「機能と構成」研究領域　領域活動・評価報告書

― 平成 16 年度終了研究課題 ―

研究総括　片山 卓也

1. 研究領域の概要

本領域は、"先進情報システムとその構成に向けて"、独創性に富んだ提案を積極的にとりあげようとするものです。具体的には、これからの社会を支える高度な機能をもった情報システムの構築を目指し、そのための構成や構築方法に関して、基本技術や先進応用事例および基礎となる理論の研究を行います。例えば、ソフトウェア、ネットワーク、プロセッサ、分散・実時間・埋め込みシステム、セキュリティ、設計・実装・進化方法論と環境、テスト・検証技術、形式的手法、高信頼性技術、ユーザインタフェースなどの研究を含みます。

2. 研究課題・研究者名

別紙一覧表参照

3. 選考方針

選考の基本的な考え方は下記の通り。

1) 選考は、「機能と構成」領域に設けた 8 名のアドバイザーの協力を得て、研究総括が行う。

2) 選考方法は、書類選考、面接選考、および総合選考とする。

・ 書類選考においては、1 提案に付き研究総括と 2 名のアドバイザーが査読し、評価する。

・ 面接選考では、可能な限り多くの提案者から直接説明を受け、質疑応答を行う。

3) 個人レベルで行う研究提案であり、萌芽的でかつ独創性に富んでおり、3 年間の研究により科学技術への大きな貢献が期待できそうなものを優先する。また、研究テーマや研究者の所属機関が特定のところに集中しないよう配慮する。

4) 審査は書類選考結果、面接選考結果および研究実施の条件等を加味し、総合的観点から行う。

4. 選考の経緯

選　考	書類選考	面接選考	採用者
対象者数	38 名	15 名	7 名

5. 研究実施期間

平成 13 年 12 月～平成 17 年 3 月

6. 領域の活動状況

・ 領域会議:　7 回

・ 研究報告会:1 回

・ 研究者訪問:研究総括が、研究開始後、全研究者を訪問。その後、研究実施場所の移動時あるいは適宜、研究者を訪問 (研究総括および／または技術参事)。

7. 評価の手続き

研究総括が各研究者からの報告および自己評価を基に、アドバイザーの協力を得て行った。また

研究終了時に当領域が開催する研究報告会（一般公開）等の参加者の意見を参考とした。

(評価の流れ)
平成 16 年 11 月　　研究期間終了（1 名）
平成 17 年 1 月　　研究期間終了（1 名）
平成 17 年 3 月　　研究期間終了（5 名）
平成 17 年 1 月　　研究報告会を東京国際フォーラムにて開催
平成 17 年 3 月　　研究報告書および研究課題別評価提出
平成 17 年 3 月　　研究総括による評価

8．評価項目
（イ） 外部発表（論文、口頭発表等）、研究を通して得られた新しい知見などの研究成果
（ロ） 得られた研究成果の科学技術への貢献

9．研究結果
平成 13 年度は 38 件の提案の中から、ソフトウェアの高信頼性化に関わる技術の研究（青木研究者、亀山研究者、中島研究者）、ソフトウェアの柔軟な運用を可能にする技術の研究（神谷研究者）、ソフトウェアノンストップ運用支援に関わる技術の研究（橋本研究者）、複雑な高性能並列プログラムを簡単に作成できる技術の研究（胡研究者）、およびマルチタスク多重処理能力を飛躍的に向上させるプロセッサの研究（田中研究者）を採択した。応募数はやや少な目であったが、採択した提案はいずれも優れたものであった。

3年間の研究期間を終了し、7名の研究者全員が当初の期待通りの研究成果を上げ、我が国の科学技術発展に大きな貢献ができたものと考える。平成 17 年 1 月 28 日に東京国際フォーラムにて開催した、広く一般を対象とした研究報告会では、多くの好意的な感想を頂いた。また、このうち4名の研究者が実装したプログラムは、既に公開ないしは近々公開を予定しており、ご利用頂きたい。

研究者毎に言えば、青木利晃研究者は「オブジェクト指向分析モデルの形式的構築法と検証法」の研究で、定理証明システムを用いてオブジェクト指向分析・設計モデルの検証を行う手法を確立し、性能面を含め十分実用に耐え得ることを確認した。また、検証コストを下げるために、証明の再利用により正しいモデルを段階的に構築する手法を提案した。

神谷年洋研究者は、「オブジェクトとメディアによるソフトウェア構造化」の研究で、ソフトウェアの動的な修正を可能にする手法（SOMA）を実現するための枠組みとして、プログラミング言語やその処理系（Soja）の開発を行った。

亀山幸義研究者は、「プログラムのメタレベルを表現・操作する言語機構」の研究で、非常に精細なコントロールを得られる限定継続の機構とメタ変数・文脈の機構について研究を行った。前者については、簡潔でかつ必要十分な規則を発見し、高いレベルで定式化を行うことができた。後者についてはメタプログラムのモデル化の基礎となる体系の定式化に成功した。

胡振江研究者は、「スケルトン並列プログラムの最適化」の研究で、少数の並列計算パターンを基本ブロックとし、これらを組合せて効率的な並列プログラムを構築するための理論的枠組を与えた。また、並列プログラミングを支援する環境を構築し、その有効性を確認した。

田中清史研究者は、「実時間マルチタスク処理を支援するプロセッサアーキテクチャ」の研究で、プロセッサコンテキストバッファ、高速割り込み応答機構および再構成キャッシュメモリなどの各種機構を提案し、その有効性の評価を行った。また、これらの新機構を搭載したプロセッサ LSI を実際に設計・製造し、評価を実施した。

中島震研究者は、「高信頼性 Web サービス」の研究で、複合 Web サービスにおける安全性ならびにセキュリティを事前に検査できる技術を開発した。特に Web サービス連携記述の解析にモデ

ル検査技術を応用する方法は、先駆的研究として高い評価を得、その論文は諸外国の多くの研究者から引用された。

橋本政朋研究者は、「ユビキタス環境を支えるプログラミング言語システム」の研究で、ユビキタスコンピューティング環境で利用されるコンピュータやプログラムに対する簡単かつ安全な実行時アップデートシステムの実現を目指し、モデル構築と Java 言語用プロトタイプシステムの実装を行った。

10. 評価者

研究総括 片山 卓也　北陸先端科学技術大学院大学情報科学研究科 教授

領域アドバイザー（所属・役職は現職）

青山 幹雄	南山大学 数理情報学部 教授（平成 13 年 4 月～）
阿草 清滋	名古屋大学 大学院情報科学研究科 研究科長・教授
市川 晴久	日本電信電話(株) NTT 未来ねっと研究所長
岩野 和生	日本アイ・ビー・エム株式会社（～平成 13 年 3 月）
菊野 亨	大阪大学 中之島センター長・大学院情報科学研究科 教授
中島 秀之	公立はこだて未来大学 学長
南谷 崇	東京大学 先端科学技術研究センター 教授
湯淺 太一	京都大学 大学院情報学研究科 教授
米崎 直樹	東京工業大学 大学院情報理工学研究科 教授

(参考)

(1) 外部発表件数

	国 内	国 際	計
論 文	19	13	32
口 頭	44	45	89
出版等	10	5	15
合 計	73	63	136

(2) 特許出願数

国 内	国 際	計
0	1	1

(3) 受賞等

- 神谷年洋：第35回市村学術賞貢献賞 (2003/04/25；大阪大学井上克郎教授，同楠本真二助教授と合同受賞)
- 中島 震：2003年度日本ソフトウェア科学会論文賞 (2004年6月)

(4) ソフトウェア公開

4件

別紙

「機能と構成」領域　研究課題名および研究者氏名

研究者氏名 (参加形態)	研 究 課 題 名 (研究実施場所)	現　　職 (応募時所属)	研究費 (百万円)
青木 利晃 (兼任)	オブジェクト指向分析モデルの形式的構築法と検証法 (北陸先端科学技術大学院大学)	北陸先端科学技術大学院大学 情報科学研究科 助手 (同上)	35
神谷 年洋 (専任)	オブジェクトとメディアによるソフトウェア構造化 (大阪大学)	独立行政法人科学技術振興機構 さきがけ研究者 (科学技術振興事業団若手個人研究推進事業 グループメンバー)	21
亀山 幸義 (兼任)	プログラムのメタレベルを表現･操作する言語機構 (筑波大学)	筑波大学大学院システム情報工学研究科 助教授 (同大学電子･情報工学系 助教授)	18
胡 振江 (兼任)	スケルトン並列プログラムの最適化 (東京大学)	東京大学大学院情報理工学系研究科 助教授 (同上)	30
田中 清史 (兼任)	実時間マルチタスク処理を支援するプロセッサアーキテクチャ (北陸先端科学技術大学院大学)	北陸先端科学技術大学院大学 情報科学研究科 助教授 (同上)	39
中島 震 (兼任)	高信頼性 Web サービス (国立情報学研究所)	国立情報学研究所 ソフトウェア研究系 教授 (日本電気株式会社ネットワーキング研究所 主管研究員)	15
橋本 政朋 (兼任)	ユビキタス環境を支えるプログラミング言語システム (産業技術総合研究所)	産業技術総合研究所 情報技術研究部門 研究員 (同研究所サイバーアシスト研究センター 研究員)	12

研究課題別評価

1　研究課題名：オブジェクト指向分析モデルの形式的構築法と検証法

2　研究者氏名：青木　利晃

3　研究の狙い：

現在、IT の進歩により、ソフトウェアの規模、及び、新規ソフトウェアの需要が共に急激に増加し続けている。しかしながら、それらのうち誤りを含むものは少なくない。今後、ますますソフトウェアの規模が大きくなり、社会システムの様々な部分が電子化されてくるであろう。これらのことを考えると、ソフトウェアの正しさを保証する手法の確立が急務である。本研究では、ソフトウェアの信頼性を向上させるために、その検証法について研究を行った。狙いは以下の 2 つである。

(1)オブジェクト指向分析・設計モデルの検証

ソフトウェアの規模が大きくなると、ソフトウェアを実現しているプログラムではなく、その設計書の役割が重要になってくる。また、ソフトウェアの複雑さや規模の管理も重要であり、それらに対処できるオブジェクト指向開発法が注目されている。オブジェクト指向開発法では、設計書を、UML などの記法を用いて分析モデル、または、設計モデルとして記述する。そこで、本研究では、そのようなオブジェクト指向分析・設計モデルに焦点を当てた。

(2)定理証明手法とモデル検査手法の適用

ソフトウェアの検証のためのアプローチとしては、定理証明手法とモデル検査手法がある。定理証明手法では、ソフトウェアの性質を推論規則などを用いて対話的に証明することにより、正しさを保証する。モデル検査手法では、ソフトウェアを有限状態で表現し、網羅的に探索して自動的に正しさを保証する。これらの手法には長所と短所がある。前者は、高い記述能力と検証能力を持っているが、対話的に証明を行うためコストが高くなる。一方、後者は、記述能力が低くソフトウェアを有限状態に抽象化する必要があるが、自動的に検証を行うことができる。

本研究では、まず、UML で記述された分析・設計モデルを、定理証明手法を適用により検証する手法を提案した。しかしながら、この手法では、対話的に証明を行うため、検証コストが高くなってしまう。そこで、コストを低くするための手段として、検証を再利用する手法を提案した。また、モデル検査技術により分析・設計モデルを検証する手法についても研究を行った。

4　研究成果：

主な研究成果として以下の 4 つがある。

(1)オブジェクト指向分析・設計モデルの定理証明システムによる検証法の提案。
(2)検証結果の再利用法の提案。
(3)計算機支援環境 F-Developer のバージョンアップ。
(4)モデル検査ツールによる分析・設計モデルの検証法の提案。

4．1　オブジェクト指向分析・設計モデルの定理証明システムによる検証法の提案

これまでに、UML のクラスダイアグラムとステートチャートダイアグラムに基づいた検証法は提案していたが、記述能力に問題があった。具体的には、クラスが持つメソッドを、オブジェクト指向的では無い、副作用無しの関数として記述していた。属性が持つ値は原始的な値のみであり、オブジェクトへの参照を持つことができなかった。また、表明などの性質も、述語論理で記述している。本来、これらの記述は、オブジェクト指向の観点から記述すべきものである。このようなメソッド定義や性質をオブジェクト指向的に記述するための言語として、様々なアクション言語と制約言語が

提案されている。そこで、それらを用いて記述されたオブジェクト指向分析・設計モデルを定理証明システムを用いて検証する手法について提案した。

この手法では、オブジェクト指向の概念を定理証明システムで扱うために、それらを実現する仕組みを作り込む必要があり、余分な証明ステップがかかる可能性がある。そこで、記述能力の他に、定理証明システムで取り扱う際に、オーバヘッドがかからないよう工夫した。評価のためにいくらかのデザインパターンを記述し、それらで成立すべき性質について検証を行った。それらの 1 つである ObserverPattern では、定理証明システム HOL で約 200 行で記述することができた。また、状態が常に反映されるという性質の証明では 4 つの補題を証明することになるが、それぞれの補題はすべて 1 ステップで証明できた。これらのことは、提案した手法が、十分な表現能力を持っていること、および、変換された記述で証明を行う場合のオーバヘッドが小さいことを意味している。

4．2 検証結果の再利用法の提案。

提案した検証手法では、検証対象の分析・設計モデルが複雑なデータ型や動作を含んでいる場合、特に、証明過程で帰納法や場合分けが必要な場合、証明ステップが多くなったり、適用する推論規則の選択が難しくなる。そこで、検証コストを下げるために、証明の再利用して正しいモデルの段階的構築する手法を提案した。この手法を、表明主導手法(Assertion Driven Approach)と呼ぶ。証明の再利用では、重複した証明を再度行う必要が無いのでコストを下げることができる。また、段階的構築では、早期に誤りが発見できるため、手戻りコストを下げることができる。

