第9回 ソフトウェア開発方法論をもっと役立たせるには？

―方法論工学への招待―

佐伯元司 東京工業大学情報理工学研究科計算工学専攻

レシピ（recipe）という言葉をご存知であろうか．「スパゲッティ200gを茹で，それにトマト水煮300gを加えて...」というやつである．つまり，料理の作り方，どの順序，タイミングで何を加えて，どうするかの手順を書いたものである．これに従うと，料理の達人でなくてもそれなりのものができるわけである．このレシピのソフトウェア作成版が，方法論（Method，Methodology）である．方法論とは，どのような手順で何を行っていけばよいかを教えてくれる．これに従えば熟練者でなくても，それなりのソフトウェアを作成できるということを目的としている．方法論は1通りではなく，種々のものが作成され，文書化されている．書店で情報工学関係の書籍コーナーへ行けば，タイトルに「構造化○○法」とか「オブジェクト指向○○法」といった語句を含んだ本をよく見かけるであろう．これらは典型的な方法論の教科書である．それではこれらの方法論は本当にソフトウェア開発に貢献しているのであろうか？　料理の「レシピ」は，なぜそのような手順になっているのかの科学的な根拠（もちろん，経験に裏付けられた根拠もあるが）がある．たとえば，なぜこのタイミングで塩を小さじ1杯入れなければいけないかは，温度や人間の辛さの感じ方などの要因からくる理由がある．それに対して，方法論はなぜこのような手順になっているのか，これで本当によいのかといった根拠が科学的にも必ずしも明確にはなっていない．方法論を科学的に，いや工学的に捉えようとする学問が方法論工学（Method Engineering）である．

方法論って何？

ソフトウェア開発では，プログラムや仕様書などの種々の文書が作成され，その作成作業は人間の高度な知的作業の上に成り立っている．したがって，熟練者と初心者との作業の質に，大きな違いが現れるであろう．この作業の質をあげるには，熟練者の作業をまねる，つまり熟練者はどのような作業を行い，どのような文書を作っているかをマニュアル化してやるのが一番であろう．このような発想で捉えられているのが方法論である．方法論は，1）記法も含めてどのような文書（以下，プロダクトという）を作ればよいか，2）どのような作業を行っていけばよいかというプロセスを示唆してくれる．方法論に従うことにより，熟練者でなくても，それなりの質を持ったプロダクトを効率的に作り上げることができるようになることを目的としている．

方法論はソフトウェア開発の種々の段階で考えられるが，実際には要求分析・設計段階でのものが重視されている．その理由は，1）要求・設計段階はソフトウェア開発プロセスの中で前段に位置し，この段階での作業の質が開発効率や開発されたソフトウェアの品質に大きな影響を及ぼす，2）要求分析・設計段階は人間の物事に対する認識力によるところが大きく，ソフトウェア開発中で最も人間の知的能力を要求される，からである．

方法論の例を1つあげよう．要求分析段階の方法論であるCoad&Yourdonのオブジェクト指向分析法[1]の作業手順を，大まかに書くと，

1）対象領域にあるオブジェクトを識別する．
2）識別したオブジェクトを用いて対象領域の構造を識別する．
3）オブジェクトの属性や演算を定義する．

としている．この作業手順に従うと，徐々にクラス図と呼ばれる図-1のようなプロダクトができていく．もちろん，料理のレシピ同様，「クラス」，「属性」，「集約」といった固有の言葉や概念が用いられている．たとえば，図-1では「リフト」がクラスであり，「位置」がリフト固有の属性，「モータ」は「リフト」と集約関係にある．方法論固有の言葉の意味や概念を知っていないとその方法論を使いこなすことはできない．しかし，料理のレシピを勉強し練習するのと同様に，方法論の勉

