様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成２５年５月６日現在

機関番号：３３９１７
研究種目：基盤研究（C）
研究期間：2010～2012
課題番号：２２５００００３６
研究課題名（和文）　観点に基づくプログラム分解と合成による編集支援方法

研究課題名（英文）　A Programming Support Method based on Decomposition and Composition using Views

研究代表者
吉田 敦（YOSHIDA ATSUSHI）
南山大学・情報理工学部・教授
研究者番号：50283495

研究成果の概要（和文）：
プログラム編集支援の要素技術であるプログラム変換は，字句を単位としつつ，前処理指令との整合性や，コメントの維持など，様々な観点で整合性を保つ必要がある。本研究では，抽象構文木の表現形式として属性付き字句系列を提案した。抽象構文木，前処理，コメントなどの各観点に対して中立なモデルであり，各観点に閉じた操作を実現できる。構文解析器およびプログラムパターン変換系を実現し,応用ツールを通じて観点を切り分けられることを確認した。

研究成果の概要（英文）：
Program transformation is a key technique for programming support. It should consistently modify programs under the restrictions of preprocess directives, comments preservation. In this research, we have proposed "attributed token sequence" as a format of abstract syntax tree, which is neutral for views, such as abstract syntax tree, preprocess directives, and comments. It enables us to build tools for each view independently. We have implemented a parser and a pattern-based program transformation tool to confirm that they enable separation of views for building applications.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
２０１０年度	900,000	270,000	1,170,000
２０１１年度	700,000	210,000	910,000
２０１２年度	500,000	150,000	650,000
年度			
年度			
総　計	2,100,000	630,000	2,730,000

研究分野：総合領域
科研費の分科・細目：情報学・ソフトウエア
キーワード：ソフトウエア工学・プログラム開発支援・プログラム変換

１．研究開始当初の背景

プログラムの保守の過程では，ライブラリの仕様変更にともなう参照箇所の修正など，プログラム全体に分散した編集作業が生じることがある。大規模なプログラムでは，依存関係が強い要素をモジュール化によってまとめ，モジュール間の結合が減るように設計することで，修正範囲が局所化するような配慮が行われる。しかし，修正の内容を予測することは難しく，修正箇所が広い範囲になることも少なくない。

特に，エラー処理やデバッグ，最適化などに関わる修正は，モジュール分割とは異なる

切り口でプログラムを見る必要があり，修正箇所はプログラム内で分散化する。一つの対策としてはアスペクト指向プログラミングを用いる方法があるが，アスペクト指向プログラミングは同じ処理が横断的に発生する場合に有効であり，処理が異なる場合には利用し難い。例えば，エラー処理は，基本的な書き方は似ているが，それぞれのエラー処理部分によって前後の文脈に合わせた記述が必要である。また，パターン置換による書換えを利用する方法も考えられるが，書換え後の妥当性の確認や前後の文脈に合わせた最適化など，分散的に存在する断片を目で確認しつつ修正することが必要であり，単純なパターン置換機能のみでは不十分である。その他，セッション管理やリポジトリの利用など，様々な横断的な要素が存在しており，それらに関わる処理は一律に管理する必要がある。

このように，ある観点を切り口としてプログラムを見ながら統一的な修正をしたいが，それぞれが前後の文脈に合わせて修正内容が異なる可能性がある場合が存在しており，現状ではそのような作業を支援する環境が整っていない。

２．研究の目的

本研究の目的は，１つのプログラムからある種の観点に合致する断片を抽出し，その断片群を編集したのちに合成することで修正を反映したプログラムを得るための環境(ツール群)を構築し，特定の観点に集中して作業できるようにすることである。そのために，観点ごとに断片を切り出す方法と編集後に合成する方法を明らかにする。観点に基づいて断片を切り出すためには，観点の定義記述が必要であり，本研究では特にプログラムのパターン記述を中心に検討を行う。合成する場合，他の断片との競合が発生する場合があり，単純に合成できるとは限らない。よって，競合の解消方法や断片を抽出する際に競合をできるだけ避ける工夫などを検討し明らかにする。

