Vol.2009-DPS-141 No.12
Vol.2009-GN-73 No.12
Vol.2009-EIP-46 No.12
2009/11/26

処理事例共有と検索による データフロー型スクリプト作成支援

中山 健[†1,†2] 市野順子[†1]
橋山智訓[†1] 田野俊一[†1]

画像処理などデータに処理を順次適用していくデータフロー型の処理の作成を，一般ユーザが容易に行えるようにすることを目的として，多数のユーザが処理事例を共有し，事例の検索やまとまった処理の抽象化，データや処理の協調的分類を行なえる情報加工環境を提案する．従来のソフトウェア再利用手法と比較した特徴は，(1) 抽象化されたプログラムだけではなく，データも含む処理事例をリポジトリとして持ち，(2) それを多くのユーザが協調的に抽象化していく過程を明示的に支援する点である．本報告では，問題の分析とプロトタイプシステムについて述べる．

Sharing and Searching Execution Examples: A User Environment for Data Flow Scripts

Ken Nakayama,[†1] Junko Ichino,[†1]
Tomonori Hashiyama[†1] and Shun'ichi Tano [†1]

One style of data processing is data flow, such as image processing, in which the given data flows through a possibly branching series of functions to get the intended result. Programmers design a script by choosing and combining predefined functions (software components) to realize the intended task. To make this design easy even for novice general users, this report proposes a framework for sharing and searching execution histories. In this framework, user design the target task by interactively repeating trying out fragment of test scripts, searching related execution hisotry, examining them, and modifying/combining them. This report discusses the framework and presents the current state of this on-going project.

1. はじめに

1.1 一般ユーザによる非定型的処理の困難さ

インターネットは一般ユーザにまで広く普及し情報検索も容易になった．WWW は多数のユーザが参加する情報共有環境であり，Web 文書をはじめとしてソフトウェアや画像など多様で大量の情報が日々更新されている．一般ユーザにも普及しているのは，Web 検索エンジンやポータルサイトのコンテンツ一覧などが充実したことによって，一般ユーザでも求める情報に容易に辿り着けるからである．

これに伴い，得た情報を個々のユーザがいろいろな目的に応じて加工したいという需要も多くなっている．需要の多い一般的な処理，例えば Web クリッピングや類似画像検索は，徐々に利用可能になってきた．しかし，専用のアプリケーションプログラムが存在するような定型的・一般的な情報加工から少しでも外れると処理のカスタマイズが必要で，ユーザ，特に一般ユーザには困難である．例えば，次のような情報抽出・整形・複数のデータの比較などである:

- ニュースサイトの Web 文書から興味のある特定部分だけを抜き出して，RSS など指定した書式に整形する．RSS の自動生成には多くの研究やサービスがある[6),8),10),14)–16)]が，これらは Web 文書の典型的な見出し等の書式や内容を認識して標準的な項目一覧を作成することが主で，ユーザが欲しいと考える内容にカスタマイズしたものは得難い．
- 画像や動画から，特定の出演者など興味のある特定部分を同定して抽出する．
- ある特定の趣味に関する Web 文書が複数あるとき，それらの一覧・比較・分類を作成する．
- 定時運行されているか監視するために，風景動画中に電車が走っているかどうか自動検出する．

現在の計算機環境では，軽量言語 (スクリプト言語) を使用できるプログラミング能力未満のユーザには，希望するデータ処理を自ら作成して行なう手段は実質上全くない．一般ユーザのプログラミング能力は平均的にはそれほど高くないが，程度の差こそあれスクリプティング能力があるユーザは多くいるはずである．このような「自力でのスクリプティング

†1 電気通信大学
University of Electro-Communications
†2 津田塾大学
Tsuda College

© 2009 Information Processing Society of Japan

Vol.2009-DPS-141 No.12
Vol.2009-GN-73 No.12
Vol.2009-EIP-46 No.12
2009/11/26

未満」レベルの一般ユーザにも，複雑ではないデータ加工については手段を提供すべきであろう．

1.2 提案手法の概要

このように，インターネットの普及と一般ユーザへの広がりという状況変化に対応して，従来のソフトウェア再利用手法をさらに発展させる必要がある．本研究は，データだけでなく処理方法も多くのユーザが共有でき，個別の目的に応じた的確な検索と修正が容易であればこの困難が解決できるという点に着目し，ソフトウェア部品検索，例示プログラミング，等の手法をデータフロー処理に特化・統合したスクリプティングの枠組みを提案する．これは粗粒度のソフトウェア部品 (内部状態を持たない関数) の組み合せによるデータフロー型スクリプト作成を対象とし，多数のユーザが処理事例を共有し，一般ユーザにも使用が容易な情報加工環境である．従来のソフトウェア再利用手法と比較した特徴は，(1) 抽象化されたプログラムだけではなく，データも含む処理事例をリポジトリとして持ち，(2) それを用途に応じて人手で，あるいは半自動的に抽象化していく過程を明示的に支援する点である．なお，条件分岐や繰り返しなどの制御構造を持たないものは，マクロと呼ぶ方がふさわしいかも知れないが，ここではこれも含めてスクリプトと呼ぶ．

