RK-001

グローバルフックを用いたプログラミング過程可視化システム

Programming Process Visualization System using a Global Hook

森田 直樹†
Naoki Morita

１．まえがき

本研究では，プログラミング演習の授業において，受講者が PC に対して行ったすべての操作を，OS のシステムメッセージよりグローバルフックを用いて取得するシステムを開発したので報告する．

プログラミング演習において，受講者や教師を支援するシステムは，数多く存在する．特にプログラミング演習では，ソースコードを編集する過程が重要である．ソースコードを作成する過程には，受講者がどのように演習に取り組んだのかを講師が判断する情報が豊富に含まれる．従来の方法は，ソースコードを編集する過程を取得するために専用のエディタが必要であった．

本研究で開発したシステムは，専用のエディタを用いることなくソースコードを編集する過程を取得することができる．具体的には，PC を操作することで発生する OS のシステムメッセージを解析することにより取得する．これにより，従来では不可能であった既存のソフトウエアを用いてもソースコードを編集する過程を取得することができる．さらに，本システムは，ソースコードを編集する過程だけでなく受講者が行うプログラミングに関するすべての過程を蓄積することができる．

２．先行研究とその問題点

C 言語や Java 言語などのプログラミングの講義では，(1)ソースコードを作成する，(2)コンパイラでソースコードを実行ファイルに変換する，(3)実行ファイルの動作を確認する，一連の工程を行う．そのため，構文エラーは，コンパイルエラーで確認することができ，アルゴリズムエラーは，実行ファイルの動作を確認することによりある程度可能である．プログラミングの授業は，多人数で行うことが多く，受講者の学習進度に差が生じやすい．そのため，講師にとって，受講者の状態を適切に把握することは困難である．これらの問題を軽減するために，受講者の理解の程度や進捗状況を講師にわかりやすい形で提供するためのさまざまな先行研究がなされている．

プログラミングは一連の工程を経る必要がある特徴を利用して，受講者が現在どの工程を行っているのかシステムが判断し講師に提供するシステム[1]~[8]がある．プログラムの基となるのは，ソースコードである．そのため，ソースコードを作成する過程には，受講者の理解の程度を講師が推測するための情報が豊富に含まれており，ソースコードを作成する過程を取得するシステム[2][3]が提案されている．また，ソースコード作成のためのヒントを与えるシステム[4]や，ソースコードの分析を助けるために抽象化し可視化するシステム[5]もある．さらには，コンパイルエラーの分析に特化したシステム[6][7]や，実行ファイルを自動的に評価するシステム[8]，学習者自身に間違いを考えさせ次回につなげる学習環境を構築する[7][9]など，さまざまな支援がなされている．

本研究では，ソースコードを作成する過程に着目する．ソースコードを作成する過程には，受講者がどのように演習に取り組んだのかを講師が判断する情報が豊富に含まれる．先行研究[2]や[3]では，専用のテキストエディタを開発することで，ソースコードの編集途中を取得することに成功している．しかし，専用のテキストエディタは，そのエディタが提供する機能しか利用することができない．たとえば，先行研究[2]や[3]が提供するエディタでは，デバッグのために変数の値を確認したり，動作の仕組みをトレースしたりすることができない．Windows アプリケーションを開発するときに用いる Visual Studio IDE では，デバッグのための機能も豊富に備える．そのため，専用のテキストエディタを用いることなく，より一般的なプログラミング環境において，ソースコードを作成する過程を取得できることが望ましい．

また，先行研究の多くは，受講者全員の進捗状況を把握することに主眼が置かれている．これにより，つまずいている学習者を早期に発見することが可能となるが，演習を繰り返していく中で状況は刻々と変化する．受講者の理解を確かなものにするためには，現在の状態をもとにコメントを返すだけでなく受講者が試行錯誤した過程を共に振り返りながら適切にコメントを返す必要がある．