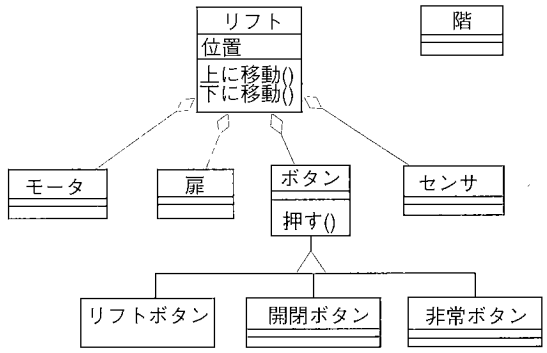


図-1　クラス図の例（UML表記）

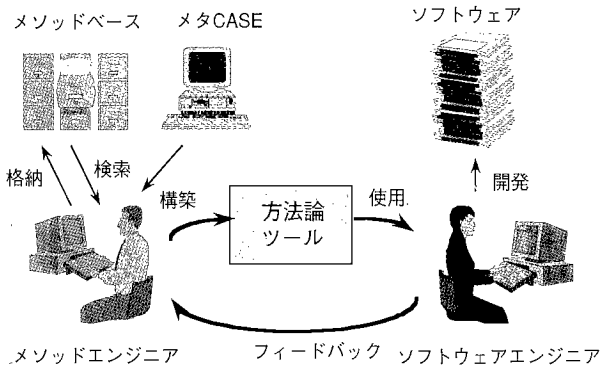


図-2　方法論工学のプロセス

強・訓練を行うことにより，ある水準のソフトウェアを効率よく開発できるようになる．

方法論は役に立っている？

これまでに，構造化分析・設計法，オブジェクト指向分析・設計法を始めとして種々の方法論が生み出されてきた[2]．果たして，本当にこれらの方法論はソフトウェア開発に役立っているのであろうか？　もし，あまり役立っていないのならばどうすればよいのであろうか[3]？　実際に現場で方法論をどのように使用しているかの事例を調査することにより，その答えの一端を知ることができる．

実際の適用例を調べてみると，教科書やマニュアルどおりに適用するのではなく，カスタマイズを行って適用した場合が多い．たとえば，教科書では状態遷移図を書かなければいけないようになっていたが，開発対象システムの動作が複雑なため，1つの状態遷移図で書くと複雑になってしまう．そこで1つの動作のシナリオを表す簡単な図を，典型的な動作例についてのみ書いて代用した事例がある．また，組織ですでに別の図の支援ツールを持っていたため，マニュアルで指定されていた図ではなく，ツールが支援する図で代替した例もある．

カスタマイズの理由としては，上記の事例のように，対象とするソフトウェア固有の性質，組織の制約や開発標準の存在以外に，ライブラリや最近はやりのパターンやフレームワークといった過去の資産の存在，といったこともあげられる．方法論の教科書やマニュアルに書かれていることはあまりに一般的すぎ，そのまま適用するのでなく，状況に応じて適当にカスタマイズして使用する，さらに一歩進んで適切な方法論を作り上げてしまうことが，方法論をもっとソフトウェア開発に貢献させる上で重要である．もちろん，どのような状況でも効力を発揮する万能な1つの方法論があれば別であるが，現実にはそのような「スーパー方法論」なるものは存在しない．重要なことは，状況に応じて適切な方法論を用いることであり，不適切なものを用いるとかえって逆効果になる．

方法論工学とは？

方法論工学とは，提唱者の1人シャーク・ブリンケンパーによると[4]，

「情報システム開発のための方法論，技術や支援ツールを構築したり，適合させたりするための工学を探求する学問分野」

ということになる．これに，方法論・ツールの管理やこれらを使いこなすための教育・訓練手法を探求するための学問分野も含めて考える．

方法論を構築する簡単で効果的な手法は，ソフトウェア開発と同様に，これまで開発された方法論を再利用することである．これまで多くの方法論が提唱されており，それらの運用経験も蓄積されている．したがって，これらもしくはその一部を部品として組み合わせて適切な方法論を作り出すのが現時点での簡単なやり方であろう．事実，Hatley法やWard法などの実時間システム用の構造化分析法は，データフロー図という既存の部品と状態遷移図という部品を組み合わせて作られた方法論である．

以上の手法に従った方法論構築・使用プロセス，つまり方法論工学プロセス（Method Engineering Process）は**図-2**のようになる．図中のメソッドベース（Method Base）は，方法論を構成するために既存の方法論を部品として格納したデータベースであり，メソッドエンジニアと呼ばれる方法論をよく熟知したエンジニアが適切な部品を検索し，それらをカスタマイズし，組み合わせ，統合することによって，状況に適した方法論を作り出す．また，メソッドエンジニアは作りあげた方法論を支援するツールの開発も行う．ソフトウェアエンジニアは，開発プロジェクトの中で，この新しい方法論とツールを使用し，その使用経験をメソッドエンジニアにフィードバックする．メソッドエンジニアは，フィードバックをもとに方法論をさらに改良したり，メソッドベースの保守も併せて行う．このようなプロセスでの重要な技術は，

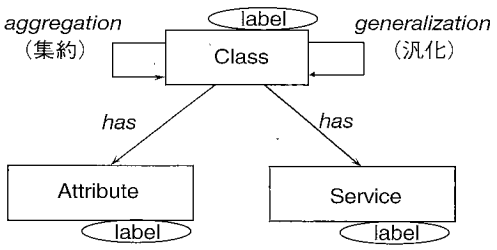


図-3　クラス図のモデル：メタモデル

1）方法論や方法論部品を記述するための技術（メタモデリング）.
2）方法論部品をカスタマイズしたり，組み合わせ・統合したりする技術.
3）どのような方法論がどのような状況に適切かを明らかにする技術.
4）支援ツールを作り出す技術.
5）上記の作業を支援するコンピュータツール（CAME: Computer Aided Method Engineering）の開発.

