ソフトウェア開発における協調支援環境Vela
——(4) 事例ベースの構成法——

2G−4

下津　直武　福井　健司　山口　高平　落水　浩一郎

静岡大学

1. はじめに

開発経験の少ない対象分野では、ソウトウェア開発事例の再利用により(類似システムの開発経験を参考にして)ソフトウェア開発が進められることが多い。我々は、このような機能を形式化するために事例型推論の適用を検討している。

事例の再利用にあたっては、

(1) 少ない手間でユーザーが事例を検索できること。
(2) 検索された事例の修正量が少ないこと。

の2点を満たすことが重要である。

このためには、ソフトウェア開発事例を開発者が遭遇する状況に応じて分類・整理しておく必要がある。

本稿では、以上の目標を達成するための事例ベースの構成法について、ＪＳＰ法を用いた事例にもとづいて実験・考察した結果を報告する。

2. 事例ベースの構成法に関する基礎考察

2.1. 状況による分類

事例収集のもっとも簡単な手段は、詳細な問題解決過程を記録してもらうことである。しかし、問題解決時の考察・決定事項等を時間順に並べた構成では、十分でない。開発時に遭遇する状況に応じて事例を分割し、共通の性質をもつものをまとめる必要がある。

このような分割の方針を検討するために、以下の実験をおこなった。

ＪＳＰによる設計記録のうち、問題解決者が思考した部分を取りだし、文単位にその活動の型を分類した。これを基本タスクと呼ぶことにする。ＪＳＰの設計プロセスについて基本タスクを洗いだした結果、以下の7種類のタスクが得られた。

(1)分析　(2)推測　(3)理解不能
(4)決定　(5)訂正　(6)確認
(7)うまくいかない

これらの基本タスクは思考の基本単位である。しかし、設計時の活動単位はこれらの基本タスクが組合わされることによって実現されている。そこで基本タスクの系列を以下の方針に従って検討してみた。

すなわち、人間は一連の思考を、「分析の必要がある」や「理解できないことやうまくいかないことがある」という状況の認識から始め、「このように決定しよう」とか「このように訂正しよう」といった状況で終了する。

このような基準をもとに基本タスクの出現系列(複合タスク)を分析・分類した。その結果、１０種類の複合タスクが得られた。

(1)分析→分析　(2)分析→決定
(3)分析→推測→決定　(4)分析→確認→決定
(5)分析→理解不能　(6)理解不能→推測
(7)理解不能→うまくいかない
(8)うまくいかない→分析→決定
(9)うまくいかない→分析→訂正
(10)うまくいかない→分析→推測→決定

こうしてえられた複合タスクは、我々の日常の設計活動時の思考の流れとよく一致するものと思われる。

2.2. 時間系列情報の保存

２．１の考察により、事例は複合タスク毎に分割されて保存されることになる。しかし、一つの事例を取りだしたとき、それまでの作業の流れ(コンテクスト)が理解できなければ、事例断片の内容を正確に理解することができない。このため、事例断片間の時間系列の情報も格納することにする。

2.3. 複合タスク毎の索引構造の付加

２．１で述べた分類法による一つの単位を考えるとき、複合タスクの概念は非常に一般的なものであり、その中に存在する事例断片群の一つを取りだすためには、さらに付加的な情報が必要である。そこで、各事例断片を取りだすための索引を設計した。ここでは、「分析→分析」、「分析→決定」、「うまくいかない→分析→決定」の３つの分類についての設計結果を示す。

(1) 分析→分析

このような複合タスクは、例えば問題を整理し理解するところに表れる。問題の整理・理解では、初心者は開発するプログラムの機能に注目することが多い。そこで、問題中にあらわれる対象物は無視してプログラムの「機能」のみを洗いだした。具体的にはプログラムの入力を出力に「変換」する記述を着目してそれを分類木の形(図１)で表した。

分類木の葉節点の下には、それぞれ該当する事例が存在する。また、１つの事例が、複数の葉節点に該当

Vela:A Software Development Environment for Cooperation Support
Naotake SHIMOZU, Kenji FUKUI, Takahira YAMAGUCHI and Koichiro OCHIMIZU
Shizuoka University

することもある。

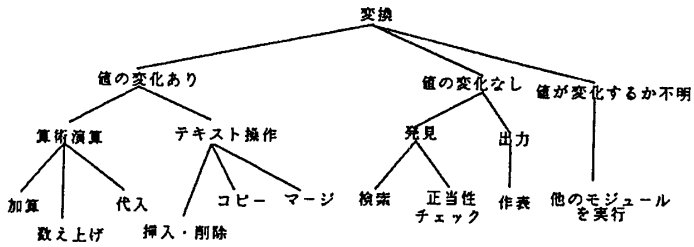

図1 プログラムの機能に関する分類木

(2) 分析→決定

ＪＳＰにおいては「入力木と出力木を分析して適切な技法を決定する」というプロセスがもっとも重要となる。そこで(2)にあたる分類木として、「技法選択」の分類木を用いた[1]。

