TOPICS

もつれたソフトウェアをときほぐす

広域システム科学系　増原英彦

階層的な分類とソフトウェア

たいていのソフトウェアの中身は，スパゲティのようになっている，と言ったら驚かれるだろうか．プログラミングの経験がない人は，もしかしたらソフトウェアの中身は整然としたものだと思っているかも知れないが，少しでもプログラミングの経験がある人ならちっとも驚かないだろう．プログラミングの入門でとりあげられるようなプログラム——例えば方程式の解を求めるものでも想像して欲しい——であっても，入力を正しく読み取る処理，解が無いような例外的事態への対処，スピードアップのための工夫，プログラムの動作確認のためのコードなどを加えてゆくうちに，色々なコードがからみあって複雑になる経験を多くの人がしたはずである．実用的なソフトウェアにおいては，その複雑なコードが何百万行も続いているのである．ソフトウェア，特にプログラミング言語の研究は，長年に渡って，そのような複雑にもつれたソフトウェアの「中身」をなるべくすっきりとさせるための方法を提案してきた．例えば，プログラムの処理手順に「構造」を持たせる「構造化プログラミング」（つまり，それ以前のプログラムには構造すらなかったのである！）は 1960 年代後半に提唱されている．同じ頃に，ソフトウェアが扱う問題領域にある「もの」をプログラムの中で直接表現しようという考えの下に「オブジェクト指向プログラミング」も提案されている．この提案は，最近になってようやく，C++ や Java などのプログラミング言語として多くの人に知られ，使われるようになってきた．

オブジェクト指向プログラミングは，人間が複雑な世界を整理する際に行う方法——階層的な分類——をプログラミングに持ち込んだことでも知られる．例えば生物を門・綱・目・科 ... と分類するように，はじめはおおまかに，そして段々と細かく分類してゆく．ソフトウェアの中身も共通する性質のものをおおまかに分類し，細かな性質の違いでさらに分類してゆくのである．図 1 は実際のソフトウェアに見られる階層的な構造を示している．

それでももつれるソフトウェア

ではオブジェクト指向プログラミングが万能かというと，決

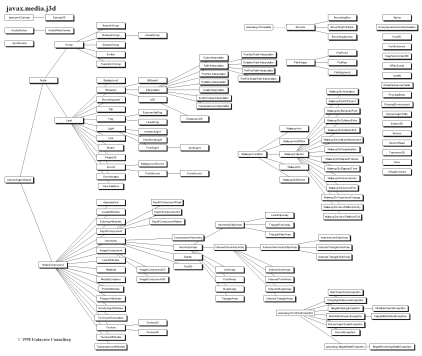

図 1：3 次元グラフィクスライブラリの階層構造（(C) 1998 Finlayson Consulting, http://java.sun.com/ より引用）

してそうではない．図 1 のようなオブジェクト指向プログラミングを使ってきれいに階層化されたように見えるプログラムであっても，その内部をよく観察するとやはり複雑にもつれあっている所が沢山見つかる．

図 2 はその現象の一端を示している．これは，商用のアプリケーションサーバのソースプログラムにおいて「例外が起きたときの処理」を行っている部分を視覚的に示したものである．縦長の箱 1 つ 1 つはクラス（階層的に分類されたプログラムの 1 単位）を表わしている．箱の長さはプログラムの行数に対応している．箱の中に引かれた赤い横線は，そこに「例外が起きたときの処理」が書かれていることを示している．つまり，このソフトウェアはでは，沢山のクラスのあちこちで，同じような処理を何度も書かなければならなかったことを意味している．

では何故こういった現象が起きるのだろうか？ 階層的分類のような新しいプログラミングの方法は，プログラムを整理するのに大きな貢献をしたことは確かである．しかし，ソフトウェアには色々な要素があり，ある観点からプログラムを整理すると，別の観点からは「もつれて」しまうことは避けられないのが現実である．

アスペクト指向プログラミング

このような「どうしてももつれてしまう現象」を解決する試みが比較的最近になって注目されるようになってきた．その中の 1 つに「アスペクト指向プログラミング」と呼ばれる新しいプログラミングの方法がある．この方法では，ある観点で（階

