柔らかい構造で変化を受容するソフトウェア

(研究課題名：オブジェクトとメディアによるソフトウェア構造化)

「機能と構成」領域　　神谷　年洋

要　旨

ソフトウェアは、ワードプロセッサや計算票といった単体のアプリケーションとして用いられる以外にも、電気製品や車など、日常的に利用される多くのモノに組み込まれるようになってきています。また、金融や物流はもとより、インターネットの情報発信、検索、通信販売サイトなど、日常的に利用されるサービスの実現に欠かせない技術になっています。アプリケーションやモノ、サービスの高機能化・高付加価値化にともなってソフトウェアは必然的に大規模化し、また、ソフトウェア同士が連携して動作したり、利用者の都合に応じたカスタマイズが必要とされる場面も増えました。同時に、ソフトウェア開発の地理的・時間的な分散化（典型的には「伽藍からバザールへと」）が進んできています。来るべきユビキタス社会において、このようなソフトウェアの連携、カスタマイズ、開発の分散・分業といった傾向はますます強くなっていくと考えられます。

本研究テーマでは、このような時代にふさわしいソフトウェアの開発・運用のための枠組みを提供するため、ソフトウェア開発・運用技術のプロトタイプであるSOMA（Solid-Object and Medium Artifact）を開発しました。この技術により、ソフトウェアを、現在のソフトウェアが持つ堅牢さを損なうことなく、開発者ばかりではなく利用者も自由・安全（間違った編集操作をしても元に戻せるという意味において）編集することを可能にします。これにより、ソフトウェアのよりよい開発、運用が可能になることを目指しています。

1．研究のねらい

従来のオブジェクト指向ソフトウェア開発技術では、ソフトウェアを、オブジェクトと呼ばれる堅牢なブラックボックスの集まりとして記述します。オブジェクトは、カプセル化によって保護される実体です。すなわち、あるオブジェクトの内部状態は、そのオブジェクトが公開しているメソッドを通じてのみ変更されます。外部から内部状態に直接アクセスすることは禁止され、オブジェクトは、自分自身の内部状態の健全性（データの整合性）を保証

することができます。カプセル化は、堅牢なソフトウェアを記述するための重要な技術のひとつとなっています。しかし、その一方で、オブジェクト間の関係を記述する方法はあまり発達しておらず、したがって本質的に変化が必要とされる構造を記述することはそれほど得意ではありません。

この弱点を補うため、デザインパターンのような設計指針、EJBなどのコンポーネント開発技術、インストーラーやプラグインといった運用時の構成管理（reconfiguration）ツール・ライブラリが開発されてきましたが、これらはいずれもすでにオブジェクト指向が導入されている部分に「あとづけ」で構造を記述する手法であるため、適用に制限があったり、開発者に負担をかけるものであったり、利用者がソフトウェアの運用時に利用することを想定していなかったりと、種々の問題があり、現在までオブジェクト指向開発技術を全面的に置き換える技術とはなっていません。

本研究テーマでは、オブジェクトの関係を記述するために、「メディア」と名付けたもうひとつの実体を導入し、オブジェクトとメディアによってソフトウェアを記述する方法を提案しました。メディアによって、ソフトウェアに含まれる（変更されることを前提とした）構造を素直に書き下すことが可能になり、また、実行環境がそのような構造の編集をサポートするための標準的な仕掛けを提供することが可能になります。

2．研究方法と成果

2．1　SOMAの提案

現在、ソフトウェア開発で盛んに用いられているオブジェクト指向開発技術は、ソフトウェアの進化を困難にする技術的な問題を抱えていました。本研究では、動的で安全（間違った編集操作をしても元に戻せるという意味において）なソフトウェアの修正を可能にするための、SOMA(Solid-Object and Medium Artifact) と名付けた新しい技術を提案しました。SOMAはソフトウェア開発技術、実行環境、運用技術をふくみます。SOMAの運用技術は、ソフトウェアの利用者が、ソフトウェアの修正を簡単、安全に行うことを可能にする操作を提供します。