図 1 に表明主導手法の概要を示す。この手法は 2 つのフェーズに分かれている。1 つ目は、個々のソフトウェア開発ではなく、ソフトウェアの集合を考えるフェーズ、すなわち、領域分析である。表明主導手法では、領域分析のフェーズで、領域に現れる共通な動作を抽出し、それにまつわる性質を証明して領域ライブラリに蓄積する。この動作と性質は、すでに検証済みのモデルからも抽出することができる。2 つ目のフェーズは、個々のソフトウェアを作成するフェーズ、すなわち、通常のソフトウェア開発である。このソフトウェア開発では、領域ライブラリから適切な動作を探し、アレンジする。そして、モデルに段階的に追加していく。このアレンジと、動作の追加では、動作やモデルの正しさが崩れないようにする。本研究では、このような領域ライブラリと、それを用いた正しいモデルの段階的構築のための基礎理論を提案した。

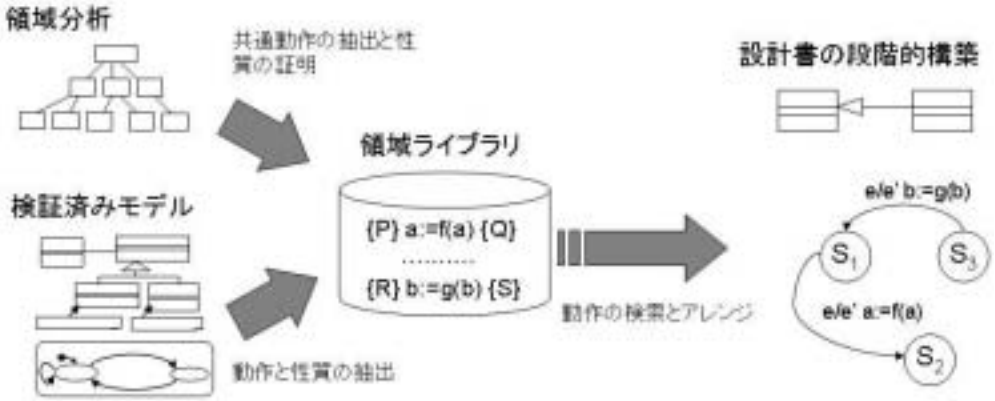


図 1 表明主導手法の概要

4．3 計算機支援環境 F-Developer のバージョンアップ。

本研究では、以上で提案した手法を支援するツール F-Developer を開発している。このツールは、2 つのサブシステム F-Prototyper と F-Verifier から構成されている。F-Prototyper では、UML で記述されたモデルをプロトタイプ実行できる。F-Verifier は、そのようなモデルを定理証明システム HOL で取り扱い可能な形式に自動的に変換して証明を支援する。さきがけ研究をはじめる段

階ではバージョンが 0.12 で非常に限られた機能しかなかったが、現在では、通常のモデル化作業に耐えられるくらいの機能が実装されている。現在は、F-Developer ver.0.2(2003 Dec.版) for Windows を最新版として公開している。

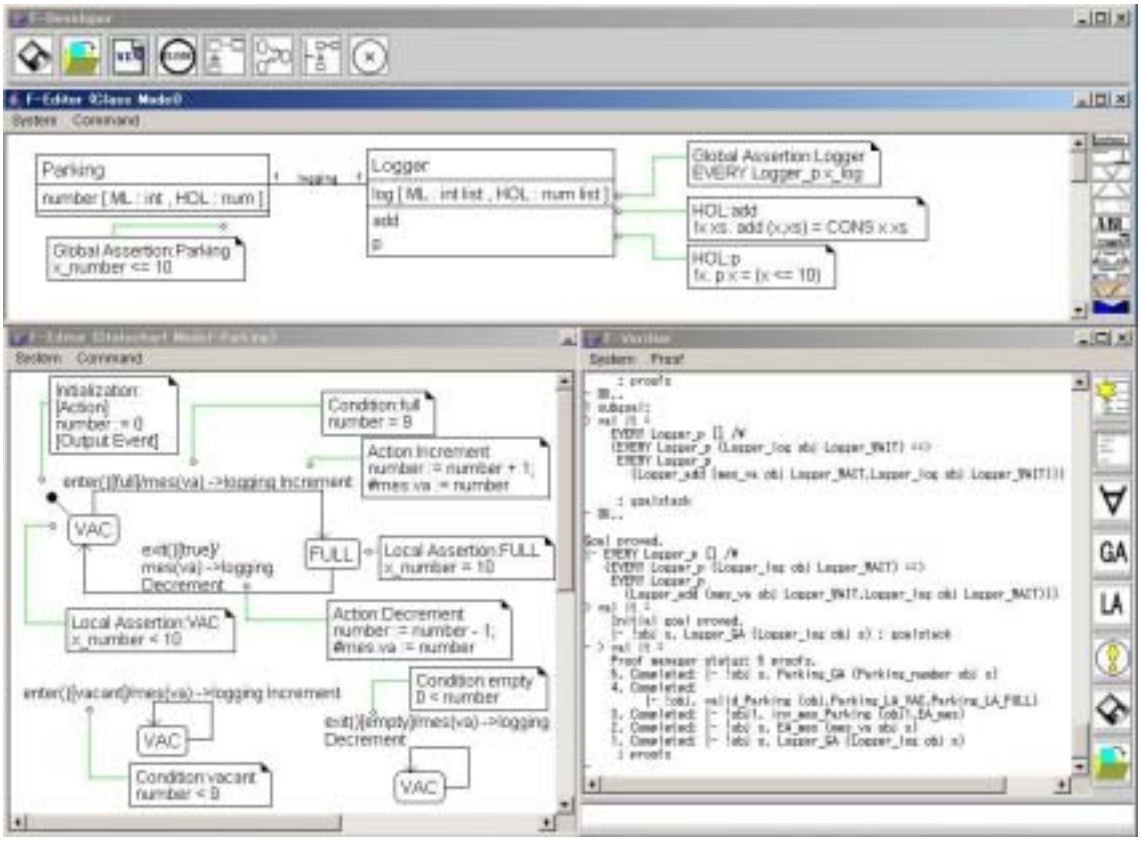


図 2 計算機支援環境の画面スナップショット

4．4 モデル検査ツールによる分析･設計モデルの検証法の提案。

モデル検査技術では、並行動作する複数の有限状態モデルの動作を網羅的に検査することにより、到達性やデッドロックなどの性質を調べる。最近では、モデル検査ツールの実装が進み、マニュアルなども整備されてきた。これにより、様々な対象に適用されるようになり、プログラムやその開発工程で作成されるモデルを対象とした検証法も提案されている。

モデル検査ツールを用いて設計モデルの検査を行う場合、調べたい性質毎に記述や検査の手法が異なる。そこで、設計モデルにおけるデータフロー解析、及び、センサーの取り扱い方に関する研究を行った。

4．4．1 データフロー解析

組み込みシステム開発、特に制御系のシステムでは、データの変換的側面、すなわち、データフローのモデル化を行う。近年、組み込みシステムの規模の劇的な増大、及び、多様化により、オブジェクト指向アプローチが取られるようになってきた。データフローの視点は、構造化分析設計アプローチで中心的な役割を担って来たが、オブジェクト指向アプローチにおいても、特に、組み込みオブジェクト指向開発では重要である。そこで、オブジェクト指向設計モデルにおけるデータフローの意味論を定義し、それを解析する手法を提案した。

オブジェクト指向設計モデルでは、オブジェクトは、互いに協調しながら並行動作している。それぞれのオブジェクトにおけるアクションの実行などによりデータフローが生じるが、それらは、オブジェクトの並行性を考慮して解析する必要がある。そこで、並行性解析に用いられるモデル検査ツール Spin により、オブジェクトの並行協調動作の結果生じるデータフローを、それらの可能な振舞いを網羅的に探索することにより解析する手法を提案した。これにより、オブジェクト指向設計モデルとデータフローを明確に対応づけ、自動解析することが可能になった。今後は、オブジェクト指向設計と Simulink などで用いられるブロック線図による設計の整合性検証などへの応用を考えている。

4．4．2 センサーの取り扱い

組込みシステムの分野では、従来から、実時間性、メモリ制約などのハードウェア制約の充足性が主な議論の対象であった。近年、組込みシステムの大規模化、多様化に伴い、組込みシステムの定義や議論の対象が変化してきた。組込みシステムと呼ばれるものは多種多様であり、通常のシステムやソフトウェアとの境界が曖昧になってきたため、それらを明確に分類することが困難なのが現状である。そこで、まず、組込みシステムで重要な役割を担っている部品、センサーに注目することにして、モデル検査を適用する手法について研究を行った。

これまでに、センサーの仕様を SCR(Software Cost Reduction)法のモード遷移表を用いてモデル化する手法、および、それをモデル検査ツール SMV で検証する手法について提案した。モード遷移表には自然数などの変数が記述されており、SMV で検証する前に、それらを抽象化する必要がある。提案した手法では、抽象解釈法の一種であるデータマッピング法で、定理証明システムを用いながら効率的に抽象化を行う手法を提案した。今後は、検査した仕様に基づいて、センサーへのアクセスメカニズム、および、制御のためのスケジューリングを考慮に入れた設計モデルをいかに構成するかについて研究を行う予定である。

5 自己評価:

本研究では，(1)オブジェクト指向分析･設計モデルを検証する手法と，(2)正しいモデルを構築する原理を明確にすることを目指した．(1)に関しては，研究を進める過程で検証対象となるモデルの記述能力と検証能力の低さが明らかになったが，アクション言語と制約言語、および、その取り扱い方を提案することにより解決することができ，目標を達成することができた．

(2)に関しては，正しいモデルを段階的に構築する規則や条件を明確にして，表明主導手法としてまとめた．これにより、基本的な原理を明らかにすることができた。当初，さらに実際のモデル構築で行われるような作業に対応する構築手法の提案を目指していたが，現時点では，明確には対応づけられていない．モデルへの要素の追加の単位がアクションや状態遷移などであり，粒度が細かいためである．今後は，デザインパターンやソフトウェアコンポーネントといった，大きな粒度の要素を扱うことができるようにする予定である．このような大きな粒度の要素の取り扱いは，現在までに提案した小さな粒度の要素に関する原理の積み重ねにより実現できそうな感触は持っている．また，提案した手法を計算機支援環境 F-Developer に実装することも今後の課題である．

主な検証技法としては定理証明技法とモデル検査技法があり，当初は，前者のみに焦点を当てていた．しかしながら，研究を進めるにつれて，オブジェクト指向分析･設計モデルの検証では，双方の使い分けが重要であることを実感した．そこで，途中から，モデル検査技法に関する研究も並行して行うことにした．そして，2 つの検証手法を提案することができた．これは当初の計画には無い成果である．これらの手法も，今後，F-Developer 上に実装する予定である．

また，このように提案した手法を F-Developer 上に実装していくことを考えると，現在のアーキテクチャでは困難であることもわかってきた．現在は，特定の検証手法用に実装されているので，複数の検証手法を柔軟に組み込めるようにはなっていない．そこで，そのようなアーキテクチャの

設計と実装を行いたいと考えている．これは，研究を進めてきて発見した新たな研究課題である．

6 研究総括の見解：

青木氏の研究は，定理証明などによる形式的検証により，正しいソフトウェアの開発を行う手法とその環境に関して研究を行ったものである．従来から形式手法によるソフトウェアの検証は研究されてきたが，検証コストが高いことにより産業界で採用されるには至っていない．青木氏は，この問題を解決するための，オブジェクト指向分析・設計モデルの検証のコストを下げることを目的として，定理証明やモデル検査による検証方法論や検証支援環境，検証結果の再利用方法論などの研究を行った．オブジェクト指向分析設計モデルの検証としては先進的な結果を出し，また，開発したツール(F-Developer)の公開など形式技術のソフトウェア開発への適用に関して優れた研究成果をあげたと評価される．

7 主な論文等：

論文

1. 青木利晃、片山卓也：オブジェクト指向分析モデルの検証支援環境、第 19 回 日本ソフトウェア科学会 全国大会論文集(CD-ROM), 2002.
2. 岡崎光隆、青木利晃、片山卓也：並行オブジェクトモデルから並行スレッドモデルへの変換法、情報処理学会 ソフトウェア工学研究会 研究報告 2003-SE-140，pp.39-46，2003.
3. 立石孝彰、 青木利晃、 片山卓也:振舞い近似手法を用いたステートチャートに対する不変性の検証、情報処理学会論文誌 Vol.44 No.6，pp.1448-1460，2003．
4. 青木利晃、片山卓也：状態遷移図の段階的構築のための論理的基盤、情報処理学会 ソフトウェア工学研究会 研究報告 2003-SE-143，pp.21-28，2003．
5. 青木利晃、岸知二、片山卓也:定理証明システムとモデル検査ツールを併用した設計モデルの検証実験、日本ソフトウェア科学会 第1回ディペンダブルソフトウェア研究会、 pp.49-58, 2004.
6. 青木利晃、片山卓也：オブジェクト指向分析モデルにおけるデータフローの形式化と解析手法、日本ソフトウェア科学会 学会誌 コンピュータソフトウェア、Vol.21, No.4,pp.1-26, July, 2004.
7. 青木利晃、岸知二、片山卓也：センサーのモデル化とモデル検査技術の適用について、組込みソフトウェアシンポジウム 2004, pp.118-125, 2004.
8. 青木利晃，片山卓也：アクション言語と制約言語を用いて記述されたオブジェクト指向設計モデルの検証法，日本ソフトウェア科学会 第2回ディペンダブルソフトウェア研究会, pp.61-70, 2005．
9. 矢竹健朗，青木利晃，片山卓也:コラボレーションに基づくオブジェクト指向モデルの検証，日本ソフトウェア科学会 学会誌 コンピュータソフトウェア，Vol.22, No.1,pp.58-76, 2005.
10. Takaaki Tateishi, Toshiaki Aoki and Takuya Katayama: Successive Behavior Approximation Method for Verifying Distributed Objects, Third International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'02), pp.439-446, 2002.
11. Mitsutaka Okazaki, Toshiaki Aoki and Takuya Katayama: Extracting threads from concurrent objects for the design of embedded systems, Ninth Asia-Pacific Software Engineering Conference 2002, pp.107-116, 2002.
12. Mitsutaka Okazaki, Toshiaki Aoki and Takuya Katayama: Formalizing sequence diagrams and state machines using Concurrent Regular Expression: Proceedings of 2nd International Workshop on Scenarios and State Machines: Models, Algorithms,

and Tools SCESM'03, pp.74-79, 2003.
13. Kenro Yatake, Toshiaki Aoki and Takuya Katayama: Collaboration-based Verification of Object-Oriented models in HOL, Proceedings of the 2nd International Workshop on Verification and Validation of Enterprise Information Systems, VVEIS 2004, pp.78-80, 2004.
14. Toshiaki Aoki and Takuya Katayama: Foundations for Evolutionary Construction of State Transition Models, the Seventh International Workshop on Principles of Software Evolution 2004, pp.143-146, 2004.

