情報処理学会論文誌 Vol. 52 No. 11 2989–3007 (Nov. 2011)

B Methodにおける高信頼ソフトウェア部品自動生成

中村丈洋[†1] 織田健[†2] 西野哲朗[†2]

ソフトウェア部品への形式仕様の付加は部品の再利用性向上に有効である．しかし，再利用によりソフトウェアを開発するためには大量の部品を用意する必要があり，それらに形式仕様を与えることは大きなコストを生じる．これに対して既存ソフトウェアから部品を生成する手法があるが，仕様と部品の整合性保証が不十分である．そこで，本稿では仕様との整合性を保証できる高信頼ソフトウェア部品を提案する．また，この高信頼ソフトウェア部品をB Methodで構築されたソフトウェアから自動生成する手法を提案する．提案する高信頼ソフトウェア部品は実装依存モデルと細分化実装で構成され，仕様として細分化モデルを持つ．細分化モデル，実装依存モデル，細分化実装はそれぞれB Methodのモデルと実装に対応した記述であり，B Methodの信頼性の定義に基づいて信頼性が定義される．この部品の信頼性の定義に基づいて部品自動生成手法を構築することで自動生成された細分化モデルが無矛盾であるための条件を明らかにし，また，実装依存モデルと細分化実装が常に整合性を満たすことを保証する．この自動生成により高信頼部品リポジトリを容易に構築することが可能になる．さらに，高信頼部品リポジトリを応用することでB Methodで記述された要求仕様に対してそれを満たす実装を出力する高信頼自動コード生成が期待できる．

Denepndable Software Component Generation with B Method

Takehiro Nakamura,[†1] Takeshi Oda[†2]
and Tetsuro Nishino[†2]

Formal specifications help us to reuse software components. But it cost to write formal specifications for all of software components. Automated component generation by slicing existent software is proposed to reduce this cost. But it's not enough in component reliability. In this paper we propose dependable software components (DSC), which can be ensured to satisfy specifications, and propose automated dependable software component generation (DSCG). A DSC is constructed with an implementation dependent model (IDM) and a sliced implementation, and a DSC has sliced model as its specification. These are similar to model and implementation in the B Method, and we can generate proof obligation from them. So we can ensure the consistency of a DSC and its specification applying the B Method. In this paper, we ensure the dependability in two angle by constructing that based on definitions of the DSC's consistency. First, we prove that sliced models generated by the DSCG has no contradiction conditionally. Second, we express the sliced model generated by the DSCG is sure to satisfy the IDM. We'll able to synthesis dependable software by composite DSCs to satisfy requirement models in the B Method.

1. はじめに

ソフトウェアの大規模複雑化にともないソフトウェアの高信頼化が求められている[3]．同時に近年の情報技術の興亡は激しく，それらへの迅速な対応も重要な課題である．ソフトウェア部品再利用はこの課題に対する有効な手段としてソフトウェア工学黎明期から開発コストの低減や信頼性の向上に大きく貢献してきた[10]．部品再利用を導入することでコーディング時の人的誤りを回避し，コーディングに要するコストを削減できる．

阿草らが提案したCASEツールプラットフォームSapidは既存のソフトウェアから部品を自動生成することで部品リポジトリの整備を容易にする[21]．Sapidのように大量の部品を管理する際には大量の部品から必要とする機能を提供するものを再利用するために部品に対して仕様を付加することが重要となる[8),14)]．DapidはSapidで細分化された断片的なプログラムにXML文書を仕様として付加することでソフトウェアの検索を容易にするものであり，これを応用して仕様と断片的なプログラム間で整合性検査を行い，信頼性を保証することが提案されている[16]．しかし，DapidのようにXMLでタグ付けされた自然言語を仕様として扱う場合，部品再利用の健全性が低く，整合性検証が型チェックなどに限定されるという問題点がある．Koalaなど[11),13)]の形式的な仕様を持った部品では仕様を計算機上で扱うことが容易になり健全性の高い部品再利用やより細かな整合性検証が可能になる．しかし，これらの部品仕様も機能名や部品名が大きな意味を持ち，部品検索がその語彙に依存するという問題がある．また，部品リポジトリを整備する研究が十分になされていないことも問題である．現在実用化されている自動コード生成[2),4)]は高度に自動化された部品再利用

†1 電気通信大学大学院電気通信学研究科情報通信工学専攻
Graduate School of Infomation and Communication Engineering, The University of Electro-Communications
†2 電気通信大学情報理工学研究科総合情報学専攻
Graduate School of Informatics, The University of Electro-Communications

表 1　提案手法と既存手法の比較
Table 1　Comparison between our proposal and previous works.

	Sapid，Dapid	提案手法	Koala
部品の適用性	○（細粒度）	○（細粒度）	×（粗粒度）
再利用の健全性	×（自然言語仕様）	○（数学的仕様）	○（形式的仕様）
部品整備コスト	○（自動細分化）	○（自動細分化）	-（部品生成無）
部品生成の信頼性	×	○（整合性保証）	-（部品生成無）

ととらえることができる．自動コード生成は人的誤りとコーディングコストを低減させることができ[12]，その有用性が大きく報じられている[20]．しかし，部品が問題領域に依存するため，目的とする問題領域の部品群が整備されていなければ自動化の恩恵を受けられない．

本稿では高信頼ソフトウェア開発に実績のある形式手法 B Method[1] をソフトウェア部品に応用した高信頼ソフトウェア部品を提案する．提案する高信頼部品では部品に数学的な仕様を付加することで Sapid などで問題となった再利用の健全性を向上させる．また，B Method で構築された既存ソフトウェアからの高信頼ソフトウェア部品自動生成手法を提案する．これにより形式的な仕様を持ったソフトウェアを自動で細分化することで，形式的な仕様を持つ高信頼ソフトウェア部品の整備コストを低減させる．さらに，細分化手法を B Method の信頼性の定義に基づいて構築，検証することで提案手法で生成される部品群が高信頼であることを保証する．これにより，問題領域の過去成果物から高信頼な部品リポジトリを容易に整備することが可能になる（**表 1**）．また，我々は本稿で提案する高信頼ソフトウェア部品のように B Method のモデルを仕様として持つ部品群を合成して要求モデルから B Method のソフトウェアを自動合成する枠組みを提案している[19]．この自動合成は低機能な部品群を合成して高度な機能を容易に実現できるため，低機能だが細粒度な部品群を大量に用意することが望ましい．本稿で提案する部品生成手法は細粒度な部品群を機械的に生成するため，細粒度な部品を大量に用意するのに適しており，提案手法をソフトウェア自動合成に応用することで，高信頼なソフトウェアを問題領域に問わず自動生成することが期待される．

2. 関 連 研 究

2.1 ソフトウェア部品

ソフトウェア部品再利用はソフトウェア開発黎明期から今日までソフトウェア開発コストの低減と信頼性向上に大きく貢献してきた．ソフトウェア部品はその用途に応じて粒度や付加される情報が大きく異なる．JAVA API ライブラリのような部品の機能の説明を部品名や自然言語の付加文書に頼る部品では部品検索の健全性が低く，欲しい部品とそうでない部品を選別することが困難である．また，異なる問題領域では同じ機能であっても機能名や部品名が異なるため，問題領域を横断する部品検索ができない．このような問題点へのアプローチとしてはライブラリやクラスなどの部品に数学的仕様を付加することがあげられる．ある変数 y を含まない数式において変数 x を変数 y に書き直しても数式の意味は変化しないように数学的仕様は名前に頼らず部品や機能の意味を与えるため，数学的仕様を用いた検索は問題領域を横断した健全性の高い部品再利用を可能にする．このような利点を有する数学的仕様であるが，数学的仕様などの形式仕様の記述は自然言語の記述に比べてコストを要する．自動逆エンジニアリング[7] による仕様付加は計算機によって実装を抽象化することで仕様を生成し，形式仕様を記述するコストをおさえることができる．しかし，ソフトウェアを開発する際には必ず仕様を記述するため，それを破棄して実装から逆エンジニアリングを行うことは部品の仕様と人間の想定する仕様との乖離を招く．また，単に形式的仕様を付加したとしても部品の実装が仕様を満たすことを保証しなければ健全な部品再利用はできない．JML[6]，VDM[9]，B Method はいずれも実行可能なプログラムに対して集合論と述語論理による仕様を付加することでプログラムが仕様を満たすことを保証する．このうち，JML と VDM がプログラムを駆動することで仕様を満たさない状態に至らないことを検証するのに対して，B Method は定理証明によって静的に検証する．本稿では ‘なぜ，その実装が仕様を満たすのか’ という定理証明過程を用いてソフトウェアを細分化することで，自動生成されたソフトウェア部品の仕様と実装の整合性の保証を試みる．そのため，本稿で扱う高信頼ソフトウェア部品は B Method に基づく記述を持つ．