次の 2 章で一般ユーザへのスクリプティング支援の背景とスクリプティング支援に求められる要件について検討したあと，3 章で本研究が対象とするデータフロー型の処理について述べる．4 章で提案手法について述べ，続く 5 章で提案手法についていくつかの観点から議論を行ない，6 章で今後の課題について述べる．

2. 一般ユーザへのスクリプティング支援の背景と要件

2.1 従来の一般的ソフトウェア開発との比較

一般ユーザが通常行いたいと考える情報加工のタスクには，注目すべき特徴がいくつかある (表 1)．求められる処理はそれほど複雑ではなく小規模である．使用される回数も多くはなく，開発過程で仕様を確定したり記録を残したりしながら手順に従うコストが相対的に大きくなる場合が多い．かけるコストと得られる利得のバランスが重要である．また，極めて高い性能や正確さは通常求められない．処理は種々多様であるが，多くのユーザをまとめてみれば，類似の傾向がある．また，一般ユーザの平均的プログラミング能力は，それほど高くない．

ソフトウェアの再利用は，ソフトウェア開発を容易にするために有望であり，ソフトウェアの部品化やコード検索等の再利用手法も多く研究されてきた．近年になってアジャイル開発のように小規模でテストを中心とした手法も出現し，最近はソフトウェア開発の際にインターネット検索などを利用して手本や手掛かりを得ることは一般的になっている．しかし，従来の手法はソフトウェア産業の中心である計画的・組織的開発の大規模ソフトウェアを主な対象として発展してきており，ソフトウェア再利用法も既存ソフトウェアを理解して変更する事などプログラミング能力が前提であった．その結果，システム開発の専門家しか使えず一般ユーザには有効ではなかった．汎用のプログラミング言語を直接扱わないという点では，求める機能を容易に得る手段として，予め用意された各種ソフトウェア部品 (コンポーネント) の組み合わせ変更やパラメータ調整等によって構成する手法が有望である．

従来の一般的なソフトウェア開発で用いられるソフトウェア部品は，主として商品として開発されたもの (commercial off-the-shelf components: COTS) で，比較的少数のソフトウェア部品で無駄なくいろいろなタスクが実現できるように緻密に設計されていて，性能も高い．一方，ソフトウェア部品は多くのユーザによっても非組織的・個別に多数作られていて利用可能ではあるが，類似機能のものが多数存在し機能が直交していない．ユーザが開

表 1 一般ユーザによる情報加工タスクの特徴
Table 1 Characteristics of end user data processing tasks.

	従来の一般的ソフトウェア開発	本研究の対象
情報加工のタスク		
種類	個別	多様だが傾向あり
規模と複雑さ	大・複雑	小・単純
要求される信頼性・性能・精度	高い	一般に高くない
多様性・利用頻度	狭い・高い	広い・低い
条件分岐・繰り返し	通常は含む	少ない
イベント処理・データベース処理	通常は含む	ない
ユーザインタラクション	通常は含む	ない
機能仕様の明示	一般に可能	非明示的なものも有り
開発者		
種類	システム開発の専門家	一般ユーザ
プログラミング能力	高い	平均すると低い
利用可能なソフトウェア部品		
開発形態	計画的	非計画的・非組織的
提供される形態	主に商用部品 (COTS)	ユーザによる公開
種類数・機能直交性	少ない・高い	多い・一般に低い
説明書	充実し高品質	乏しく低品質
全体としての記述力	狭い (問題領域専用)	広い
更新頻度	低い	高い

© 2009 Information Processing Society of Japan

Vol.2009-DPS-141 No.12
Vol.2009-GN-73 No.12
Vol.2009-EIP-46 No.12
2009/11/26

発したこのようなソフトウェア部品は，仕様の不統一，ドキュメントの不足などの理由で，検索と使用までに必要な理解の労力と手間が大きい．統一的な分類が難しいうえ，定常的に更新され，数も非常に多いため内容記述を作成する付加的コストが大きくなる．利用するための使用方法説明文書や利用環境の入手が個別で難しく，ソフトウェアやデータがどういうものあるいは内容でどういう使い方をすべきなのか等の情報が得にくい．自然言語で書かれる場合は，外国語が読めないと理解できないという問題も生じ，有効活用が難しい．非組織的・個別に作られて公開されているデータについても状況は同じである．

