頼れるWebシステム構築のためのソフトウェアモデリング

（研究課題名：Webアプリケーション指向ソフトウェアモデリング）

「機能と構成」領域　結縁 祥治

要　旨

インターネットに接続された計算機上で実行される Web アプリケーションは，データベース情報アクセスをはじめとしたソフトウェアとして一般的に用いられるようになっています．Webアプリケーションは，その特性としてコンポーネントを組み合わせてその振舞いが決まります．さらに，各コンポーネントはネットワークあるいは他のコンポーネントと相互作用を及ぼしながら計算が進行します．このようなソフトウェアは，CPU が単一ないし複数の制御フローを逐次的に実行していく従来のソフトウェアとは異なったメカニズムで動作します．コンポーネントは機能ごとに分散して記述されるため，従来のソフトウェアに対する手法がそのままでは適用できません．本研究は，このような分散された記述にわたる相互作用的な計算メカニズムを表現する**通信プロセスモデル**をWebアプリケーションのソフトウェアモデルとして用いることで，Webアプリケーションの振舞いを解析する厳密な方法を与えました．さらに，信頼性を向上を目的とした振舞いモデルを構築するための研究を行いました．

1．研究のねらい

本研究では，Webアプリケーションの記述と振舞いを結びつけるソフトウェアモデリング手法の確立を目的とします．Webアプリケーションは，従来の計算機プログラムとは以下の２点で大きく異なります．

(1) 単一の言語で記述されないこと：Webアプリケーションにはさまざまなプログラム言語やスクリプトが混合します．このため，各々のプログラム言語から統合可能な形で相互作用を取り出せるようにする必要があります．

(2) 制御フローが単一プログラム上で完結しないこと：Webアプリケーションはイベント駆動のメカニズムで実行されます．このため，従来の関数的な概念に基づく入出力関係ではモデル化が不十分です．

このような異なったパラダイムのアプリケーション構築の技法に対しては，新たな概念に基づく抽象的ソフトウェアモデリングが必要です．このためのキーとなる概念として，同期イベント通信と振舞い合成に基づいて振舞いモデルを構築します．具体的手法として Milner, Parrow および Walker のπ計算に基づいて，Webアプリケーションの動作モデルを開発します．この計算モデルはWebアプリケーションの記述の構造から直接的に振舞いをモデル化することを可能にします．

数学的な手法に基づいた形式的意味論によるソフトウェアモデリングを行うことで，既存の言語で記述された Web アプリケーションに対してソフトウェアモデルを構築します．形式的意味に基

づくことは，並行的に相互作用を及ぼすコンポーネントの振舞いを定式化する上で重要な意味を持ちます．逐次的な実行に比べて，並行的な計算では実行状態を正確に把握することが難しいため，形式的意味に基づいてすべての状態をもれなく把握可能なことは基礎的な振舞いの意味づけとして有用な性質です．

このモデリングに基づいて Web アプリケーションに対してソフトウェアとしての信頼性の向上を目指します．さらに，振舞いの明確化による振る舞いの安定性の向上を図る具体的技法を提案し，実際の構築環境のプロトタイプを示します．