研究課題別評価

1　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

2　研究者氏名：神谷　年洋

3　研究の狙い：

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。

4　研究成果：

4．1　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、（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）動的で安全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。

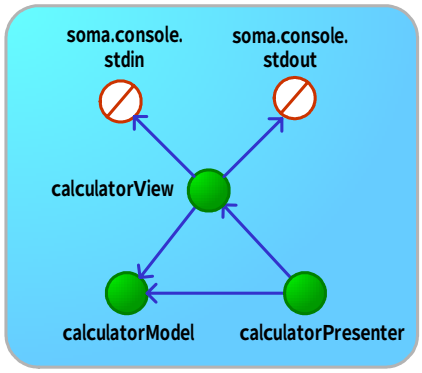


図1　SOMA モデルで表現した MVC 電卓

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、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。

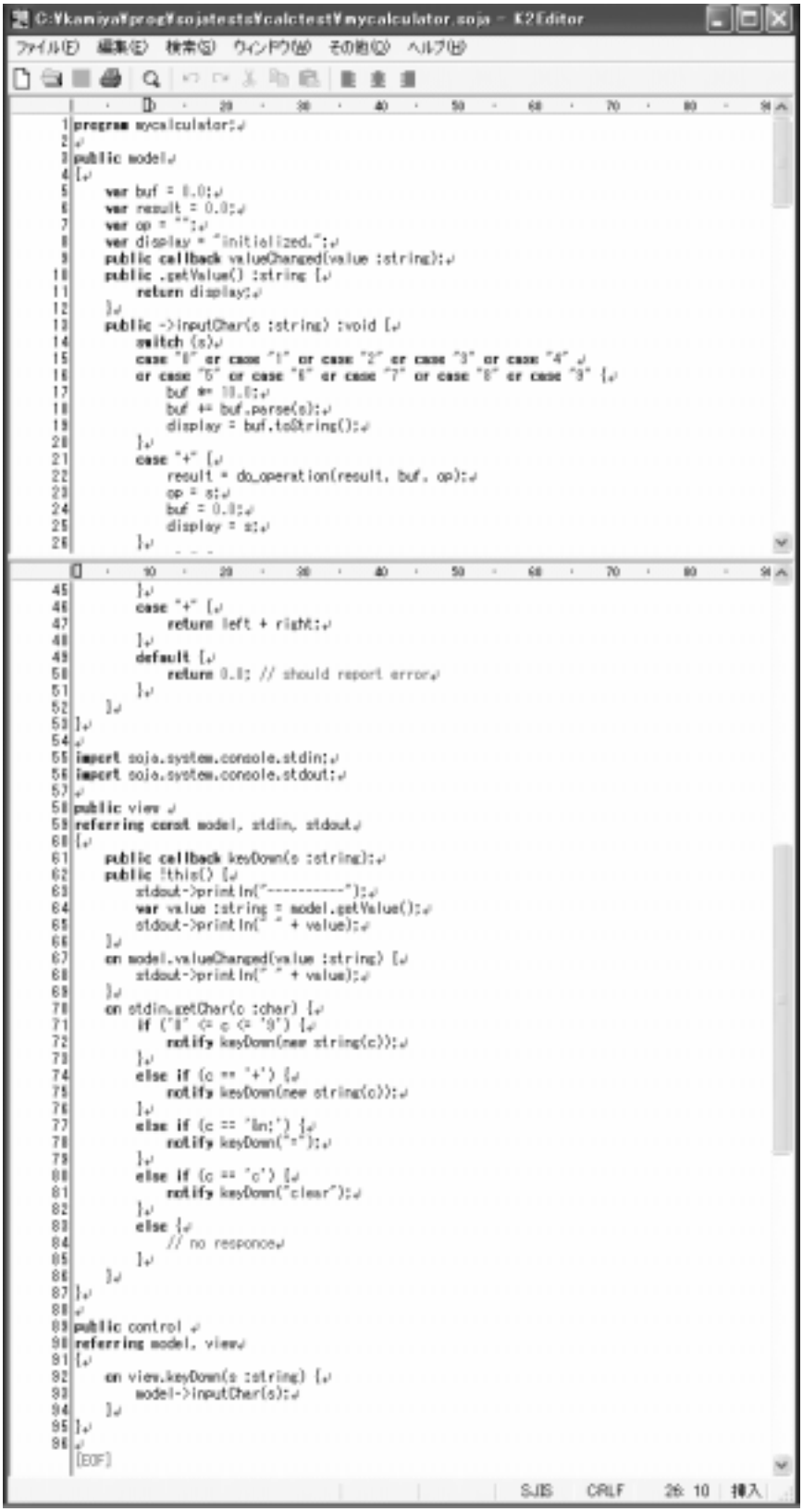


図2　MVC 電卓のソースコード(一部)

4．2　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。

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。

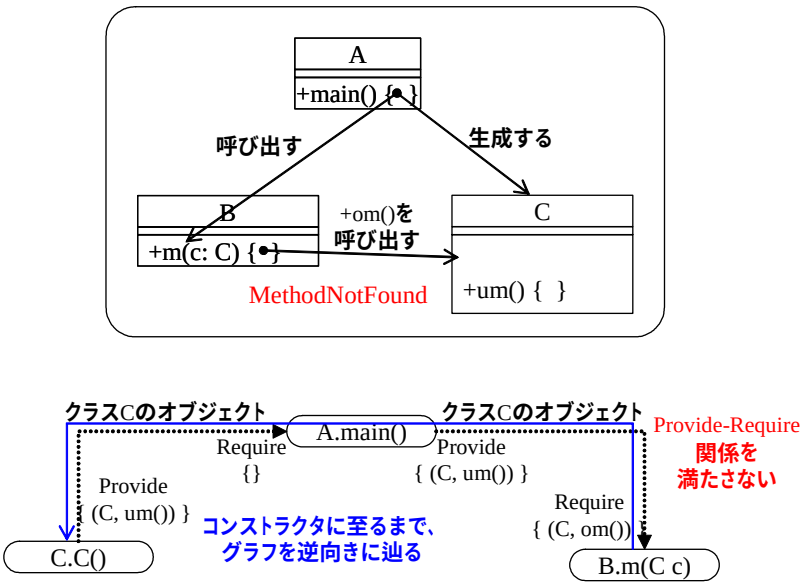


図3　提案手法によるインターフェイス互換性問題の検出と原因箇所の特定

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。

4．3　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。

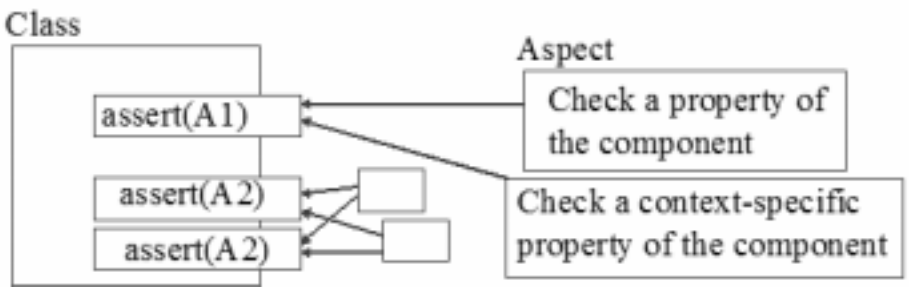


図4　アスペクトによるアサーション

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。

4．4　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。

5　自己評価：

本研究では、当初、クラスライブラリのようなツールキットの開発を目指しましたが、研究・開発の課程で従来のソフトウェア開発技術（特にプログラミング言語）では問題とは見なされていなかった事柄が大きな問題となることが判明し、最終的には、それらの問題の解決も含め、プログラミング言語処理系の開発を行いました。

具体的には、問題とはソフトウェアを利用者が動的に変更することを想定した構成管理や、ソフトウェアの変更を前提とした例外処理、アサーションといったものです。前者の構成管理は、主にソフトウェアの利用者に対して表面化するようなソフトウェアの安全性（ソフトウェアが正しく変更できるか、間違った変更操作を取り消すことができるか）や利便性（簡単に変更できるか）に関わる

問題であり、変更されることを想定していないソフトウェアでは問題となっていないような内部状態の取り扱いや、どのような変更操作を提供すべきかを研究しました（4．2項参照）。また、後者の例外処理やアサーションは、主にソフトウェアの開発者に対して表面化するような、進化を前提としたソフトウェアの記述方法に関わる問題であり、制約や例外処理のような、あるソフトウェア（部品）が周り（の部品）とのやりとりをする際の処理を、周りが動的に変更される可能性のあるソフトウェアではどのように記述できるのか、記述するべきかを研究しました（4．3項参照）。

結果として、本研究課題で提案したソフトウェアの動的な修正を可能にする手法（SOMA と命名）を実現するための枠組みとして、Soja と呼ばれるプログラミング言語やその処理系を開発しました。また、既に存在するソフトウェアに対して、どのようにして動的な修正や構成管理を適用することができるかという問題を研究しました。現在、Java で記述されたソフトウェアのオブジェクト（インスタンス）を分析し、必要に応じてソフトウェアに修正を加えるためのツール IIAnalyzer(仮称)を開発中であり、平成17年4月末にリリースする予定にしています。

6 研究総括の見解：

オブジェクト指向ソフトウェア開発技術では，ソフトウェアはオブジェクトという情報隠蔽されたモジュールによって記述され，モジュールの再利用性や記述の堅牢性が高められている反面，オブジェクト間の関係記述が困難であり，ソフトウェアの進化が行いにくい．神谷氏は，オブジェクト間の関係記述のためにメディアと言う概念を提案し，オブジェクト＋メディアによってソフトウェアを記述する方法論 SOMA を提唱し，SOMA によりソフトウェアを記述するための言語環境 soja の開発を行った．提案した概念はユニークで有効であると考えられる．新しいアイディアの提案を積極的に行った研究であり，十分に評価できる．

7 主な論文等：

(1) 特許

発 明 者 ：神谷 年洋
発明の名称：類似部分文字列検出方法、類似部分文字列検出装置、類似部分文字列検出プログラム、および該プログラムを記録した記録媒体
出 願 人 ：科学技術振興機構
出 願 日 ：平成 14 年 6 月 28 日

(2) 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).

(3) 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).

(4) 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, 蔵川 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).

(5) **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**.

(6) **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).

研究課題別評価

1　研究課題名：プログラムのメタレベルを表現・操作する言語機構

2　研究者氏名：亀山　幸義

3　研究の狙い：

コンピュータ上でプログラムを実行するとき、プログラムのコード自身のほかに、環境や制御などの情報を利用します。環境は、変数とその値の対応であり、制御はプログラムカウンタ、すなわち、次に実行するプログラム中の番地の情報です。環境や制御は、通常のプログラム言語では、プログラムから直接操作可能な対象物ではなく、直接操作ができないレベル（メタレベル）にあるもの、すなわち、メタレベルの概念です。

メタレベル概念をプログラムの中から直接操作する機能、つまり、『メタレベル機構』を使うことにより、従来は極めて複雑な記述が必要だったプログラムを、簡潔にわかりやすく記述できることが知られており、研究されてきました。しかし、プログラム言語の安易な拡張は信頼性を損ねる可能性があります。たとえば、メタレベル機構より更に強力なものとして、プログラムの『自己反映』があります。これは、インタープリタなどの言語処理系をプログラムの中から変更できるため、「どんな計算でも出来る」機構となっており、静的な解析や検証をすることが困難です。

本研究は、メタレベル機構による高機能性・高記述性を、高信頼性と両立させた形で利用できるようにすることを目標としました。ここでポイントとなるのは、考察の対象とするメタレベル機構を、通常のプログラム言語の意味論に即した概念に対応するもののみに限定することです。このような概念は、自己反映計算のような無制限の拡張と異なり、拡張した言語の意味を、元の言語の意味から構築できると言う利点があります。本研究では、このように、意味論の助けを借りて、形式的体系によるモデリングと、それに対する検証方法の確立を行うことを目的としました。

本研究では、応用上重要なメタレベル機構として、精細な制御を実現する限定継続（delimited continuation）とその発展形、および、メタ変数と文脈を取り上げて、この目的を達成するための研究を行いました。

4　研究成果：

4．1　非局所脱出の体系に関する研究

『さきがけ』研究の前段階として、非局所脱出（non-local exit）の機構の定式化と理論的解析に関する研究を行っており、『さきがけ』研究の期間に論文出版をしました。