多数の一般ユーザにより作成・開される情報は相互の整合性確保や機能分担などの調整がほとんど行なわれないが，ソフトウェアの種類もデータも，そして利用者の数も非常に多く，総体として傾向があると期待される．これはこれらの潜在的有用性が高く，これらを有効に活用できれば，個々の利用者が必要とするソフトウェアやデータが容易に得られるという協調の利点が得られ，ソフトウェア再利用の有効性が高いと考えられる．つまり，一般的でない特殊な事をしようとしない限り，これらを使って実現できる可能性が高い．

2.2 一般ユーザ向けスクリプティング支援の要件

一般ユーザ向けスクリプティング支援には，次のような要件がある:

- データが容易に把握できるような種々の可視化等のユーザインターフェースがあること．
- 検索は定型的なものだけでなく，種々の部分的な手掛かりから行なえること．従来は，利用者が自らの利用意図を表現する手段に乏しいことにより，検索や理解が困難であった．必要とされる検索様態は多様であり，検索意図の簡潔で統一的な表現方法を確立し，いろいろな検索手段を提供すべきであろう．再利用手法が有効であるためには，所与のソフトウェア部品群が対象問題領域において (a) 高い記述力をもつとともに，利用者がある機能を必要としたとき，それを実現する適切な構成を決定するスクリプティングが容易でなければならない．具体的には，(b1) 関連しそうなものの検索，(b2) 見つかったものを理解して目的に応じて変更する事が必要である．
- 試行が容易にできること．一度の試行を行なうために必要なスクリプトの準備，試行用データの準備，結果の確認が容易に，しかも短い時間でできること．
- 目標の処理やデータと類似していると思ったとき，そこから目標へ向かってどのように近付いていけばよいかが容易に分かること (山登り)．
- ユーザが容易に論理を把握したり記憶したりできる複雑さには限界がある．自らが理解した事や作ったものを，完成された状態だけでなく途中の過程においても整理して記録しておき，後から参照・変更できること．
- 多人数で作業するときは，他人の知識も共有できること．ただし，他人は同じ観点で知識を記録しているとは限らず，場合によっては悪意によって間違った知識を登録する場合もあるので，取捨選択が可能であること．

蓄積されている処理事例やソフトウェア部品の種類は多い方が良いが，効率よく的確に検索ができるという前提が満たされている必要がある．また，一般的な要件としては，登録されている事例の品質が高いこと (たとえば，入力として HTML が必要な処理部品に対して画像を入力するなど処理結果がナンセンスなものは低品質である) や，できるだけ多様な目的の処理を含んでいることなどが望まれる．

3. データフロー型の処理

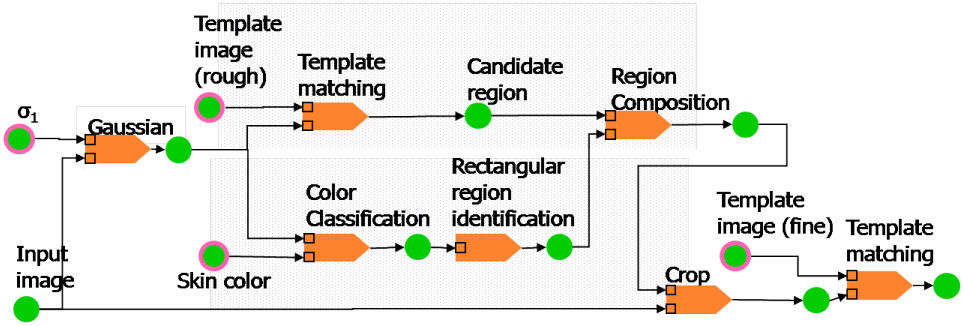


図 1 データフロー型の処理の例
Fig. 1 An example of data flow style processing.

画像処理などデータに処理を順次適用して加工していくデータフロー型の処理は，一般ユーザが求める処理の主要なものの 1 つである．データフロー型の処理は，1 個あるいは複数個のデータを処理部品に入力し，処理結果をさらに次の処理部品に入力するという連鎖で目標の処理を達成する．本研究では単純のために，処理部品には内部状態が無く出力は入力データによってだけ決まる関数型で，出力は 1 つだけのものを考える．入力データの個数は処理部品ごとに固定とし，複数ある場合は何番目の入力なのかが区別される．

図 1 は，画像処理の事例を模式的に表わしたものである．丸印がデータを，長方形が処理部品を，そして有向辺がデータの流れをそれぞれ表わす．この例は処理事例なので変数は無く，データはすべて値を持っている．同様に処理部品は単に種類を表しているだけではな

© 2009 Information Processing Society of Japan

Vol.2009-DPS-141 No.12
Vol.2009-GN-73 No.12
Vol.2009-EIP-46 No.12
2009/11/26

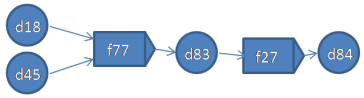


図 2 処理による依存関係の例
Fig. 2 Dependency among data and a function.