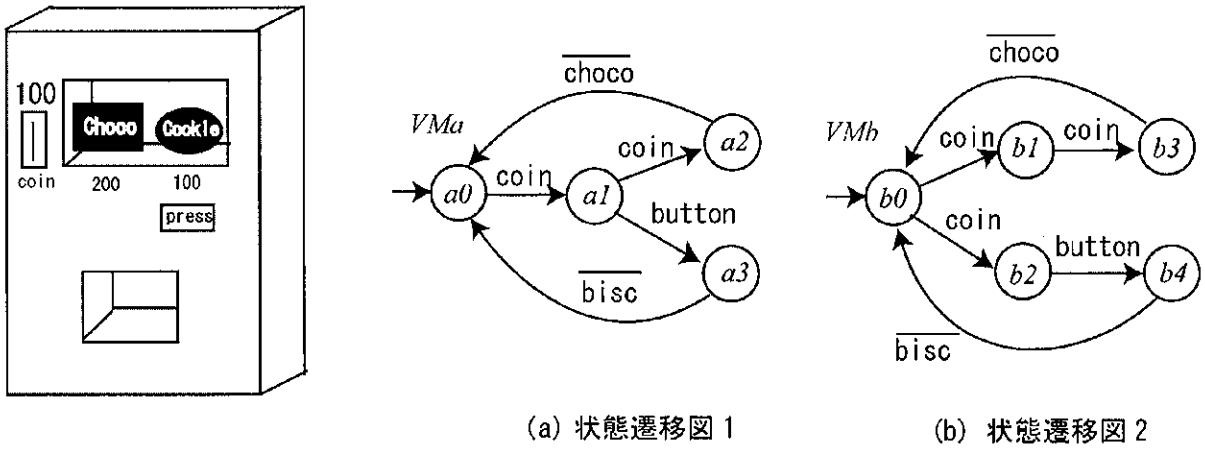
2．研究方法と成果

2．1　通信プロセスモデル

本節では，本研究の基本的な計算の枠組みとなる通信プロセスモデルについて説明します．通信プロセスモデルは，複数のプログラム(プロセス)が互いに同期通信を行いながら計算が進行する計算モデルです．その意味付けの特徴は，あるプロセスが計算の途中で行う通信のパターンによって意味づけを行う点です．従来のソフトウェアが基本的に計算が始まる前の入力値と計算が終了したときに得られる最終結果に基づく関数をもとに記述されているという点と比較すると，プログラムの状態を外部からより詳しく観測することが可能になります．例えば，図 1 のようなお菓子の自動販売機を考えます．(a)では，100 円を入れると次はボタンを押してビスケットを買うか，さらに 100 円を入れてチョコレートを買うかを選択することができます．これに対して(b)では，最初の 100 円を入れた時点で，ビスケットを買うか，さらに 100 円を入れてチョコレートを買うかということが決まります．(b)のような自動販売機は一般に故障していると思われますが，共に 100 円の入力に対してビスケットという出力，200 円の入力に対してチョコレートという出力という入出力関係では(a)と区別がつきません．これは，100 円を入れた時点での状態が単純な入出力関係では無視されてしまうためです．ここでは，お客と自動販売機が並行に動作して，お金の挿入，ボタンの押下げ，お菓子の取り出しといった相互作用を行っています．通信プロセスモデルはこのような計算途中の中間状態を一つ目の通信が終了した時点で次に可能になる通信を観測することで，上記の 2 つの自動販売機を区別します．Web アプリケーションでもリンクのクリックによる入力，ページレンダリングによる出力で同様の現象が発生します．

通信プロセスモデルとして，よく知られた代表的な枠組みとして，Milner の CCS(Calculus of Communicating Systems)と Hoare の CSP(Communicating Sequential Processes)があります．これらの体系では，プログラムの振舞いをいくつかの基本的な演算子を使って表します．例えば，CCS では，a という通信をしてから P となるプログラムを $a.P$ と表現します．$P+Q$ は P あるいは Q のいずれか一方を実行することを表します．通常の再帰表現と併せて，先の自動販売機の振舞いは以下のように表すことができます．ここで，coin は 100 円効果を挿入する動作，button はボタンを押す動作，$\overline{\mathsf{choco}}$，$\overline{\mathsf{bisc}}$ はそれぞれチョコレートとビスケットを出す動作とします．

$$VMa = \mathrm{coin}.(\mathrm{coin}.\overline{\mathrm{choco}}.VMa + \mathrm{button}.\overline{\mathrm{bisc}}.VMa)$$
$$VMb = \mathrm{coin}.\mathrm{coin}.\overline{\mathrm{choco}}.VMb + \mathrm{coin}.\mathrm{button}.\overline{\mathrm{bisc}}.VMb$$