2.2 ソフトウェア部品自動生成

ソフトウェア部品を利用して開発を行うためにはソフトウェア部品リポジトリに部品を集積しなければならない．部品の集積方法としては 1 度開発したソフトウェアを分析してそこから再利用性の高い機能を抽出して部品化することが考えられる．ソフトウェア部品自動生成はこの作業を自動化することで部品リポジトリの構築を容易にし，より多くのソフトウェアドメインで部品再利用手法を適用可能にする．また，人手による部品生成時のバグの混入を防ぐうえでも部品自動生成は有効な手段である．阿草らの提案する CASE ツール Sapid はプログラムスライシングを応用することでソフトウェア部品を自動生成する[17]．プログラムスライシングはプログラムを要約する手法の 1 つであり，プログラム中で興味がある文の働きを理解しやすいように理解に不要な部分を取り除く．本稿ではプログラムスラ

イシング手法の 1 つである Weiser の手法[15] を用いるが，これは各命令文をノード，プログラムを有効グラフととらえ，注目するノード i と注目する変数群 V に対してノード i における変数群 V の状態を説明するのに必要十分なノード群（プログラム）を出力する．また，阿草らは部品自動生成において，その仕様も細粒度解析する Dapid を提案した[18]．しかし，Dapid の扱う仕様はタグ付けされた自然言語でありコードとの整合性検証が型チェックなどに限定される．

2.3 B Method

B Method はソフトウェアの仕様記述から実装工程までを網羅する形式手法である．B Method の記述は ‘モデル’，‘リファインメント’，‘実装’ に分けられる．モデルはソフトウェアの仕様であり，集合論と述語論理による制約と実行順序の概念がない非決定的な代入で記述される．実装は命令型言語同様に実行順序を持ち結果が一意に定まる代入で記述される．B Method では実装がモデルを満たすことを定理証明によって保証する．しかし，モデルは実装手段を考慮しない抽象的な記述であるため，要求を直接書き下したモデルと実装手段を明示した実装ではソフトウェアのとらえ方に大きな差が生じる．これを埋めるのがリファインメントである．モデルの概念を実装に書き下すことが困難である場合に実装を意識したモデルをリファインメントとして記述して，その記述から実装を書き下すことでモデルの概念を実装に書き下すことが容易になる．3.4 節で理由を述べるが，本稿ではリファインメントの書式を入出力として扱わないため，以下ではモデルと実装の構造を説明する．B Method のモデルは命題と操作の組で表される．k 個の操作を持つ B Method のモデルは **図 1** (a) のような記述であるが，命題 C_A, P_A, I_A, $Q_{A1}, \ldots, Q_{Ak}$，代入 U_A, $V_{A1}, \ldots, V_{Ak}$ に対して式 (1) に示す組で表される．

$$(C_A, P_A, I_A, U_A, ((Q_{A1}, V_{A1}), \ldots, (Q_{Ak}, V_{Ak}))) \tag{1}$$

ここで，命題 C_A，P_A，I_A はそれぞれパラメータ制約（CONSTRAINTS 節），プロパティ制約（PROPERTIES 節），不変条件（INVARIANT 節）である．代入 U_A は初期化を表し，命題と代入の組のリスト（$(Q_{A1}, V_{A1}), \ldots, (Q_{Ak}, V_{Ak})$）は各操作の代入とその事前条件（PRE 節）を表す．B Method において命題は集合論と述語論理により記述される．また，B Method のモデルでは代入は等価代入文（$X := E$）や ANY 文，IF 文，SELECT 文などで構成され，それぞれの文は同時代入（||）で結合される．すなわち，モデルの操作には代入順序の概念が存在しない．ANY 文，SELECT 文はともに非決定的な代入を行う文である．ANY 文は非決定的選択文といい，非決定的な値を生成し，その値を用いた計算を記述する．また，SELECT 文は条件選択であるが，複数の条件が同時に真になる場合に

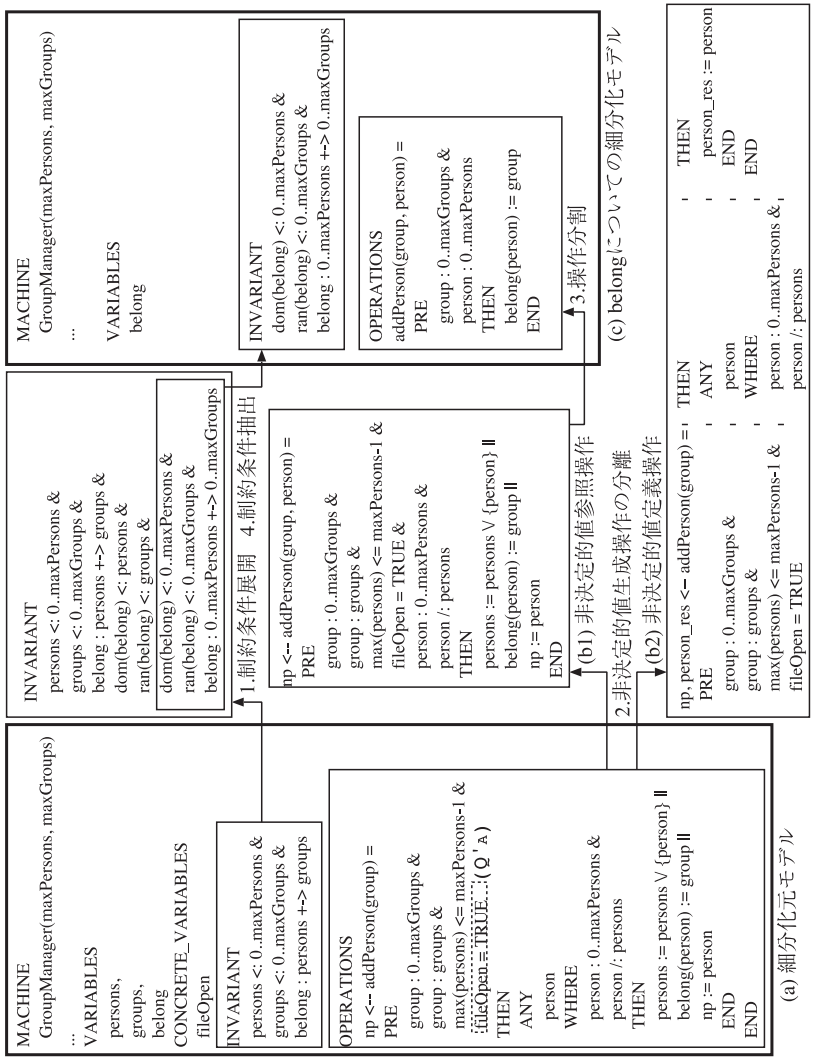


図 1　グループ管理のモデルと細分化例
Fig. 1　Model of group management system, and its sliced models.