く，その処理部品がそれぞれの入力データによって 1 度実行されたという事を表わす．したがって，データは外部から与えられて入る有向辺が無いか，処理の結果生成されて入る有向辺が 1 本かのどちらかの場合しかない．一方，データは複数の処理の入力となり得るので，データから出る有向辺は複数の場合もある．また，処理結果で得られたデータが前段階に戻るループは存在しない．

ある入力データを処理結果して結果を得たとき，この処理を介して入力データに依存関係がある．同様に，ある処理結果を入力してさらに次の処理をした場合，前後の処理同士は中間結果を介した関係があると見なせる．ソフトウェア部品への入出力データやソフトウェア部品自体についてユーザが部分的な知識を持っていれば，他の部分についても推測ができる．たとえば，図 2 の処理が行なわれたとき，もしユーザが「データ d18 は貿易統計である」と知っていたとすると，他のデータやソフトウェア部品も「貿易統計関係である可能性が高い」と推測できる．また,「ソフトウェア部品 f77 は，f27 の前段に使われるものだ」という推測も可能である．

4. 提 案 手 法

4.1 試行錯誤による開発

ソフトウェア開発は，意図する機能や実装・使用するデータなどに関する断片的知識を出発点として求める実装を得るまでの探索的最適化過程である．ユーザ (開発者) は目的に応じてリポジトリ内の種々の類似事例を検索・取捨選択・修正・総合し，試行結果の妥当性を手掛かりとして対話的・探索的に機能実現を行う．たとえ部分的であっても，求める機能と類似した例示的仕様や実装事例が見付けられれば，それを手本とすることで実装や保守が容易になる．

ソフトウェア部品の適用履歴をデータおよびソフトウェア部品の相互関係と考え，リポジトリをコード領域 (スクリプト=ソフトウェア部品の組み合わせ方) だけではなく，トレース領域 (処理事例および関連するデータ) にする．これにより，内容の深い理解をしなくても検索結果を試行でき，処理結果を参考にして機能を判断する事が容易になる．試行用入力

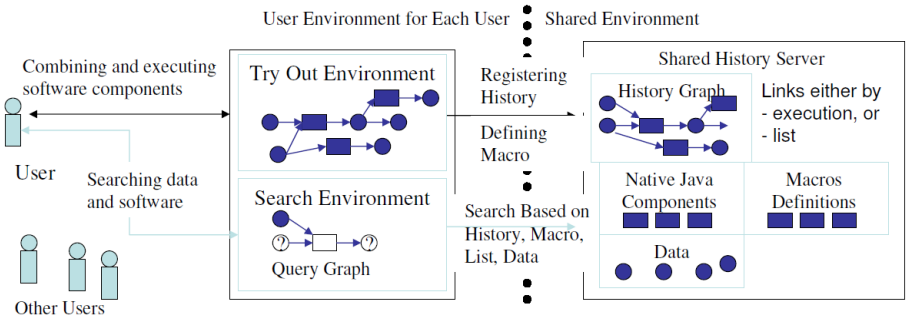


図 3 システム構成
Fig. 3 Overview of the prototype system.

データも豊富に得られ試行が促進される．データ検索とコード検索は相補的であり，データを手掛かりとしてコードを検索，あるいはその逆もできる．利用頻度の情報も得られ，使用され方を反映した処理事例マイニングが可能となる．

再利用に十分なリポジトリを予め準備するのは現時的ではない．そこで，複数の開発者が個々の開発事例を分類・共有すれば，協調的に追加・類型化される事により，ソフトウェア部品群に未知の部品が非組織的・定常的に追加され続けているような場合でも開発者は絶えず蓄積・洗練され続ける豊富な処理事例と分類が手本にでき，他利用者の使用経験を参考にして開発・保守が容易になる．

4.2 処理事例の共有・分類・検索

システムは，すべてのユーザが共有する部分と，個々の一般ユーザが使用するユーザ環境に分かれている (図 3)．ユーザで共有する部分にはデータや処理部品が共有されているほか，(1) データフロー型の処理事例リポジトリ (実行履歴) と，(2) ユーザがいくつでも自由に作成・変更できるブックマークを保持している．処理事例リポジトリは有向グラフ (履歴グラフ) で表現され，使用されたデータがどれであるかも併せて保存されている．ブックマークは，ユーザが処理方法を抽象化する過程と結果で得た知見を表現するために使用する．ブックマークには，データや処理部品，履歴グラフの一部 (部分グラフ)，処理部品やデータの変数を含む変数化グラフが登録できる．他のユーザが作成したブックマークも参照できるので，ユーザ間で協調してデータや部分グラフを分類することもできる (処理やデータの類似関係の抽象化)．データや処理事例間の類似や対応等の関係を表現する共通の枠組みを提供し，多くの開発者がこれらを追加・分類し続けるリポジトリを共有する．このリポ