(a) 状態遷移図 1　　(b) 状態遷移図 2

図 1．簡単な自動販売機

VMa と *VMb* は，途中の通信状態をすべて比較する双模倣等価性(bisimulation equivalence)によって区別されます．*VMa* と *VMb* の各状態を最終状態とした場合，オートマトンの基本意味である言語において等価ですが，双模倣等価性では各状態において次にどの通信が可能であるのかということまで一致しなければ等価とはなりません．このため，*VMa* における状態 *a1* に対応する状態,すなわち，次に button と coin の両方が可能な状態が *VMb* にありません．このように双模倣等価性に基づいて通信の可能性を観測することでより細かく振舞いを分類することができます．さらに，通

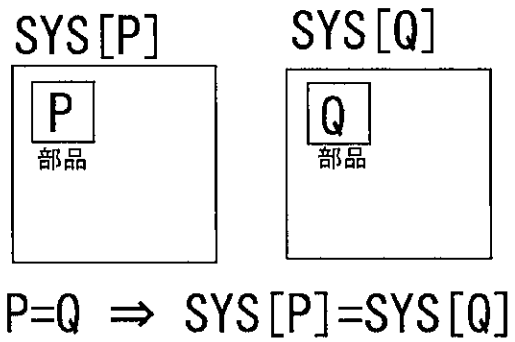


図 2 ：合同性に基づくモジュラリティ

信の可能性や観測する通信を制限することによって，目的に応じた等価性が定義されます．

通信プロセスの等価性は，通信プロセスを記述する演算子に対して合同性を持つように定義されます[1]．合同性によって，プログラムモジュール P と Q が等価である場合には，そのモジュールを

[1] このような構造は**プロセス代数**とも呼ばれます．

システムのどのような部分に埋め込んでも等価であることが保証されます．このことは，コンポーネントがプログラムの部品として組み込んだ場合に全体が正常に機能するという基本的な性質に対応します．(図2)

本研究では，Webのコンポーネントを通信プロセスで表し，その振舞いと記述の関係を通信プロセスモデルの構文と振舞い意味論でモデル化することによって振舞いを解析する手法を得ることを目標とします．ここでは，通信プロセスモデルの体系として，上にあげたCCSを通信の際にポート自身も通信で受け渡すことができるようにしたπ計算を対象とします．

2．2　Webオートマトン:Webアプリケーションの振舞いモデル

Webアプリケーションの振舞いを抽象的に意味づけるためのモデルとして，Webオートマトンを提案しました．Webオートマトンは，ページ間遷移を状態遷移とみなす静的なWebシステムのモデルであるLinkオートマトンをMVC[2]にモデルに基づくWebアプリケーションに拡張した振舞いモデルです．本研究においてはMVCアーキテクチャに基づくフレームワークとしてよく用いられている Apache Struts(アパッチ・ストラッツ)フレームワークを抽象的な振舞いモデルとして表現します．Webオートマトンは Webアプリケーションの抽象的な実行の概念を表現します．Webオートマトンの応用として Webアプリケーションのテスト系列の生成手法について研究しました．

Struts フレームワークの基本動作の概略は以下の通りです．request チャネルを通して外部からHTTP リクエストを受け取り，ActionForm オブジェクトを作成し，Action オブジェクトを発行します．Action オブジェクトはこの際にデータベースなどのバックエンドのビジネスロジックにアクセスします．Action オブジェクトの結果は ActionForward オブジェクトで返されます．その結果を構成ファイル(struts-config.xml)を参照して得られるActionMapping記述に従ってJSPを発行し，ブラウザなどのクライアントに結果を返します．この Web アプリケーションの振舞いをオートマトンの状態遷移としてモデル化しました．

Webオートマトンは$\langle View, \delta, \iota, iv, Trans, FV \rangle$という6つ組みで表現されます．$View$はJSPによって生成される画面の集合，$\delta$は$View$の各要素を生成するJSPに含まれる変数の集合，$\iota$は$View$の各要素に含まれる変数に成立する条件の集合，$iv \in View$は初期ビュー，$Trans$はビュー間の遷移関係，$FV \subseteq View$は最終状態の集合を表します[3]．Web オートマトンは EFA(Extended Finite Automata)の一種であり，URL リクエストによる変数の更新とページ遷移をモデル化します．Webオートマトンにおける$View$は JSP の変数に代入される値によって無限になります．しかし，遷移のパターンはその構成から本質的には有限です．状態遷移のパターンによって値を有限的に分類することによって，有限の$View$によって振舞いを有限的に特徴づけることができます．