非局所脱出は、多くの現代的プログラム言語で実装されている例外処理の機能に相当します。たとえば、Java の try/catch, Lisp 系言語における catch/throw, ＭＬ言語における exception の機構などです。これは、プログラムの制御を切り替えるという意味でメタレベル機構の一種と考えられます。

ここで取り扱ったのは、非局所脱出機構を持つプログラムの型システムと停止性の関係です。ループなどがない単純なプログラムでも、非局所脱出の機構を何の制限もなく導入すると、停止しないプログラムを書けてしまいます。脱出先を表すタグが動的な範囲（extent）である場合には停止しないプログラムが存在することを足がかりにして、タグが静的な範囲を持ち、適当な型付け規則があるときには、非局所脱出機構を導入したプログラムが停止性（強正規化可能性、どのような順番で計算しても、実行がいつか停止すること）を持つことを証明しました。ここで用いた型システムは通常のプログラム言語とも両立するものとなっており、Java の try/catch など広い範囲で応用がきくものです。

停止性そのものは極めて理論的な成果ですが、停止性を満たすかどうかは、型を持つ計算体系が適切に構築されたかどうかの一種の試金石となっています。非局所脱出の体系においても、

停止性の証明をうまく構成するためには、むしろ計算規則が弱すぎる（非局所脱出の機構を使う規則が弱い）という逆説的な結果が得られ、言語設計への示唆が得られました。また、従来不明だった『脱出タグをあらわす型』の論理的意味付けが得られました。これらの成果は TCS 誌で発表しました。

静的なケースで停止性が成立するという結果は、脱出先を表すタグが動的か静的かに依存して、プログラムの停止性が変わってくることを意味しています。そこで、動的なタグを持つ体系について型システムを若干拡張した形で定式化し、その論理的意味付けを考察し、APLAS'02 国際会議で発表しました。

4．2 精密な継続を扱う機構の公理化

『継続』（continuation）は、元来は、表示的意味論においてジャンプ命令を解釈するために発明された概念であり、「計算の残りの部分」を意味しています。プログラムの実行のある時点Aで、その時点での継続を保存し、計算が進んだ時点Bにおいて、その保存した継続Aを呼び出すとすれば、AからBへのジャンプ命令をあらわしていると考えられます。このように「現在の継続」を保存し、後に呼び出せるようにすることにより、非局所脱出、ループ、コルーティンなど多様な制御を表現することができます。「現在の継続」を保存する機能は、Scheme 言語の call/cc（call with current continuation）や Standard ML/NJ 言語の callcc として実現されており、これらに対する研究も多くの蓄積があります。

call/cc による継続の機能は非常に強力と考えられていましたが、近年、call/cc ではうまく記述できないプログラムのパターンがあることもわかってきました。call/cc の機構は、(1)「残りの計算の全て」を一括して保存することしかできず、「残りの計算の一部」を指定することができない、(2)保存した継続を呼び出すと、その時点の継続が捨てられてしまう（継続が入れ替わる）、の2つの欠点があり、これらの欠点により表現できないプログラムが存在しています。そこで、より精密な継続を表現するため、限定継続（delimited continuation）の機構が提案されました。これは、部分継続、あるいは、合成可能継続という別名を持つことからわかるように、残りの計算のうちの一部を指定して保存することができ、さらに、保存した継続を呼び出すとき、現在の継続は捨てられずに2つの継続が合成されるというものです。本研究では、限定継続の機構のうち最も広く使われている shift/reset 機構について研究を行いました。

$$1+(2\times(\mathrm{call/cc}\ \lambda k.\ 3+(k\ 4)))\ \rightarrow\ 1+(2\times(3+(k\ 4)))\quad \text{where}\quad k=(1+(2\times\Box))$$
$$\rightarrow\ 1+(2\times 4)\ \rightarrow\ 9$$
$$1+\langle 2\times(\mathrm{shift}\ k.\ 3+(k\ 4))\rangle\ \rightarrow\ 1+\langle 3+(k\ 4)\rangle\quad \text{where}\quad k=\langle 2\times\Box\rangle$$
$$\rightarrow\ 1+\langle 3+\langle 2\times 4\rangle\rangle\ \rightarrow\ 12$$

図1 call/cc と shift/reset の違い（<…>が reset をあらわす）

従来、shift/reset 機構については、CPS 変換（関数型プログラム言語のコンパイラで使われる変換）による意味論、応用例、効率的な実装などが知られていましたが、検証のための規則はわかっていませんでした。また、操作的意味論は informal なものとして与えられていただけであり、CPS 変換による意味論に照らして十分であるかどうか等はわかっていませんでした。したがって、実装は存在しても本当に正しい実装かどうかの証明を与えることができませんでした。

本研究では、shift/reset を含む2つのプログラムが等しいことを証明する公理（等式）の集合を求めることに成功しました。この公理の集合は、既に与えられていた CPS 変換による意味論に対して健全かつ完全（必要十分）なものであり、かつ、変換によらずに直接的に shift/reset に対して推論が可能なものであるため、ソフトウェア検証への応用が可能なものです。この結果を得るためには、CPS 変換に対して、適切な逆変換を見つけ出すという点が鍵となりました。

当初与えた公理の集合はかなり複雑なものでしたが、さきがけ研究の領域会議などの場で当領域3期生の長谷川真人氏と議論を重ね、同氏の協力を得ることにより、公理を大幅に簡単にすることに成功しました。さらに従来よく研究されてきた call/cc の理論との関係を明確にしました。この成果は ICFP'03 国際会議で発表しました。

その後、私たちの公理に基づき、操作的意味論も（ある程度）正当化され、shift/reset に対する研究が一段と進むきっかけとなりました。実際に、その後の CW'04 ワークショップでは、限定継続に関する研究発表が5件中3件あり、また、ICFP'04 国際会議でも別の著者らによる限定継続の定式化の研究が発表されるなど一種のブームのような状況となりました。

4．3 継続の階層化へ

上記の研究は理論的には非常にすっきりしたものでしたが、shift/reset を使ったプログラムの検証には不十分なものでした。なぜなら、shift/reset が1種類しかないことが前提になっているため、shift/reset 機構を異なる目的で使っている2つのプログラムを組み合わせることができないからです。たとえば、shift/reset の代表的利用例であるバックトラッキングと複数解の収集は、それぞれ単独では問題なく動作するものですが、それぞれのプログラムを合成して1つの大きなプログラムの中で使おうとすると2種類の使い方が干渉してしまい、期待した結果が得られません。したがって大規模プログラムにおいては事実上使えない機能となっていました。この解決策として、shift/reset をレベル付けして、異なる使い方には異なるレベルを与えるという階層化が提案され、レベルの数（階層の数）に応じて繰り返し CPS 変換を施すという変換により、意味論が与えられていました。しかし、この変換は極めて複雑であり、これに対する理論的解析はほとんどなされておらず、したがって、階層化した shift/reset もあまり使われていませんでした。

本研究では、この階層化された shift/reset 機構について、変換によらずに直接に検証が容易になる規則の発見を目指しました。最初は、2種類の shift/reset（レベル2までの shift/reset）を持つプログラム言語に対する健全かつ完全な検証規則の集合を発見し、CW'04 ワークショップで発表しました。しかし、ここで得られた集合は、極めて複雑な形をした多数の公理から構成されていたため、直接検証に使うことは困難で、その応用は限定的でした。

その後さらに研究を継続し、最終的に、任意のレベル n の shift/reset に対して、簡潔な検証規則の集合を発見しました。この検証規則の健全性および完全性の証明（必要十分な規則であることの証明）では、階層化された CPS 変換により変換されたプログラムを、型システムの構造により整理したことと、それに基づく逆 CPS 変換の発見が本質的に貢献しています。簡潔さの目安として、次の表に公理の個数の比較表を掲載します。ここでは公理の数だけを示していますが、公理の形自体も前節のものとほぼ同等（レベルに関する添え字を除くと完全に一致する）になっています。

表1 公理（等式）の個数

	λ計算に対する公理	reset に対する公理	shift に対する公理
前節のもの（レベル1）	3個	2個	3個
本節のもの（レベルn）	3個	3個	3個

この成果を CSL'04 国際会議で発表し、論理の立場からの討論を行いました。また、その後、上記の表のうち、reset に対する公理を更に簡単化して2個とすることに成功し、CSL'04 論文の雑誌版として学術雑誌に投稿しました。

4．4 メタ変数と文脈に関する研究

入力としてプログラムを受け取ったり、プログラムを生成したりするプログラムを、簡単のために

『メタプログラム』と総称します。メタプログラムは特別なものではなく、コンパイラなどの言語処理系、プログラム変換システムなどのほか、動的な web ページを作成するためのプログラムなど、様々なソフトウェアが含まれます。一方、メタプログラムの正しさを系統的に示したり、正しいメタプログラムを構成したりするための方法論は十分に研究されているとはいえません。そこで、本研究では、通常のレベル（オブジェクトレベル）とメタレベルとの区別がある簡単なプログラム言語を設計し、その上でメタレベルプログラミングがどのように行われるか基礎的な考察をしました。

メタプログラムにとって本質的なことは、当然ながら、オブジェクトレベルとメタレベルを区別することです。それぞれのレベルの変数、プログラム、環境、継続などの概念は異なるものとなるため、この区別に基づいたメタレベル機構が必要となります。ここでは、メタプログラムの定式化の第一歩として、メタレベルの変数（メタ変数）やそれに対する操作を定式化しました。メタ変数は通常の変数（オブジェクトレベルの変数）と同様に、λ抽象したり値を束縛したりすることができますが、メタ変数に対する代入は通常の代入と異なり、通常の変数のキャプチャを許すという特徴があります。

$$(\lambda X.\ \lambda x.\ \ X)\ (xy) \ \rightarrow\ \lambda y.\ \ xy$$
$$(\lambda X.\ \lambda Y.\ \ X)\ (xY) \ \rightarrow\ \lambda Z.\ \ xY$$

図2　キャプチャのある代入と、キャプチャを避ける代入
（通常の変数は小文字 x, y で表し、メタ変数は大文字 X, Y, Z であらわす）

この現象を、より一般的に定式化するため、メタレベルのメタレベル、すなわち、メタメタレベルや、メタメタメタレベルなど、無限のメタレベル階層を持つ体系の中で、通常の代入とキャプチャを許す代入の2通りの代入を導入しました。このような体系は極めて複雑になるように予想されましたが、適当な型システム（論文では arity と呼んでいます）の導入により、計算体系として必要な性質（合流性、正規形に関する性質、停止性などの性質）を全て満たす体系が得られることを示しました。特に、正規形に関する性質は、「計算が途中でスタックすることがない」ことを保証する重要な性質です。

また、この体系においては、私たちが従来から研究していた、プログラム文脈をうまく表現できること、特に、文脈の合成を自然に表現できることがわかりました。プログラムの文脈というのは、1つだけ穴のあいたプログラムのことであり、プログラム中のある部分に着目したとき、それを取り囲むプログラムと言う意味合いです。文脈は、モバイル計算など異なる環境で動作するプログラムをモデル化する際に、『環境』の側を表現するのに使うことができると言う応用があります。この場合、移動するプログラムの側にあらわれる自由変数が、移動後の環境で値を持つと言う効果を表すため、単なる代入では表現できず、キャプチャを許す代入が必要になります。

ここで述べた成果は、京都大学佐藤雅彦教授、五十嵐淳講師、千葉大学櫻井貴文助教授との共同研究によるものです。従来からの文脈に関する成果は、雑誌 JFLP やコンピュータソフトウェアで発表し、ここで述べたメタ変数と文脈に関する成果発表は CSL'03 国際会議で行いました。

4．5　その他の研究

さきがけ研究は特定の目標を掲げて実行してきましたが、この期間内にも「正しさの保証」という大テーマに関連する他の研究も行ってきました。その1つとして、大学院生の中島一氏とともに、モデル検査における抽象化の研究を行いました。具体的な検証例を取り上げ、そこで用いられる抽象化写像を、より細かいいくつかの基本的抽象化に分解して正しさを保証する方法や、実時間システムに対してモデル検査の効率を改善するため一種の静的解析を組み合わせて対象範囲を絞りこむ方法の研究などを行い、研究成果を発表しました。

5 自己評価:

さきがけ研究における私のテーマは、プログラム言語のメタレベル機構の正しさについての基礎研究です。特に、非常に精細なコントロールを得られる限定継続の機構とメタ変数・文脈の機構について研究を行い、意味論のガイドに基づき、型システム、論理の手法を用いて研究を行いました。前者(コントロールの機構)については、階層化された形での一般的な限定継続機構の検証に関して、必要十分で、かつ、簡潔な規則を発見し、後者(メタ変数)については、メタプログラムのモデル化の基礎となる体系の定式化という成果を得ました。基礎理論という性質上、研究成果が当初の目標としていたレベルに到達したかどうかの判定は簡単にはできませんが、少なくとも、限定継続機構の定式化については、当初予想していたレベルの定式化(レベル1の定式化)を超えて、任意のレベルの定式化を行うことができた点は、大きな成果であると考えています。また、メタ変数の体系については、文脈を自然な形で合成できる体系は世界的にも他にはないことからすると、当初の目的はある程度達成できたと考えています。

本研究の特徴は、メタレベル機構を定式化・モデル化する際に、従来のプログラム理論(とくに形式意味論)と全く異なる枠組みを用意するのではなく、従来の理論のうち利用できるものは利用し、従来理論と調和する形で定式化を行った点です。たとえば、限定継続機構として提案されたもののうち、ＣＰＳ変換という標準的な変換に基づいて意味が規定されているオペレータ(shift/reset オペレータ)を研究対象として取り上げました。この選択が正しかったことは、その後の同種の研究発表が全て shift/reset に関するものであることからもわかります。メタレベル機構は極めて強力な言語拡張の手段ですが、安易な拡張によりプログラム言語の信頼性を損なうことが多いことが知られています。本研究では、もともと意味論に存在している概念のみを扱うことにより、プログラム言語の信頼性を確保しつつ、言語拡張することについて一定の道筋をつけたと考えています。

本研究の基礎的成果をもとに、次のステップの研究として、ここで得た理論を基にして、実際に検証するためのシステムを実装し、具体的な例題を検証することが考えられます。より具体的には、種々のプログラム変換や証明からのプログラム抽出などを一種のメタプログラムとして定式化し、適当な論理体系の上で検証をおこなうことが今後の研究課題です。

6 研究総括の見解:

プログラムの実行を制御する機構をプログラム内部から操作して，プログラムの実行メカニズムを柔軟かつ強力にするためのメタレベル機構に関して，亀山氏は，限定継続とメタ変数・文脈に関して，理論的な研究を行ったものである．このような機構は，注意して導入しないとプログラムの理解や解析を困難にし，正しいプログラムの構成を阻害する危険がある．亀山氏は，精密な理論的研究を展開し，非局所脱出の体系，shift/reset 機構の公理化，継続機構の階層化などに関して研究を行い，安全なメタレベル機構の開発に関して優れた成果をあげている．研究成果も一流の国際会議などで公表し，また，関連の国際会議でプログラム委員長などをするなど，大変高い研究成果が出たと評価される．

7 主な論文等:

(1) 書籍(編)

1. Yukiyoshi Kameyama, Peter J. Stuckey (editors), FLOPS 2004 Proceedings, Lecture Notes in Computer Science Volume 2998, Springer, 307 pages, April, 2004.