はそれらに対応する代入のうち 1 つが非決定的に選択される．B Method の実装は**図 2** (a) のように記述される．実装はモデルと同様に命題と操作の組で表現することができ，k 個の操作を持つ実装はプロパティ制約 P_I，輸入 R_I，不変条件 I_I，代入 U_I, $V_{I1}, \ldots, V_{Ik}$ に対して式 (2) に示す組で表される．

$$(P_I, R_I, I_I, U_I, (V_{I1}, \ldots, V_{Ik})) \tag{2}$$

輸入 R_I は ‘別名．モジュール名’ という書式でモジュールの機能を取り込む記述である．B Method で実装に用いられる基本語彙はきわめて小さく，端末やファイルとの I/O はモジュールを輸入することで行われる．実装における代入は非決定性を含まない代入文で構成され，それぞれの文は逐次代入（;）で結合される．

B Method ではモデルから実装までの記述が整合性を満たすことを定理証明によって保証する．この定理証明の命題を ‘証明責務’ と呼ぶ．証明責務にはモデルの初期化，モデルの操作，実装の操作に関する証明責務があり，それぞれ式 (3)，式 (4)，式 (5) のように表される．B Method では証明責務が真であることを初期化と各操作ごとに定理証明することでソフトウェアの信頼性を保証する．これらの式において代入 V，命題 I に対して $[V]I$ は V が I を満たすための最弱事前条件といい，V を実行後に I を満たすことを示す命題である．最弱事前条件は右結合であり $[V_{Ij}]\neg([V_{Aj}]\neg I_I)$ における () は省略することができる．

$$C_A \wedge P_A \Rightarrow [U_A]I_A \tag{3}$$

$$C_A \wedge P_A \wedge Q_{Aj} \wedge I_A \Rightarrow [V_{Aj}]I_A \tag{4}$$

$$(C_A \wedge P_A \wedge I_A \wedge Q_{Aj}) \wedge (P_I \wedge I_I) \Rightarrow [V_{Ij}]\neg[V_{Aj}]\neg I_I \tag{5}$$

3. 高信頼ソフトウェア部品自動生成

本章では B Method の枠組みを応用した信頼性保証が可能な高信頼ソフトウェア部品を提案し，B Method で構築された既存のソフトウェアから高信頼で細粒度な高信頼ソフトウェア部品群を自動生成する手法の概要を説明する．

3.1 高信頼ソフトウェア部品

3.1.1 定　　義

本稿で提案する高信頼ソフトウェア部品は**図 3** に示す部品のように実装依存モデルと細分化実装によって構成される．さらに，高信頼部品は仕様として細分化モデルを持つ．図 3 の部品リポジトリに示すように各部品はその部品の仕様である細分化モデルごとにリポジトリに格納される．このように複数の部品に対して 1 つの細分化モデルが仕様として与えられるのは，細分化モデルが数学的仕様であり同じ細分化モデルに対してファイルによる実

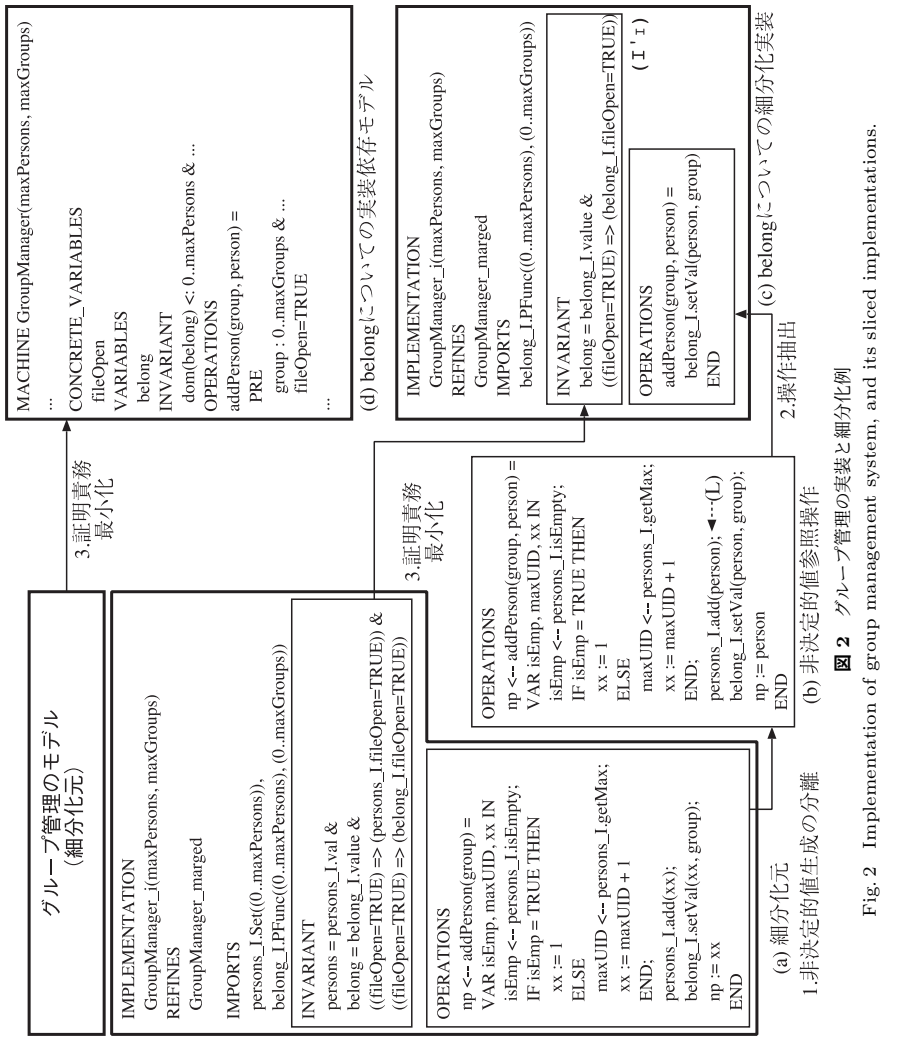


図 2　グループ管理の実装と細分化例
Fig. 2　Implementation of group management system, and its sliced implementations.

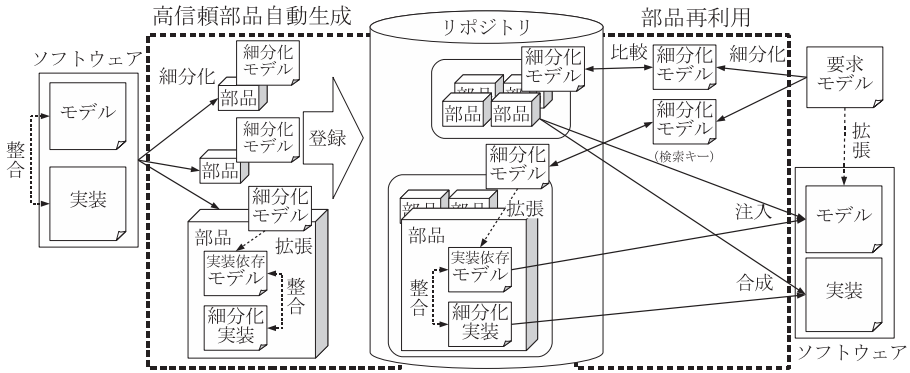


図 3　高信頼部品自動生成と部品再利用の流れ
Fig. 3　Outline of dependable component generation and reuse.

装やデータベースによる実装など複数の実装方法が存在するためである．細分化モデルはパラメータ制約 C'_A，プロパティ制約 P'_A，不変条件 I'_A，事前条件 Q'_A，代入 V'_A の組により式 (6) のように表される．

$$(C'_A, P'_A, I'_A, Q'_A, V'_A) \tag{6}$$

細分化モデルは図 1 (c) のように B Method のモデルに倣って記述される．実装依存モデルは細分化モデルと同様の構造を持ち，パラメータ制約 C'_D，プロパティ制約 P'_D，不変条件 I'_D，事前条件 Q'_D，代入 V'_D の組により式 (7) のように表される．

$$(C'_D, P'_D, I'_D, Q'_D, V'_D) \tag{7}$$

部品において細分化モデルと実装依存モデルの命題は $C'_D \Rightarrow C'_A$，$P'_D \Rightarrow P'_A$，$I'_D \Rightarrow I'_A$，$Q'_D \Rightarrow Q'_A$，という関係を持ち，V'_A と V'_D は等しい．これにより，細分化モデルに実装依存の制約を論理積で追加したものが実装依存モデルであり，本稿ではこのことから図 3 のように実装依存モデルは細分化モデルに対する ‘拡張’ であるとする．細分化モデルは実装依存モデルに拡張されることにより操作を適用できる状態や適用後の状態が限定される．細分化実装はプロパティ制約 P'_I，輸入 R'_I，不変条件 I'_I，代入 V'_{ref}, V'_{def} の組により式 (8) のように表される．

