99-01080

ソフトウェア構成管理ツールと連動したデバグ支援システム

松　下　　誠　　大阪大学大学院基礎工学研究科助手

1　まえがき

ソフトウェア保守では、既存のソフトウェアに対する多くの変更が発生する。しかし、ソフトウェアに変更を加える際に欠陥を作り込む確率が50%から80%の間であることが Hetzel の研究［8］で示されており、ソフトウエアに変更を加えた場合には変更した部分だけではなく変更していない部分に関しても動作確認が必要となる。また、近年のソフトウエア開発の大規模化、複雑化に伴い、ソフトウェア保守もより大規模かつ複雑なものとなっており、ソフトウェアの理解性および保守性を向上させることで保守活動を効率良く行うことが必要となる。

実際のソフトウェア保守では、ソフトウェアに変更を加えた後で、ソフトウェアが仕様通りの性能で動作することを確認するために回帰テスト［4、14］が活用されている。

また、実際のソフトウェア開発、保守において、ソフトウェアの理解性、保守性を向上させるために、作成したプロダクト（ソースプログラムや付随する文書）に対する様々な修正を正しく認識し、管理することを目的としたバージョン管理システム［1、13］が広く利用されている。

バージョン管理システムを用いて開発されたソフトウェアの保守段階では、仕様通りの性能で動作するバージョン（本稿では以降、基準バージョンと呼ぶ）が存在する。従来のソフトウェア保守では、機能追加、変更要求に伴い、この基準バージョンに対して様々な変更を加えることで、多くのバージョンが新しく作成される。ソフトウェアに対する変更は追加、変更した機能のテストが全て成功した段階で終了する。本稿では以降、変更終了時のバージョンを最新バージョンと呼ぶ。ソフトウェアの変更後には、最新バージョンの動作確認を回帰テストによって行う。回帰テストで欠陥が発見されると、欠陥の原因となるエラーの特定を行い、それを修正する。しかし、変更していない機能に欠陥が発見された場合、その原因となるエラーを特定することは容易ではない。

そこで、基準バージョンから最新バージョンの間に行われた変更内容を用いてデバッグを行う研究が行われている［11、15］。しかし、テストを行う前に、テスト入力の設定やテスト結果を比較するための正常な出力の設定といったテストツール［9］を利用するための環境設定を状況に応じて手作業で行う必要がある。また、テストを行う度に、ソフトウェアの基準バージョンに変更を適用してコンパイルを行う必要がある。さらには、テスト後に行うエラーの修正作業については不十分なため、実際のソフトウェア保守作業に既存の手法を適用することは難しい。

そこで本研究では、これらの問題点を解決するためのデバッグ手法 DMET の提案を行い、それに基づくデバッグ支援システム DSUS の構築を行う［10］。

今回提案するデバッグ手法 DMET は、ソフトウェア変更後の動作確認において変更していない機能に対して欠陥が見つかった場合のデバッグ作業を支援する。DMET ではバージョン管理システム、テストツールの利用を前提とし、対象とするソフトウェアを限定することで、テストツールの環境設定の自動化、および、デバッグ作業の支援を実現する。DMET はテスト手法、表示手法、反映手法の３つの手法から構成されている。１つ目のテスト手法では、最新バージョンから基準バージョンまでバージョンを遡りながら自動的にテストを行うことで、欠陥の原因となるエラーを含むバージョン間を特定する。２つ目の表示手法では、特定されたバージョン間の差分をソースプログラムの最新バージョン上で強調する。３つ目の反映手法では、デバッグ時に行った変更を欠陥のある古いバージョンに反映する。

今回構築したデバッグ支援システム DSUS は、簡易エディタとしての機能を備えており、ソフトウェア保守における一連のデバッグ作業を支援する。また、DSUS はシステム内部でバージョン管理を行うため、利用者はエラーの修正作業に専念できる。さらに、DSUS を用いた評価実験を行うことで、テスト作業において欠陥の原因特定に成功した場合に、従来よりデバッグ時間が短縮されることを確認した。また、実験を通じて一連のデバッグ作業を支援できることを確認した。

2　デバグの重要性

2.1　ソフトウェア保守

ソフトウェアの保守とは、本稼働中のソフトウェアの運用の継続を可能にするため、あるいはそれらを改善するための工程である［6］。ソフトウェアにかかる全費用のうち保守の占める割合は約80%にものぼり［3］、ソフトウェアに携わる人が費やす総時間の約65%が、保守やそれに関連する作業に費やされている［2］。このことから、ソフトウェアライフサイクルにおけるソフトウェア保守の重要性を認識できる。