(2) 学術雑誌

1. Yukiyoshi Kameyama, Masahiko Sato, “Strong Normalizability of the Non-deterministic Catch/Throw Calculi”, Theoretical Computer Science 272 (1-2), pp. 223-245, 2002.
2. Masahiko Sato, Takafumi Sakurai, Yukiyoshi Kameyama, “A Simply Typed Context Calculus with First-Class Environments”, Journal of Functional and Logic Programming

2002(4), pp. 1-41, 2002.

3. Azza A. Taha, Masahiko Sato, Yukiyoshi Kameyama, “A Second Order Context Calculus”, **コンピュータソフトウェア** 19:3, pp. 2-19, 2002.
4. **中島一、亀山幸義、「抽象化と精密化による実時間モデル検査の改善」、情報処理学会論文誌：プログラミング** Vol. 45, No. SIG12 (PRO23), pp. 11-24, 2004.

(3) **査読つき国際会議**

1. Yukiyoshi Kameyama, Masahito Hasegawa, “A Sound and Complete Axiomatization for Delimited Continuations”, Proc. Eighth ACM International Conference on Functional Programming (ICFP’03), pp. 177-188, 2003.
2. Masahiko Sato, Takafumi Sakurai, Yukiyoshi Kameyama, Atsushi Igarashi, “Calculi of Meta-Variables”, Proc. 17th International Workshop on Computer Science Logic (CSL’03), Lecture Notes in Computer Science Volume 2803, pp. 484-497, 2003.
3. Yukiyoshi Kameyama, “Axiomatizing Higher-Level Delimited Continuations”, Proc. Fourth ACM-SIGPLAN Continuation Workshop (CW’04), pp. 49-53, 2004.
4. Yukiyoshi Kameyama, “Axioms for Delimited Continuations in the CPS Hierarchy”, Proc. Annual Conference of the European Association for Computer Science Logic (CSL’04), Lecture Notes in Computer Science Volume 3210, pp. 442-457, 2004.

(4) **口頭発表**

1. Yukiyoshi Kameyama, “Dynamic Control Operators in Type Theory”, Second Asian Workshop on Programming Languages and Systems, Daejon, Korea, Dec., 2001.
2. Yukiyoshi Kameyama, “Partial Continuation and CPS-Translation”, Workshop on Foundation of Software, Hangzhou, China, Sep., 2003.
3. **亀山幸義、中島一、『ケーススタディ：モデル検査と定理証明を用いた信号制御システムの検証』、システム検証の科学技術シンポジウム、産業技術総合研究所システム検証研究ラボ、**2004 **年** 2 **月**.

研究課題別評価

1 研究課題名：スケルトン並列プログラムの最適化

2 研究者氏名：胡 振江 (HU, Zhenjiang)

3 研究の狙い：

With the increasing popularity of parallel programming environment such as PC cluster, more and more people, including those who have little knowledge about parallel architectures and parallel programming, are hoping to write parallel programs to solve their daily problems. This situation eagerly calls for models and methodologies that can assist programming parallel computers effectively and correctly.

Data parallel model turns out to be one of the most successful ones for programming massively parallel computers. To support parallel programming, this model basically consists of two parts: (1) a parallel data structure to model a uniform collection of data which can be organized in a way that each element can be manipulated in parallel; and (2) a fixed set of parallel skeletons on the parallel data structure to abstract parallel computation structures of interest, which can be used as building blocks to write parallel programs. Typically, these skeletons include element-wise arithmetic and logic operations, reductions, prescans, and data broadcasting.

This data parallel model not only provides programmers an easily understandable view of a single execution stream of a parallel program, but also makes the parallelizing process easier because of explicit parallelism of the skeletons. For instance, in high performance Fortran 90/95, the parallel data structure is array and the parallel skeleton is FORALL; in the parallel language NESL, the parallel data structure is sequence and the most important parallel skeletons on sequences are apply-to-each and scan; and in the BMF (Bird Meertens Formalisms) parallel model, the parallel data structure is typically parallel list, and the parallel skeletons are mainly map and reduce.

Despite these promising features, the application of current data parallel programming using skeletons suffers from several problems, which prevent it from being practically used. Firstly, because parallel programming relies on a set of parallel primitive skeletons for specifying parallelism, programmers often find it hard to choose proper ones and to integrate them well in order to develop efficient parallel programs to solve their problems. Secondly, the skeletal parallel programs are difficult to be optimized, and the major difficulty lies in the construction of rules meeting the skeleton-closed requirement for transformation among skeletons. Thirdly, skeletons are assumed to manipulate regular data structures. For irregular data structures like nested lists where the sizes of inner lists are much different, the parallel semantics of skeletons would lead to load unbalance which may cancel the effect of parallelism in skeletons.

This project aims to solve these problems based on the theory of Constructive Algorithmics, investigating what kinds of recursive structures are suitable for capturing parallel computation on parallel data structures, and constructing rules for manipulating such recursive structures.

4 研究成果：

Our main contribution is a novel framework supporting efficient parallel programming

using skeletons. We have designed and implemented a self-optimizing C++ parallel skeleton library, with which users, with little knowledge of parallel programming, can program a wide class of parallel algorithms without the need to be concerned with details of parallel architectures and data communications among processors. In addition, we have proposed a programming methodology for systematic development of efficient skeletal parallel programs. In the following, we shall detail three important results.

4.1 A Generic Parallel Skeleton for Parallel Programming

We have proposed a new powerful parallel skeleton, which can significantly ease skeletal parallel programming, efficiently manipulate both regular and irregular data, and systematically optimize skeletal parallel programs.

We have defined a novel parallel skeleton that cannot only efficiently describe data dependency in a computation through an accumulating parameter, but also exhibit nice algebraic properties for manipulation. It can be considered as a higher order list homomorphism, which abstracts a computation requiring more than one pass and provides a better recursive interface for parallel programming.

We have given a single but general fusion rule, based on which we construct a framework for systematically optimizing skeletal parallel programs. Inspired by the success of the shortcut deforestation for optimizing sequential functional programs in compilers, we gave a specific shortcut law for fusing composition of skeletal parallel programs, but paying much more attention to guaranteeing the skeleton-closed property. Our approach using a single rule is in sharp contrast to the existing approaches based on a huge set of transformation rules developed in a rather ad-hoc way. Furthermore, we proposed a flattening rule to deal with both regular and irregular nested data structures efficiently. Compared to the work by Blelloch where the so-called segmented scan is proposed to deal with irregular data, our rule is more general and powerful, and can be used to systematically handle a wider class of skeletal parallel programs.

4.2 A Self-Optimizing Skeleton Library

We have designed and implemented a parallel skeleton library that guarantees efficient combinations of skeletons. Our idea was to associate each skeleton not only with an efficient parallel implementation but also with an interface for efficient combination with other skeletons. This interface contains information about how the skeleton consumes and produces its data. This idea is not new in the functional community, where we have seen the success of shortcut deforestation (fusion) in optimizing sequential programs in compilers. However, as far as we know, we are the first to introduce this idea to the design of parallel skeleton libraries.

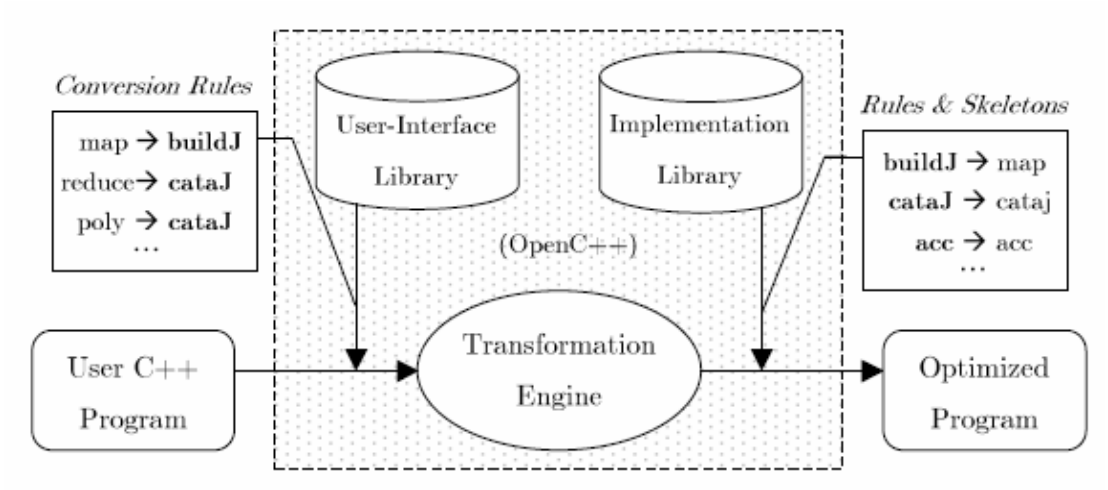


Figure 1: A Fusion-Embedded Skeleton Library

Our skeleton library for skeletal parallel programming in C++ (Figure 1) has the following new features. First, we need only a single optimization rule, and this rule can be applied to skeletal parallel programs in any way while guaranteeing the same result and termination. Second, our library allows new skeletons to be introduced without any change to the existing optimization framework, and ensures their efficient combination with existing skeletons in the library. This remedies the situation where transformation rules must take combinations of the skeletons with existing ones into account. Third, our library is simple to use. From the programmers' point of view, as our library does not introduce any new syntax, a programmer who knows C++ should have no trouble in using it. We construct a structured interface for the skeletons as well as apply a general optimization rule concisely and quickly with the help of the reflection mechanism provided with Open C++. We found it very useful to use meta programming in implementing the transformation, which, we believe, is worth greater recognition in the skeleton community.

4．3 From List Skeletons to Tree Skeletons

Trees are useful data types, widely used for representing hierarchical structures such as mathematical expressions or structured documents like XML. Due to irregularity of tree structures, developing efficient parallel programs on trees is much more difficult than developing efficient parallel programs on lists. Unlike linear structure of lists, trees do not have a linear structure, and hence the recursive functions over trees are not linear either (in the sense that there are more than one recursive calls in the definition body). It is this nonlinearity that makes the parallel programming on trees complex.

We have given a systematic method for parallel programming using tree skeletons, by proposing two important transformations, the tree diffusion transformation and the tree context preservation transformation. The tree diffusion transformation is an extension of the list version. It shows how to decompose natural recursive programs into equivalent parallel ones in terms of tree skeletons. The tree context preservation transformation is an extension of the list version too. It shows how to derive associative operators that are required when using tree skeletons.

In addition, to show the usefulness of these theorems, we have demonstrated the first formal derivation of an efficient parallel program for solving the party planning problem

using tree skeletons.

5 自己評価:

The three and half years are short, but fruitful and enjoyable. I am pleased that the results we have achieved are really encouraging; our new parallel skeleton library, together with a set of nontrivial applications, has proved that the theory of constructive algorithmics, i.e., the program calculation theory, is practically useful for better solving problems in parallel programming, which has not been well recognized so far.

Our main contribution is a novel framework supporting efficient parallel programming using skeletons. First, we have designed and implemented a C++ parallel skeleton library, with which users can code their parallel algorithms as if they used other library functions without need to concern about details of parallel architectures and data communications among processors. Second, we have proposed a programming methodology, which is useful for systematically developing efficient skeletal parallel programs from initial straightforward specifications. Third, we have implemented a programming environment, with which one can develop and run skeletal parallel programs efficiently. This framework can greatly help easing parallel programming using skeletons, efficiently manipulating both regular and irregular data, and systematically optimizing skeletal parallel programs.

It is our hope that the results of this work would lead to a future standard framework for skeletal parallel programming that can greatly help easing parallel programming using skeletons, efficiently manipulating both regular and irregular data, and systematically optimizing skeletal parallel programs. In practice, we would expect the first performance-guaranteed parallel skeleton library in C++, which is really useful for solving practical problems. We wish it to be a convincing witness of usefulness of constructive approach in parallel programming. In theory, we would expect a unified algebraic（constructive）model for structuring data, control, and communication skeletons in development of efficient parallel programs.