© 2009 Information Processing Society of Japan

Vol.2009-DPS-141 No.12
Vol.2009-GN-73 No.12
Vol.2009-EIP-46 No.12
2009/11/26

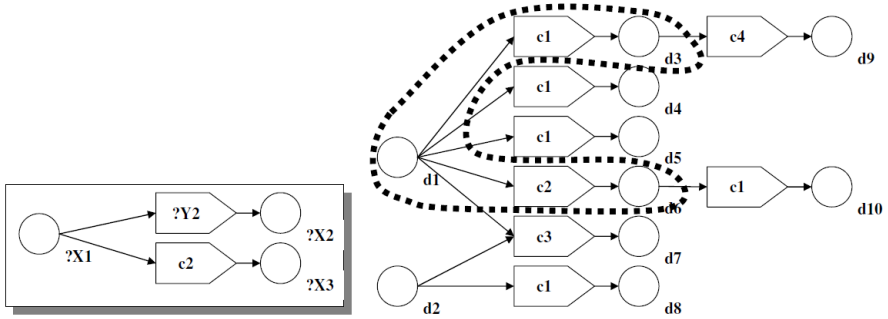


図 4 問い合せグラフ (左) とマッチ結果 (右) の例
Fig. 4 Examples of query graph and matching result.

ジトリには，各開発者の様々な視点からの多様な類似構造が埋め込まれていくことになる．ユーザは，ユーザ環境を通して共有部分のデータや処理部品をブラウズし，適当と思うものを組み合せて処理を実行できる．どのように組み合せて実行したかは，実行の結果生成されたデータと共に処理事例に追加されていき，これもユーザ間で共有される．

また，履歴の検索，ブックマークの検索，そしてそれらを組み合せた検索もできる．履歴に対する問い合わせは，データや処理部品の一部に変数化したものを含む問い合せグラフを用いて表現し，履歴グラフ中のマッチする部分が返される (図 4)．この例では "?" で始まる名前が変数を表わしている．ブックマークに対しては，指定したものが含まれているブックマークを探すという検索ができる．検索では，例えば次のような事ができる:

- **試行の構造に基づく検索:** データ d2 が生成されたものの場合，他のどのデータを元にしてどのような処理で生成されたかの由来を知る．
- **分類構造に基づく検索:** 数枚の笑顔データが共通して含まれる分類を知ることにより，笑顔データと推測される他のデータを探す．同様に，数個の処理部品からそれらと類似する処理部品を探す．
- **試行あるいは組み合せ構造に基づく検索:** 処理部品 f43 の入力の前処理として，どのような部品が使われる事が多いかを知る．
- **組み合せ構造に基づく検索**処理部品 f43 を組み合せに用いている部品定義を探す．
- **自動分類・実行比較による検索**処理部品を使って，データや処理部品自体の分類を自動生成する．逆に，分類を与えてそれに近い分類を生成するような処理部品を検索する．

変数化したデータや処理部品を含むグラフは，問い合せグラフとして用いることができる

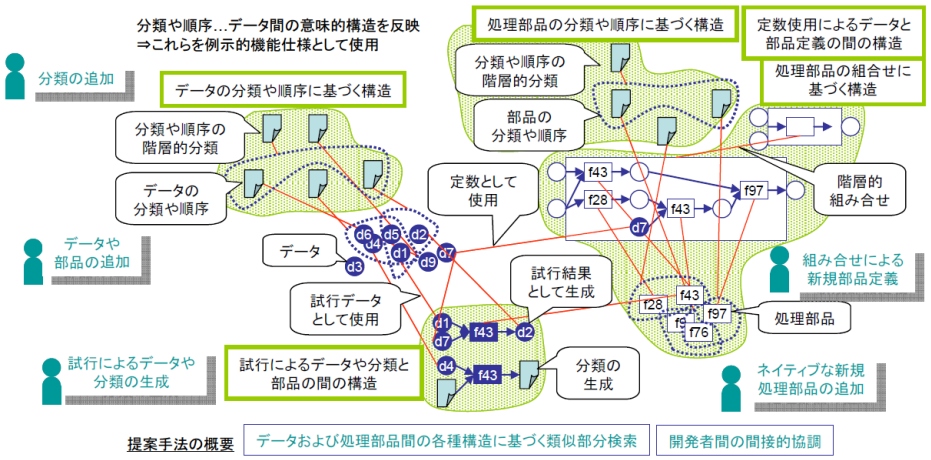


図 5 提案手法の枠組み
Fig. 5 The overview of the proposed framework.