SOMAでは、ソフトウェアをソリッド・オブジェクト（以降、オブジェクト指向の「オブジェクト」と区別する必要がないかぎり、単に「オブジェクト」と表記）とメディアと呼ぶ2種類の実体によって記述します。大きくは、オブジェクトがアプリケーションの機能を、メディアがアプリケーションの境界や構造を記述するのに用いられます。表1に、オブ

ジェクトとメディアそれぞれの特徴を示します。これらに加えて補助的に、オブジェクトが欠落していることを示す「プロキシ」と呼ばれる実体も用いられます。

表1　ソリッド・オブジェクトとメディアの特徴

	ソリッド・オブジェクト	メディア
役　　割	アプリケーションの機能、すなわち、アルゴリズムやデータ	アプリケーションの境界・構造
内　　容	インスタンス変数、メソッド、依存する他のオブジェクトの型	内包するオブジェクト、オブジェクト間の結合
情報隠蔽・開示	ブラックボックスであり、カプセル化によって保護される	ホワイトボックスであり、メディアの操作によって記述を修正することができる

単体のアプリケーションは、オブジェクトとメディアにより、メディアによって示される境界の中に、機能を持ったオブジェクトが含まれ、オブジェクトは自身が依存するオブジェクトへ結合している、というモデルで記述されます。

図1に、モデル・ビュー・コントローラで構成された電卓アプリケーションをSOMAでモデル化した図を示します。角が丸い四角はメディア、丸はオブジェクト、斜めの線が入った丸はプロキシ、オブジェクトやプロキシに添えられた文字はそれらの名前、矢印はオブジェクトの結合（向きは依存しているオブジェクトから依存されているオブジェクトへ）を示します。

この図で、例えば、ビューオブジェクト（calculatorView）モデルオブジェクト（calculatorModel）や入出力（stdin, stdout）に依存しているため、これらへの結合を持ちます。このうち、

