルでは，解析管理モデル，プロセス管理モデル，およびオブジェクト管理モデルを考える．それぞれの階層を超えた関連は，必ずしも一対一ではない．

解析モデルは，プロダクトの性質やツールの性質を解析し，その質を決定したりツール導入時における性質の決定をしたりするために用いる．基底モデルは，プロセスの動的な性質を定義するために使われる．さらに，ツール相互作用モデルは，ツールの集合の能力を規定しプロセスの個々の作業を規定するために用いる．

実現モデルは，論理的なモデルを実際に環境として実現する際に必要な機能を導出するために使うモデルである．解析管理モデルは，解析ツールの起動やその導入などを管理する機能を表す．プロセス管理モデルは，プロセスの実行とそこにおける各作業で使われるツールあるいはその集合の起動を管理する機能を表す．オブジェクト管理モデルは，プロダクト，プロセスプログラム，ツールをすべてオブジェクトとして定義し管理する機能を表す．

一方，SDA では，個々のソフトウェア開発環境を目的やユーザに応じて適応していく機能をも考慮する．これは，Gandalf プロジェクトでも考慮されていたが，目標とするものは，環境生成のための環境であった．これにより，ソフトウェアエンジニアが自分の要求に合致した環境を生成すればよい．ただし，Gandalf でもそのような究極的なシステムの実現は難しく，構文向きエディタが生成的に作成された程度であった．SDA でも，図-4 に示すように，組織，ソフトウェア開発プロジェクト，および個々のソフトウェアエンジニアに応じて適応した環境を生成していく[12]．

SDA では，現在，設計プロセスの経験を実際にプロセスとして記述することから始め，いくつかのプロトタイプを使ってその実際的な実験をしている．次の段階として，SDA 環境のアーキテクチャを追究することを始めている．ここでは，モデルに応じた階層的アーキテクチャと，完全な分散的アーキテクチャとが第一近似として考えられる．