従来のソフトウェア保守活動において、ソフトウェアに変更を加えるために行われる作業は基本的に次の３つである。

１．ソフトウェアおよびなされるべき変更の理解

２．システムの変更を実現するためのソフトウェアの変更

３．変更後のソフトウェアの動作確認

Fjeldstad と Hamlen によって行われた調査によると、ソフトウェアの変更においてソフトウェアを理解することが保守作業の中で最大の割合を占めることがわかる［7］。従って、ソフトウェア保守でソフトウェアおよび行われた修正を把握、理解すれば、発見された欠陥の原因を究明するために役立つ。

2.2　回帰テスト

実際のソフトウエア保守では、欠陥の修正や機能追加、変更によってソフトウェアに対して変更を加える。回帰テストはこのようなソフトウェアに変更がなされた後で動作確認のために適用される工程である。変更後には、変更を加えた機能の動作確認だけではなく、変更前の基準バージョンでは正しく働いていた機能の動作作確認も重要である。

2.3　バージョン管理

ソフトウェア開発過程において作成されるプロダクト（ソースコード、マニュアルや付随する文書等）の新規作成や、既に作成されたプロダクトの修正等、プロダクトに対する作業がどのように行われているかを識別し、組織化した上で管理するために、バージョン管理システムが広く用いられている。バージョン管理システムでは、プロダクトの修正作業を開発者間で調整することによって間違いのない修正を行うことを可能とする［1、5、13］。ここでいうバージョンとは、開発作業の進行と共に変化するプロダクトのある状態を指す。直観的には、ある変更による結果作成されたプロダクトそれぞれがバージョンとなる。バージョン管理されていないプロダクトは、ある特定のバージョンしか存在しないプロダクトと考えることができ、その変更は上書きによってのみ行われると考えることができる。

2.4　関連研究とその問題点

ソフトウェアに対して変更を行った場合、以前は正常に動作していた機能が動作しないことがある。これはソフトウェアに変更を加えることで、欠陥を作りこんだことが原因だと考えられるが、その原因となるエラーを特定し、エラーを修正することは容易ではない。

そこで、 Ness とNgo［11］、Zeller［15］によってソフトウェアの欠陥の原因を特定するための研究が行われている。

しかし、どちらの手法も基準バージョンのソースプログラムを基本として考えており、テストを行うためには毎回そのソースコードに修正を適用して、コンパイルを行わなければならない。また、テストを行う前に、ソフトウェアへの入力の設定、結果を比較するための正常な出力の設定といった、テストツールを利用するための設定を手作業で行う必要がある。さらには、欠陥の原因を特定するためのテスト作業にのみ重点が置かれており、デバッグ作業までは考慮されていないため、実際の保守作業に適用することは難しい。

3　デバッグ手法 DMET

本節では、２．４節で述べた既存の手法の持つ問題点を解決するためのデバッグ手法 DMET について述べる。DMET では変更していない機能に対する一連のデバッグ作業を支援するために、テスト作業に関して既存の手法における前提を改善し、修正作業に関して特定された差分を表示するための方法が考えられている。

DMET では対象とするソフトウェアをある程度限定することで、既存の手法では手作業で行っていたテストツールの環境設定を自動的に行う。また、バージョン管理システムを利用して実行可能プログラムもバージョン管理すること

により、テストの度に行っていたコンパイルを行わない。さらに、テスト作業から修正作業尾までのデバッグ作業を支援することで、実際のソフトウェア保守に適用することができる。

DMET はテスト手法、表示手法、反映手法の３つの柱から構成される（図１参照)。

テスト手法では、テストツールを利用して自動的にテストを行い、欠陥の原因を含むバージョン間を特定する。DMET では変更前の基準バージョンで正しく動作していた機能の動作確認に重点を置き、回帰テストを導入する。回帰テストにおいて欠陥が発見された場合には、欠陥の原因を特定するテスト（本稿では以降、局所化テストとする）によってその原因を含む実行可能プログラムのバージョン間を特定する。

表示手法では、テスト手法で特定された実行可能プログラムのバージョン間の差分をソースプログラムの最新バージョン上で表示する。DMET ではソースプログラムの最新バージョン上で必ずデバッグを行う。従って、特定されたバージョンから最新バージョンまでに行われた変更によって、差分に修正を加える必要がある。