(3) うまくいかない→分析→決定

うまくいかない状況に関して、「トラブル発生への対処」の分類木を用いた[1]。

これらの分類木をもとに簡単な検索実験をおこなった。実験は簡単な検索プログラムをつくり、被験者に検索をおこなわせた。実験の方法としてはあらかじめ、被験者に与える問題の解決にもっとも有用であると思われる事例を選んでおく。次に被験者にその問題をみせて、検索プログラムを使ってその問題の特徴を代入してもらう。こうして、検索された事例にあらかじめ選んでおいた事例が含まれていれば、有効な事例が検索されたといえる。

2.4. 評価

この実験の結果、次の点が考察できた。

(1) 機能に関する分類木に対して、技法選択プロセスの分類木のほうが有効な事例だけを検索できる割合が高かった。これは開発プロセスに依存して得られる分類木の方がより有効であることを示している。

(2) 有効と考えた事例の他に現在の問題とは関係ない事例が検索されたこともあった。これは、分類木の洗練化が十分進んでいなかったためである。

3. おわりに

本実験で使用した索引木には、さらに改善されるべきノードが幾つか含まれ，その改善プロセスを支援する環境を必要である。そのためには、「再構成」、「洗練化」、「一般化」などの機能を実現していかなければならない(図2)。

再構成とは，最終的には，事例ベースから有効なチャンクを構築することであり，ユーザから助言として与えられた別のビューから，既存の索引構造および事例ベースを再構成する操作などが主となる．従って，スロットを独立したフレームとして見なせる機能などが必要となるが，ＲＬＬ[2]がこの機能を提供している言語であり，ＲＬＬを出発点として再構成機能を考察していく予定である。

また，洗練化とは，再構成より得られたチャンクの構造を洗練化することである．このような知識構造洗練化システムとしてはＭＯＲＥ[3]があり，「異なった結論部が同一条件部を持つ知識構造には，洗練化される余地が大きい」などの戦略が整理されており，このＭＯＲＥを出発点として洗練化機能を考察していく予定である。

最後に，一般化とは，洗練化より得られたチャンクの適用範囲を広げることである．この機能の実現には，帰納的学習の操作が利用可能であり，「条件削除」「定数化」「区間拡大」「上位概念置換」などの操作の適用可能性について考察していく予定である。

謝辞

本研究はＳＤＡコンソーシアムの補助金と、科研費重点領域研究(1)(課題番号02249109)の一部の援助のもとにおこなわれた。記して謝意を表する。

参考文献

[1] 中尾, 山口, 落水: ソフトウェアプロセスの分析と形式化: 第41回情報処理学会全国大会予稿集.

[2]R.Greiner: RLL-1: A Representation Language Language, Stanford heuristic Programming Project HPP-80-9 (1980).

[3]G.Kahn, S.Nowlan and J.McDermott : MORE : AnIntelligent Knowledge Acquisition Tool, Proc. of IJCAI'85, 1, pp.581-584 (1985)

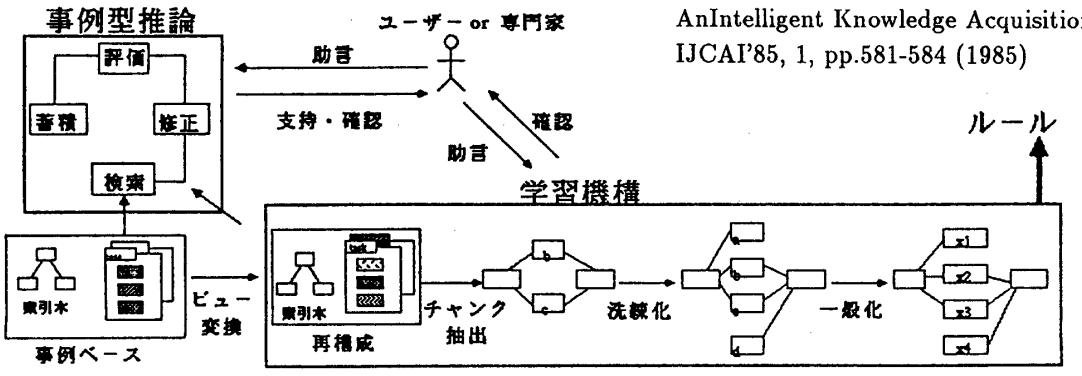

図2　知識獲得生成フェーズの構成