$$(P'_I, R'_I, I'_I, V'_{ref}, V'_{def}) \tag{8}$$

細分化実装は図 2 (c) のように実装に倣って記述される．代入 V'_{ref} は ‘参照部’ といい大域変数を参照し計算結果を局所変数に格納する代入である．また，代入 V'_{def} は ‘代入部’ といい V'_{ref} で計算された局所変数の値を大域変数に格納する代入である．ただし，V'_{ref} で大域変数を変更してはならず，また，V'_{def} で大域変数を参照してはならない．参照部と代入部をこの順で逐次代入により結合した代入 V'_{ref}; V'_{def} が細分化モデルの代入 V'_A に対応する代入である．細分化モデルは同時代入のみで構成されるため，結合順序によらず代入結果はつねに等しい．これに対して，細分化実装は逐次代入のみで構成されるため，結合順序によっては代入結果が異なる．この問題を本稿では ‘副作用’ と呼ぶ．副作用を解消するために本手法では細分化実装の代入を参照部と代入部に分け，結合時には参照部が代入部より前に来るようにすることで大域変数の代入前に代入される値がすべて計算されていることを保証する．たとえば，ある部品 1，2 において，細分化モデルの操作がそれぞれ V_1，V_2，細分化実装の操作がそれぞれ (V_{ref1}, V_{def1})，(V_{ref2}, V_{def2}) であるとき，細分化モデルの操作を同時代入で結合した操作 $V_1 \parallel V_2$ と細分化実装の操作を逐次代入で結合した操作 V_{ref1}; V_{ref2}; V_{def1}; V_{def2} は代入結果が等しい．モデルが初期化と複数の操作を持つのに対して細分化モデルは操作を 1 つだけ持ち，初期化を持たない．本手法では初期化の代入文も一般的な操作と同様に扱い，部品自体は初期化を持たない．そのため，高信頼ソフトウェア部品には初期化をするだけの部品と初期化が不明で操作を行う部品が存在する．高信頼ソフトウェア部品では細分化モデル，細分化実装，実装依存モデルから証明責務を生成し，それが真であることを証明することで部品の信頼性を保証する．式 (9)，式 (10)，式 (11) はそれぞれ初期化を行う細分化モデル，操作を行う細分化モデル，細分化実装に関する証明責務である．

$$C'_A \wedge P'_A \Rightarrow [V'_A] I'_A \tag{9}$$

$$C'_A \wedge P'_A \wedge Q'_A \wedge I'_A \Rightarrow [V'_A] I'_A \tag{10}$$

$$(C'_D \wedge P'_D \wedge I'_D \wedge Q'_D) \wedge (P'_I \wedge I'_I) \Rightarrow [V'_{ref};\ V'_{def}] \neg [V'_A] \neg I'_I \tag{11}$$

3.1.2　高信頼ソフトウェア部品の再利用

我々は B Method を応用した高信頼なソフトウェア部品を用いた自動コード合成フレームワークを提案している[19]．このフレームワークは ‘自動コード合成’ と ‘部品自動生成’ から構成され，本稿で提案する高信頼ソフトウェア部品自動生成手法はこのフレームワークの部品自動生成として利用することを想定したものである．自動コード合成は図 3 右側のように要求する機能を記述した要求モデルを入力として高信頼ソフトウェア部品を再利用することで B Method のソフトウェアを自動で生成する．本稿の主題は高信頼ソフトウェア部品自動生成であるため自動コード合成の詳細は割愛するが，自動コード合成は以下の手順により B Method のモデルを入力として B Method のモデルと実装を出力する．ここでは問題を単純化するた

め，要求モデルの 1 操作のみに注目し，これを要求機能 $(C_K, P_K, I_K, Q_K, V_K)$ としてモデルの代わりに用いる．自動コード合成では要求機能を本稿で提案するモデル細分化で細分化し，得られた k 個の細分化モデル群 $(C'_{Ki}, P'_{Ki}, I'_{Ki}, Q'_{Ki}, V'_{Ki})$ $(1 \leqslant i \leqslant k)$ をそれぞれ検索キーとして部品検索をする．この際，式 (12) を満たす細分化モデル $(C'_A, P'_A, I'_A, Q'_A, V'_A)$ を仕様として持つ部品群を検索することで検索キーと同じ操作を持ち，検索キーの操作が実行可能な条件で操作が実行可能な部品群が得られる．本稿では割愛するが，この際に部品の変数を検索キーの変数に置き換えることと計算量を考慮した一致判定が必要である．

$$(C'_{Ki} \wedge P'_{Ki} \wedge I'_{Ki} \Leftrightarrow C'_A \wedge P'_A \wedge I'_A) \wedge (V'_{Ki} = V'_A) \wedge (Q'_{Ki} \Rightarrow Q'_A) \tag{12}$$

このように数学的に部品を検索し，部品の変数名を要求モデルの変数名に置換することで要求モデルと操作名や変数名の異なる部品を再利用できる．部品検索によって各細分化モデルごとに部品が得られるが，細分化モデルは集合論と述語論理に基づく記述であるため同じ細分化モデルを仕様として持つ部品でも I/O やユーザインタフェースなどの実装が異なる．そのため，各検索キーで検索された部品群からたかだか 1 つの部品を利用者が選択する．部品検索時に検索キーの変数名を用いて部品の変数名を置き換えるが，これにより変数名を細分化モデルの変数名に統一した実装依存モデルを $(C'_{D1}, P'_{D1}, I'_{D1}, Q'_{D1}, V'_{D1}), \ldots, (C'_{Dk}, P'_{Dk}, I'_{Dk}, Q'_{Dk}, V'_{Dk})$，細分化実装を $(P'_{I1}, R'_{I1}, I'_{I1}, V'_{ref1}, V'_{def1}), \ldots, (P'_{Ik}, R'_{Ik}, I'_{Ik}, V'_{refk}, V'_{defk})$ とする．このとき，$C'_{D1} \wedge \cdots \wedge C'_{Dk}$ のように各部品の実装依存モデルの制約を論理積で結合することでモデルの制約を生成し，$P'_{I1} \wedge \cdots \wedge P'_{Ik}$ のように細分化実装の制約を論理積で結合することで実装の制約を生成する．また，$V'_{ref1}; \ldots; V'_{refk}; V'_{def1}; \ldots; V'_{defk}$ のように参照部が代入部の前に実行されるよう逐次代入で細分化実装の代入を結合することで実装の操作を得る．

この自動コード合成は数学的仕様である細分化モデルを用いて部品を検索するため高い健全性を持ち，さらに，高信頼ソフトウェア部品により細分化モデルと細分化実装の整合性を保証できるため，得られたソフトウェアの整合性を保証する際には実装依存モデル間の整合性や実装間の整合性のみを保証すればよく，整合性検証のコストを低減させることができる．このような自動コード合成を運用するためには運用する問題領域ごとに大量の高信頼ソフトウェア部品を持つ部品リポジトリを構築する必要がある．自動コード合成フレームワークに図 3 左側のように本稿で提案する高信頼部品自動生成を応用することにより，B Method で記述された既存のソフトウェアから高信頼ソフトウェア部品を自動生成することが可能になり，より多くの問題領域で迅速に高信頼なソフトウェアを構築することが期待される．

3.2　高信頼ソフトウェア部品自動生成の概要

本節では提案する高信頼ソフトウェアの概要を説明する．高信頼ソフトウェア部品自動生成は図 3 左側のようにモデルと実装からなる B Method のソフトウェアを入力として，それらを細分化した細分化モデル，実装依存モデル，細分化実装からなる高信頼ソフトウェア部品群を出力する．さらに，高信頼なソフトウェアを入力としたとき，出力される部品が高信頼であることを保証する．すなわち，入力ソフトウェアにおいて式 (3)，式 (4)，式 (5) の証明責務が真であるならば，得られる部品群について式 (9)，式 (10)，式 (11) に示した証明責務が真になることを保証する．この細分化手法は図 3 右側の部品再利用においても要求モデルから細分化モデルを生成する際に利用されるため，細分化モデルは実装を参照することなくモデルから生成できる必要がある．そのため，本手法では高信頼部品自動生成をモデルから細分化モデル群を生成する ‘モデル細分化’ と細分化モデルに合わせてソフトウェアから部品を抽出する ‘実装抽出’ に分割する．