反映手法では、テスト手法で特定されたバージョンまでバージョンを遡りながら、デバッグ時に最新バージョン上で行った変更をそれらに適用する。本稿では以降、デバッグ時に行った変更を適用する必要がある最新バージョンからテスト手法で特定されたバージョン間までのバージョンをまとめて反映必須バージョンと呼ぶことにする。反映必須バージョンに対してデバッグ時の変更を適用することで、それ以降のデバッグを効率良く続けることが可能となる。最新バージョンのソフトウェア V にテスト T を行うことで欠陥 O が発見されたとする。この場合、テスト手法によって欠陥の原因となるエラーを含むバージョン間を特定し、ソースプログラムの最新バージョン上で表示して実際にデバッグを行う。ところがデバッグ後のソフトウェアの動作確認において、同一のテスト T で新たな欠陥 O’が発見されたとすると、もう一度テスト手法を利用する必要がある。反映必須バージョンにデバッグ時に行った変更を適用していなければ、欠陥 O を含んでいたバージョンは１度目と同じ出力をするため、新たに発見された欠陥に対する原因の特定を効率良く行うことが困難となる。従って、反映必須バージョンに最新バージョン上での変更内容を適用することが重要となる。反映は、まずソースコードに対して行い、その情報を利用して実行ファイルにも反映を行う。

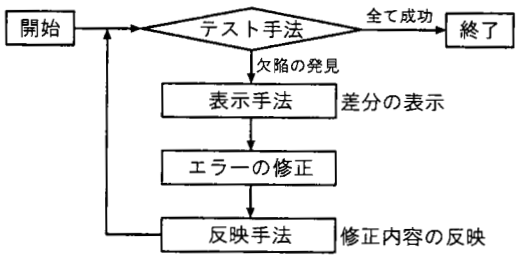

図1：デバック手法の流れ

４　デバッグ支援システムＤＳＵＳ

本節では，３節で述べたデバッグ手法ＤＭＥＴに基づくデバッグ支援システム（ＤＳＵＳ）について説明する．

4.1　システムの機能

デバッグ支援システムＤＳＵＳ（ＤｅｂｕｇｇｉｎｇＳＵｐｐｏｒｔＳｙｓｔｅｍ）は，バージョン管理システムから得られるバージョン情報をもとにＤＭＥＴに基づいてデバッグを効率良く行うことができるシステムである．ＤＳＵＳの特徴としては次に示す８つの点が挙げられる。

- ●ソフトウェアを開発するための言語に依存しない。
- ●バージョン管理システムでのバージョンの登録，取り出しをシステム内部で自動的に行うため，システムの利用者は変更作業，デバッグ作業に専念できる．バージョンに関する情報は準備した機能によって参照することができる。
- ●エディタ機能を実装しており、ソフトウェアに対する変更、テストおよびデバッグのソフトウエア保守における一連の作業の流れを支援できる。
- ●ソフトウェアに欠陥が発見された場合、テスト手法に基づいて欠陥の原因を含むバージョン間を特定できる。
- ●入力ファイル、ソフトウェアの基準バージョンを指定することで、テストツールを利用してソフトウェアの正常出力を自動的に構成する。テストではこれらの情報をもとにテストツールの環境設定を自動的に行う。
- ●テスト手法によって原因を含むバージョン間が特定された場合、表示手法に基づいてソースプログラムの最新バージョン上でそのバージョン間の差分を強調表示することができる。差分の表示方法は、差分の中で削除したコードについてはコードを参照する機能を設け、挿入したコードに対しては残っているコードを強調表示する。図２では、

エディタ上で表示されているソースプログラム中で、挿入したコード部分の背景が薄赤色で表示されている。また、ソースプログラムの各行の横についている薄青色のボタンをクリックすることで、削除したコードの内容を参照できる。参照する際に表示されるウィンドウの画面写真を図３に示す。

●デバッグ時に実際に変更を行った場合、反映手法に基づいて反映必須バージョンにデバッグ時の変更を反映することでその後のテスト、デバッグを効率良く行うことができる。

●ソフトウェアの変更作業を行うモードとデバッグ作業を行うモードの２つがあり、回帰テストの結果によってシステム内部で自動的にモードが切り替わる。システムの起動当初は変更作業モードである。

GUI 部においては、システム内部で行っている処理の中で利用者に提供可能な情報を表示するためのメッセージウインドウを実装している。（図４参照）

4.2 システム構成の概要

DSUS はシステムの中核をなす DSUS_{main} とバージョン管理ツール RCS、テストツール DejaGnu、利用者のインタフェースである GUI によって構成される。図５はシステムの各構成要素のつながりを表したものである。