３．研究の方法

研究を進めるうえで基盤となる環境を作るために，プログラムの構文解析器とパターン変換系の構築を行った。書換え対象のプログラミング言語は，C 言語とした。また，C 言語本来の構文木，前処理指令，コメントとスペースの 3 つの観点に着目し，それぞれ独立に編集するための方法を検討し，実装を行った。

４．研究成果

(1) プログラムの構文解析器について

この研究の土台として，まず，抽象構文木を構成し，それを操作する環境が必要である。その最初の段階として，構文解析器を構築した。この構文解析器は，生成する抽象構文木がそのあとで書換えられることを前提に構築する必要があり，従来の情報の抽出に用いられる抽象構文木の形式では情報が不足するので，新たに考える必要がある。

C 言語では，標準的に前処理指令が使われており，一般的に，前処理前の状態は構文規則を満たさず，構文解析は困難である。しかし，プログラマは前処理前の記述を編集しており，その編集を支援するためには，前処理前の状態での解析結果が必要となる。

前処理後であれば構文解析は可能であるので，前処理後を解析し，そのあとに逆の変換を行う方法も考えられる。これは，プログラムの静的解析など，情報を抽出するときには有効な方法である。しかし，書換えの場合，前処理の変換による写像と逆写像を，書換えと同時に更新していく必要がある。その処理は極めて複雑であり，また，実現できない場合も生じる。そこで，前処理前のプログラムでは，構文規則から逸脱する箇所は限定的であることから，部分的に精度が異なる抽象構文木を構成することで，前処理前の記述をそのまま構文解析する方法を検討した。

全体の精度を下げた解析であれば，実現できる可能性があり，かつ，最も精度が低い状態は単純に字句が並んだ状態である。そこで，字句の並びから，粗く構文要素を抽出し，そのあと，さらに詳細に解析していくことで，全体として解析が可能と考えた。また，字句の並びをベースとすれば，その書換えは字句の並びの編集であり，書換えの実現も容易になる。そこで，構文解析結果を保持できる属性付き字句系列を提案した。

属性付き字句系列は，基本的にはプログラムのテキストを構成する字句を並べたものであり，それらを再結合すると元のテキストを再現できる。さらに，各字句には，その字句の型を属性として持つ。この型は，字句解析における字句の種類とほぼ同じである。また，空文字で構成される字句に非終端要素の開始と終了の型を割り付けることで，文や宣言などの非終端要素も表現できる。この空文字の字句は仮想字句と呼ぶ。

属性付き字句は，次の書式で記述する。

型名（*属性値**）〈テキスト〉

*属性値*には，*型名*以外の値を記述することを想定しているが，実装した解析器では，対応関係を示す識別記号を括弧や仮想字句に割り当てており，これが唯一の属性値となっている。この識別記号は，対応する括弧の組や，

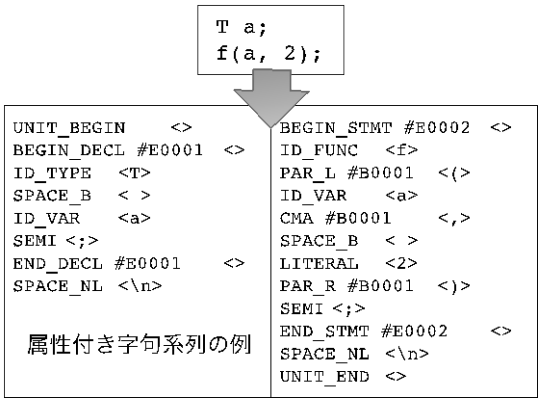

図 1 属性付き字句系列の変換例

仮想字句の開始と終了の組に対して，同じ記号を割り当てることで，組み合わせを正確に把握するために用いる。また，カンマについては，括弧内または宣言や文を区切る要素であることから，その区切る対象と同じ識別記号を割り当てている。これにより，後述するパターン変換系において，正確にパターンの適合範囲を決定することができる。図 1 に属性付き字句系列の変換例を示す。