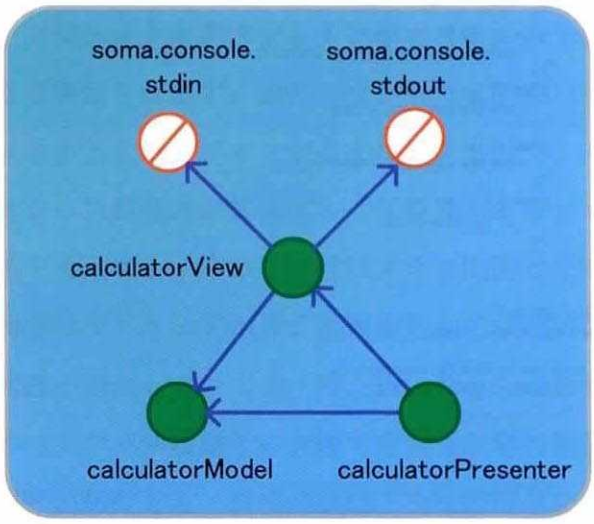

図1　SOMAモデルで表現したMVC電卓

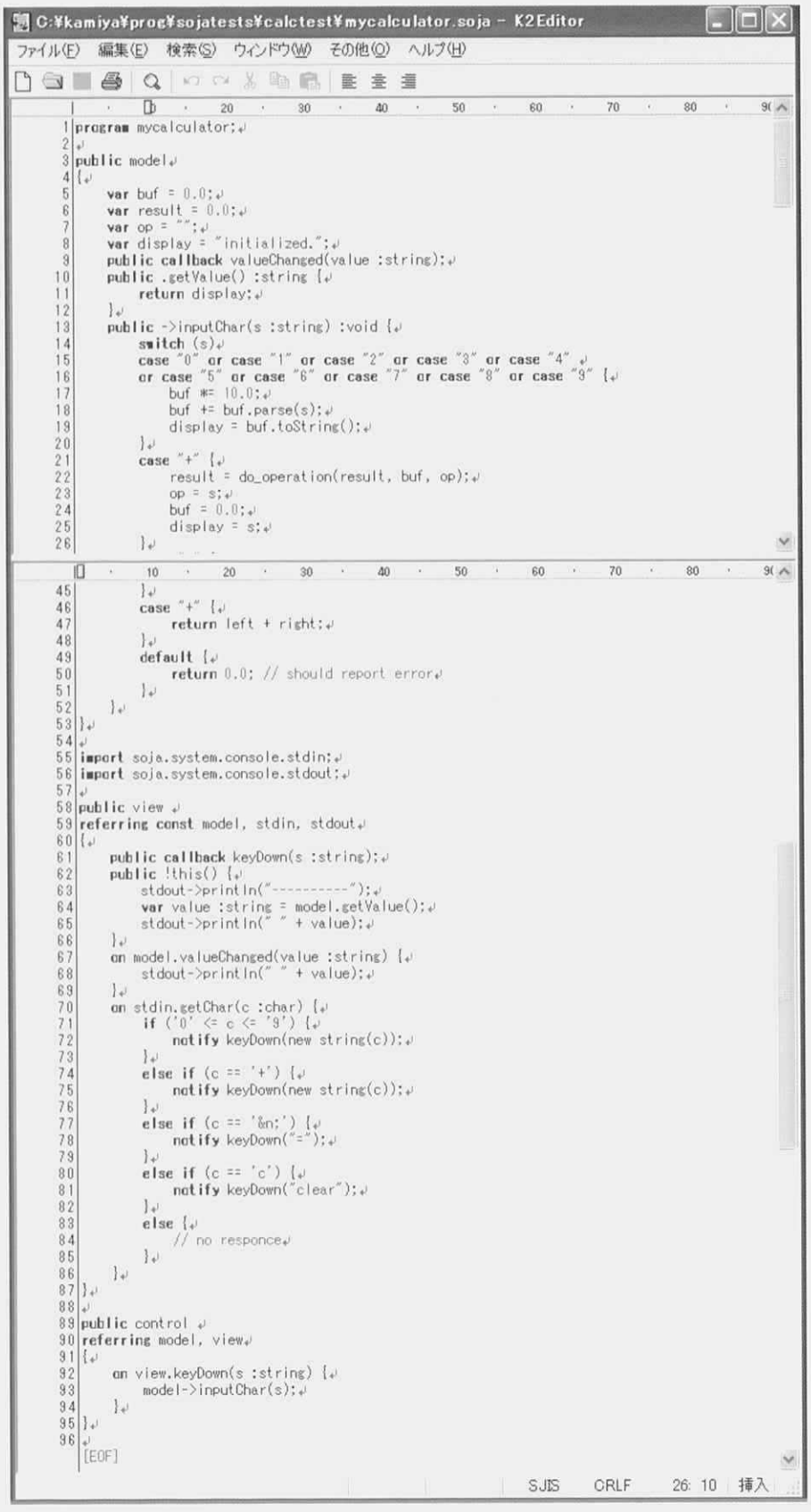

図2　MVC電卓のソースコード（一部）

stdin, stdoutがプロキシとなっていますが、これは、このメディアはこれらのオブジェクトを含んでおらず、実行時に別途与える必要があるということを意味しています。

利用者がアプリケーション（複数）やOSといったソフトウェアを運用するときに、これらに連携を行わせたり、カスタマイズすることをサポートするため、SOMAではメディアを対象とした操作を提供しています。操作は3種類あり、それぞれ切断（cut)、接合（join)、境界算出（boundary computation）と名付けられています。

試験的に、電卓プログラムを、SOMAを既存のプログラミング言語C++を用いて、専用のライブラリを提供する方法で実装してみたところ、ソースコードのサイズが3千行超になってしまい、電卓の機能を実現する部分と、SOMAのための部分が混ざってしまい、可読性が損なわれることが判明しました。そのため、SOMAを自然に記述することを目的として、プログラミング言語sojaを策定し、言語処理系およびランタイムライブラリを開発しました。Sojaを用いて記述した電卓プログラムのソースコードを図2に示します。ソースコードの記述内容は、図1のモデルを骨組みとして、オブジェクトの内部に関する記述を付け加えたものとなっています。