３．本研究の目的

本研究の目的は，専用のエディタを用いることなくソースコードを編集する過程を取得し，かつ，コンパイルや実行ファイルの実行などのプログラミングに関するすべての動作を記録することである．さらに，理解の修正が必要な受講者には，受講者と共に試行錯誤した過程を振り返りながら適切なコメントを返すことができるように，その振り返りポイントを素早く探せるようにすることである．

本研究で想定するプログラミング演習の環境は，OS は Windows とし，実行ファイルを作成する方法は，Visual Studio IDE を用いてビルドする，または，Visual Studio コマンドプロンプトからコンパイラを起動しソースコードをコンパイルする方法をとる演習とする．

Visual Studio は，Microsoft 社製のソフトウエア開発ツールであり，アプリケーションを効率よく開発することができる．そのため，初心者からプロのプログラマまで利用する．Visual Studio を用いて実行ファイルを作成する方法は，ソースコードの作成からビルドまでのすべての工程を Visual Studio IDE 上で実施する方法と，ソースコードは別

† 東海大学 情報通信学部 通信ネットワーク工学科，
Tokai University

途作成し Visual Studio コマンドプロンプトからソースコードをコンパイルする２つの方法がある．

Visual Studio IDE 上で実行ファイルを作成する方法は，ソースコードを作成する前にプロジェクトを準備したり，Visual Studio IDE の操作を習得したりする必要がある．そのため初心者には，プログラミング以外の要素も習得する必要はあるが，眼には見えないメモリ空間上の変数の値を可視化したり，自分が作成したソースコードをどのような順番で実行しているのかトレースしながら確認したりすることができる．

Visual Studio コマンドプロンプトからソースコードをコンパイルする方法は，プログラミング以外の最低限の操作のみで演習を行うことができ，プログラミング初心者を対象とした授業で用いられることが多い．

本研究では，これら二つの方法において，プログラミング演習を行うすべての過程を取得することを目的とする．具体的には，受講者の PC 画面とその取得した画面の特徴をあらわすための情報を取得する．画面を取得するタイミングは，マウスのクリックイベントや，文字列の入力が確定したキーイベントが発生するタイミングである．特徴量として取得する情報は，マウスイベントの際は対象のウインドウ名であり，キーイベントの際はソースコードである．これらにより，受講者の演習過程の振り返りの際は，取得した画像を基に行うことができ，また，振り返りに必要な画像の検索は，同時に取得したテキスト情報に対して行うことができる．

４．グローバルフックを用いたプログラミング過程可視化システム

4.1 システム構成

本システムは，プログラミング過程取得モジュール，プログラミング過程蓄積モジュール，プログラミング過程再現モジュールから構成される．

プログラミング過程取得モジュールは，受講者の PC 上で動作し，受講者が PC を操作するイベントの取得とその時の受講者の PC 画面をキャプチャした画像をプログラミング過程蓄積モジュールに送信する．プログラミング過程蓄積モジュールとプログラミング過程再現モジュールは，Web サーバ上で動作し，蓄積モジュールは，プログラミング取得モジュールから送られたデータをサーバ上に蓄積し，再現モジュールは，Web ブラウザ上で受講者の PC 画面を再現するための HTML を形成する．

4.2 プログラミング過程取得モジュール

本モジュールは，受講者の演習過程を取得するアプリケーション OperationLog.exe とグローバルフックを行うための OperationLog.dll，Visual Studio IDE の機能を拡張するアドイン，Visual Studio コマンドプロンプト上での操作を取得するバッチファイル cl.bat から構成される．