構文解析時に前処理指令を取り扱うために，観点を切り替えて解析を行った。すなわち，前処理指令を構成する部分字句列のみに着目し，前処理指令の構文解析を行い，次に前処理指令を構成する部分字句列を空白と同様に無視して，C 言語本来の構文解析を行う方法である。

C 言語本来の構文解析では，人間がプログラムの一部のみを見ても，プログラムを正しく理解できる点に着目し，字句の構成に関するヒューリスティックな知識を書換えルールとして整備した。解析時には，文レベルまでの粗い構文解析を適用し，そのあとで書換えルールを適用することにより，精度を高めた。解析処理の全体の流れを図 2 に示す。

GNU Core Utils や FreeBSD のソースプログラムの解析実験を行い，実用的な精度で解析が可能なことを確認した。また，ウェブ上で公開されているプログラム断片は，その実体としては HTML タグを含むテキストであるが，それらもタグの字句定義を加えることで，解析ができることを確認した。これは HTML という観点とプログラムの 2 つの観点を取り扱えることを意味する。

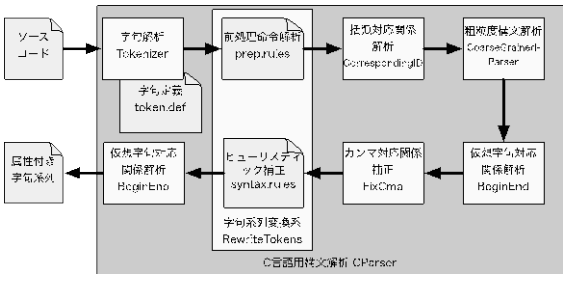

図 2 解析器内部の流れ図

前処理前のプログラムでは，識別子の定義が与えられないことが多いことから，記号表を構成しないで解析を進める戦略を採用した。しかし，精度を高めるためには，部分的な記号表が必要なことから，不確定な情報を許容する記号表を用いて，プログラム全体を解析する過程で，補正していくフィルタ型プログラムを構成した。最終的に構成された記号表を用いて再解析することで，さらに精度を向上できることを確認した。

(2) プログラムパターン変換系について

プログラムの書換えを直感的に表現する方法として，プログラムの本来の記述をそのまま使用できるパターン変換記述を提案した。基本的には，書換え前の断片と書換え後の断片を記述し，必要に応じて，パターン変数を用いて抽象化も表現できる。従来のプログラム変換系は，抽象構文木に対する操作を記述していたので，利用者が本来やりたい編集内容を抽象構文木に対する操作に翻訳する必要があったが，提案方法では，その手間がなくなり，容易に記述可能である。ただし，細かな操作が記述しにくい欠点がある。

パターン変換系は，このパターン変換記述を構成する 2 つの断片を構文解析し，字句系列に対するパターン変換ルールに翻訳する。書換え対象のプログラムを属性付き字句系列に変換したあと，このルールを適用することで書換えを実現する。ルールの処理は，字句系列に対するパターン変換が可能な専用の処理系を構築し，それを用いて実現した。この処理系は，ルールを文字列に対する正規表現に翻訳し，実装言語である Perl の正規表現エンジンで書換え処理を行うように実装した。

パターン変換の記述例と実行例を図 3 に示す。これは前処理条件分岐指令で分断された文から前処理分岐指令を外に移動させる処理である。この変換によって可読性が高まるとともに，別のパターン変換を行うときの前段階としても使用できる。