各構成要素は以下のような動作を行う。

●DSUS_{main}

システムの中核として GUI からの操作の一括管理，RCS によるバージョン管理、DejaGnu の環境設定および DejaGnu によるテスト実行の管理を行う。

●GUI

DSUS の利用者が実際に作業を行うエディタ部分を指し、利用者はこのエディタを通じて、ソフトウェアの変更およびテスト、デバッグを行う。利用者からはこのエディタしか見えず、バージョン管理、テスト実行はシステム内部でのみ行われる。

●RCS

実際にバージョンの記録や呼出しを行う。

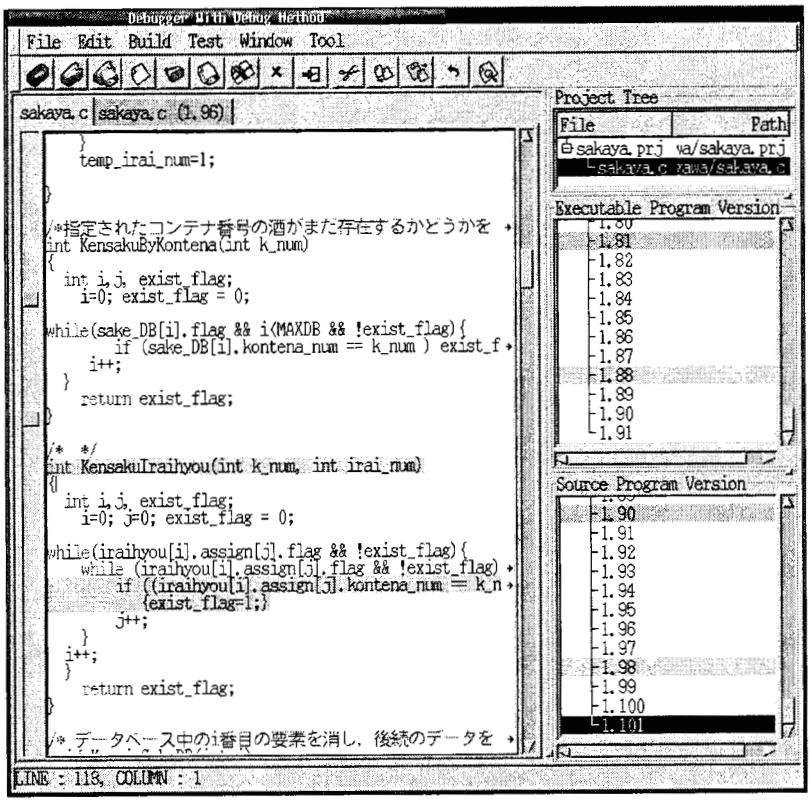

図2：デバック時のメインウィンドウ

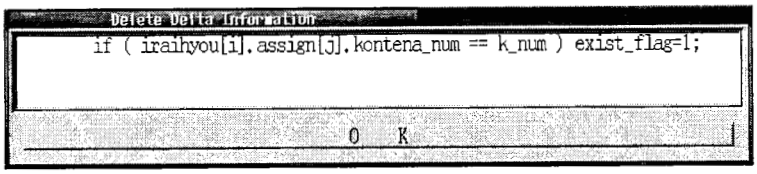

図3：削除コード参照ウィンドウ

●DejaGnu DSUS$_{main}$ によって設定された環境に基づいて実際にテストを実行し、その結果を DSUS$_{main}$ に渡す。

5 DMET の評価実験

本節では、DSUS を用いて行った DMET の評価実験［12］について述べる。本実験の目的は DMET を用いることで変更していない機能に対するデバッグ作業を効率良く行えるかを評価することである。