2.2　インターフェイスの解析による自動的な構成管理手法

上述のSOMAは、利用者による運用時のソフトウェアの編集、例えば、2つのアプリケーションを組み合わせたり、一部の部品を入れ替えたりといった作業を実現する技術です。この技術には、利用者が誤った操作をしたときに、それを事前に中断させたり、あるいは回復する手段も含まれています。このような誤りの検出は、インターフェイスの互換性に基づいて行われます。すなわち、あるオブジェクト間の結合において、依存される側のオブジェクトが、依存する側のオブジェクトが要求するインターフェイスを備えていない場合、そのような結合はエラーでると判定されます。通常、ソフトウェアはバージョンアップにより進化するため、同じオブジェクトもバージョンが違えばインターフェイスが異なっていることが普通です。さらに、異なった開発者が提供する「互換性がある」オブジェクトを組み合わせる状況も想定され、インターフェイスの互換性は大きな問題となります。現在の構成管理技術は、このような動的なソフトウェアの修正を前提としていないため、ソフトウェアの運用においていくつかの問題が発生しています。例えば、ひとつの開発組織から提供されている単一のソフトウェアのパッケージは簡単に利用できても、他の開発組織が提供するアプリケーションを利用するのは困難である、特定のソフトウェア部品のバージョンが異なるためにアプリケーションが動作しない、といった問題が起きています。

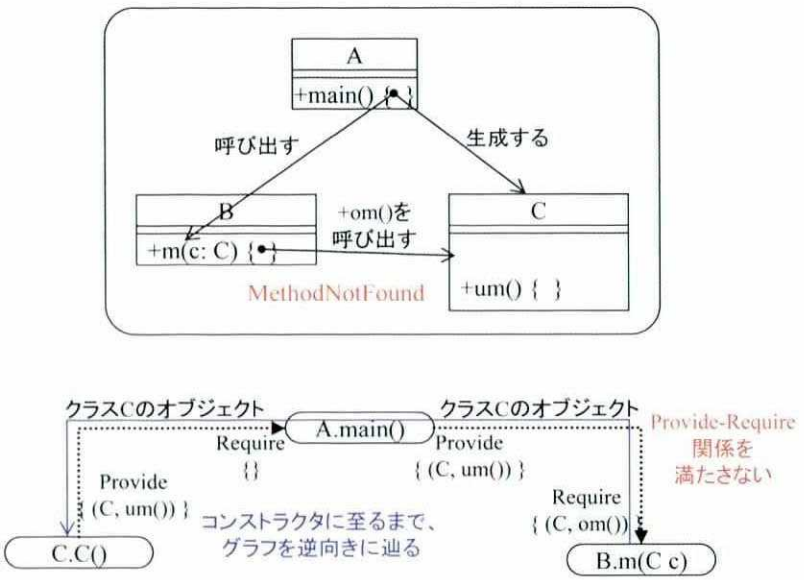

図３　提案手法によるインターフェイス互換性問題の検出と原因箇所の特定

この研究では、インターフェイスのバージョン番号や宣言（「このインターフェイスはこういう機能を提供する」という開発者による記述）ではなく、インターフェイスの利用（あるインターフェイスを備えるオブジェクトがこのように利用されているという状況）から互換性を検査する手法を提案しました。これにより、従来の互換性検査（インターフェイスの名前やバージョン番号、宣言に基づいたもの）より正確な検査が行えるようになります。アプローチとしては、あるインターフェイスを持つオブジェクトが利用されている場所で要求されているメソッドを、そこに到達するオブジェクトを生成する場所で用いられているクラス定義が提供するかを調べ、不足があればエラーを報告します。

この手法により、ソフトウェア編集作業を行う前に、インターフェイスの互換性を検査できるだけではなく、互換性の問題を起こす部分の特定が可能になります。これにより、ソフトウェアの実行環境は、互換性問題に対するより良いサポートを利用者に提供することが可能になります。

この研究では、プログラミング言語Javaで記述されたプログラムを想定していますが、提案された互換性検査の手法は、SOMAに対しても同様に適用可能なものとなっています。