ほか，処理の出力に相当するデータがすべて変数化されているものは実行可能で，スクリプトとして扱われる (処理の抽象化)．

提案手法の枠組みを図 5 に示す．提案手法は，キーワードや分類によるソフトウェアやデータの検索等と対立するものではなく，例えばキーワードで探したデータを最初の手掛かりとして，引き続き提案手法でソフトウェア開発を続けるといったように，相補的な利用を想定している．

4.3 プロトタイプの実装

プロトタイプシステムは，基本となる実装[7),9)] を拡張した．処理事例等を表わす有向グラフは Prolog の事実節で表現されており，問い合せも Prolog の単一化によって実現している．処理部品自体は，画像処理を対象問題領域として Java で実装し，実行トレースが Prolog の事実節の形式で出力される．Prolog の単一化を用いたことによりグラフ問い合せの表現力は高いがスケーラブルではない．既存の Java クラスも，もインターフェースを統一するラッパーで対応でき，移行の手間は大きくない．プロトタイプシステムの画面例を図 6 に示す．

© 2009 Information Processing Society of Japan

Vol.2009-DPS-141 No.12
Vol.2009-GN-73 No.12
Vol.2009-EIP-46 No.12
2009/11/26

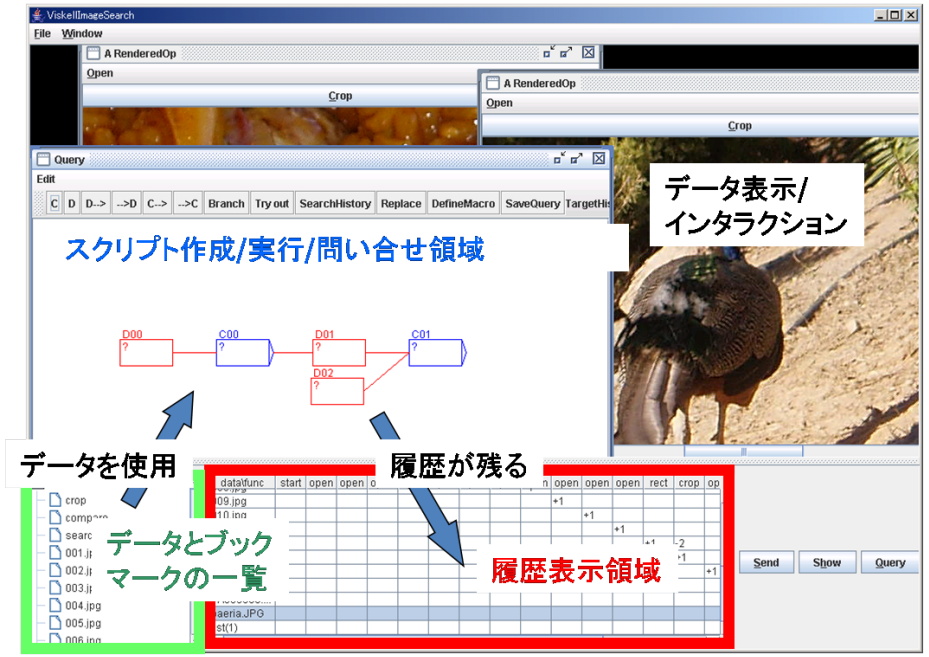


図 6 プロトタイプシステムの画面
Fig. 6 The prototype system.

5. 議論および関連研究

5.1 ソフトウェアの再利用

Java のクラスをソフトウェア部品と考え，ソースコードから使用関係を抽出して汎用性が高いかどうかを推測する研究がある[3),4)] が，コード領域を対象としたものである．また，画像処理を対象として，専門家による処理事例を蓄積しておいて，初心者が検索によって処理方法を知る試み[11)–13)] があり，これは処理の過程の画像をデータベースの版として保存している．検索は，処理の有向グラフに対する正規表現に近いものを定義している．いずれも，分類や抽象化の機能は含まれていない．

ソフトウェア部品ではなく，プログラミング言語のソースコードを対象として検索する手法には，例えば，例となるソースコードが蓄積されているリポジトリーから，作成中のソースコード自体を問い合わせとして使用して，構造が類似している例を取り出し，その理由も同時に提示するシステム[2)] がある．コード領域を対象としたもので，分類や抽象化の機能も含まれていない．プログラミング言語の理解が必要であることから，一般ユーザ向きではない．ただし，(1) 作成中のコード自体が問い合せなので，問い合せ言語を覚えなくてよい，および，(2) リポジトリは構文解析が可能なソースコードからなら容易に自動作成できる，という利点は本研究と共通する．ソフトウェアの構造ではなく，ユーザごとにどのようなコンポーネントを使う傾向があるかをフィルタリングして推薦するもの[5)] もあるが，データを手掛かりとしたものではない．