Struts フレームワークで構成された図書管理システムの Web オートマトンの遷移グラフを図3

---

[2] MVC=Model View Control

[3] ここで最終状態を定義に含めたのは，このモデルの能力を示すために，テスト系列の生成に応用するためです．通信プロセスモデルとしてモデル化する際には最終状態の区別は本質的に必要ありません．

に示します．このシステムは，図書データの検索と更新を行います．更新をするためにはあらかじめ登録されたアカウントでログインしなければなりません．一回の検索，更新が終了する View を最終状態として定義しています．

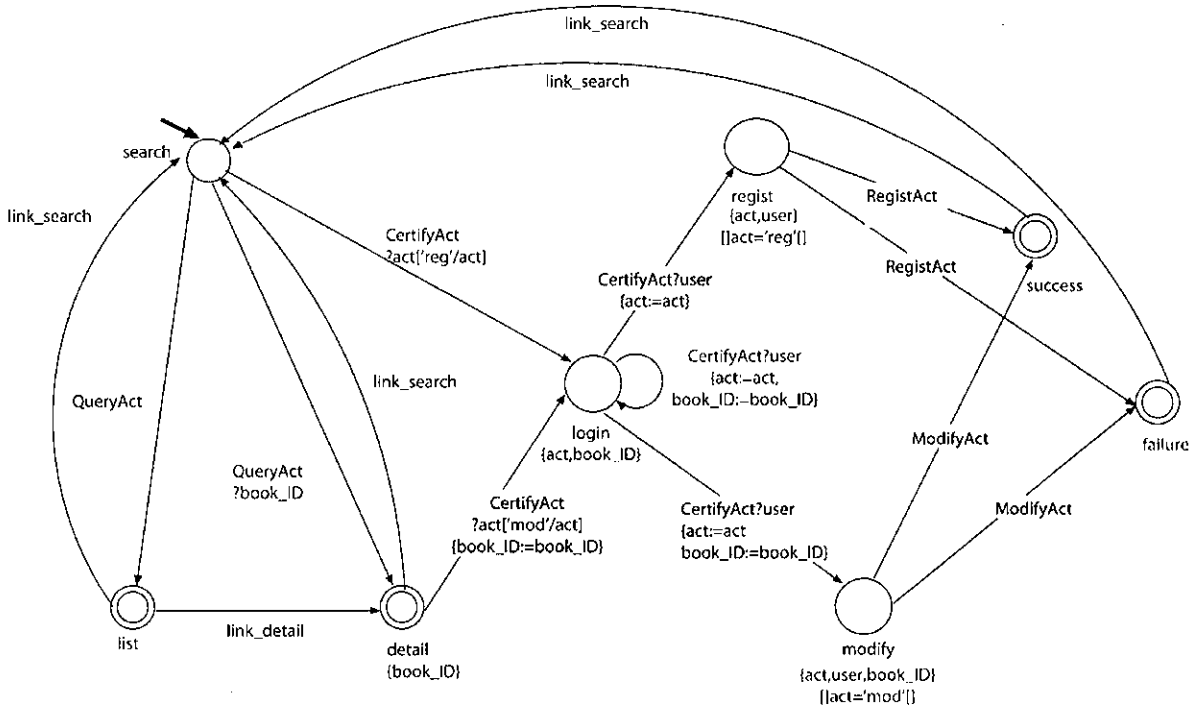


図３：Web オートマトンの例

Web オートマトンによるテスト系列生成

Web オートマトンの応用として，Web オートマトンの表す受理系列を Web アプリケーションのテスト系列として有効なテストを生成する手法を提案しました．Web オートマトンにループが存在して初期 View から到達可能である場合，Web オートマトンの生成する系列は無限になります．ここで，テストという側面から見ると同じループを回る場合はテストとしてはよく似たテストであるという観測から，ループを回る回数で系列を分類し，Web アプリケーションに対するテスト基準を提案しました．通常のテストでは，ループを２回以上回らない系列が用いられます．図３の Web オートマトンの場合のテスト系列の数を表１に示します．ここでループ基準T^nはループを n 回実行する系列の集合を意味します．