4.2.1 演習過程として取得する情報とタイミング

Windows PC は，マウスやキーボードを用いて操作を行う．それらの操作の結果，アプリケーションが動作し，その結果が画面に反映される．そのため，受講者の PC 画面をキャプチャし蓄積することで，受講者が行った演習過程を後から振り返ることができる．本研究で開発した OperationLog.dll は，OS に対して SetWindowsHookEx 関数を用いてグローバルフックを行いウインドウメッセージを監視する．これにより，マウス操作やキー操作を操作対象となるウインドウに反映させながら，その操作イベントを取得することができる．本 OperationLog.dll が監視する項目は，WM_LBUTTONDOWN（マウス左ボタン押し下げ），WM_LBUTTONDBLCLK（マウス左ボタンダブルクリック），WM_RBUTTONDOWN（マウス右ボタン押し下げ），WM_CHAR（キーボードからの文字の入力）である．

本研究で開発した OperationLog.exe は，DLL が監視中に該当操作を取得したタイミングで受講者の PC 画面のキャプチャを行う．さらに，その該当の操作が，画面全体のどの部分に対しての操作であるかを確認しやすくするために，図 1 に示すように，キャプチャ画像の対応箇所に赤枠でハイライト処理を施す．これにより，キャプチャした画像を後から振り返った時に，ウインドウ全体に対してどの部分に行った作業であるのかを確認することができる．さらに，OperationLog.exe は，取得した画像を後から検索しやすくするために，GetWindowText 関数や GetModuleFileNameEx 関数を用いて，操作対象となるアプリケーション名やファイル名を取得する．これらの情報は，ユーザ識別情報やキャプチャ画像などと総合的に関連付けられ UDP でプログラミング過程蓄積モジュールへ送られる．

4.2.2 ソースコードの編集過程を取得する

プログラムの基となるのは，ソースコードである．そのため，ソースコードを作成する過程には，受講者の理解の程度を講師が推測するための情報が豊富に含まれている．OperationLog.exe を用いることで，ソースコード作成時の受講者の画面も取得することができる．しかし，蓄積してある大量の画像データの中から受講者にコメントを返すための適切な画像を探しだすことは困難である．そのため，ソースコードを編集している場合には，ソースコードも取得し画像データと関連付けて蓄積する手法をとる．これにより，本研究では未実装であるが，先行研究のようにシステムが受講者の演習状況を自動で判断するリソースとして利用することができる．

- **テキストエディタ上のソースコードの取得法**

メモ帳や Terapad などのテキストエディタは，ウインドウ内に Edit クラスのエディットコントロールが配置されており，ユーザは，その領域内で文字列の編集を行う．エディットコントロール内にある文字列は，WM_GETTEXT メッセージにより取得できる．この特徴を利用して，OperationLog.exe は，テキストエディタになりすまして OS に WM_GETTEXT メッセージを発行し，受講者が作成しているソースコードを取得する．取得したソースコードは，同時に取得したキャプチャ画像と関連付けを行い，プログラミング過程蓄積モジュールへ送信する．

- **Visual Studio IDE 上のソースコードの取得法**

Visual Studio IDE 上のコードエディタは，VsTextEditPane クラスから構成される．そのため，コードエディタ上で編集されるデータは，メモリ空間に存在するデータを基に描画データとして，つまり，画像としてユーザに提供される．そのため，役割ごとに色分けを施したり，あるまとまりを

折りたたんだ状態で表示したりすることができる．その一方で，テキストエディタのように WM_GETTEXT メッセージではコードエディタ上のソースコードを取得することができない．本研究では，コードエディタ上のソースコードを自動保存するアドインを開発することにより，ソースコードの作成過程をテキストデータとして取得できるようにした．

4.2.3 コンパイルの結果を取得する

ソースコードの作成に区切りがついた段階で，構文エラーや記述ミスがないかを確かめるために，また，実行ファイルを作成するためにコンパイルを行う．コンパイラの出力結果は，受講者の取り組みに対して客観的なフィードバックとなる．そのため，エラーがない場合は，または，エラーがあっても自分の力で間違いを修正できる受講者は，自分の理解を修正したり，理解を確実なものにしたりできる．一方で，プログラミングに苦手意識を持つ受講者は，やみくもに変更し始める要因となりえる．そのため，演習過程を振り返った際にコンパイラが出力したメッセージを確認できることは重要である．