モデル細分化は入力モデルの制約条件 C_A，P_A，I_A と細分化する操作 (Q_A, V_A) の組 $(C_A, P_A, I_A, Q_A, V_A)$ を入力として，その操作を細分化した細分化モデル群 $(C'_{A1}, P'_{A1}, I'_{A1}, Q'_{A1}, V'_{A1}), \ldots, (C'_{Al}, P'_{Al}, I'_{Al}, Q'_{Al}, V'_{Al})$ を出力する．これは 1 操作の細分化であるが，モデルの初期化と操作をすべて細分化することで入力モデルの細分化モデル群が得られる．初期化を細分化する際には Q_A を真にして，V_A を初期化 U_A にする．一般的に部品の粒度が小さいと 1 部品の機能が低機能になり 1 部品だけを再利用して得られる利益が小さく，一方で粒度が大きいと 1 部品の機能は高機能になるが再利用可能な場面が限定される．部品の粒度にはこのようなトレードオフがあるが，本稿で提案した高信頼ソフトウェア部品は 3.1.2 項に示した自動合成により低機能な部品群を合成して高度な機能を低コストで得られるため粒度が小さいことによるデメリットが解消される．そのため，高信頼ソフトウェア部品の粒度は細粒度であることが望ましい．提案する部品生成手法では部品の粒度を ‘1 操作における 1 変数の状態変化’ とする．すなわち，細分化元の操作の代入 V_A が変数群 $X_1, \ldots, X_m$ を変化させるとき，細分化モデルの代入群 $V'_{A1}, \ldots, V'_{Al}$ はそれぞれ $X_1, \ldots, X_m$ のいずれか 1 つの V_A の代入後の状態を導出する．この粒度では m 変数の状態を変化させる操作からは m 個の細分化モデルが得られるが，提案手法では条件分岐を条件ごとに分割するなどして部品をさらに細粒度化する．また，3.1.1 項に示したように高信頼ソフトウェア部品は証明責務が真であることで信頼性を保証できるため，証明責務が真であるソフトウェアを細分化して得られる部品群の証明責務は真であることが望ましい．提案するモデル細分化手法は初期化から得られた細分化モデルについて式 (9) が真になることを

保証する．また，操作から得られた細分化モデルの証明責務を真にするためには定理証明と同等の計算量が必要であり，本手法では近似的な計算によりこの計算量を軽減するため，多くの場合に証明責務が真になることを期待できるが，偽になる可能性もある．このため，式 (10) の証明責務が真であることを作業者が証明し，偽である場合には抽出した制約条件に不足があるため，これを修正する．このように自動生成される細分化モデルの無矛盾性を完全に保証することはできないが，多くの場合に無矛盾な細分化モデルが生成されるように細分化手法を構築する．これを保証するために本研究では細分化手順で得られる細分化モデルにおいて証明責務が真になるための条件を証明する．モデル細分化手順は 4 章で説明する．

実装抽出は入力モデルの制約と操作の組 $(C_A, P_A, I_A, Q_A, V_A)$，入力モデルの操作に対する実装操作と実装の制約と輸入の組 (P_I, R_I, I_I, V_I)，細分化モデル $(C'_A, P'_A, I'_A, Q'_A, V'_A)$ を入力として，実装依存モデル $(C'_D, P'_D, I'_D, Q'_D, V'_D)$ と細分化実装 $(P'_I, R'_I, I'_I, V'_{ref}, V'_{def})$ を出力する．ここで，実装依存モデルは細分化モデルの拡張であり，実装依存モデルと細分化実装は式 (11) の証明責務が真になる．この際に問題となるのは ‘細分化モデルの操作と等価な代入の実装操作からの抽出’ と ‘証明責務が真になる記述の細分化’ である．本手法では実装の操作から細分化モデルの操作と等価な代入文を抽出するためにプログラムスライシングを応用する．また，入力された細分化モデルと実装から証明責務を生成し，それが真であることを保つよう最小化し，そこから細分化実装と実装依存モデルの制約条件を得ることで証明責務が真になることを保証する．ただし，証明責務最小化は計算として定理証明を含むため，その作業の一部を人間が行う必要がある．実装抽出手順は 5 章で説明する．

図 1 (a) のモデルと図 2 (a) の実装を入力として，操作 addPerson を変数 belong に注目して提案手法を適用するとモデル細分化により図 1 (c) の細分化モデルが得られ，実装抽出により図 2 (c) の細分化実装と図 2 (d) の実装依存モデルが得られる．この例題における導出手順は 6 章の手法適用例で説明する．

3.3　対象とする問題領域

提案する高信頼ソフトウェア部品とその自動生成手法は運用される問題領域として集合論と述語論理のみで制約や信頼性を記述できる問題領域を想定している．すなわち，ユーザインタフェースや処理時間に対する制約や信頼性などが問題とならない領域を想定している．これは高信頼ソフトウェア部品の信頼性を集合論と述語論理に基づいた証明責務で定義しており，集合論と述語論理で記述できない問題は信頼性を保証できないためである．また，3.1.2 項で紹介した部品再利用において細分化モデルを検索キーとした部品検索は高い健全性を持つが，同様の理由により集合論と述語論理で記述できない要求に対しては部品検索の健全性を保証できない．

3.4　文法に対する制限

本手法では入力されるモデルと実装を以下のように制限する．

(1)　表明付き代入（ASSERT）は利用できない．
(2)　リンク不変条件は ‘抽象変数 =(実装変数の式)’ の形式に限定される．
(3)　操作呼び出しで呼び出される操作は提案手法でそれ以上細分化できない操作でなければならない．
(4)　WHILE ループの中で大域変数への代入を行ってはならない．
(5)　実装の操作において大域変数に代入した後にその大域変数を参照してはならない．
(6)　実装の操作において大域変数への代入時に参照される式は局所変数として格納できるものに限定する．ただし，代入される大域変数自身を参照することは可とする．

(1) の制限は記述の構成要素を限定することで問題を単純化する．ASSERT は証明責務の証明補助に用いる命題であり，これがなくても機能の記述には問題がない．(2) の制限はモデルと実装の大域変数の対応付けを容易にするためである．これによりモデル変数と実装変数の対応が多対多である実装を書けないが，詳細化を十分に行ったリファインメントを入力モデルとすることで 1 対多，あるいは 1 対 1 で記述できると考えられる．(3) の制限は呼び出される操作の細分化を考えないことで問題を単純化するためである．モジュールの細分化により呼び出される操作を細分化できると考えられるが，B Method におけるモジュールの細分化は本研究を応用することでできると考えられる．(4)，(5)，(6) の制限は部品を合成した際の副作用を解消するためである．(4)，(5) の制限は局所変数を活用することで解消できると考えられる．また，(6) の制限はモジュールにおけるデータ表現を制限するが代替モジュールを用意することで解消できる．また，B Method ではモデルに対して段階的詳細化を行うため，モデルと実装のほかにリファインメントが登場するが，本稿の提案手法ではリファインメントの書式を入出力として扱わない．そのため，本研究で部品自動生成への入力となるソフトウェアのモデルは十分に詳細化が行われたリファインメントをモデルとして書き下したものを想定している．モデルとリファインメントの間の証明責務はモデルと実装の間のそれとほぼ同じであるため，本研究を応用してリファインメントを細分化し部品に含めることが考えられる．

4. モデル細分化

4.1 概　　要

モデル細分化ではモデルの制約と操作の組 $(C_A, P_A, I_A, Q_A, V_A)$ を細分化して細分化モデル群 $(C'_{A1}, P'_{A1}, I'_{A1}, Q'_{A1}, V'_{A1}), \ldots, (C'_{Ak}, P'_{Ak}, I'_{Ak}, Q'_{Ak}, V'_{Ak})$ を得る．V_A が初期化であるならば細分化モデルは式 (9) の初期化の整合性を満たし，V_A が操作であるならば細分化モデルは式 (10) の操作の整合性を満たすことが期待される．$V'_{A1}, \ldots, V'_{Ak}$ は 3.2 節で示した粒度で V_A を分割して得られる代入である．また，C'_{Ai}，P'_{Ai}，I'_{Ai}，Q'_{Ai} は V'_{Ai} で用いられる変数間の関係を定めた制約である．以下にモデル細分化手順を示す．

(1) **制約条件展開**：制約条件 C_A，P_A，I_A，Q_A に対して推論を行い，入力モデルでは明示されない変数間の関係を明示した制約条件 C_{rsn}，P_{rsn}，I_{rsn}，Q_{rsn} を得る．制約条件展開は ‘等価な変数の特定’ と ‘制約条件抽出’ において与えられた変数間関係を特定するのに必要となる．詳細は 4.2 節で説明する．

(2) **非決定的値生成の分離**：事前条件と代入の組 (Q_A, V_A) から非決定的値生成操作 (Q_{AnyDef}, V_{AnyDef}) と非決定的値参照操作 (Q_{AnyRef}, V_{AnyRef}) を得る．非決定的値生成操作は V_A が含む ANY 文で局所変数に格納される非決定的な値を生成してそれを戻値として返すだけの操作である．非決定的値参照操作は V_A で ANY 文により生成される値を操作の引数として受け取り，V_A が含む値の生成以外の代入を行う操作である．詳細は 4.3 節で説明する．