２．３　アスペクトによるアサーション

ソフトウェアの再利用は、大規模なソフトウェアの開発を可能にし、ソフトウェアの開発期間を短縮し、同時に（エラーの少なさの観点から）品質を向上させる重要な技術です。ソフトウェアの再利用をサポートするために、さまざまな部品や部品化の技術が提案されて

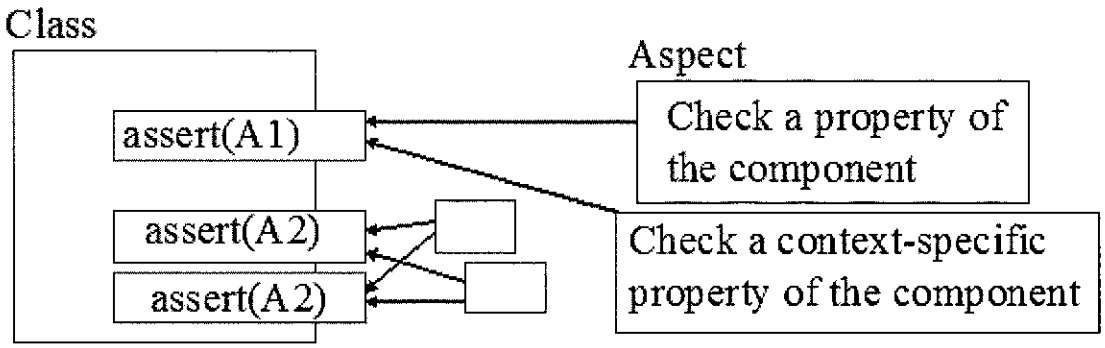

図4　アスペクトによるアサーション

います（SOMAもソフトウェア部品化の技術を含んでいます）。ソフトウェア部品の汎用性と、特定用途への対応は、どちらもソフトウェア部品の再利用しやすさの観点からは重要ですが、従来、この２つはトレードオフの関係にありました。すなわち、汎用性を高めようとすると、特定用途に対応した機能（以降「ドメイン知識」）を部品から追い出すことになり、逆に、特定用途に対応させようとすると、部品にドメイン知識を含めることになります。

この研究では、このトレードオフを解消するため、ドメイン知識を、それぞれの部品ではなく部品の間を繋ぐ技術に持たせるというアプローチを取ることを提案しました。具体的には、特定用途に応じたチェックルーチンを、オブジェクトではなく、アスペクトに持たせることで、オブジェクト自体の汎用性を損なうことなく、チェックルーチンを変更することが可能になりました。

この研究は、SOMAの枠組みにおいては、オブジェクト間の通信を実現する部分に応用されています。オブジェクトは、メソッド、コールバック、通知メッセージの３種類の呼び出しを備え、また、それぞれについて例外の送出と捕獲を実現する必要がありました。この研究の成果により、プログラミング言語の記述として、これらに対して見通しよく統一的な記法を与えることができました。

２．４　そのほか

SOMAは概念の提案だけではなく、実際に動作するツールの提供を目指して、言語処理系や実行環境（Soja）の開発も行いました。SOMAのモデルは従来のオブジェクト指向のモデルとは異なるため、既存のライブラリを直接用いるのは困難が多いことも判明し、従来のソフトウェアをSOMAによって書き直すための調査も行いました。これらの過程で、漸進的パーザー、ソースコード縮退によるソースコード理解、コードクローンを用いたソースコード調査などの研究を行いました。

3．今後の展望

現在、来るべきユビキタス社会の到来に向けて、dynamic reconfigurationやend-user developmentといった、利用者による動的なソフトウェアの編集（修正から開発まで）を可能にする研究分野が興りつつあります。これに応えるように、ソフトウェア開発技術からも、従来のコンポーネント技術やエージェント技術、Intentional Programmingといったものにくわえて、アスペクト指向、EJB3.0、Code Insertionといった新しい技術が登場してきています（SOMAはこれらの技術のひとつであり、開発者と利用者に共通の可視化を特徴としています)。これらの技術は、コンピュータ科学の初期に提案された「ソフトウェアが実行時に自分自身を修正する」あるいは「データとして書いたものがコードとして実行できる」といったアイデアを、コンピュータが社会の基盤として組み込まれている現在の社会が必要としている、安全性や進化、理解容易性、保守性、スケーラビリティといったさまざまな要求に応えられるよう発展させたものです。