が必要であり，現在，産・官・学の連携のもと，欧米で盛んに研究・開発が進められている[6]．以下では，各々の技術の開発状況を簡単に紹介する.

メタモデリング

従来，方法論はマニュアルなどに自然言語で記述されていた．方法論や方法論部品をメソッドベースに格納するには，それらをモデル化し何らかの形式言語で記述しなければならない．方法論をもう一段高いレベルでモデル化することをメタモデリングという．方法論は，「オブジェクト」，「属性」といった固有の言葉，概念を含んでおり，これらをベースにモデル化を行い，記述する．記述言語としては，実体関連図，Zなどの形式的仕様記述言語などが使用されている．現状では，どのようなプロダクトを作ればよいかという方法論のプロダクト的な側面での記述が盛んに行われており，実体関連図がよく使用されている．その理由は，実体関連図によるプロダクトの構造の記述が，そのままリポジトリの記述につながるからである.

図-3にクラス図（正確には記法を含んでいないので，オブジェクトモデルといった方がよいであろう）の一部を実体関連図で記述した例を示す．図中の長方形が実体であり，方法論中の概念を表す．矢印が概念間の関連を表す．たとえば，「クラス（Class）」は「属性（Attribute）」や「操作（Service）」を「持っている（has）」．このようなメタモデルは，クラス図が持っている本質だけを取り出して，一般化して表したものであり，オブジェクト指向方法論共通の記述でもある.

方法論の統合

方法論部品をなんでも手当たり次第に組み合わせて統合すれば有益な方法論ができあがるというわけではない．料理でも，「茹でたスパゲッティ200gに茹でたうどん100gを混ぜ合わせ...」などという，同種の材料（スパゲッティとうどん）を混ぜ合わせて作るようなレシピはあまりない．このようなレシピに従って作っても味覚の面から優れたものができないからである．これと同様に，同じような方法論（部品）を組み合わせても有益な方法論はあまり得られない．たとえば，クラス図と実体関連図を統合したような方法論は有益であるとはいえないであろう．それぞれ相手にない特徴を持つ方法論同士を，相手の不足しているところを補うように組み合わせて初めて有益な方法論ができあがる．つまり，有益な方法論を作り出すための，部品の組合せの規則が考えられる.

前節でも述べたように，方法論はプロダクトの側面とプロセスの側面の2つを持っている．方法論を統合する場合にこの2つの側面に着目して，図-4に例示するような2つの統合方法がある．(a）では，プロダクトの側面から，クラス図，状態遷移図（ステートチャート），事象トレース図，データフロー図といった4種類のプロダクトをある規則に従って結合した例で，結果として Rumbaughらの OMT（Object Modeling Technique）が得られる．また (b）のように，オブジェクト指向分析法を用いてクラス図を作成し，そのクラス図をもとにZなどの形式的仕様記述言語で記述するといった方法論は，プロセスの側面から2つの方法論を統合したと考えられる．前者は，クラス図が持っていない記述能力，つまりシステムの動作を記述する能力，機能の詳細を記述する能力を他の図で補おうとして得られた方法論である．後者はクラス図では詳細部分が厳密に記述できないという点と，分析段階でいきなりZのような形式的仕様記述言語で記述することは難しいという点を解決するために，プロセスに応じて適した方法論を組み合わせた例である.

方法論の特性

あるソフトウェア開発プロジェクトに対してどのような方法論が適切あるいは不適切なのであろうか？　また，それをどうやって判断すればよいのであろうか？　もちろん，これまでにも，「データフロー図だけでは実時間システムの開発を行うのは難しい」といった一般的な経験則はいくつかいわれている．データフロー図は，システムの機能とその機能間を流れるデータをモデル化するだけであるため，実時間システムの本質であるシステムの時間的な動作を記述することができないからである．このため，状態遷移図や事象トレース

図といったシステムの動作系列を直接表現できるような手法がよく用いられる．また，納期が厳格で重圧になってくるようなプロジェクトでは，あまり多くの種類のプロダクトを作成しないような方法論がよく使用され，成功している[6]．このような（経験）規則はもっとあるはずである．これらをもっと収集し，体系化できないであろうか？