6 研究総括の見解:

胡氏の研究は，データに内在する並列性を並列データ構造として捉え，それを処理するための並列処理スケルトンを少数用意し，それらの組み合わせによって並列処理プログラムを作成しようとするものである．このような方式で並列プログラムを構成するための理論体系を整備し，それにもとづいてスケルトンライブラリを構成し，現実の問題に対して並列プログラムが効率的に構成できることを具体的に示した．他の類似研究に比べると，胡氏の研究結果は扱える並列データ構造の種類が多く，また，スケルトンとその組み合わせ機構が数学的により整理されており，系統的な展開が可能なことである．国際的にも高い評価を得ており，優れた研究成果をあげたと評価される．

7 主な論文等:

1. Kiminori Matsuzaki, Zhenjiang Hu, Kazuhiko Kakehi, Masato Takeichi, Systematic Derivation of Tree Contraction Algorithms, to apper in Parallel Processing Letters, 2005.
2. Hideya Iwasaki, Zhenjiang Hu, A New Parallel Skeleton for General Accumulative Computations, International Journal of Parallel Programming, 32 (5): 389-414, October 2004.
3. Tetsuo Yokoyama, Zhenjiang Hu, Masato Takeichi, Deterministic Second-order

Patterns, Information Processing Letters, Vol. 89, No.6, Elsevier, 2004. pp. 309-314.
4. **横山 哲郎**, **胡 振江**, **武市 正人**, **決定論的 2 階パターンとプログラム変換への応用**, **コンピュータソフトウエア**, 21 (5): 71-76, 2004.
5. Dana Na Xu, Siau-Cheng Khoo, Zhenjiang Hu, PType System : A Featherweight Parallelizability Detector, Second ASIAN Symposium on Programming Languages and Systems (APLAS 2004), Taipei, Taiwan, November 4-6, 2004. LNCS 3302, Springer Verlag. pp.197-212.
6. Kiminori Matsuzaki, Kazuhiko Kakehi, Hideya Iwasaki, Zhenjiang Hu, Yoshiki Akashi, A Fusion-Embedded Skeleton Library, International Conference on Parallel and Distributed Computing (EuroPar 2004), Pisa, Italy, 31st August - 3rd September, 2004. LNCS 3149, Spinger Verlag. pp.644-653 .
7. Kiminori Matsuzaki, Zhenjiang Hu, Kazuhiko Kakehi, Masato Takeichi, Systematic Derivation of Tree Contraction Algorithms, 4th International Workshop on Constructive Methods for Parallel Programming (CMPP 2004), Stirling, Scotland, UK, 14 July, 2004. pp. 109-124.
8. Kazuhiko Kakehi, Zhenjiang Hu, Masato Takeichi, List Homomorphism with Accumulation, 4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03), Lubeck, Germany. October 16-18, 2003. pp. 250-259.
9. Mizuhito Ogawa, Zhenjiang Hu, Isao Sasano, Iterative-free Program Analysis, 8th ACM SIGPLAN International Conference on Functional Programming, (ICFP 2003), Uppsala, Sweden: 25-29 August 2003. ACM Press. pp.111-123.
10. Tetsuo Yokoyama, Zhenjiang Hu, Masato Takeichi, Deterministic Second-order Patterns and Its Application to Program Transformation, International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR 2003) Uppsala, Sweden: 25-27 August 2003. pp. 165-178. Revised version appears in LNCS 3018, 2004. Springer Verlag. pp. 128-142
11. Kiminori Matsuzaki, Zhenjiang Hu, Masato Takeichi, Parallelization with Tree Skeletons, International Conference on Parallel and Distributed Computing (Euro-Par 2003), Klagenfurt, Austria, 26th - 29th August 2003. Lecture Notes in Computer Science 2790, Springer Verlag. pp.789-798. An extended version appears as Technical Report METR 2003-21, Department of Mathematical Informatics, University of Tokyo, 2003.
12. Zhenjiang Hu, Tomonari Takahashi, Hideya Iwasaki, Masato Takeichi, Segmented Diffusion Theorem (invited paper), 2002 IEEE International Conference on Systems, Man and Cybernetics (SMC 02), Hammamet, Tunisia, October 6-9, 2002. IEEE Press.
13. Wei-Ngan Chin, Zhenjiang Hu, Towards a Modular Program Derivation via Fusion and Tupling, The First ACM SIGPLAN Conference on Generators and Components (GCSE/SAIG 2002), Pittsburgh, PA, USA, October 6-8, 2002. Affiliated with (PLI 2002). Lecture Notes in Computer Science 2487, Springer Verlag. pp.140-155.
14. Zhenjiang Hu, Hideya Iwasaki, Masato Takeichi, An Accumulative Parallel Skeleton for All , 11th European Symposium on Programming (ESOP 2002), Grenoble, France, April 8 - 10, 2002. Lecture Notes in Computer Science 2305, Springer Verlag. pp.83-97.
15. **横山 哲郎**, **篠埜 功**, **胡 振江**, **武市 正人**, **変換戦略の記述に基づくプログラムの自**

動生成システムの実装, **情報処理学会論文誌**, Vol. 43, No. SIG 3(PRO 14), pp. 62-77, March 2002.

研究課題別評価

1　研究課題名：実時間マルチタスク処理を支援するプロセッサアーキテクチャ

2　研究者氏名：田中清史

3　研究の狙い：

近年LSIの集積度が上がり、従来のプロセッサよりも大規模な回路が1チップに搭載可能となってきた。このことから、オンチップマルチプロセッサ化、システムオンチップ化が次世代のキーテクノロジーとして注目されている。また、シングルプロセッサの機構としても、特別な付加ハードウェアを必要とする各種機構が現実的に可能になってきた。しかし、従来の高速化を達成するための研究では、シングルタスクの高速化に重点が置かれてきた。本研究では、実時間（リアルタイム）汎用マルチタスク環境におけるプロセッサの有効実行性能を向上させるための機構をターゲットとする。

計算機はマルチタスク環境で使用されているが、これは各実行タスクが時分割でプロセッサを使用することにより実現されている。1つのタスクが一定時間プロセッサを使用した後、あるいは外部からの割り込みが発生した時、プロセッサコンテキストが入れ換えられ、他のタスクによるプロセッサ使用に移行する。現状のプロセッサにおいてプロセッサコンテキストを入れ換える際、入れ換え前の汎用レジスタおよびプログラムカウンタ、状態レジスタなどの専用レジスタの内容をメモリに退避し、入れ換え後のタスクのために汎用レジスタおよび専用レジスタの内容をメモリから読み出してセットするといった、メモリベースでの手続きが実行される。このような方式ではキャッシュミスなどの要因により、コンテキスト切り替えに要する時間が各タスクの有効実行時間に対して無視できない大きさになる。更には、リアルタイム性が要求されるタスクの場合には、コンテキスト切り替えの時間を含んだ最悪実行時間によってそのリアルタイム性を保障するため、このコンテキスト切り替えの処理時間を削減することによりタスク実行に余裕を持たせることが可能となる。また、組み込みシステムをターゲットとした場合、敏速な割り込み応答が要求されるため、割り込み処理への高速なコンテキスト切り替えが重要となる。

本研究では、従来のマルチスレッドアーキテクチャを拡張することによりタスク切り替えの際のレジスタ値やアドレス空間などのプロセッサコンテキストを入れ換える時間をゼロにするアーキテクチャ、およびキャッシュメモリを再構成することによりメモリアクセス時間を削減する方法を提案し、実際にプロセッサの回路を設計し、LSIを開発・評価してきた。

4　研究成果：

近年、インテルのPentium 4、IBMのPOWER5など、マルチスレッド型のプロセッサが出現してきた。マルチスレッドアーキテクチャは複数のスレッド（タスク）の実行コンテキストを内蔵し、それらを瞬時に切り替えて実行することにより、メモリアクセスレイテンシの隠蔽および命令実行ユニットの稼働率を向上させる。本研究で提案するプロセッサは基本的にblock-multithreading方式を利用し、スレッドの優先度を考慮した切り替えを行う。

（1）プロセッサコンテキストバッファ

大規模データを対象とするアプリケーションの普及により、メモリアクセスの待ち時間はプログラムの全実行時間の75%にもおよぶとされている。このことはスレッド実行が頻繁にキャッシュミスを発生させることを意味する。したがって、数個のスレッドコンテキストを内蔵していても、メモリアクセスの全ての待ち時間を隠蔽することは困難である。一つのスレッドコンテキストを格納する領域を本研究ではプロセッサコンテキストセット（PCS）と呼ぶが、これを多数用意することはハードウェアのサイズおよびコストの面から困難であり、また実行ユニットに接続するための選択器（セ

レクタ) の規模が大きくなるため動作速度が低下する要因となる。そこで本研究では、小規模のハードウェアで仮想的に切り替え対象のスレッドコンテキスト数を増加させる“プロセッサコンテキストバッファ (PCB) ”を提案した。

(2) 高速割り込み応答機構

割り込みが発生した際には、一般に割り込みハンドラは実行中のタスクのコンテキストをメモリに保存し、割り込み処理終了時に保存したコンテキストをメモリから読み出すことによって元のタスク実行に復帰する。このメモリベースの保存・復帰はキャッシュミスの原因となり、特に割り込みに対する応答時間が重要であるリアルタイムシステムでは問題となる。本研究ではマルチスレッドアーキテクチャを応用し、割り込み専用 PCS を用意することにより上記のオーバヘッドを削除する。すなわち、あるタスクの実行中に割り込み要求が発生した場合、そのタスクの PCS の内容はそのままにしておき、割り込み専用 PCS に切り替えて割り込み処理を行う。同様に割り込み処理終了時には、単に元の PCS に切り替えることによって元のタスク実行を再開する。

(3) 再構成可能キャッシュメモリ

従来のキャッシュメモリは、アプリケーションの参照するデータの時間的・空間的局所性にその効率が依存していた。今日の大規模数値計算やデータベース、メディアプロセッシングなどではデータ参照に局所性が無いものが多く、高速なキャッシュメモリの有効利用が困難になっている。またマルチスレッド型プロセッサにおいてスレッド同士でキャッシュを共有する場合、スレッド間でキャッシュ内の使用領域の競合が頻発し、キャッシュミスが増加、すなわちキャッシュ効率が低下する傾向がある。そこで本研究では、キャッシュメモリ管理法を動的に再構成することにより、データ参照効率を向上させることを提案した。提案した動的再構成により、優先度ベースパーティショニングおよびキャッシュメモリを利用した FIFO バッファの使用が可能となる。

(4) プロトタイププロセッサ：PRESTOR-1

本研究では、提案する各種機構を持つプロセッサを実際に設計し、LSI として研究開発した。本プロセッサを PRESTOR-1 (PRESTO RISC-1) と呼ぶ。開発した LSI チップの写真を図1に示す。

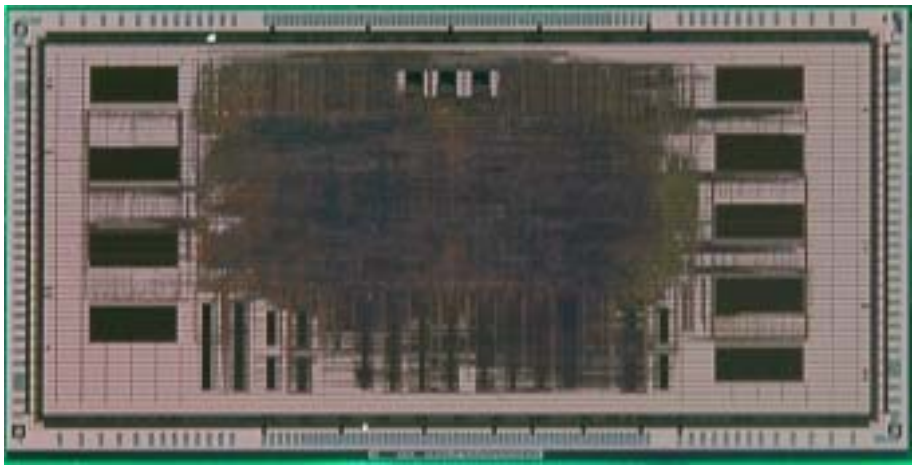


図1 PRESTOR-1 チップ写真

(5) 基本性能評価

ハードウェア記述言語 VHDL で記述された PRESTOR-1 の RTL シミュレーションにより各種機能の評価を行った。高速割り込み応答機構の評価において、割り込み発生／処理終了で毎回コンテキストの入れ替えを行う従来手法に対して、一回の割り込みあたり 230 サイクルを削減可能であることがわかった。これは主に、コンテキストの退避／復帰のためのストア／ロード命令実行

のオーバヘッドとキャッシュミス削減の効果である。また、再構成可能キャッシュメモリが提供するFIFO バッファの使用に関する RTL シミュレーションにおいて、FIFO バッファへのプリフェッチ効果により、最大 83%のプログラム実行時間の削減が確認された。

5 自己評価:

本研究では、マルチタスク環境でリアルタイムアプリケーションを効率良く実行するプロセッサアーキテクチャの確立を目指した。このため、プロセッサコンテキストバッファ、高速割り込み応答機構および再構成キャッシュメモリなどの各種新機構を提案し、その有効性の評価を行った。また、最大の目標である、これらの新機構を搭載したプロセッサ LSI の開発・製造・評価を実施した。

一方、研究開発したプロセッサ LSI 上で基本性能評価などは実施できたが、時間の関係で実用規模のアプリケーションを用いた評価が残った。また、評価の結果、キャッシュメモリの動的再構成に関して、実行プログラムのメモリアクセスの振る舞いを考慮した更なる効率化を行う余地があることが判明した。更に、提案した高速化機構を最大限利用するためのプログラミングインタフェースを確立する必要がある。

今後の課題としては、提案するプロセッサアーキテクチャを用いた並列・分散環境の構築および、リアルタイム OS との連携により、リアルタイム性を向上させるシステムの実現が挙げられる。

6 研究総括の見解:

田中氏の研究は，マルチタスク環境でリアルタイムアプリケーションを効率よく実行するプロセッサアーキテクチャに関して，プロセッサコンテキストバッファ，高速割り込み応答機構，再構成キャッシュメモリなどの機構を提案し，その有効性の評価を行ったものである．具体的には，PRESTOR-1 と呼ばれるプロセッサーチップを開発した．基本性能に関しては，RTL シミュレーションにより評価を行い，予定された性能が得られることを確認している．実用的には，効率化に関しては改善点があることが判明しているが，限られた研究期間のなかで，構想から実装までを行い，一定の成果をあげたことは大いに評価できる．個人研究のなかでかなりの規模のものを構築した力量は評価に値する．

7 主な論文等:

1. Khairuddin Khalid and Kiyofumi Tanaka, Implementation of FIFO Buffer Using Cache Memory, **情報処理学会 計算機アーキテクチャ研究会**, Vol.2002, No.150, pp.83－88, 2002 **年**.
2. Kiyofumi Tanaka, Fast Context Switching by Hierarchical Task Allocation and Reconfigurable Caches, Proc. of International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems 2003, IEEE Computer Society Press, pp.20－29, 2003.
3. Khairuddin Khalid and KiyofumiTanaka, Evaluation of Cache Memory as FIFO Buffer, **情報処理学会 計算機アーキテクチャ研究会**, Vol.2003, No.27, pp.91－96, 2003 **年**.
4. Kiyofumi Tanaka and Tomoharu Fukawa, Highly Functional Memory Architecture for Large-Scale Data Applications, Proc. of International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems 2004, IEEE Computer Society Press, pp.109－118, 2004.
5. **今井俊晴**, **田中清史**, **高速フィルタリングを支援する高機能メモリコントローラ**, **情報処理学会 計算機アーキテクチャ研究会**, Vol.2004, No.123, pp.89－94, 2004 **年**.
6. Kiyofumi Tanaka, PRESTOR-1: A Processor Extending Multithreaded Architecture, International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, 2005.