ループ基準	T^0	T^1	T^2
系列の数	9	１３２	１９４７

表１：生成されるテスト系列数

２．３　通信プロセスモデルによるモデル化へ

Web オートマトンのような抽象的な振舞いを Web アプリケーションから直接生成するために Web 本研究では，Web アプリケーションを表す通信プロセスモデルとして非同期π計算という体

系を対象とします．ここでの目的は，Webアプリケーションの振舞いモデルをプログラムから直接生成することです．

MVC のアーキテクチャに基づいてそれぞれの記述ごとに通信プロセス項に翻訳し，そのまま振舞い合成して全体の振舞いを定義します．この変換は以下の手順で行います．

(1) ビューに対する翻訳：各 JSP からアンカータグ，リンクタグおよびフォームタグを構文的に抽出し，アンカー，リンク，フォームの各パーツを振舞い合成します．

(2) モデルに対する翻訳：FormBean をオブジェクトとして変換し，setter メソッドおよび getter メソッドを実装します．

(3) コントロールに対する翻訳：Action オブジェクトを Struts-config.xml にしたがって変換します．Action オブジェクトは，ビューの中のフォームタグからの通信によって起動され，結果によってビューを起動します．

それぞれの変換ではコンポーネント単位の Web オートマトンを構成し，非同期π計算の項に変換します．

変換ターゲットとして Pierce と Turner によって提案されている非同期π計算言語である Pict を用いました．図4にログイン手続きを行う変換結果を示します．Pict 処理系によって抽象的に実行することができます．

ここでは，ログインページにおけるリンクの振舞い，ページ遷移，データベースとの通信をπ計算の同期通信としてモデル化しています．さらに，Pict 言語はタイプづけされた言語であるため，通信内容を表すタイプづけをすることができます．

2．4　その他の研究

通信プロセスモデルの柔軟性を高めるために，時間経過を考慮した通信プロセスモデルの体系においてどのような演算子がどのような遷移的動作意味を持てば，双模倣等価性が代数を構成するかについて研究を行いました．通信プロセスモデルでは遷移的動作意味は構造的操作意味定義(Structural Operational Semantics，以下，ＳＯＳ)で定められます．ＳＯＳは一般に次のような形をした推論規則の集合で表されます[4]．

$$\frac{X_1 \xrightarrow{a} X_1',\ X_2 \xrightarrow{b} X_2'}{f(X_1, X_2) \xrightarrow{c} t[\vec{Y}]}$$

このような形の推論規則が一定の条件を満たすとき，一意的に定義される遷移関係から定められる双模倣等価性が定義される演算子に対して合同性を持つことを示しました．この結果を応用すれば演算子構成が柔軟な通信プロセスモデルを構成することができます．さきがけ研究においてこの結果を直接応用するにはいたりませんでしたが，通信プロセスモデルにおける振舞い等価性について，さらに，時間の概念について，今後の見通しに対する知見を得ることができたと考えています．