ソフトウェアの現在のもうひとつの大きな流れとして、ソフトウェアの機能の中で、コンテンツが果たす役割が重要になってきた（あるいは、コンテンツの役割が重要であることが認識されてきた）というものがあります。ソフトウェアの高機能化に伴いビジネスルールが取り込まれたり、Googleに代表されるような大規模なデータベースを用いた分析が必要とされたり、ゲームのように高度なグラフィクスや音楽が主要な機能であるソフトウェアが登場してきています。SOMAは、開発者と利用者が共通に持つソフトウェアを操作するためのビューを提供することで、プログラム開発者とコンテンツ開発者の分業・分散がより進んだときに、両者の間でソフトウェアを交換し合う、編集しあうといった作業に役立つことでしょう。

今後、研究期間に開発した言語処理系および実行環境であるSojaを発展させるためには、どのようにライブラリを設計すべきかといった、より広い見地からの議論が必要であると考えています。また、この数年に登場した、新しい開発技術であるEJB3.0やJava Genericsを取り入れることも考えています。

4．成果リスト

記　事

1．神谷 年洋，“コードクローンとは，コードクローンが引き起こす問題，その対策の現状”，電子情

報通信学会誌 Vol. 87, No. 9, pp. 791-797 (2004-9).

受　賞

1．「コードクローン検出システム」, 第35回市村学術賞貢献賞 (2003/04/25). 大阪大学大学院基礎工学研究科 井上克郎 教授, 同 楠本真二 助教授とともに.

論 文 誌

1．佐々木亨, 肥後芳樹, 神谷年洋, 楠本真二, 井上克郎, "プログラム変更支援を目的としたコードクローン情報付加ツールの実装と評価", 電子情報通信学会論文誌, Vol. J87-D-I, No. 9, pp. 868-870 (2004-9).
2．肥後芳樹, 植田泰士, 神谷年洋, 楠本真二, 井上克郎, "コードクローン解析に基づくリファクタリングの試み", 情報処理学会論文誌, no. 45, vol. 5, pp. 1357-1366 (2004-5).
3．泉田聡介, 植田泰士, 神谷年洋, 楠本真二, 井上克郎, "ソフトウェア保守のための類似コード片検索ツール", 電子情報通信学会論文誌 D-I, Vol. J86-D-I, No. 12, pp. 906-908 (2003-12).
4．植田泰士, 神谷年洋, 楠本真二, 井上克郎, "開発保守支援を目指したコードクローン分析環境", 電子情報通信学会論文誌 D-I, Vol. J86-D-I, No. 12, pp. 863-871 (2003-12).
5．門田暁人, 佐藤慎一, 神谷年洋, 松本健一, "コードクローンに基づくレガシーソフトウェアの品質の分析", 情報処理学会論文誌, vol. 44, No. 8, pp. 2178-2188 (2003-8).
6．Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue, "CCFinder: A Multi-Linguistic Token-based Code Clone Detection System for Large Scale Source Code", IEEE Trans. Software Engineering, vol. 28, no. 7, pp. 654-670, (2002-7).
7．[論文誌] 山本哲男, 松下　誠, 神谷年洋, 井上克郎, "ソフトウェアシステムの類似度とその計測ツール SMMT", 電子情報通信学会論文誌, vol. J85-D-I, no. 6, pp. 503-511. (2002-6).