図-5 に示すのは，階層的アーキテクチャの一例[13]であり，型付きのオブジェクト管理層を中心にして，

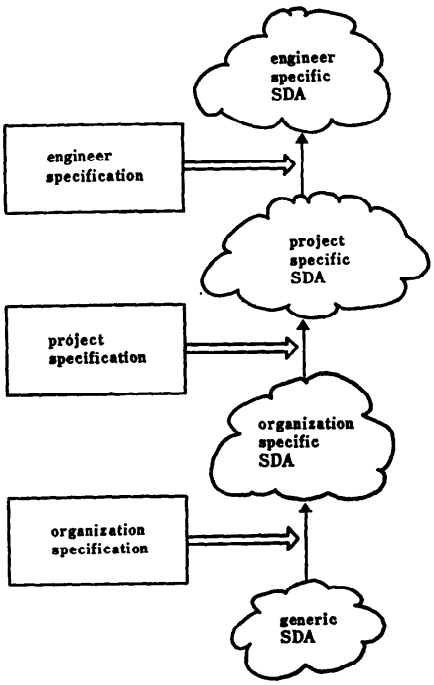

図-4 SDA 適応過程

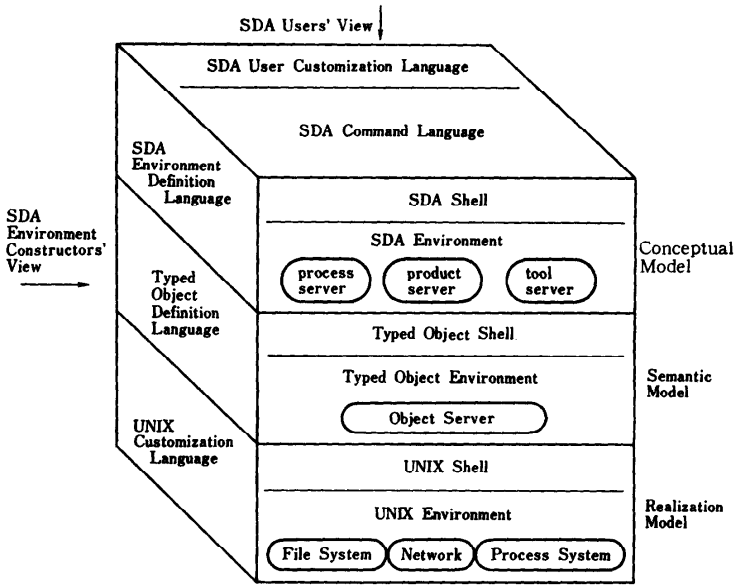

図-5 SDA の階層的アーキテクチャの例

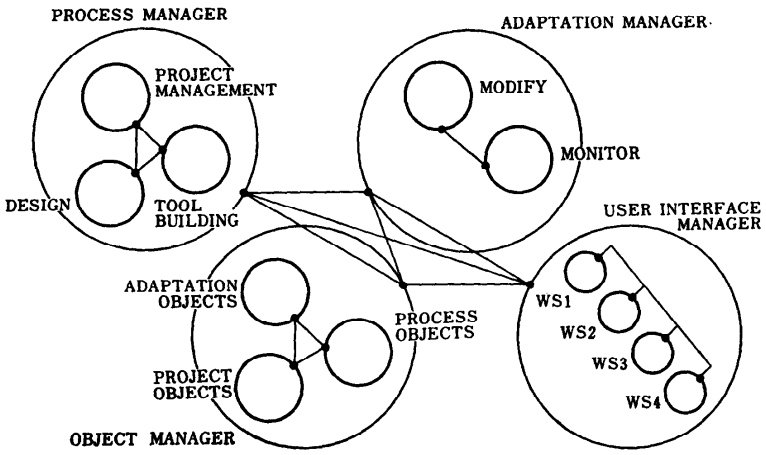

図-6 SDA の分散型アーキテクチャの例

概念モデルレベルではプロセスの記述を行っている．それぞれの階層に対して，その機能や構造を記述する言語が提供されているので，適応はそれを介して行う．

一方，図-6に示すのは完全な分散型アーキテクチャである．そこでは，プロセスマネジャ，オブジェクトマネジャ，およびユーザインタフェースマネジャが相互に通信しながら独立して稼働している．個々のマネジャの内部は，再び複数個のプロセスから成り，それが相互に通信しあっている．そこでは，例外事象が生じると，マネジャに付属した通信プロセスにその事実が伝達され，さらに他のマネジャの通信プロセスを介してそのマネジャの管理下にあるプロセスへ伝達される．3個の独立したマネジャのほかに，さらに適応マネジャが独立して稼働している．これは，オンラインで各プロセスの動きを監視しながら，必要に応じて環境の適応を実施していく．

プロセスマネジャ内には，プロジェクトマネジャ，

デザインプロセス，およびツールビルディングプロセスが存在する．オブジェクトマネジャ内には，プロジェクトオブジェクト，プロセスオブジェクト，および適応オブジェクトが存在する．ユーザインタフェースマネジャ内には，ワークステーションを介してアクセスしてくるユーザとの相互通信などを管理するプロセスが存在する．適応マネジャ内には，モニタリングプロセスと，モディフィケーションプロセスが存在する．このような内部に存在するプロセスは，先に述べたような機能により，他のマネジャ内のプロセスと相互通信しながら，その機能を果たしていく．

SDA の今後の課題として，アーキテクチャの追究，それに基づく総合的プロトタイプの開発，実際の設計プロセスへの適用とその評価などが考えられる．

4. ヒューマンベースト方式—Leonardo

米国テキサス州オースティンにある MCC (Microelectronics and Computer Consorthium) は，米国のメーカを中心とした研究コンソーシアムであるが，その中の部署の一つである STP (Software Technology Program) は，Laz Belady が率いている．ここでは，新しいソフトウェア開発環境のプロジェクトとして，Leonardo プロジェクトを推進している．

Leonardo では，ソフトウェア開発が基本的には人間であるエンジニアの作業に基づくものであること，特に上流工程では人間の創造性に深くからんだ作業が必要であることを重視し，そのために人間の本質を追究しつつ，開発環境を構築していこうという方針を採っている．そこでは，ソフトウェア工学のアーキテクチャとして，図-7 に示すような階層を考える．ここで，関係層では，オブジェクト間の関係を自由に操作できる必要があり，そのために planetext システムを作成し，これを使って例題の設計を行う設計者の意思決定の記録を採った[14]．これは，その記録の解析を行い，優れた設計者（スーパデザイナ）の意思決定機能を探り出すきっかけをつかもうという目的をもっている．

また，分散層では，分散システムの設計のために使う言語 Raddle とそれを支援するツール VERDI とを開発した．Raddle は，N-party 会話モデルに基づく設計言語である[15]．

また，STP の提唱により，IEEE/ACM のシンポジウムとして，CSCW (Computer-Supported Cooperative Work) が始まった．これは，ソフトウェア開発のような共同作業に対して計算機システムの支援をいかに行うかを追究する問題で，人間の特性と関連した興味のある課題である．N-Party 会話モデルは，当然この課題に当てはめることができるであろう．また，Xerox のパロアルト研究所では，Colab システムと称して，ワークステーションを使った共同作業支援環境の実験を行っている[16]．

また，Leonardo で提案されている新しいソフトウェアプロセスとして，演劇出演サイクルモデルがある[17]．これは，図-8 に示すような基本サイクルが演劇出演プロセスにおいてみられるものであり，これを技術移転も含めたソフトウェア開発プロセスに適用しようという試みである．このモデルは，個々の人間というよりも集団としての人間を考えているものであるが，発想の自由さ，新しさに注目すべきものがある．