研究課題別評価

1 研究課題名：高信頼性 Web サービス

2 研究者氏名：中島 震

3 研究の狙い：

インターネットの発展と共に Web サービスの技術が新しい業務サービスの基盤として登場した。当初は、ひとつの機能を提供する単独 Web サービスが主流であったが、最近では、複数の Web サービスを統合して新しいサービスを提供する複合サービスの技術に注目が集まっている。複合化によって多数のビジネスパートナーと連携したサービス提供が可能になる。

Web サービスの世界では、異なるベンダが開発したソフトウェア基盤がネットワーク上で情報交換する。そのため、W3C (World Wide Web Consortium) や OASIS (Organization for the Advancement of Structured Information Standards) といった中立組織を中心とする技術標準化の活動が活発になっている。Web サービス複合化の技術も、ベンダ提案段階から OASIS での標準化活動に進み、WS-BPEL (Web Service Business Process Execution Language) と呼ぶ一種の分散協調システム記述言語が提案される段階に至った。複合サービスを WS-BPEL のプログラムとして表現する。

複合サービスを表現する連携のことをオーケストレーションと呼ぶことがある。多数の演奏者である Web サービスの全体調整を行うことが特徴であり分散協調システムとみなせることを示す簡潔な用語である。一般に、分散協調システムは動作振る舞いが複雑なため、処理が進行しないデッドロックによるシステム停止などの不具合を除去することが難しい。ある複合サービスが実行中に停止すると、関連する Web サービス提供者にも影響を与える。これを安全性の問題と呼ぶ。WS-BPEL の記述を対象として安全性の観点から不具合がないことを確認する必要がある。

次にセキュリティに関する問題を考える。企業が活動の効率化を目的として、外部のコンサルタント会社等に、従来は社内で行っていた業務を外部委託することが増えている。コンサルタント会社は顧客会社の情報を入手してはじめて有用なアドバイスができる。

Web サービス技術を用いたコンサルタント業務は便利な反面、問題も大きい。顧客企業は必要な情報を Web サービスとして提供し、コンサルタント会社はノウハウを組み込んだ業務ロジックを WS-BPEL のプログラムとして表現する。顧客企業の情報や市場動向調査レポートなどの一般情報を全て Web サービスの技術を用いて収集することが可能になる。

顧客会社からみると、コンサルタントに必要とはいえ、社外秘情報が外部に出るため、取り扱いに慎重になる。コンサルタント業務を表現する WS-BPEL のプログラムが何らかの誤りによって、大切な情報を流出することがないという証拠を、コンサルタント会社に求める。WS-BPEL 記述を対象としてセキュリティの観点からの不具合がないことを確認しなければならない。

本研究課題では、WS-BPEL のプログラムを解析し、安全性ならびにセキュリティという上記の 2 つの観点からの不具合がないことを、インターネット上で実行する前に確認する技術を検討した。

4 研究成果：

4.1 安全性の解析

当初の研究目的は、WSFL (Web Service Flow Language) のプログラムを対象としてデッドロック等の不具合があるかないかを解析する技術を確立することであった。本研究課題の開始時点では、WSFL と呼ぶ Web サービス連携記述言語が提案された段階で、まだ WS-BPEL は、その基になった BPEL4WS (Business Process Execution Language for Web Service) も含めて提案されていなかった。

WSFL の言語仕様を調査すると、業務の作業連鎖を記述するワークフローの考え方を Web サ

ービスに転用したものであることが判明した。WSFL は外部 Web サービス起動をノードとし、この起動順序を指定する制御フローならびに外部 Web サービスと交換するデータのやりとりをデータフローで表現する一種のネットワーク指向並行記述言語である。

そこで、WSFL プログラムの構成要素を、並行システム記述言語である Promela を用いて表現する方法を考案した。これによって、Promela を入力とするモデル検査ツール SPIN を用いて、デッドロックの有無などを自動解析する方法を実現できる。モデル検査の方法で WSFL プログラムの振舞いチェッカが実現できることを示した。

また、いくつかの実験を行った結果、WSFL の言語仕様に不備があり、デッドロックを除去することが難しいことを指摘した。デッドロックを除去する方法を提案し、提案方式が問題を解決していることを具体的な実験を行うことで確認した。

4.2 セキュリティの解析

Web サービスは開放型ネットワークであるインターネットを土台としているため、標準化でもセキュリティに関する議論が活発である。Web サービスの基本通信メッセージである SOAP を暗号化し通信路で情報が漏えいしないことを保障する技術として WS-Security が、また、指定 Web サービスへのアクセスを許可するか否かをあらわすアクセス制御ポリシー表現、ならびにアクセス制御実現のためのプロトコルなどが WS-Authorization として提案されている。

ところが、暗号技術とアクセス制御の技術だけでは、情報漏えいの問題を扱うことができない。従来から情報システムセキュリティの分野で、ラティスに基づく情報フロー制御の方法が提案されていた。本研究課題では、WS-BPEL を対象として、情報フロー制御の方法を用いて情報漏えいの有無を解析する方法の適用可能性を検討した。これは、現在の Web サービス技術体系では未だ取り扱っていない問題である。冒頭のコンサルタント会社のような先進的な Web サービスを実現する上で必須の技術になると考え、先行して研究を進めた。

ラティスに基づく情報フロー制御の方法では、セキュリティからみた情報の重要さを導入し、重要さを表すセキュリティラベルに順序関係を与える。この順序関係を dominates 関係と呼び、これがラティスとなることが方式名の由来である。代表的な例では、「秘密文書」、「公開文書」などを考えることができ、「秘密文書」は「公開文書」よりも dominates 関係が大きい、あるいは支配的であるという。次に、利用主体と資源の双方にセキュリティラベルを与え、dominates 関係を満たす方向だけに情報の流れを許可する。機密関係の高い方向にだけ情報の流れを許可するため Flow-Up と呼ぶ。

Flow-Up だけでは、高いレベルの利用主体が読んだデータは2度と読み出すことができないという問題がある。アクセスの都度、dominates 関係の支配的な方向に情報が動き、最終的に、誰も読み出すことができない「ブラックホール」のような超高機密レベルに落ち込む。この問題を避けるために、クラス低下 (Declassification) と呼ぶ方法を導入する。

今、利用主体 P1 が資源 T1 から読み出したデータを資源 T3 に書き込む場合を扱い、次の関係が満たされているとする。ここで、L (P1) は P1 が持つセキュリティラベルを示す。

L (P1) dominates L (T1)
L (P1) dominates L (T3)
L (T3) dominates L (T1)

3つめの dominate 関係を考えるかぎり資源 T1 から T3 へのフローは許可される状況にある。しかし、利用主体 P1 がデータをアクセスするために、P1 が T1 から得たデータのセキュリティラベルが表面上、P1 と同一になる。そのため、2番目の関係から P1 から T3 へのフローが禁止され、T1 から T3 へのフローを禁止すべきと結論してしまう。条件が厳しすぎるという問題である。

この問題を解決するために、クラス低下により、利用主体 P1 のラベルを一時的に下げる。次のような関係を満たす一時的なラベル DCL が見つかればよい。

L (P1) dominates L (DCL)

L (DCL)　　dominates　L (T1)
L (T3)　　dominates　L (DCL)

1番目は DCL が P1 よりも小さいこと、2番目は DCL によって T1 からのフローが許可されること、3番目は DCL から T3 へのフローが許可されることを示す。この例では、L (T1) に一致するように L (DCL) を選べばよい。一般的には数多くの dominates 関係を満たすように、DCL の値をうまく選ぶ必要がある。

ラティスに基づく方法は、上の例を一般化することで次のように整理することができる。ある実行経路上に現れる全てのアクセスごとに dominates 関係を集め、すべての dominates 関係を満たすような一時ラベル DCL が存在するかを調べる。DCL の値を具体的に求めることができれば、情報フローに誤りがなく、したがって、情報漏えいがないといえる。一方、DCL の値がなければ情報漏えいの問題があると結論することができる。

上に述べたように、情報漏えいの問題は、流れるデータに付随するセキュリティラベルの値がラティスを形成し、その値を比較することで集めてきた dominates 関係が解を持つか否かを判定することである。この処理は2つの問題に分割することができる。実行経路を網羅的に生成し当該経路上の dominates 関係を集めてくる1番目の処理、さらに、集めてきた dominates 関係の制約を解く2番目の処理である。

本研究課題で提案する手法は、1番目の処理に 4.1 節で検討したモデル検査の方法を用いる。すなわち、セキュリティラベルを付加できるように拡張した WS-BPEL を対象として、4.1 節で用いたモデル検査の方法を適用する。2番目の制約計算処理は大小関係の比較演算で実現できることがわかった。

以上、本研究によって、Web サービス連携の記述を対象とする情報漏えいの解析がはじめて可能となる。

5 自己評価:

本研究課題では、並行システムのモデル検査やラティスに基づく情報フロー制御といった科学的な裏づけのある研究成果が、ビジネス主導の技術である Web サービスの世界でも重要な役割を果たすことを具体的に実感できた。

研究開始当初は、WSFL のプログラムを対象にして検討を進めた。この研究成果は Web サービス連携記述の解析にモデル検査の方法を適用した先駆的な研究として一定の評価を得て、諸外国の研究者から論文を引用されている。一方、WSFL は同時期に提案された XLANG と統合され、BPEL4WS として主要ベンダが共同提案して、現在の WS-BPEL になった。

本研究課題のように、ビジネスが密接に絡むソフトウェア技術を研究対象とする場合、実用的な貢献が明確である反面、急なビジネス環境の変化によって、計画の変更を余儀なくされた。幸い、WS-BPEL は WSFL の特徴であるネットワーク指向並行処理記述を部分言語として持つため、WSFL の方式を WS-BPEL の安全性解析を行う方法に適用することができ、研究が無駄にならずにすんだ。

本当に実用的な解析ツールを開発するためには、解決すべき課題が多く残っている。また、本研究課題の中で、ラティスに関する制約処理をモデル検査と統合する必要があり、これを一般化することで新しい研究課題をみつけた。今後、実用的なツールの実現ならびに、新たな課題を解決するための基礎研究を行いたいと考えている。

6 研究総括の見解:

中島氏の研究は，インターネット上のウェブサービスの安全な構成法に関するものであり，研究期間中2つの研究成果を出した．複数のウェブサービスを組み合わせて複合サービスを構成する場合には，全体として調和して動作し，デッドロックなどを起こさないことが必要である．中島氏はウェブサービスに関する国際標準言語WSFLの仕様にデッドロックに関する問題点があることを，

モデル検査技術によって明らかにした．この結果は国際的に高い評価を得た．中島氏の2番目の研究成果は，ウェブサービスにおけるセキュリティに関するものであり，数学的束理論とモデル検査を用いて，安全にデータを移動させる方法を研究した．いずれの研究も，現実の問題を最新の情報科学の手法とツールを用いて解決したもので大変優れた研究成果であると評価される．

7 主な論文等：

(1) 論文誌

1. 中島 震，玉井 哲雄 ：EJBコンポーネントアーキテクチャのSPINによる振舞い解析，コンピュータ・ソフトウェア，Vol.19, No.2, pp.2-18 (2002年3月)．
2. 中島 震：Web サービスフロー記述のモデル検査検証，情報処理学会論文誌，Vol.44, No.3, pp.942-952 (2003年3月).
3. 中島 震：コンポーネントフレームワーク振舞い解析への多値遷移システムの応用，コンピュータ・ソフトウェア，Vol.21, No.2, pp.32-36 (2004年3月).

(2) 国際会議

1. S. Nakajima : Verification of Web Services Flows with Model-Checking Techniques, International Symposium on Cyber World (CW 2002), pp. 378-385 (2002年11月).
2. S. Nakajima : Behavioural Analysis of Component Framework with Multi-Valued Transition Systems, Asia-Pacific Software Engineering Conference (APSEC 2002), pp.217-226 (2002年12月).
3. S. Nakajima : Model-Checking of Safety and Security Aspects in Web Service Flows, International Conference on Web Engineering (ICWE 2004), pp. 488-501 (2004年7月).