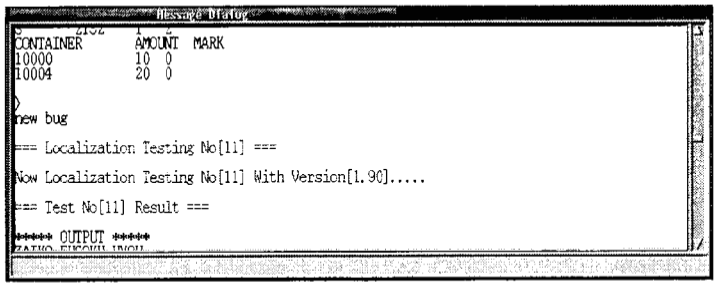

図4：メッセージ表示ウィンドウ

表1：実験で用いたソフトウェアに関するデータ

	バージョン数	テスト回数	含原因
ソフトウェア A	28	25	○
ソフトウェア B	91	10	○
ソフトウェア C	72	70	×

5.1 実験の概要

実験の被験者10人を２つのグループ G_1、G_2 にわける。一方のグループ G_1 の被験者は DMET に基づく DSUS を利用する。もう一方のグループ G_2 の被験者は回帰テスト、バージョン情報の参照が行えるシステムを利用する。実験においては、それぞれのグループの被験者が欠陥の作り込まれた酒屋問題を取り扱う３つのソフトウェアを用いてデバッグ作業を行う。本実験で利用したソフトウェアに関するデータを表１に示す。

5.2 結果と考察

表２に DMET を利用したグループ G_1 の実験結果を示し、表３に DMET を利用しないグループ G_2 の実験結果を示す。

表中の A、B、C はそれぞれ実験で利用したソフトウェアを表している。A、B のソフトウェアは DMET のテスト手法で特定されたバージョン間の差分に欠陥の原因が含まれるが、C のソフトウェアは DMET のテスト手法で特定されたバージョン間の差分に欠陥の原因が含まれない。

欠陥の原因特定に成功したソフトウェア A、B に対するデバッグ時間の合計において、５%有位水準によるウェルチの検定によって有位差が見られた。しかしながら、欠陥の原因特定に失敗したソフトウェア C に対するデバッグ時間には有位差が見られるほどの差はなかった。

従って、この結果から DMET を利用することで欠陥の原因特定に成功した場合に従来よりも効率良くデバッグ作業を行えることがわかった。また、実験を通じて一連のデバッグ作業を支援できることを確認した。

表２：DMET を利用したグループ G_1のデバッグ時間（単位：分）

	A&B	C
被験者 1	75	25
被験者 2	62	60
被験者 3	65	57
被験者 4	79	34
被験者 5	54	21
平均	67.0	39.4

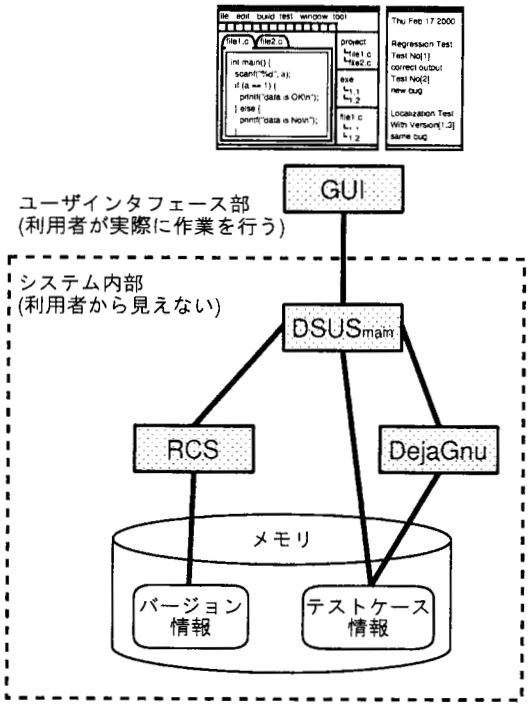

図5：DSUSのシステム構成

表３：DMET を利用しないグループ G_2 のデバッグ時間（単位：分）

	A&B	C
被験者 6	237	20
被験者 7	107	5
被験者 8	237	15
被験者 9	69	5
被験者 10	165	8
平均	163.0	10.6

6　まとめ

本研究では、ソフトウェア保守においてソフトウェアに変更を加えることで変更していない機能に欠陥が発見された場合の差分を用いたデバッグについて、その基本的概念と既存の研究についてその前提条件、テスト作業および修正作業の方法、問題点を述べた。そして、これらの問題点を解決するためのデバッグ手法 DMET の提案を行った。DMET は欠陥の原因を含むバージョン間の差分を特定するテスト手法、特定された差分を最新バージョン上で表示する表示手法、修正作業の内容を反映必須バージョンに反映する反映手法の３つから構成されており，デバッグ作業におけるテスト作業から修正作業を支援する。また、デバッグ手法 DMET に基づくデバッグ支援システム DSUS の構築について述