(3) **等価な変数の特定**：$x = dom(r)$ における x，r や $Z = X \cup Y$ における Z，X，Y のように等号（=）で関係を定義された変数群を ‘等価な変数’ と定義し，等価な変数の集合群 $\{\{e_{11}, \ldots, e_{1a}\}, \ldots, \{e_{s1}, \ldots, e_{sb}\}\}$ を得る．ここで，e_{ij} は変数であり，任意の i，x，y について e_{ix} と e_{iy} は等価な変数である．制約条件展開によって任意の 2 変数間の関係が展開されているため，‘=’ で関係づけられた式を見つけることは容易である．

(4) **操作分割**：操作の代入 V と等価な変数群 $\{e_1, \ldots, e_a\}$ を入力として細分化された代入群 $V'_1, \ldots, V'_l$ を出力する．ここで，V は非決定的値生成の分離で得られた非決定的値生成操作，あるいは非決定的値参照操作であり，$V'_1, \ldots, V'_l$ は等価な変数群 $\{e_1, \ldots, e_a\}$ に対する代入文を条件分岐の条件ごとに抽出した代入文である．詳細は 4.4 節で説明する．

(5) **制約条件抽出**：展開された制約条件 C_{rsn}，P_{rsn}，I_{rsn}，Q_{rsn} と操作分割で得られた代入 V' を入力として細分化モデルの制約条件 C'，P'，I'，Q' を得る．ここで，C'，P'，I'，Q' は C_{rsn}，P_{rsn}，I_{rsn}，Q_{rsn} が含む V' で用いられる変数間の制約である．詳細は 4.5 節で説明する．

(6) **検証と修正**：得られた C'，P'，I'，Q' および V' から細分化モデルを生成する．V' が初期化から得られた代入であるならば証明責務はつねに真である．V' が操作から得られた代入であるならば証明責務が偽になる場合があるため，式 (10) の証明責務を定理証明器などの補助を受けて利用者が証明する．証明責務が偽であるならば手順 (1) の推論において証明に必要な命題が推論に失敗しているため，それを制約条件に追加する．

V_A を 3.2 節で示した粒度で分割する際，同様の分割が実装においても可能でなければならない．ここで特に問題になるのがモデルにおける非決定的な選択である．モデルの代入文は SELECT 文のような非決定的な条件分岐を含むが，実装では必ずどちらか一方の条件でしか実装されない．そのため，どちらで実装されていてもよいように代入の細分化を行う必要がある．また，3.2 節で定めたように部品の粒度の基準は ‘1 操作における 1 変数の状態変化’ であるが，モデルにおいて同じ内容を保持する式は実装において 1 変数で表現されることが考えられるため，等価な式を構成するモデル変数に対する操作は分割できない．これらのことから，手順 (3) のように ‘等価な変数’ を特定し，等価な変数ごとに操作を分割する．この粒度に沿った操作分割は手順 (4) で行われる．本手法ではさらに細粒度化するために手順 (2) を定めた．ANY 文の値を生成する部分と値を用いて計算する部分は手順 (4) で分割することができないため，手順 (2) で分割することで細粒度化する．手順 (5) は操作で用いられた変数群に関する制約を抽出することで細分化モデルの制約を得る．以降の節では各手順の詳細を説明し，この手順で得られる細分化モデルの信頼性を保証する．

4.2 制約条件展開

制約条件展開は命題 G を入力として，G において暗黙的な変数間関係を明示した命題 G_{rsn} を出力する．制約条件展開は入力モデルのパラメータ制約 C，プロパティ制約 P，不変条件 I，事前条件 Q に対して適用される．任意の変数間関係を推論することは定理証明と同様に計算の停止性を保証できないが，本手法では与えられた命題に対して推論ルールを有限回適用することで，すべての暗黙な関係を推論できる保証はないが有限時間で結果を得る．**表 2** は制約条件展開の推論ルールを B Method の表記で表したものである．‘⇒’ で表したルールは左辺のパターンに対して右辺の式を推論し制約条件に追加する．また，‘⇔’ で表したルールは左辺のパターンに対して右辺の式を，右辺のパターンに対して左辺の式を推

表 2　推論規則一覧（抜粋）
Table 2　List of reasoning rules (extracts).

dom(r) <: X & ran(r) <: Y	⇔	r : X +-> Y
dom(r) = X & ran(r) <: Y	⇔	r : X --> Y
a = b & bRx	⇒	aRx
S <: Y & S ={} & y : Y	⇒	y /: S
r[{x}]={} & r[{x}]<:ran(r)	⇔	x /: dom(r)
r が単射 & y:r[{x}]	⇔	y = r(x)
r が単射 & y/:r[{x}]	⇔	y /= r(x)
r が関数	⇒	r~が単射

論し制約条件に追加する．この表において R は任意の二項演算子を表している．r:X+->Y のように右辺に写像の定義を持つルールは写像の値域と定義域を集合として扱うための推論である．実際にはこれと同様の推論を>+>，>->などの他の写像についても行う．y/:S を右辺に持つルールは集合 S の要素でないことが暗黙的に示された要素 y について，それを明示する推論（集合要素推論）である．x/:dom(r) を右辺に持つルールは定義域として x を含まない写像 r において，x の像が空集合になることを示している（写像集合交換）．写像集合交換と集合要素推論を組み合わせることで，issued:copies+->users，copy/:dom(issued)，x:users という制約条件から x/=issued(copy) という推論が可能になる．

4.3　非決定的値生成の分離

4.1 節に述べたように非決定的値生成の分離では事前条件と代入からなる操作 (Q, V) を入力として，非決定的値生成操作 (Q_{AnyDef}, V_{AnyDef}) と非決定的値参照操作 (Q_{AnyRef}, V_{AnyRef}) を出力することで操作を細粒度化する．分割元の操作を**図 4** (a) のように操作戻値 res，操作引数 $args$，事前条件 Q，制約 W で規定される値を持つ局所変数群 $v1, \ldots, vk$ についての代入 V_{any} で構成される ANY 文，ANY 文以外の代入文 V で表す．このときの非決定的値生成操作と非決定的値参照操作をそれぞれ図 4 (b)，図 4 (c) に示す．非決定的値生成操作は元の操作と同じ事前条件 Q を持ち，V_{any} の代わりに局所変数 $v1, \ldots, vk$ を戻値にする代入を持った ANY 文を V_{AnyDef} とした操作である．この際，局所変数ごとに戻値となる変数を定義する必要があるが，本稿ではこれを局所変数名の末尾に_res を加えたものとする．非決定的値参照操作は事前条件 Q_{AnyRef} として元の事前条件 Q と ANY 文の制約 W の論理積 $Q \wedge W$ を持ち，代入 V_{AnyRef} として ANY 文の代入 V_{any} と代入 V の同時代入 $V_{any} \parallel V$ を持つ．この際，引数として ANY 文で定義された変数群を追加する．

上記の出力において非決定的値生成操作は ANY 文の制約で規定された非決定的な値を生成してそれを戻値とする操作であり，元の操作が含む代入文はいっさい持たない．また，非決定的値参照操作は元の操作では生成された値を引数として受け取り，元の操作が含む代入を行う操作である．そのため，非決定的値生成操作は ANY 文を持ち，非決定的値参照操作は ANY 文を持たない操作となる．非決定的値参照操作の事前条件に ANY 文の WHERE 部を加えることによって，非決定的値生成操作の局所変数と非決定的値参照操作の引数を対応付けている．なお，ANY 文に類似する文法として充足代入文，要素代入文，局所定義文があるが，これらはすべて ANY 文で表現できるため ANY 文に書き換えたうえで分割する．実装抽出では本節で行った分割と同様の分割を行う必要がある．しかし，ANY 文はモデルで用いることはできるが実装で用いることができないため，5.3 節のように実装において非決定的値生成の分離を行う手順を与える．

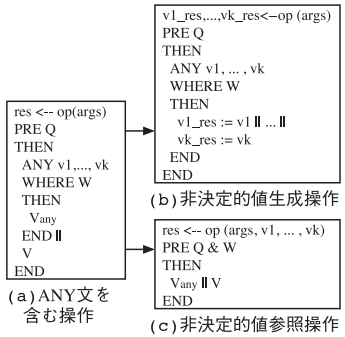


図 4　ANY 文を含む操作の分割
Fig. 4　Slicing of the ANY statement.