実行時の呼び出し関係のトレースを記録するものは，古典的な[1)] をはじめとして多くの研究があるが，これらは実行速度の向上など性能評価を主眼としており，プログラム作成の支援を目的とするものではない．

5.2 具体例による抽象度と手間の交換

ソフトウェアは，仕組みの理解よりも処理結果が意図に沿うかどうかの判断の方が遥かに容易である．具体例の対話的試行錯誤の手間と引き換えにプログラミングの抽象度を下げられ，一般ユーザに適している．極端な場合で言えば，ソフトウェアの仕組みをほとんど理解していなくても，多くの候補ソフトウェアについて試行を繰り返せば，求めるソフトウェアと考えられるものを発見できる．

もちろん，具体例によって判断する限りでは，期待する動作をいくら多くのデータに対して得たとしても確実とは言えないが，実際上はそれで十分な場合も多く，必要があれば候補ソフトウェアを絞っておいて動作の理解に注力すればよい．この方法によれば，ユーザの理

© 2009 Information Processing Society of Japan

Vol.2009-DPS-141 No.12
Vol.2009-GN-73 No.12
Vol.2009-EIP-46 No.12
2009/11/26

解の度合いに応じて，ソフトウェアを理解するという抽象的な作業と試行の手間を交換できることになる．試行によってソフトウェアを探す場合には，網羅的に候補を列挙できることが前提となる．

具体例による方法は，仕様書に基づくソフトウェア開発手法よりも適用できる問題領域が広いという利点もある．パターン認識等の機能仕様は，例えば「笑顔の写真の集合」という非明示的な方法で規定するしかないが，仕様書から詳細化を行なう従来のソフトウェア開発手法では，非明示的機能仕様の表現が困難である．事例に基づく提案手法では，これが可能となる．

5.3 リファクタリングとの関係

一般的なオブジェクト指向言語は継承に基づくクラス階層を持ち，システムが扱うデータや機能をどのクラスに所属させるかを十分に考えて設計する．プログラムのリファクタリングはこの過程である．適切な設計を行なっておけば，クラス間の機能やデータの直交性が高まり，他のシステムを実装する際にも少ない変更で再利用できるからである．

しかし，少ない変更で再利用できるということは，それぞれのシステム固有の実装に関する情報はそれぞれのクラスには無く，それらの組み合せ方に集約されているということになる．クラスライブラリの説明書はそのクラスについての説明は豊富だが，そのクラスが他のクラスとどのような関係を持って使われるのかという説明は，一般にはあまり豊富に提供されているとは言えない．また，説明が提供されていても，それを試行するための完成したプログラムや試行用データは，さらに入手が困難である．

6. おわりに

インターネットの普及と一般ユーザへの広がりという状況変化に対応して，従来のソフトウェア再利用手法をさらに発展させるため，データだけでなく処理方法も多くのユーザが共有でき，個別の目的に応じた的確な検索と修正が容易であれば，この困難が解決できるという点に着目し，ソフトウェア部品検索，例示プログラミング，等の手法をデータフロー処理に特化・統合したスクリプティングの枠組みを提案した．今後検討すべき課題には，次のようなものがある．

- 検索機能をもつ関数型リフレクティブ言語としての体系化
  現在のプロトタイプシステムでは，処理事例やブックマークの検索結果はデータやサブグラフとして提示されるだけである．検索を行なうという処理自体も履歴には記録されているものの，それを利用する手段は特に提供していない．しかし，検索結果を単に提示するだけでなく,「一部を変数化，置換，他のグラフと合成するなどして実行可能なグラフを作成し，それを自動実行して次の試行を行なう」「実行結果によって次に行なうべき問い合わせやブックマークを生成する」などが可能になれば，ある程度以上のプログラミング能力のあるユーザなら試行が効率的に行える．
- 処理事例リポジトリやブックマークのマイニング
  現在のプロトタイプシステムでは，処理事例に対する検索しか提供していない．しかし，特定のソフトウェア部品の種類やデータに注目すれば，その種類によく入力されるデータや頻繁に共に使われる部品など，利用状況の統計的な情報が得られる．この情報やブックマークをマイニングすれば，典型的な利用法や，利用法同士の相互の関係などが得られる可能性がある．
- 推薦スクリプティングユーザインターフェース
  マイニングで情報が得られたならば，スクリプトを作成しつつある一般ユーザに対し，データやソフトウェア部品，あるいはそれらを部分的に組み合せたものを手掛かりとして，それをどう発展させるのが典型的で適切かの推薦を提示できる可能性がある．提示には，ユーザの思考を妨げず容易に理解・利用できるユーザインターフェースが必要である．特に，一般ユーザが使えるようにする事を目標としているので，単純で直観的なインターフェースであることが望ましい．
- 処理事例取得の工夫
  現在のプロタイプシステムでは，ユーザが行なった試行はすべて記録され，それがユーザの意図に合って成功したのかあるいは失敗したのかの情報は，ユーザが明示的に分類しない限り記録されない．逆に，実行時にエラーが出たような明らかな場合は，試行は記録されない．成功/失敗の情報を，ユーザのインタラクションを阻害することなく自然に取得できるような工夫が必要である．どのユーザが，どういう目的で行なったものかという情報も記録できればよいだろう．
- 評価方法
  提案手法の有効性は，基本的には対象とする問題領域 (たとえば画像処理) の特性に依存するため，評価方法が問題となる．しかも，有効性はリポジトリの規模の大きさと質の高さに応じて大きくなると考えられ，ある規模以上の実験対象が無いと評価が難しくなる可能性がある．