上記のような規則を収集するためには，方法論にはどのような特性があり，それらが実際のソフトウェア開発プロジェクトの特性とどのように関係しているかを明らかにする必要がある．この関係が明確になることによって初めて，プロジェクトにどのような方法論が適切かが明らかになる．方法論とプロジェクトの関係を調べるには，1）方法論自身を分析する，2）実際のプロジェクトに適用し，このプロセスや結果を分析する方法がある[5]．

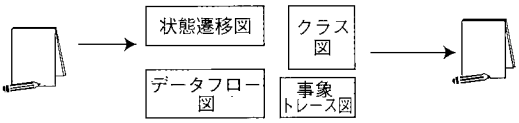
方法論は多数提唱されているが，ある程度の訓練を積まないと十分には使いこなせない．どれが使う側の人間にとって簡単で，どれが難しいのであろうか？その答えを出そうとする試みがある．方法論の複雑度（Method Metrics）という考え方である．方法論は「クラス」，「状態」などといった特有の概念を含んでおり，一般にはこれらが多ければ多いほどその方法論について知っておかなければいけないことが多くなり，難しくなるであろう．このような概念の数や概念間の関連の数などをもとに，方法論の複雑度を定義しようというわけである．

実際の開発プロジェクトに方法論を適用した事例は学会発表の場でもかなり報告されてきており，事例研究も徐々に進んでいる．たとえば，文献7）では実際の12のプロジェクトにオブジェクト指向方法論を適用した際に共通に見られた問題点（オブジェクトの多面的な見方ができない，メッセージパッシングだけではオブジェクト間の単純なインタラクションしか表現できないなど）を抽出し，分析している．しかし，このような事例研究が増えているにもかかわらず，分析方法や結果のまとめ方が主観的，アドホックであったり，分析の観点がさまざまであったり，それゆえこれらの成果を共有し，方法論とプロジェクトとの関係を明確にするには至っていないのが現状である．今後，適用事例を収集し，適用事例や分析結果を，WWW等を利用して共有できる仕組みを作り上げることが重要である．

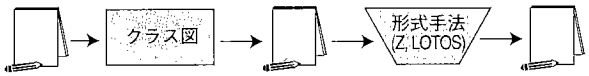
方法論をもっと役立てるための5カ条

最後に本稿で述べてきたことを，方法論をより効果的に使用するという観点でまとめる．

1）開発プロジェクトの特性を把握すること．
2）自分が使用しようとしている方法論の特性をよく理解すること．プロジェクトの特性にマッチしているかどうかを考えること．



（a）プロダクト主導型の統合



（b）プロセス（作業手順）主導型の統合

図-4　方法論の統合

3）方法論を過信しないこと．ソフトウェア開発中で偶発的に生じる事象に対しては，方法論は無力である．たとえば，作業中に急に納期が短縮されてしまったなどというアクシデントについては方法論では対処できない．方法論の守備範囲をしっかり認識すること．
4）採用している方法論を開発チームの全員が理解していること．方法論は作業標準でもある．作業の進捗状況をお互いに認識しあい，他人が作成した文書（中間生成物も含めて）を理解する上で有益である．
5）方法論を使用したときの開発経験を蓄積し，利用できる形にしておくこと．このようなデータは，状況に応じた方法論を選択する上での重要な手がかりにもなるし，改善にも不可欠である．

方法論工学は，本格的な研究・開発は始まったばかりである．方法論を使いこなすための鉄則は，方法論自身をよく知ることであり，これを助けてくれるのが方法論工学の技術である．欧米に対し遅れをとっている我が国でも，方法論工学の第一歩でもある，方法論適用経験の蓄積とこれらの情報共有から早急に始め，方法論をもっと効果的にソフトウェア開発に適用するための技術を整備していかなければならない．

参考文献
1）Coad,P. and Yourdon, E.: Object-Oriented Analysis, Prentice Hall (1990).
2）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).
3）Saeki, M., Osterweil, L., Aksit, M., Roland, C. and Kruchten, P. : Are Methods Really Useful?, ICSE-20 Panel, Proc. of 20th ICSE, Vol. II, pp.18-23 (1998).
4）Brinkkemper, S. : Method Engineering: Engineering of Information Systems Development Methods and Tools, Information Science & Technology, Vol. 38, No.4, pp.275-280 (1996).
5）本位田真一, 青山幹雄, 深澤良彰, 中谷多哉子（編）：オブジェクト指向・分析, 共立出版 (1995).
6）Method Engineering: Principles of Method Construction and Tool Support, Chapman & Hall (1996).
7）Aksit, M. and Bergmans, L. : Obstacles in Object-Oriented Software Development, Proc. of ACM OOPSLA'92, pp.341-358 (1992).

（平成10年11月3日受付）