会議 (査読つき)

1．Takashi Ishio, Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue, "Assertion with Aspect", SPLAT'04 (Software-Engineering Properties of Languages for Aspect Technologies), Lancaster, UK, (Mar. 22, 2004).
2．Yoshiki Higo, Yasushi Ueda, Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue, "On Software Maintenance Process Improvement Based on Code Clone Analysis", LNCS 2559 Product Focused Software Process Improvement, pp. 185-197, Springer, The 4th International Conference on Product Focused Software Process Improvement (Profes 2002), Rovaniemi, Finland, (December 9-11, 2002).
3．Yasushi Ueda, Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue, "On Detection of Gapped Code Clones using Gap Locations", Proc. of the IEEE 9th Asia-Pasific Software Engineering Conference (APSEC 2002), pp. 327- 336, Queensland, Australia, (December 4-6, 2002).
4．Toshihiro Kamiya, "SOMA: A Paradigm to Evolve Software Based on Separation of Concerns", Proc. of the IPSJ SIGSE/ACM SIGSOFT 5th International Workshop on Principles of Software Evolution (IWPSE 2002), pp. 124-128, Orland, Florida, (May 19-22, 2002).

そのほか

1．早瀬康裕, 神谷年洋, 松下　誠, 井上克郎, "インタフェースのprovide-require 関係の解析に基づいた自動的な構成管理手法の提案", 電子情報通信学会技術研究報告, SS2004-7, Vol.104, No.242, pp.7-12 (2004-8).

2．肥後芳樹，神谷年洋，楠本真二，井上克郎，“コードクローン情報を用いたリファクタリング支援ツール”，電子情報通信学会技術研究報告，SS2004-1, Vol.104, No.7, pp.1-6（2004-5）.
3．石尾　隆，神谷年洋，楠本真二，井上克郎，“アスペクトを用いた表明の記述”，情報処理学会研究報告，2004-SE-144, pp.75-82,（2004-3-18）
4．神谷年洋，石尾　隆，井上克郎，“プログラミング言語のスコープ階層を反映した構造化解析器”，日本ソフトウェア科学会研究会資料シリーズNo.28 第1回ディペンダブルソフトウェアワークショップ（DSW2004）論文集，pp. 159-168（2004-2-24）.
5．神谷年洋，石尾　隆，植田泰士，楠本真二，井上克郎，“ソースコード縮退によるソースコード理解”，オブジェクト指向最前線2003 情報処理学会OO2003シンポジウム予稿集，pp. 65-68,（2003-8-21）.
6．肥後芳樹，神谷年洋，楠本真二，井上克郎，“類似コード片を用いたリファクタリングの試み”，情報処理学会研究報告 2003-SE-143, pp. 29-36,（2003-7-17）.
7．Toshihiro Kamiya, “On an Object-and-Connection Modeling as a Separation of Concerns Based on Knowledge and Task Abstraction Levels”, The 1st International Workshop on Designing Human-Software Interaction（DHSI2003）, Boulder, Colorado,（May 26-29, 2003）.
8．内田眞司，門田暁人，神谷年洋，松本健一，工藤英男，“コードクローンによるソフトウェア解析”，平成14年電気関係学会関西支部連合大会シンポジウム講演，Nov. 9, 2002（採録決定）.
9．Monden, D. Nakae, T. Kamiya, S. Sato, and K. Matsumoto, “Software quality analysis by code clones in industrial legacy software”, Proc. of the 8th IEEE Symposium on Software Metrics（METRICS2002）, pp. 87-94, Ottawa, Canada,（June 4-7, 2002）.
10. Yasushi Ueda, Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue, “Gemini: Maitenance Support Environment Based on Code Clone Analysis”, Proc. of the 8th IEEE Symposium on Software Metrics（METRICS2002）, pp. 67-76, Ottawa, Canada,（June 4-7, 2002）.
11. 吉田正和，藏川　圭，中小路久美代，神谷年洋，“リファクタリング指標の構築に向けたオブジェクト指向システムの前進的開発における進化メトリクスのケーススタディ”，情報処理学会研究報告 Vol.2002, No.23, pp.95-102（2002-3）.
12. 神谷年洋，“静的および動的構造化としてのオブジェクト・メディアモデル”，電子情報通信学会技術研究報告 SS2001-52, Vol. 101, No. 674, pp. 17-23, 電子情報通信学会 ソフトウェアサイエンス研究会，福山大学，（2002-3-5）.