(3) 前処理指令に関する書換えについて

前処理指令を含むプログラムを書き換えるときは，前処理指令に対する観点と C 言語本来の構文木での観点を混ぜた状態で考える必要があり，複雑である。図 3 もその例であり，文の構成と，前処理分岐指令の両方を取り扱う必要がある。ただし，図 3 は簡単な処理であり，より複雑な書換えにおいても適切に前処理指令を取り扱えるか検証する必要がある。そこで，前処理指令の代表であるマクロ定義について逆置換する機能を，前処理分岐指令に対しては，それを含む

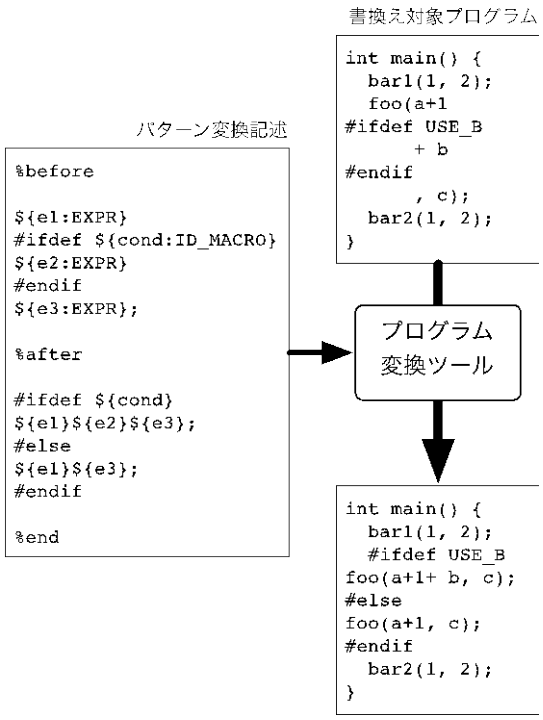

図 3 プログラムの変換例

断片を整合性を保ちつつ書き換える方法を題材として，前処理指令に関する処理と，構文木に対する操作が独立に記述できるか検討した。

マクロの逆置換とは，マクロを定義したときに，そのマクロで置き換えるべき記述を自動的に探し，マクロの参照に書き換える処理である。ただし，マクロに置き換えるべき箇所の書き方は一律ではなく，表現の違いを考慮する必要がある。そこで，マクロの定義から，逆置換をする字句系列に対するパターン変換ルールを生成するツールを構成した。マクロの定義そのものも解析し，表現の異なる記述に変換して，複数のルールを生成することで，表現の違いにも対処した。マクロは一般的に式レベルで書かれることから，式レベルの構文解析も行った。検証実験の結果，2つの観点の間で不整合が起こることなく書換えが実現できることを確認した。

前処理の分岐指令を含む記述を書き換えるときは，その分岐命令をまたがった書換えに対処する必要がある。一つの方法は，すべての分岐の組み合わせに基づいて，分岐のないテキストをすべて生成し，書換えを適用してから合成する方法が考えられるが，現実的なプログラムでは，組み合わせ爆発が起こるので，実用的ではない。また，合成後の前処理指令の位置を決定することが難しいことも判明した。そこで，一般的に，パターンが適合する範囲が狭いことと，そこに含まれる分岐数も少なくなることに着目し，パターンの適合範囲を求めるときだけ，分岐先を調べながら進め，いずれかの分岐で適合した時点で，その範囲に含まれる分岐指令を範囲外に移動させる変換を実現した。極端に適合範囲が広いなど，前提に合わないパターンを用いない限り，組み合わせ爆発の発生を抑えながら書換えが実現できることを確認した。分岐指令を範囲外に移動させる変換は，可逆的な等価変換となるよう定義し，書換え後は可能な限り元の位置関係に戻るようにした。

(4) コメントと空白の維持について

ツールによってプログラムを書き換えたときに，コメントや空白が失われることは，実用性の点で問題がある。空白は，スタイルの整形ツールや開発環境の整形機能などを用いることで補正できるが，コメントを失うと取り戻すことができない。