5. 今後のソフトウェア開発環境の動向

計算機システムの高性能化は，ソフトウェア開発環境の土台が十分の能力を保証できることを意味する．そこで，こんどはその土台の上にいかに知恵を絞って開発環境を構築するかという問題が重要になる．単にツールインテグレーションという発想をするだけでなく，ソフトウェア工学あるいはソフトウェアエンジニアの立場から柔軟な発想をもって，環境構築を目指さなければならない．

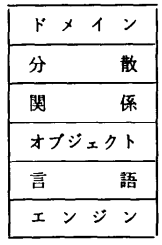

図-7 ソフトウェア工学の階層

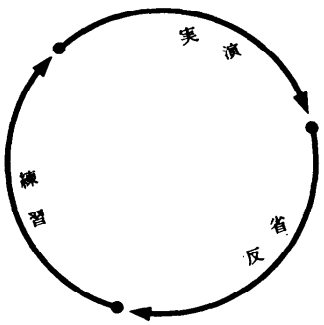

図-8 演劇出演の基本サイクル

今後の環境構築のために重要となってくる要素技術は，次のようなものであろう．

- オブジェクト管理システム

どのプロジェクトでも共通に必要となるものは，オブジェクト管理システムである．これは，型の導入など検索機能の向上と効率との問題があり，今後十分な検討が必要になってくる．

- ユーザインタフェース

これは，人間の作業を軽減させるという意味で，今後も重要な要素技術となる．

- 意思決定支援システム

ソフトウェア開発の上流工程の支援を行うためには，今後，意思決定支援システムの追究が必要であろう．

- 人間の共同作業支援システム

計算機システム，特に個人用のシステムが増えるにつれ，共同作業を支援する機能は必須のものとなる．

参 考 文 献

1) Special Issue on Gandalf Project, The Journal of Systems and Software, Vol. 5, No. 2 (May 1985).
2) ソフトウェア生産工業化システム（Σ システム）概要説明書，情報処理振興事業協会 (Apr. 1985).
3) Taylor, R. N., Osterweil, L. et al.: Foundations for the Arcadia Environment Architecture, Proc. of 3rd ACM SIGPLAN/SIGSOFT Software Engineering Symposium on Practical Development Support Environments, ACM SIGPLAN Notices, Vol. 24, No. 5 (Feb. 1989).
4) Dowson, M.: ISTAR—An Integrated Project Support Environment, Proc. of 2nd ACM SIGPLAN/SIGSOFT Software Engineering Symposium on Practical Development Support Environments, ACM SIGPLAN Notices, Vol. 2, No. 1 (Jan. 1987).
5) Kishida, K. et al.: SDA: A Novel Approach to Software Environment—Design and Construction, Proc. of 10th ICSE, IEEE Computer Society, Singapore (Apr. 1988).
6) Marks, P.: What is Leonardo?, Technical Report 141-86, MCC (Apr. 1986).
7) Osterweil, L.: Software Process is Software Too, Proc. of 9th ICSE, IEEE Computer Society, Monterey (Apr. 1987).
8) Hudson, S. E. and King, R.: The Cactis Project: Database Support for Software Environments, IEEE Trans. on Software Engineering. Vol. 14, No. 6 (June 1988).
9) Weilden, J. C. et al.: PGRAPHITE: An Experiment in Persistent Typed Object Management, Proc. of 3rd ACM SIGPLAN/SIGSOFT Software Engineering Symposium on Practical Development Support Environments. ACM SIGPLAN Notices, Vol. 24, No. 5 (Feb. 1989).
10) Young, M., Taylor, R. N. and Troup, D. B.: Software Environment Architectures and User Interface Facilities, IEEE Trans. on Software Engineering, Vol. 14, No. 6 (June 1988).
11) ソフトウェア設計環境の新しい統合技術，第1回 SDA シンポジウム，SDA コンソーシアム (Nov. 1987).
12) Penedo, M. H. and Riddle, W. E.: Guest Editor's Introduction: Software Engineering Environment Architecture, IEEE Trans. on Software Engineering, Vol. 14, No. 6 (June 1988).
13) 山岡，斎藤：設計環境のアーキテクチャ，情報処理学会プログラミング言語研究会 (1988).
14) Potts, C. and Bruns, G.: Recording the Reasons for Design Decisions, Proc. of 10th ICSE, IEEE Computer Society, Singapore (Apr. 1988).
15) Evangelist, M. et al.: Using Raddle to Design Distributed Systems, ibid.
16) Stefik, M. et al.: Beyond the Checkboard: Computes Support for Collaboration and Problem Solving in Meetings, CACM, Vol. 30, No. 1 (Jan. 1987).
17) Marks, P.: System Development as a Performing Art——A People-Centered View of Leonardo, Technical Report, STP-365-86, MCC (Dec. 1986).

（平成元年2月28日受付）