(3) 依頼原稿

1. 中島 震：書評－G.J.Holzmann著 The SPIN Model Checker，コンピュータ・ソフトウェア，Vol.21, No.2, pp.61-69 (2004年3月).
2. 中島 震：組込みソフトウェアへのモデル検査の応用，情報処理，Vol.45, No.7, pp.690-693 (2004年7月)．

(4) 表彰

1. 2003年度日本ソフトウェア科学会論文賞受賞 (2004年6月)．

研究課題別評価

1　研究課題名：ユビキタス環境を支えるプログラミング言語システム

2　研究者氏名：橋本　政朋

3　研究の狙い：

近年、ユビキタスコンピューティング（Ubiquitous Computing）という概念が注目されるようになってきている。ユビキタスコンピューティングとは、1980 年代後半に Xerox の Palo Alto 研究所で提唱された概念である。生活空間の至る所に埋め込まれたコンピュータやセンサなどが連携することで、利用者に煩雑な操作を一切強いることなく、適切な情報サービスを時と場所を選ばず提供することを目的としている。この 10 年以上も前に生み出された概念は、ここ数年のコンピュータ小型化技術、ネットワーク技術やセンサ技術などの著しい発展により急速に現実味を帯びてきている。この研究ではユビキタスコンピューティングを本格的に実現・普及させてゆく上で鍵となると考えられる技術のうち、ソフトウェアの無停止運用技術に注目し、さらに停止要因の中でも特にソフトウェアのアップデートを想定する。アップデートはバグの修正や機能拡張などのため今後も必須であり続け、特にセキュリティのためにも益々頻発化すると考えられる。このアップデート作業は通常、対象ソフトウェアの再起動を余儀なくするため、そこかしこに存在する数多のコンピュータが連携して情報サービスを行うようなユビキタス環境では、そのような再起動は無視できないサービス低下を招くと予想される。しかし、一般にプログラムの実行を止めずにアップデートを行うことは不可能であるため、アップデートに対し制約を課すなどの特化を行わなくてはサービス終了なくアップデートを行うことはできない。今までに幾つかの提案はあったが、残念ながら様々な問題がありユビキタスコンピューティング環境への適用には難があった。この研究では、ユビキタス環境への適用を考慮し、サービスの終了を伴わないアップデートを実現することを目指している。ユビキタス環境への適用のために、計算資源に対してスケーラブルで、処理の大部分が自動化可能でかつ、ヘテロ環境対応であることを要件とする。ユビキタス環境で稼動するソフトウェア、特にサーバに対してこのようなノンストップ運用を支援することで、ユビキタスコンピューティングの本格的普及の一助となることを目指している。

4　研究成果：

この研究では、スケーラブルかつ自動化可能かつヘテロ環境対応であることをユビキタス環境への適用要件と考え、それを満たす、サービス終了を伴わないアップデート方式を提案し、そのモデル化を行った。また、その方式が実際の利用に耐えうるものであることを示すため、プロトタイプシステムの実装を行い、実際のソフトウェアに対する適用実験を行った。

4．1　アップデートモデル

実行時にアップデートを行うためのシステムは今までにも幾つか提案されているが、システムが扱える変更に対する制限が厳し過ぎたり、適用に関する制限を低減させる代償としてプログラマに負担を強いたり、アップデートの適用によりプログラムの実行に異常を来す可能性があったりなど、実際の利用には難があった。特に安全性に関しては問題で、実行時のアップデートを曖昧性の無い形式的なモデルとして定義し、実行に異常を来さないことを証明する必要がある。以降では提案モデルについて説明し、従来法に対する改善点、安全性の確認法について概要を説明した後、実装について触れる。

プログラムの実行は、プログラムカウンタやメモリの状態を含んだ実行状態の変化の系列として表される。

$$P: S_0 \rightarrow S_1 \rightarrow \cdots \rightarrow S_p \rightarrow \cdots \rightarrow S_i \rightarrow \cdots$$

今、状態が S_i の時にプログラム P をアップデートすると仮定する。この時、プログラムの変更がどのようなものであれ、S_i 以前の実行系列、$S_0 \rightarrow \cdots \rightarrow S_i$ に影響を与えなければ、プログラムを変更しても実行状態 S_i からの実行の継続は安全と言える。しかし一般には S_i 以前の状態系列に何らかの影響を与えてしまう可能性がある。プログラムの変更により影響を受ける最初の実行状態が S_p だったとする。この場合は新しいプログラム P'の実行は S_p から開始すれば安全と言えるが、S_p が S_i に近いとは限らない。S_0 に近ければ再起動した方がよいであろう。従来法にはそれを避けるため実行状態 S_i を直接変換するプログラムをシステムに与えるものもあった。しかし、そのような変換プログラムの作成は一般には自動化不可能である。提案モデルではこのような場合にも対処するため、S_p からの実行系列がプログラム変更前の実行系列中の S_p と S_i との間のいずれかの実行状態 S_q と等価な実行状態へ至るならば、S_i から実行を継続することとした。

$$\begin{array}{l} P: S_0 \rightarrow S_1 \rightarrow \cdots \rightarrow S_p \rightarrow \cdots \rightarrow S_q \rightarrow \cdots \rightarrow S_i \rightarrow \cdots \\ \qquad\qquad\qquad\qquad\quad \| \qquad\quad \| \\ \qquad\qquad\qquad\quad P': S_p \rightarrow \cdots \rightarrow S_q \qquad\quad S_i \rightarrow \cdots \end{array}$$

S_p から実行系列が全く変わってしまう場合はその時点からの実行を続ける他はないが、自動化は可能である。ここで注意が必要なのは例えば上の場合、P'への移行により S_p と S_q との間でプログラムが部分的に実行されることになるため、P での実行に加えてその部分が二重に実行されることになるということである。例えば通信でメッセージを送る処理がそのような部分にあって、二重送信を避けたいと思うなら、利用者があらかじめその扱いを指定しておく必要がある。このようにモデルを設定することで、従来提案されてきた方式と比べて、より多くの場合で安全な実行時のアップデートが自動で可能となる。例えば C 言語と対象としたあるシステムでは、main 関数の変更があると再起動するしかなかったが、提案法ではそのような場合でも条件が揃えば実行時のアップデートが全自動で可能である。

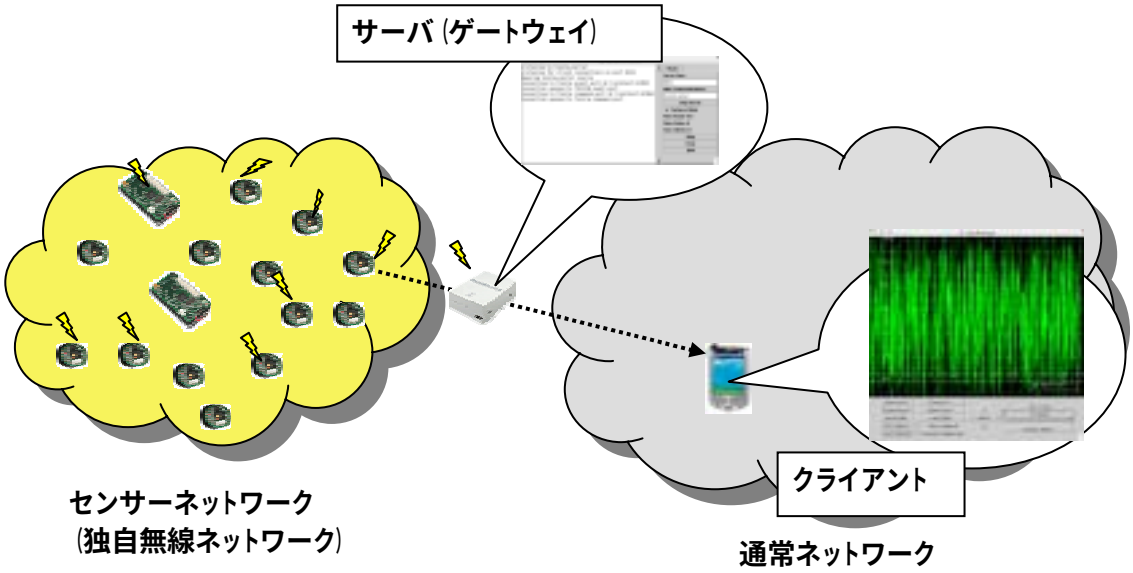
4.2 プロトタイプシステム

上述のモデルに基づき、Java 言語で記述されたプログラムを対象とする実行時アップデートシステムのプロトタイプを構築した。Java 言語システムを利用することでヘテロ環境に対応している。スケーラビリティに関しては、モデルでは全時点での実行状態を保存し比較できるようになっていたが、実装では保存する実行状態の数や部分的な再実行のステップの上限などをパラメタとして計算資源に応じて与えることになる。また、変更による影響を静的に解析する必要も出てくる。この解析に関しては従来からのフロー解析技法が利用できる。

プロトタイプシステムは、差分解析システムと実行時システムとから構成される。アプリケーションプログラムは実行時システムの管理下で実行する必要がある。差分解析システムは文字通り新旧プログラム間の差分を調べ、そこから実行時アップデートに必要な情報を抽出する。プロトタイプシステムでは、詳細に差分を調べるために構文木レベルの比較を行なう。一般には木構造同士の比較は計算量が大きいため、なるべく精度を落とさずに節を減らす工夫（対応付けが確定した枝の刈り取り、木の枝の折り畳みと展開、複数節の圧縮、部分木の先行解析）を行なっている。実行時システムはアプリケーションプログラムの実行を管理し、実行時アップデートを実現する。今回の Java 言語用システムは、JPDA（Java Platform Debugger Architecture）と BCEL（Byte Code Engineering Library）とを用いて実装されている。アプリケーションプログラムの通常実行時の速度低下はほとんど認められない。但し、クラスファイルのサイズが約2～3割程度増加する。

4．3　適用実験

ユビキタスコンピューティングでは、利用者の状況や、環境の状況を把握するために、センサシステムを用いる。その中でも近年注目されているのが、センサネットワークと呼ばれるシステムである。各種センサが搭載された非常に小型で低消費電力のコンピュータ（センサノード）が無線でアドホック通信することでセンサ値情報を交換する。センサノードは消費電力を押さえるため、特殊な無線と特殊なプロトコルを用いて通信する。また、センサ値を用いた処理は、他の処理能力の高いコンピュータで行われることが多いため、センサネットワークの通信と通常のネットワークとの仲介をするセンサゲートウェイと呼ばれるサーバがよく利用される。



図．センサネットワークゲートウェイへの適用

今回は、Mica Mote（Crossbow 社製）と呼ばれるセンサネットワークシステムのための OS である TinyOS（U.C.Berkeley）の配布物に含まれるセンサゲートウェイソフトウェア（SerialForwarder：約1万2千行）に対してプロトタイプの適用を行った。8つのバージョンに対して更新の分析を行ったところ、実質的に意味のある更新があったのは最後の2つのバージョン間のみであり、通信処理を行うメソッド内でのフラッシング処理の追加と例外処理の追加であった。これらの更新は比較的取り扱いが易しく、全自動で実行時更新が可能であることが確認できた。クラスファイルは約3割のサイズ増加であった。図では、センサネットワークからのセンサ値がゲートウェイを経由し、単純にセンサ値をグラフ出力するクライアントへ送られる様子が示されている。このように通信を行っていても実行時アップデートが可能とするために、通信状態を保存・再構成できる特殊な Socket ライブラリを用いている。

5　自己評価：

本研究では、特にユビキタスコンピューティング環境でよく利用されるコンピュータやプログラムに対して、アーキテクチャやプログラミング言語を問わず、誰でも簡単に使いこなせ、しかも安全な実行時アップデートシステムを実現することを目指した。あらゆる更新に対して全自動で実行時にアップデートを行うことは不可能であるので、動作原理をあらかじめ設定し、その上での安全性を確認するためのモデルを構築する必要があるが、このモデルに関しては極力制約をなくそうと考えたため、研究期間後半に至るまでなかなか仕様が決まらなかった。このため、全般に進行に遅がでてしまったが、結果としては扱える更新パターンは当初の想定を越えるものとすることができ

た。また、実装に関しても、プログラミング言語非依存性までは実現できなかったが、当初考えていたよりも完成度の高い Java 言語用プロトタイプシステムを実現することができた。実験構想初期では、スレッドの処理や通信の処理が複雑となるため、SerialForwarder というプログラムそのものを扱うのは困難と考え、かなり制約を加えた上での適用を想定していたが、研究期間を延長して実装を進めたことで、制約なしに適用することができた。尚、全般（特に論文）に遅れが出てしまったことは反省すべき点と考えている。

6 研究総括の見解：

橋本氏の研究はプログラムの更新による計算機の停止を回避する方式に関するものである．通常の更新では，更新に伴いプログラムは一旦停止し，更新後のプログラムが最初から実行を開始する．プログラムの停止を伴わずに更新を行う試みはこれまでにもあったが，橋本氏の方法は，従来のものに比べてスケーラビリティや自動化に優れ，適用範囲が広いものである．方式の提案とそのためのメカニズム，プロトタイプシステムの構築に関する研究を行った．今後のユビキタス技術に有用な研究であり，研究成果をなるべく早期に取りまとめ公表することを望むものである．

7 主な論文等：

(1) 論文

1. 中島秀之、橋本政朋。 日常生活のための知的情報基盤．情報処理学会誌．Vol.43，No.5．2002．
2. 橋本政朋．実行状態移送に基づく実行時更新システムの実現（仮）．（投稿予定）

(2) 口頭発表

1. 橋本政朋．プログラムの実行時更新を支援するプログラミング言語系の構築に向けて．PPL2002．

(3) ソフトウェア

1. SUS-X/Java: A Software Update System for the Java™ Programming Language（仮）（公開予定）