既存の研究では，抽象構文木の要素としてコメントを保持する方法が提案されているが，どのように適切な位置に移動させるかは明示されていない。そもそもコメントと構文要素の関係は明示的には表現されず，プログラマが意味を解釈してその関係を理解している。また，その際に，コメントの出現位置には一定の習慣があり，関係を決めるときのヒントになっているが，その位置関係は構文木とは関係なく，文字と行で構成されるテキストの視点で決められている。したがって，既存の研究のように，構文木に直接結びつける方法は不適切であり，本来は，構文木の構成と，構成要素とコメントの位置関係は独立したモデルで定義されるべきである。

そこで，コメントはその前後で書換え後も残る字句のうち距離(その字句にたどりつくまでに無視する字句の数)が最短のものに付随して移動するというモデルを設定し，プログラムの書換えと，コメントの維持が独立して記述できるか確認をした。

プログラムパターン変換系に対し，パターン変換記述を解析した直後に，このモデルに基づく解析を行い，コメントや空白に適合するパターン変数を埋め込むフィルタを挿入したところ，ほぼ自然な形で維持できていた。(図 4) ただし，小規模の事例であり，また，要素が削除されたときに，本来であれば付随して削除されるべきコメントも残る問題など，実用面ではまだ完全ではない。

作成したフィルタは，パターン変換系に挿入するときに，挿入箇所以外での修正は必要なく，また，フィルタ自体は構文木の構成に関する情報は使用していない。したがって，コメントに対するビューのみで独立して実装できたことを意味する。

(5)総括

本研究の目的は，観点を分離してとらえ，それぞれの観点での書換え結果を合成する

ことで，全体の書換えを実現することである。研究では，3 つの観点に立ち，実現方法を検討した。その結果，属性付き字句系列を土台とし，特定の観点に関する字句だけを見て書き換えることで，当初予想していた合成という手順を用いることなく，書換えを実現できた。

本研究の手法は，island grammar と呼ばれる部分解析の手法と類似しているが，island grammar が解析木を作ることが目的であるのに対し，本研究ではさらに書換えを行う点で，既存の研究とは異なる。