- Visual Studio コマンドプロンプトの場合

受講者がコンパイルしたタイミングでコンパイルメッセージなどの情報をプログラミング過程蓄積モジュールへ送るために，コンパイルコマンドと同じ名前のバッチファイル cl.bat を記述した．バッチファイルに記述した内容は，通常のコンパイル処理に加え，コンパイルメッセージをファイル化し，ソースコード，メッセージファイル，ユーザ識別情報をプログラミング過程蓄積モジュールへ送信するための処理である．

- Visual Studio IDE の場合

Visual Studio IDE を用いてコンパイルを行う場合には，コンパイルメッセージは，該当のフォルダに HTML ファイルとして自動保存される．本研究で開発したアドインは，コンパイルメッセージが記述された HTML ファイルをユーザ識別情報と共にプログラミング過程蓄積モジュールへ送信する．

4.3 プログラミング過程蓄積モジュール

プログラミング過程蓄積モジュールは，Web サーバ上で動作する．本モジュールは，受講者の PC 上で動作するプログラミング過程取得モジュールから送られたデータを，ユーザ情報ごとに，画像データとその画像に関するウインドウ情報やソースコードなどのテキスト情報を時系列で管理する．

4.4 プログラミング過程再現モジュール

Web サーバ上で動作するプログラミング過程再現モジュールへのアクセスは，Web ブラウザを用いる．本モジュールが提供する情報は，HTML で構成されており 3 階層構造となる．

1 階層目である TOP 画面は，受講者全員の現在の状況が確認できるリストが提供される．リストは，テーブルタグを用いて構成されており，その要素は，学生証番号，最後に取得した画像に関連付けられたテキストデータである．テキストデータがソースコードの場合には，カーソル行の 1 行が表示される．これらの情報は，Ajax を用いて常にプログラミング過程再現モジュールと通信を行う．これにより，再読み込みの操作を行うことなく情報を更新し提供することができる．操作の過程を確認するには，学生証番号をクリックすることにより 2 階層目へ移動する．

2 階層目で提供される画面は，TOP 画面で選択した受講者のテキスト情報を時系列から逆順でリスト化したものである．リストの構成要素は，プログラミング過程蓄積モジュールが取得したタイムスタンプ，画像に関連付けられたテキスト情報であり，その情報がソースコードの場合には，編集行を，コンパイル項目の場合には，コンパイラのメッセージが表示される．コンパイラのメッセージに対しては，エラー時は赤色，成功時は青色で色分けが施される．確認したい時間帯のタイムスタンプをクリックすることにより，受講者の PC 画面のキャプチャ画像が表示される 3 階層目へ移動する．

3 階層目で提供される画面は，図 1 に示すような受講者の PC 画面のキャプチャ画像を，サムネイル形式で複数枚表示したり，1 枚ずつ表示したりすることができる．画像は，ボタンにより進めたり戻したりすることができる．

4.5 プログラミング演習過程再現例

図 1 は，プログラミング過程蓄積モジュールが蓄積した画像群の一部（12 枚分）を示したものである．この図は，sample という名前で用意された空のプロジェクトに対し，ソースコードを入力するまでの準備の工程（図中番号 1～8）と，「#include」とプログラミングを入力し始めた工程（図中番号 8～12）を取得した画像である．これらにより，受講者が演習過程に視覚した情報を確認することができる．再現画像からはマウスカーソルの動きを確認することができないが，操作対象となるウインドウは赤枠で囲まれているため，その部分にフォーカスをあてて確認することができる．

4.6 試用結果

プログラミングの第 1 回目の講義で利用した．受講者にとっては初めてのプログラミングであり，講義内容は，printf 関数を用いて表示する方法を題材に，プログラムができるまでの仕組みや，プログラミングの工程を体験することである．この日の演習環境は，メモ帳を用いてソースコードを記述し，コマンドラインからコンパイルコマンドを用いてコンパイルする形態をとった．