4.4　操 作 分 割

操作分割では与えられた代入 V と注目する等価な変数群 $\{e_1, \ldots, e_a\}$ を入力として，細分化された代入 $V'_1, \ldots, V'_l$ を出力する．$V'_1, \ldots, V'_l$ は変数群 $\{e_1, \ldots, e_a\}$ を変化させる代入文を条件分岐の条件ごとに抽出した代入文である．このため，1 つの代入 V から 1 つの等価な変数群 $\{e_1, \ldots, e_a\}$ について操作分割した結果は複数得られる．ただし，2.3 節のモデルの代入文で説明したように条件選択 SELECT は非決定的な記述が可能であり，排他的でない条件 P_1，P_2 について P_1 ならば代入文 S_1，P_2 ならば代入文 S_2 が与えられているとき，実装においては必ずしも P_1 において代入文 S_1 が行われるとは限らない．そのため，本手法では排他的でない条件選択については分割を行わないこととする．本手法では分割

ルールを単純化するために，最も基本的な文法についてのみ分割ルールを用意し，他の文法はこの基本的な文法に書き換えることで分割する．たとえば，`x,y:=E1,E2` のような複数変数への等価代入は `x:=E1||y:=E2` のような同時代入と等価代入に書き換える．また，有限非決定的選択文（CHOICE），IF 条件文，場合分け（CASE）は条件選択（SELECT）に書き換える．以下に操作分割を関数 slice とした操作分割の再帰的定義を表す．ここで，関数 slice は代入 V と注目する変数群 $\{e_1, \ldots, e_a\}$ を受け取り，分割結果を代入文のリストとして返す．以下の slice を用いた表現では引数として注目する変数群を省略する．

等価代入　slice($x := E$) に対して，変数 x が注目する変数群に含まれるならば，結果は $x := E$ である．含まれないならば結果は空リストである．

同時代入　slice(V_1 || ... || V_n) に対して，slice(V_1), ..., slice(V_n) が条件選択を持たないならば結果は slice(V_1) || ... || slice(V_n) である．そうでないならば，slice(V_1), ..., slice(V_n) のうち排他的でない条件を持つものどうしを同時代入で結合した代入文のリストが結果である．ただし，同じ SELECT 文から分割された SELECT 文は同時代入で結合せずに 1 つの SELECT 文にまとめる．これは排他的な条件で発火する代入文はそれぞれ別な部品になることを意味する．

非決定的選択（ANY）　slice(ANY$v_1, \ldots, v_l$ WHERE W THEN V) を考える．W を論理積で分割して同じ変数を含む命題どうしを論理積で結合した命題を $W_1, \ldots, W_k$ とする．$1 \leqslant i \leqslant k$ について V から W_i が含む局所変数群 X_i を含む代入文を抽出し同時代入で結合したものを V_i とする．また，$V_1, \ldots, V_k$ のうち注目する変数群に含まれる変数を変更する代入を $V_{j_1}, \ldots, V_{j_s}$ とする．このとき，出力は ANY $X_{j_1}, \ldots, X_{j_s}$ WHERE $W_{j_1} \wedge \cdots \wedge W_{j_s}$ THEN V_{j_1} || ... || V_{j_s} である．

条件選択　slice(SELECT P_{11} THEN V_{11} ... WHERE P_{1a} THEN V_{1a} ... WHERE P_{s1} THEN V_{s1} ... WHERE P_{sb} THEN V_{sb}) を考える．ここで，任意の i に対して $P_{i1}, P_{i2}, \ldots$ は互いに排他的でない条件であり，$i \neq j$ である j に対して P_{ix}，P_{jy} は互いに排他的である．条件 P_{ix}，P_{iy} が排他的でないとは事前条件 Q に対して $Q \Rightarrow P_{ix} \wedge P_{iy}$ が真であることをいう．$SELECT_i$ を SELECT P_{i1} THEN slice(V_{i1}) WHERE P_{i2} THEN slice(V_{i2}) ... としたとき，条件選択の分割結果は $SELECT_1, \ldots, SELECT_s$ である．ただし，slice(V_{ix}) が空であるならば P_{ix} も SELECT 文から取り除き，すべての遷移条件が取り除かれるならば $SELECT_i$ は出力に含まれない．

B Method のモデルでは条件の評価も同時に行われるため，互いに排他的でない遷移条件 P_1，P_2 とその代入 V_1，V_2 を持つモデルは P_1，P_2 どちらにも属する状態に対して

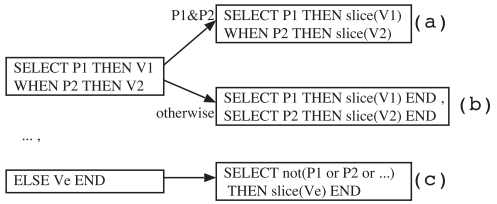


図 5　SELECT 文の分割
Fig. 5　Slicing of the SELECT statement.

V_1，V_2 のどちらの代入を行ってもよいことを表す．すなわち，このような排他的でない遷移条件を持つ場合には遷移条件 P_1 に対する実装が V_1 であるとは限らない．そのため，本手法では排他的でない遷移条件は分割しない．**図 5** に示すように，条件 P_1，P_2 が互いに排他的でないときは条件 P_1，P_2 を分割することはできず，(a) のように 1 つの SELECT 文になる．条件 P_1，P_2 が互いに排他的であるときは (b) のように条件 P_1，P_2 についての SELECT 文がそれぞれ作られる．ELSE 節についてはすべての条件と排他的であるため，図 5 (c) のようにすべての条件の論理和の否定を条件として SELECT 文を作る．

4.5　制約条件抽出

制約条件抽出は命題 G と 4.4 節の分割で得られた代入 V'，および分割元の代入 V を入力として命題 G' を出力する．G は 4.2 節の制約条件展開で展開された命題群 C_{rsn}，P_{rsn}，I_{rsn}，Q_{rsn} であり，命題 G' は V' で用いられる変数間の関係を定義する証明責務が真になる命題である．ここで，関数 $vars(V')$ を代入 V' に含まれる変数群，関数 $chs(G, v)$ を変数群 v とプリミティブな値や型のみで構成される G に含まれる命題とする．また，V' で変化する変数群 Y'，V で変化するが V' で変化しない変数群 Y''，とする．このとき，G が C_{rsn}，P_{rsn}，Q_{rsn} であるとき，$G' = chs(G, vars(V'))$ である．また，G が I_{rsn} であるとき，$G' = chs(G, vars(V') - Y') \wedge chs(G, vars(V') - Y'')$ である．

仮に I_{rsn} に対する出力を $G' = chs(G, vars(V'))$ と定義してしまうと G' に含まれる命題 G_i が Y' と Y'' 間の関係を定義するとき，最弱事前条件 $[V']G_i$ が $[V]G_i$ と異なる命題になるため証明責務が偽になりうる．ただし，初期化の代入中では変数を参照できず $vars(V') - Y'' = vars(V')$ が成り立つため，V が初期化である場合には出力を $G' = chs(G, vars(V'))$ と定義しても証明責務は真である．提案手法では最弱事前条件をなす制約条件において V' で変化する変数 Y' と V で変化するが V' で変化しない変数 Y'' 間

の関係を定義する命題を抽出しないことでこの問題を解決している．たとえば，不変条件として $W \subseteq ran(r)$ を持ち，引数 w に対して $W := W \cup \{r(w)\}$ と $r := r \cup \{w \mapsto f(w)\}$ という 2 つの代入文を持つ操作を変数 W に関して細分化することを考える．このとき，本手法に入力されるモデルの証明責務は真であるため，$W := W \cup \{r(w)\} \,||\, r := r \cup \{w \mapsto f(w)\}$ という代入 V に対して $[V](W \subseteq ran(r))$ が真であることが保証される．操作分割では $W := W \cup \{r(w)\}$ という代入 V' が得られるが，この代入文が含む変数 r は細分化元の V で変化するため，$W \subseteq ran(r)$ は制約条件抽出において抽出しない．仮にこの制約を抽出すると最弱事前条件 $[W := W \cup \{r(w)\}]\,(W \subseteq ran(r))$ が真になるとは限らないため部品の信頼性を保証できない．