べた。DSUS は対象とするソフトウェアの開発言語を限定しないため、幅広く活用できる。また DSUS ではシステム内部でバージョン管理、テスト実行を行うため、利用者がエラーの修正作業に専念できる環境を提供する。最後に、DSUS を用いて DMET の評価実験を行った。実験により、DMET を利用することで従来よりもデバッグ作業を効率良く行えること、一連のデバッグ作業を支援できることがわかった。従って、実際のソフトウェア保守におけるデバッグに適用できると考える。今後の課題をとしては次に示す３つが挙げられる。

- ●今回得られた実験データをもとに欠陥の原因を特定するテストアルゴリズムの改善し、欠陥の特定にかかる時間およびテスト回数の削減を図る。
- ●テストツールによる実験結果の比較を完全一致で行っており、現在の日時が出力されるような場合や簡単な出力文を加えた場合に欠陥出力と認識されるため、比較出力にマスクをかけられるようにすることでより柔軟な比較が行えるようにする。
- ●現状の DMET は変更していない機能に対するデバッグだけを対象としているため、変更した機能に対するデバッグや新規開発におけるデバッグにも役立つように拡張する。

参考文献

［1］Babich, W. A.,“Software Configuration Management,”Addison-Wesley, Reading, Massachusetts, 1986.

［2］CASE 1988-89, Sentry Market Research, Westborough, MA, pp.13-14, 1989.

［3］Cashman, P. M. and Holt, A. W.,“A Communication-Oriented Approach to Structuring the Software Maintenance Environment,”Software Enginnering Notes, 5, No.1, pp.4-17, January 1980.

［4］Dogsa, T. and Rozman, I.,“CAMOTE-Computer Aided Module Testing and Design Environment,”Proceedings of the Conference on Software Maintenance-88, Phoenix. Ariz., pp.404-408, 1988.

［5］Estublier, J. and Casallas, R.,“The Adele Configuration Manager,”In Walter Tichy, editor, Configuration Management, pp.99-133. John Wiley and Sons, Ltd., Baffins Lane, Chichester, West Sussex PO191UD, England, 1994.

［6］FIPS Publication 106,“Guideline on Sftware Maintenance,”Federal Information Processing Standards, U. S. Department of Commerce/National Bureau of Standards, June 1984.

［7］Fjeldstad R. K.,“Application Program Maintenance Study-Report to Our Respondents,”Proceedings of GUIDE 48, The GUIDE Corporation, Philadelphia, 1979.

［8］Hetzel, W.,“The Complete Guide to Software Testing,”QED Informaion Sciences, Wellsley, Mass., 1984.

［9］IEEE.,“Test Methods for Mesureing Conformance to POSIX,”ANSI/IEEE Standard 1003.3-1991, ISO/IEC Standard 13210-1994.

［10］Matsushita, M., Teraguchi, M., and Inoue, K.,“Effective Testing and Debugging Methods and Its Supporting System with Program Deltas”, In Proceedings of International Symposium on Principles of Software Evolution, pp.281-288, 2000.

［11］Ness, B., and Ngo, V.,“Regression containment through source code isolation.”, In Proceedings of the 21st Annual Internatial Computer & Applications Conference (COMPSAC ‘97), IEEE Computer Society Press, pp.616-621, 1997.

［12］田原、寺口、松下、井上、“プログラムの差分情報を用いたデバグ手法の評価実験”、情報処理学会第61回全国大会 5W-3, pp.1-327-1-328, 2000.

［13］Tichy, W. F., RCS-A System for Version Control, Software-Practice and Experience, Vol.15, No.7, pp.637-654, 1985.

［14］Yau, S. and Kishimoto, Z.,“A Method for Revalidating Modified Programs in the Maintenance Phase,”In Proceedings of the 11th Annual Internatial Computer & Applications Conference (COMPSAC‘87), pp.272-277, Tokyo, Japan, 1987.

［15］Zeller, A.,“Yesterday, my program worked. Today, it does not. Why?”, Proceedings of the 7th European Software Engineering Conference and 7th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (ESEC/FSE ‘99), Toulouse, France, September 1999.

＜ 発　表　資　料 ＞

題　　名	掲載誌・学会名等	発表年月
プログラムの差分情報を用いたデバグ手法の評価実験	情報処理学会弟61回全国大会講演論文集 5W-3,pp. 1-327-1-328	2000年10月
Effective Testing and Debugging Methods and Its Supporting System with Program Deltas	Proceedings of International Symposium on Principles of Software Evolution, pp. 281-288(November 2000)	2000年11月