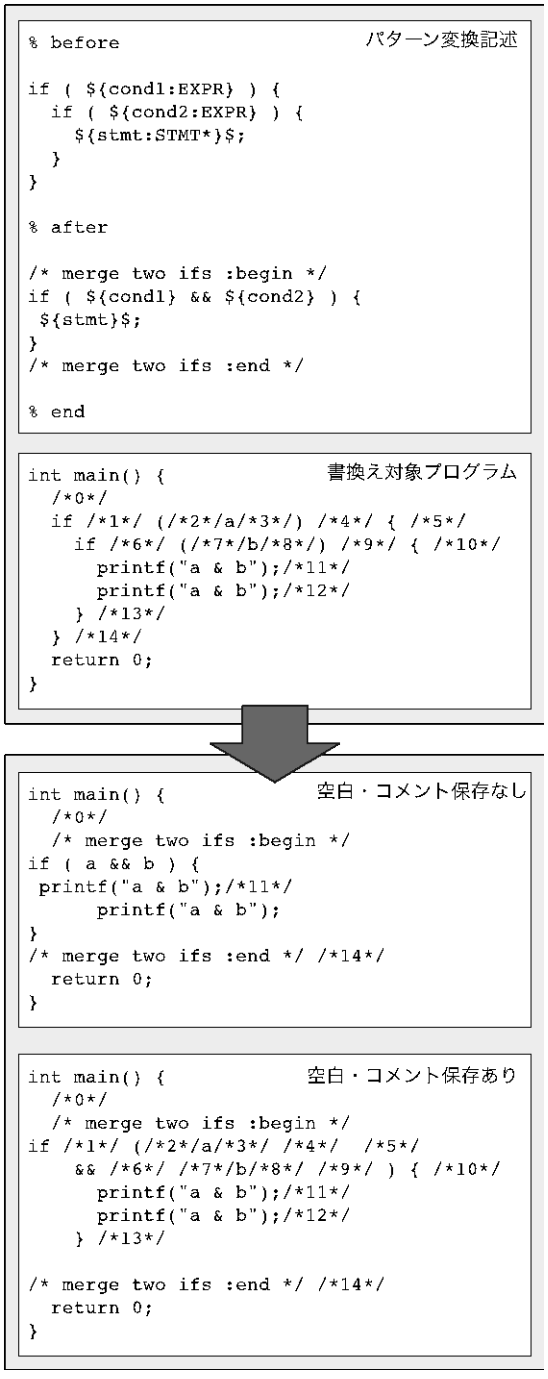

図 4 コメントを保存した変換

この研究では実用性という面においても成果が得られた。これまで，TXL や srcML など，プログラムの書換え系は提案されているが，書換え時の前処理指令やコメントの取扱いについては検討されておらず，プログラムの最適化や，プログラムから情報を抽出するといった，それらを無視しても使える分野にしか適用できなかった。本研究では，これらを積極的に扱い，独立した観点としてとらえて操作することで，適切に維持しながら書換えができることを示せた。

一方，当初，想定したものの実現に至らなかったものとして，データフローや制御フローをビューとして扱うことが挙げられる。制御フローについては，前処理の分岐指令の扱いを応用することで実現は可能である。データフローは，スライスを求めて，そのスライスを書換えることになるが，スライスに含まれなかった箇所とどう整合を取るかが難しい。

今後の課題としては，書換え系の実用性の検証である。実用化に必要な技術は得られたが，小規模の実験の中でしか検証できていない。本研究で構築したプログラム解析器はオープンソースに適用することで，現実的な規模のプログラムに対応できることを示すことができたが，書換えに関しては現実的な書換えの題材を用意することが難しい。一つの方法が，オープンソースのリポジトリの編集履歴から，繰り返し適用される書換えパターンを抽出し，それらを再適用して，実際の書換え結果とどの程度の差異が生じるか確認することである。また，このような書換えパターンの抽出は典型的なバグの修正など，新たな展開につながることが期待できる。

５．主な発表論文等

〔雑誌論文〕（計 11 件）

①前林達也，吉田敦，蜂巣吉成，張漢明，野呂昌満，「前処理プログラムに対する記号表の構成手法」，情報処理学会論文誌，Vol.54，No.2，pp. 912-921，2013,査読有.
②蜂巣吉成，吉田敦，張漢明，野呂昌満，「文脈を考慮した HTML4 から HTML5 への書き換え方法の提案」,ソフトウェア工学の基礎 XIX -- 日本ソフトウェア科学会，FOSE 2012，pp. 21-26，2012，査読有.
③前林達也，吉田敦，蜂巣吉成，張漢明，野呂昌満，「前処理プログラムに対する記号表の構成手法」，ソフトウェアエンジニアリングシンポジウム 2012 論文集，Vol.2012，pp. 1-6，2012，査読有.
④立道昂太，吉田敦，蜂巣吉成，張漢明，野呂昌満，「Web ページ記述内のプログラム断片に対する DOM tree を用いた構文木の構成手法」，ソフトウェアエンジニアリン