プログラミングは，複数の工程を順を追って実施する必要がある．講師が予測できる注意事項は，あらかじめ学生に説明することができるが，初心者ならではの素朴な間違いは，講師の予測をはるかに超える．このような間違いに関しては，取得した操作過程をスクリーンで再現しながら受講者全員に説明することができた．

講義が終わった後で，質問を受けた．その際に，その学生の演習過程を一緒に振り返りながら解説を行った．一見，でたらめに作成したとしか思えないプログラムであってもその過程を逆に再現することにより学生が間違えるきっかけとなった原因を確認でき，適切なコメントを返すことができた．

５． まとめと今後の課題

本研究では，プログラミング演習の授業において，受講者が行ったすべての操作を監視し取得するシステムを開発した．具体的には，特別なエディタを用いることなくソースコードの編集過程を取得したり受講者の PC 画面を取得したりすることができ，必要に応じて Web ブラウザで演習過程を再現できるシステムである．

本システムには，プログラミング演習の過程がテキストベースの情報として豊富に蓄積される．これらの情報を分析することにより，さまざまな支援を行える可能性がある．これらのデータの有効活用や，本システムを長期的に利用し定量的な有効性の検証を行うことを今後の課題とする．

謝辞

本研究は，文部科学省科学研究費助成金（若手研究（B ）No.24700912）からの補助金を受けた．

参考文献

[1] 内藤広志，斎藤隆，水谷泰治，「プログラミング演習の進捗モニタリングシステム」，FIT2008，pp.319-320

[2] 片桐由裕，立岩佑一郎，山本大介，高橋直久，「受講者の操作履歴の分析機能を用いたプログラミング指導者支援システム」，信学技報，ET2009-83，p181-186

[3] 井垣宏，斎藤俊，井上亮文，中村亮太，橋本真二，「プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案」，情報処理学会論文誌，Vol.54 No.1，pp.330-339

[4] 桑原恒夫，玉城幹介，山田光一，中村喜宏，満永豊，小西納子，天野和哉，「個人進度別教育支援システム（MESIA）における行き詰まり生徒の支援機能とその評価」，電子情報通信学会論文誌，D-Ｉ Vol.J83-D-Ｉ No.9，pp.1013-1024

[5] 水穂良平，土田康太，浜名隆広，佐藤国正，「PSF 手法に基づくプログラム作成過程における遷移の可視化」，信学技報，SS2005-54，pp.37-42

[6] 倉澤邦美，鈴木恵介，飯島正也，横山節雄，宮寺庸造，「プログラミング演習における一斉指導のための学習状況把握支援システムの開発」，信学技報，ET2004-104，pp.19-24

[7] 榊原康友，松澤芳昭，酒井三四郎，「コンパイルエラー修正時間に着目した学習分析指標の提案と内省学習効果分析への適用」，情報処理研究会報告，Vol.2013-CE-118 No.8，pp.1-8

[8] 小西達裕，鈴木浩之，伊藤幸宏，「プログラミング教育における教師支援のためのプログラミング評価機構」，電子情報通信学会論文誌，D-Ｉ Vol.J83-D-Ｉ No.6，pp.682-692

[9] 知見那彦，櫨山淳雄，宮寺庸造，「失敗知識を利用したプログラミング学習環境の構築」，電子情報通信学会論文誌，D-Ｉ Vol.J88-D-Ｉ No.1，pp.66-75

(1)該当フォルダの選択　(2)「追加」の選択　(3)「新しい項目」の選択　(4) テンプレートの選択

(5) 「c++ファイル」の選択　(6)入力欄の選択　(7) ファイル名の入力完了　(8) コードエディタの選択

(9) 「#」入力時　(10) 「i」入力時　(11) 「n」入力時　(12) 「c」入力時

図１　プログラミング過程蓄積モジュールが蓄積した画像群

（プロジェクトにソースコードファイルを新規作成しプログラムを記述する工程の一部）