図2：アプリケーションサーバソフトウェアにおける例外処理の出現（Colyer, et al., “Using AspectJ for component integration in middleware”, in Companion of OOPSLA’03 より引用）

層的に）分類されたプログラムがあったときに，色々な場所に「散らばって」（あるいは「もつれて」）しまうような処理のために特別な手段を用意しようというものである．詳しいことまでは書けないが，プログラムが動く際に内部で用いられている名前や処理の性質などを契機にして処理が行われるようにすることで，これまで「もつれて」しまっていた処理を1まとめにするのがこの方法のアイデアである．現在，アスペクト指向プログラミング言語としてJava言語を拡張して設計されたAspectJ言語などが開発され，研究者だけでなく，産業界でも使われ始めている．

「関連」を扱うための機構

我々の研究グループでは，このアスペクト指向プログラミング言語をより良くするために基礎理論を作ることや機能の拡張をいくつか行っている．その1つが芝浦工業大学の古宮研究室と共同で開発した「連想アスペクト」機構である．この機構は小さなアプリケーションプログラムを組み合わせて，1つの大きなアプリケーションプログラムを作るような場合に必要となり，かつ，これまでは「もつれて」しまっていた処理を簡単にまとめられるようにするものである．

例えば，プログラミング言語の統合開発支援環境（IDE）のようなアプリケーションを，テキストエディタと，コンパイラを組み合わせて作る（図3）ことを考える．このような場合，「ファイルが保存されたらコンパイルする」「コンパイルをする前に関連するファイルを保存する」「コンパイルエラーの起きた場所をエディタで表示させる」といったアプリケーション間の連携が必要となる．これまでの方法では，どんなにコンパイラやエディタのプログラムが階層的に整理されていたとしても，結局は沢山の場所を変更することになってしまう．さらに，エディタが開いている複数のファイルとコンパイラを対応づけるといった，アプリケーション間の対応を維持するためには沢山のコードを書かなければいけない．

我々の提案した連想アスペクト機構を用いると，エディタとコンパイラを連携させるための処理を独立したコードとして記

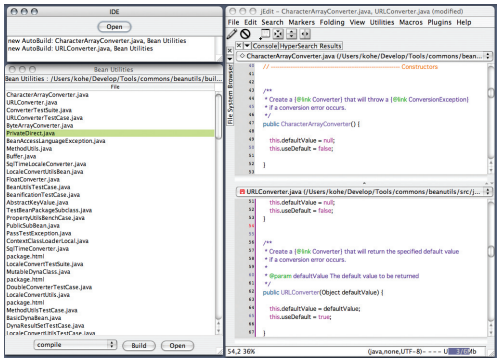

図3：連想アスペクトを用いて作られた統合開発支援環境の実行画面。右側がエディタで左側がコンパイラが読み込むプログラムの一覧表示である．

図4：オブジェクト指向プログラミングと従来のアスペクト指向プログラミングおよび連想アスペクトによるアプリケーション記述の比較。長方形はクラスを表わし、青・緑・赤色はそれぞれ、エディタ・コンパイラ・連携のためのコードを表わす．

述できるようになる．さらに，オブジェクトどうしの対応を管理するための仕組みがあるので，複数のファイルとコンパイラの間の対応を取るための労力を大きく節約できるようになる．同じアプリケーションプログラムを（図4左）オブジェクト指向言語Java，（同中）従来のアスペクト指向言語AspectJ，（同右）連想アスペクトの3種類の言語による記述の比較をすると，「もつれていた」コード（赤い部分）が，アスペクト指向プログラミングによって独立した単位としてほぐされ，さらに連想アスペクトによって簡潔になっていることが見てとれるだろう．これを可能にする連想アスペクトの処理系をAspectJコンパイラの拡張として実現し，オープンソースで公開している．

おわりに

新しいプログラミング方法の提案は，プログラミング言語の設計者と，それを利用してソフトウェアを作る開発者の両方が揃ったときに初めて普及するものである．その意味ではアスペクト指向プログラミングは設計者と開発者の両者が積極的に研究開発を行っている．プログラミング言語の世界では新しいプログラミングの方法が定着するには10年単位の時間がかかるので，その努力が結実するにはまだ時間がかかると思われるが，今後の発展が楽しみな分野である．