グシンポジウム 2012 論文集, Vol.2012, pp. 1-6, 2012, 査読有.
⑤吉田敦 , 蜂巣吉成 , 沢田篤史, 張漢明, 野呂昌満, 「属性付き字句系列に基づくソースコード書換え支援環境」, 情報処理学会論文誌, Vol.53, No.7, pp. 1832-1849, 2012, 査読有.
⑥ Sawada, A, Noro, M., Chang, H.-M., Hachisu, Y. and Yoshida, A., "A Design Map for Recording Precise Architecture Decisions", Proceedings of the 18th Asia-Pacific Software Engineering Conference (APSEC2011), pp. 298-305, 2011, 査読有.
⑦沢田篤史, 野呂昌満, 蜂巣吉成, 張漢明, 吉田敦, 長大介, 浦野彰彦, 「ソースコードインスペクションツールのためのソフトウェアアーキテクチャの設計と進化」, 日本ソフトウェア科学会 編, コンピュータソフトウェア, Vol. 28, No. 4, pp. 4_241-4_261, 2011, 査読有.
⑧蜂巣吉成, 吉田敦, 野呂昌満, 沢田篤史, 張漢明, 「HTML 要素の状態を考慮した CSS の拡張方法の提案」, 電子情報通信学会論文誌 D, Vol. J94-D, No. 11, pp. 1931-1934, 2011, 査読有.
⑨蜂巣吉成, 野呂昌満, 沢田篤史, 張漢明, 吉田敦 , 「コンパイル方式による XQuery 問い合わせプログラム生成方法」, ソフトウェア工学の基礎 XVIII -- 日本ソフトウェア科学会, FOSE 2011, pp. 21-30, 2011, 査読有.
⑩曽我展世, 吉田敦, 蜂巣吉成, 沢田篤史, 張漢明, 野呂昌満, 「表現の違いを考慮したマクロ逆置換方法の提案」, ソフトウェアエンジニアリングシンポジウム 2011(SES2011)論文集, 情報処理学会シンポジウムシリーズ, Vol. 2011, pp. 1-6, 情報処理学会, 2011, 査読有.
⑪吉田敦 , 蜂巣吉成 , 沢田篤史, 張漢明, 野呂昌満, 「属性付き字句系列に基づくプログラム書換え支援環境の試作」, ソフトウェアエンジニアリング最前線(ソフトウェア・エンジニアリング・シンポジウム 2010 予稿集), pp.119-126, 2010, 査読有.

〔学会発表〕(計 5 件)
①張漢明, 野呂昌満, 沢田篤史, 吉田敦 , 蜂巣吉成 , 横森励士 アーキテクチャ指向開発における形式手法の適用に関する考察, 情報処理学会 組込みシステム研究会, 長崎県対馬交流センター, 2013 年 3 月 13 日.
②蜂巣吉成 , 吉田敦 , 「プログラミング学習における誤り訂正問題の自動生成方法の提案」, 電子情報通信学会 ソフトウェアサイエンス研究会,休暇村志賀島, 福岡県福岡市, 2013 年 3 月 6 日.
③張漢明, 野呂昌満, 沢田篤史, 横森励士, 吉田敦 , 蜂巣吉成 , 「並行システム記述におけるフォールトパターンに関する考察」, 電子情報通信学会 ソフトウェアサイエンス研究会, 広島市立大学, 広島市, 2012 年 11 月 1 日.
④張漢明, 野呂昌満, 沢田篤史, 吉田敦 , 蜂巣吉成, 横森励士, 「パターンに基づく CSP 記述の検査に関する考察」, 電子情報通信学会 ソフトウェアサイエンス研究会, 北陸先端科学技術大学院大学, 石川県能美市, 2011 年 10 月 28 日.
⑤張漢明, 野呂昌満, 沢田篤史, 蜂巣吉成, 吉田敦, 「階層分割に基づく組込みソフトウェアの振舞い検証の支援について」, 情報処理学会 組込みシステム, 公立はこだて未来大学, 北海道函館市, 2010 年 8 月 9 日.

〔図書〕(計 0 件)

〔産業財産権〕
○出願状況 (計 0 件)

○取得状況 (計 0 件)

〔その他〕
ホームページ等

６．研究組織
(1)研究代表者
吉田 敦 (YOSHIDA ATSUSHI)
南山大学・情報理工学部・教授
研究者番号：50283495

(2)研究分担者
蜂巣　吉成 (HACHISU YOSHINARI)
南山大学・情報理工学部・准教授
研究者番号：30319298

(3)連携研究者

なし