---

[4] 前提に遷移の否定を含んだり，変数ではない一般的な項を含むなどいろいろなバリエーションが提案されています．

```
def loginPage[login:^^[^String ^String ^[] ^[] ^[] ^[] ^[] ^[]]]
  = login?req =
    ( new mailId:^String
      new passwd:^String
      new loginSubmit:^[]
      new newAccount:^[]
      new forgotPassword:^[]
      new authLogin:^^[^String ^String ^[]]
      new newAccountLink:^[]
      new forgotPasswordLink:^[]
      new authLoginActionCont:^[]
      new newAccountCont:^[]
      new forgotPasswordCont:^[]
      ( req![mailId passwd loginSubmit
             newAccountLink forgotPasswordLink
             authLoginActionCont newAccountCont forgotPasswordCont]
      | mailId?id =
         passwd?pw =
          loginSubmit?[] =
           ( new req:^[^String ^String ^[]]
             ( authLoginAction![authLogin]
             | authLogin!req
             | req?[idParam pwParam cont]
                 = (idParam!id | pwParam!pw
                    | cont?[] = authLoginActionCont![])
             )
           )
      | newAccount?[] = newAccountCont![]
      | forgotPassword?[] = forgotPasswordCont![]
      ))

and authLoginAction[authLogin:^^[^String ^String ^[]]]
    = authLogin?req =
      ( new r:^Bool new idch:^String new pwch:^String
        new c:^[]
        ( req![idch pwch c]
        | idch?id = pwch?pw = askDBAuth![id pw r]
        | r?b = if b then print!"Forward OK"
                     else print!"Forward NG"
        )
      )

and newAccountPage[newAccount:^^[]]
    = newAccount?req = ()

and forgotPasswordPage[forgotPassword:^^[]]
    = forgotPassword?req = ()

and askDBAuth[id:String pw:String r:^Bool]
    = r!true

run ( loginPage![login] | login!loginReq
    | loginReq?[id passwd submit nalink fplink cont1 cont2 cont3] =
      ( id!"syuen" | passwd!"1234" | submit![]
      | cont1?[] = print!"AuthLogin"
      | cont2?[] = print!"New Account"
      | cont3?[] = print!"Forgot Password"
      ))
```

図4　Pictによる出力結果の記述

3．今後の展望

Webアプリケーションはここ数年で大変な拡大を遂げ，多くの概念とソフトウェア構築技術が提案されています．このことは，従来のプログラミングモデルがWebアプリケーションには直接的には不十分であることを示しています．ここでは，主にソフトウェア工学的なアプローチからプログラミング記述に手を加えています．このようなアプローチは，実際のプログラム構築に対して即効性のある手法ですが，検証という点では最終的なプログラムの正しさはプログラマーの書くプログラムコードを対象としなければならないため限界があります．この点について，本研究では限定的であるにせよ，直接的にコンポーネントの構造を反映した振舞いの抽象化を与えることができました．

通信プロセスモデルによる記述で明らかになる興味深い点は，Strutsフレームワークなどの記述が分散的で複数のコンポーネントにわたって制御が横断的に移動するにもかかわらず，継続を通信プロセスモデルにおける計算メカニズムでモデル化した場合，その中心的な制御は非常に逐次的に捉えられるという点です．本研究の時間的範囲では理論的展開をもって示すことができませんでしたが，今後，型理論を適用することでこの制約を明らかにできると考えています．

本研究ではサーバーサイドプログラムに限定し，特定のフレームワークに対する研究を進めてきました．これは問題を単純化するための制限であり，通信プロセスモデルにおける振舞い合成はサーバーサイド，フレームワークに限定されません．本研究による性質は最も簡単な到達可能性をに基づくテスト基準の提案にとどまっています．さらに変換については手作業で行っており，これを自動的に変換するツールを現在作成中です．

今後は，Springなど他のフレームワークに対する応用，さらに，最近よく使われるようになってきたリッチクライアントプラットフォームのプログラミングに対する応用に通信プロセスモデルを適用して，サーバーとクライアントが協調して動作するWebアプリケーションのモデル化を行うことが考えられます．このようなシステムでは，コンポーネント間の分散性がより高まるため，プログラムの性質を導くためにはより有効な手段になると考えられます．通信プロセスモデルにおける振舞い合成の概念は，サーバーとクライアント間のコンポーネント合成と，サーバー内部でのコンポーネントを区別しません．あるいは，フレームワークの組み合わせ方についても区別しません．しかし，実際のプログラミング開発においては，デバッグなどの過程でこれらの合成を区別する方が有用な情報を得られる可能性もあります．このような場合に演算子をＳＯＳで新たに定義することで通信プロセスモデルを拡張してより詳細な性質を示すことができると考えられます．