今後，プロトタイプによる実験と評価を通じて提案手法を改良し，有効なシステムとして実現していきたい．

© 2009 Information Processing Society of Japan

Vol.2009-DPS-141 No.12
Vol.2009-GN-73 No.12
Vol.2009-EIP-46 No.12
2009/11/26

謝辞 有益な議論をさせて頂いた大須賀 昭彦・田原 康之・中川 博之の各氏に，謹んで感謝の意を表する．

参 考 文 献

1) Graham, S., Kessler, P. and Mckusick, M.: Gprof: A call graph execution profiler, *ACM Sigplan Notices*, Vol.17, No.6, p.126 (1982).
2) Holmes, R., Walker, R., Murphy, G. et al.: Approximate structural context matching: An approach to recommend relevant examples, *IEEE Transactions on Software Engineering*, Vol.32, No.12, p.952 (2006).
3) Inoue, K., Yokomori, R., Yamamoto, T., Matsushita, M. and Kusumoto, S.: Ranking significance of software components based on use relations, *IEEE Transactions on Software Engineering*, Vol.31, No.3, pp.213–225 (2005).
4) Inoue, K., Yokomori, R., Fujiwara, H., Yamamoto, T., Matsushita, M. and Kusumoto, S.: Component Rank: Relative Significance Rank for Software Component Search, *In Proceedings of the 25th international conference on Software engineering*, IEEE Computer Society, pp.14–24 (2003).
5) McCarey, F., Cinnéide, M. and Kushmerick, N.: Rascal: A recommender agent for agile reuse, *Artificial Intelligence Review*, Vol.24, No.3, pp.253–276 (2005).
6) MyRSS.jp: MyRSS.jp, http://myrss.jp/.
7) Nakayama, K., Shibata, M., Yazawa, S., Kobayashi, Y., Maekawa, M. and Okamoto, T.: Sharing and Searching History: Collaborative Component and Data Reuse, *Communications and Information Technologies, 2006. ISCIT'06. International Symposium on*, pp.580–583 (2006).
8) NTT Resonant: goo RSS 作成，http://fm.goo.ne.jp/.
9) 中山 健，谷沢智史，鄭 捷聰，野中貴俊，小林良岳，前川 守：関数型ソフトウェア部品の適用履歴を用いたスクリプティング支援環境，情報処理学会論文誌. プログラミング， Vol.47, No.2, p.100 (20060215).
10) 新納浩幸，佐々木稔：Web ページ内の目的部分の自動抽出，情報処理学会自然言語処理研究会, 2003-NL162-6, pp.30-40 (2003).
11) 川島享， 田幡勝，金森吉成，増永良文：画像オブジェクトの版管理モデル，電子情報通信学会論文誌 D， Vol.79 (1996).
12) 田幡， 有次， 金森：半構造データモデルによる画像処理履歴の管理，情報処理学会論文誌 データベース， Vol.41, pp.64–75.
13) 田幡勝，有次正義，金森吉成：2000-DBS-122-30 半構造データモデルによる画像処理履歴の問合せの性能評価，情報処理学会研究報告. データベース・システム研究会報告， Vol.2000, No.69, pp.229–236 (2000).
14) 南野朋之，奥村 学：RSS 自動生成のためのタイトル生成，言語処理学会第 11 回年次大会, pp.57–60 (2005).
15) 南野朋之，奥村 学：なんでも RSS - HTML 文書からの RSS 自動生成，人工知能学会第 19 回全国大会 (JSAI2005) (2005).
16) 南野朋之，齋藤 豪，奥村 学：繰り返し構造に基づいた Web ページの構造化，情報処理学会論文誌, Vol.45, No.9, pp.2157–2167 (2004).

© 2009 Information Processing Society of Japan