4．成果リスト

論文誌

1. Atsushi Mizuno, Ken Mano, Yoshinobu Kawabe, Hiroaki Kuwabara, Shoji Yuen, Kiyoshi Agusa, "Name-passing style GUI Programming in the π-calculus-base language Nepi" Electric Notes of Theoretical Computer Science (Elsevier) , Volume 139 Issue 1, pages 145-168, 2005 (ARTS'04)
2. Mohammad Sharaf Aun, Shoji Yuen, Kiyoshi Agusa,"Towards Assuring Quality Attributes of Client Dynamic Web Applications: Identifying and Addressing the Challenges" Journal of Web Engineering , Vol.4 , 2005 , pages 144-164
3. Shoji Yuen, Keishi Kato, Kiyoshi Agusa, "Web Automata: A Behabioral Model of Web Applications based on the MVC", コンピュータソフトウェア , 22 巻 , 2005 , pages 44-57
4. Irek Ulidowski, Shoji Yuen ,"Process languages with discrete relative time based on the Ordered SOS format and rooted eager bisimulation", The Journal of Logic and Algebraic Programming (Elsevier) , Vol.60-61 , 2004 , pages 401-460
5. 桑原 寛明,結縁 祥治,阿草 清滋, "時間付きπ計算によるリアルタイムオブジェクト指向言語の形式的記述", 情報処理学会論文誌 , 45 巻 , 2004 , pages 1498-1507
6. Mohammad Sharaf Aun, Shoji Yuen, Kiyoshi Agusa, "An Approach for Debugging Client Dynamic Web Applications", 情報処理学会論文誌 , 45 巻 , 2004 , pages 2373-2383
7. 渥美紀寿,山本晋一郎,結縁 祥治,阿草清滋, "FCDG に基づいたコーディングパターン", コンピュータソフトウェア , 21 巻 , 2004 , pages 27-36

解説論文

1. 結縁祥治, "通信プロセスモデルと形式意味論に基づくソフトウェアのモデル化", コンピュータソフトウェア , 22 巻 , 2005 , pages 22-43

会議(査読つき)

1. H. Kuwabara, S. Yuen, and K.Agusa, "Congruence properties for a Timed Extension of the π-calculus", in Supplemental Volume of DSN2005, 2005, pages 207-214
2. Atsushi Mizuno, Ken Mano, Yoshinobu Kawabe, Hiroaki Kuwabara, Shoji Yuen, Kiyoshi Agusa, "Name-passing style GUI Programming in the π-calculus-base language Nepi", in Preliminary Proceedings of ARTS2004, Technical Report No.2004/28, 2004, pages49-66
3. S. Yuen, K.Kato, D.Kato, S.Yamamoto, K.Agusa: "Testing Framework for Web Applications based on the MVC model with Behavioral Descriptions", ICITA04, 2004, 11-4:pages 1-6

口頭発表

1. 加藤大樹, 結縁祥治, 阿草清滋, "高信頼性Webアプリケーションのための振舞い合成手法", 第２回ディペンダブルソフトウェアワークショップ, DSW05, 2005, pages 41-50
2. 桑原寛明, 結縁祥治, 阿草清滋："π計算に対する時間拡張と代数的意味論", ソフトウェア工学の基礎XI, FOSE2004, 2004, pages 97-108
3. 末次亮, 結縁祥治, 阿草清滋："時間オートマトンの遷移制約記述に基づくAIBOプログラムスケルトンコードの生成手法", 組込みソフトウェアシンポジウム2004, 2004, pages126-133
4. 深谷直彦,結縁祥治,阿草清滋, "実行可能なメモリモデルに基づくJava並行プログラムのモデル検査",ソフトウェア工学の基礎ワークショップFOSE2003 , 2003 , pages 227-238
5. 桑原寛明, 結縁祥治, 阿草清滋, "時間付きπ計算によるリアルタイムオブジェクト指向言語の形式的記述", 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, 2003, pages 67-73

特　許

1. 特願2004-165578 "ウィジェット操作方法, 装置, プログラムおよびこのプログラムを記録した記録媒体", 出願日：平成１６年６月３日