入力される制約条件として 4.2 節の制約条件展開が適用された制約条件を与えているが，これは変数間の関係の取りこぼしを解消するためである．入力モデルの制約条件にはすべての変数間の関係が明示されてはおらず，命題から推論される暗黙的な関係が存在する．このような制約条件に対して上記のような命題の抽出を行うと変数間の関係に取りこぼしが生じるが，4.2 節の制約条件展開で任意の変数間の関係が明示することによりこれを解消する．

4.6　モデル細分化手法の信頼性保証

高信頼な入力モデルを細分化したとき，つねに高信頼な細分化モデルが得られるならばその細分化手法の信頼性は高いといえる．3.1.1 項に示したように高信頼ソフトウェア部品の信頼性は部品の証明責務が真であることで保証される．本節では整合性が保証されたモデル，すなわち，式 (3)，式 (4) が真であるモデルに対してモデル細分化を適用して得られる細分化モデルにおいて証明責務が真になることを証明し，これにより提案したモデル細分化手法の信頼性を保証する．細分化モデルの証明責務は式 (9) に示した初期化に関する証明責務，および，式 (10) に示した操作に関する証明責務である．4.6.1 項では入力モデルの初期化を細分化して得られる細分化モデルにおいて初期化に関する証明責務が真になることを示す．また，4.6.2 項では入力モデルの操作を細分化して得られる細分化モデルにおいて操作に関する証明責務が真になることを示す．ただし，4.6.2 項に示すように操作に関する証明責務が真になるか否かは制約条件展開に依存するため，得られた細分化モデルに対して人間が証明責務を証明しなければならない．この信頼性と人間の負担については 7.2.1 項で説明する．

4.6.1　初期化の整合性保証

2.3 節の式 (3) にモデルにおける初期化の証明責務を示した．この式においてパラメータ制約 C，プロパティ制約 P には不変条件とは異なりモデル変数が登場せず，パラメータとプロパティがどのような型に属するといった制約のみが記述される．そのため，$C_A \wedge P_A$ が偽であるときを考慮することは重要ではない．この初期化の整合性保証では $[U_A]I_A$ のみに注目する．細分化モデルの初期化と不変条件をそれぞれ U'_A，I'_A としたとき，初期化に関するモデル細分化手法の信頼性を表す命題を式 (13) に定義する．式 (13) が成り立つことを証明 1 に示す．

$$[U_A]I_A \Rightarrow [U'_A]I'_A \tag{13}$$

証明 1　式 (13) の証明

任意の命題 P，任意の変数群 v について $P \Rightarrow chs(P, v)$ であることは明らかである．$[U_A]I_A$ は不変条件 I_A に対して代入文 U_A による変数置換を行った命題である．そのため，任意の変数群 v について式 (14) が成り立つ．ここで注目する等価な変数群を ss としたとき $U'_A = slice(U_A, ss)$ であるため，任意の変数群 v を $vars(slice(U_A, ss))$ とすることで制約条件抽出の定義から式 (15) が得られる．式 (15) の右辺の U_A のうち，実際に変数置換を行っているのは I'_A が含む大域変数群に対して代入を行っているものだけである．ここで，$slice(U_A, ss)$ が置換する変数群 ss と I'_A が含む大域変数群 $vars(slice(U_A, ss))$ が等しいならば式 (16) が成り立つ．初期化では大域変数は代入文の左オペランドにしか現れないため，U'_A が含む大域変数と U'_A が置換する変数群はつねに等しい．すなわち，$vars(slice(U_A, ss)) = ss$ が成り立つ．よって，式 (16) が成り立つ．以上により，式 (13) が保証される．

$$[U_A]I_A \Rightarrow [U_A]chs(I_A, v) \tag{14}$$

$$\Rightarrow [U_A]chs(I_A, vars(slice(U_A, ss))) = [U_A]I'_A \tag{15}$$

$$\Rightarrow [slice(U_A, ss)]chs(I_A, vars(slice(U_A, ss))) = [U'_A]I'_A \tag{16}$$

4.6.2　操作の整合性保証

2.3 節の式 (4) にモデルの操作における証明責務を示した．この式のパラメータ制約 C とプロパティ制約 P を 4.6.1 項と同様の理由で省略し，$Q_A \wedge I_A \Rightarrow [V_A]I_A$ に注目する．細分化モデルの事前条件，不変条件，代入をそれぞれ Q'_A，I'_A，V'_A としたとき，操作に関するモデル細分化手法の信頼性を表す命題を式 (17) に定義する．

$$(Q_A \wedge I_A \Rightarrow [V_A]I_A) \Rightarrow (Q'_A \wedge I'_A \Rightarrow [V'_A]I'_A) \tag{17}$$

式 (17) が成り立つことを証明 2 に示す．この証明では任意の変数間の関係を命題 P から推論する推論器 $rsn(P)$ を想定する．4.2 節の制約条件展開がこの推論器の役割を担うが，実際には理想的な推論器を実装することは不可能である．そのため，操作の整合性保証は制約条件展開に依存する．

証明 2　式 (17) の証明

任意の変数間の関係を命題 P から推論する推論器 $rsn(P)$ を想定する．この推論器を用いたとき，$X \Rightarrow Y$ が真であり，ある変数群 v' に関して $X' = chs(rsn(X), v')$, $Y' = chs(rsn(Y), v')$ であるならば，$(X \Rightarrow Y) \Rightarrow (X' \Rightarrow Y')$ は真である．これは，推論器 rsn によって変数群 v' 間の関係が推論されたことで，v' に関する命題 Y' を証明するのに必要な仮定 X' が得られるためである．ここで，X を $Q_A \wedge I_A$, Y を $[V_A]I_A$ に置き換えることでは目的とする命題にはならないことに注意する．なぜなら，$[V'_A]I'_A$ が $[V_A]rsn(I_A)$ に存在しない命題を持つ場合には $Y' = chs(rsn(Y_A), v')$ の前提が崩れるからである．$[V'_A]I'_A$ が $[V_A]rsn(I_A)$ には存在しない命題を持つ場合とは，V_A では $y := y'$ と変化した変数 y が V'_A では $y := y$ となり変化せず，I'_A が変数 y を含む命題を持つ場合である．4.5 節の制約条件抽出において V_A では変化するが V'_A では変化しない変数を v' から除くことで $[V'_A]chs(rsn(I), v')$ が $[V_A]rsn(I_A)$ には存在しない命題を持たないようにしている．これにより，$Y' = chs(rsn(Y), v')$ の前提が成り立ち式 (17) が保証される．

5. 実装抽出

5.1 概　　要

実装抽出は細分化モデルに合わせて実装から細分化実装と実装依存モデルを抽出する．ただし，実装抽出は操作ごとに行われるため，式 (1) のような入力モデルと式 (2) のような入力実装から各制約条件と注目する操作のみを抽出した組 $(C_A, P_A, I_A, Q_A, V_A), (P_I, R_I, I_I, V_I)$ を入力モデルと入力実装の代りに用いる．実装抽出はこの組と細分化モデル $(C'_A, P'_A, I'_A, Q'_A, V'_A)$ を入力として，実装依存モデル $(C'_D, P'_D, I'_D, Q'_D, V'_D)$ と細分化実装 $(P'_I, R'_I, I'_I, V'_{ref}, V'_{def})$ を出力する．ただし，入力モデルと実装において式 (5) の実装の操作に関する証明責務が真である．出力される実装依存モデルにおいて V'_D は V'_A そのものであり，C'_D，P'_D，I'_D，Q'_D は細分化モデルの制約と実装依存の制約の論理積である．また，実装抽出で出力される実装依存モデルと細分化実装は式 (11) の証明責務が真である．これにより部品の信頼性が保証される．また，実装依存モデルが細分化モデルの拡張となるため部品が仕様を満たすことが保証される．以下に実装抽出手順を示す．

(1)　**大域変数の対応付け**：モデルに合わせて実装を抽出するためにモデルの大域変数に対応する実装の大域変数を抽出する．詳細は 5.2 節で説明する．

(2)　**非決定的値生成の分離**：モデル細分化の ‘非決定的値生成の分離’ で得られる非決定的値生成操作，非決定的値参照操作に対応するように実装の操作を分割する．詳細は 5.3 節で説明する．

(3)　**操作抽出**：2.2 節で紹介した Weiser の手法を応用して実装操作から細分化モデルの操作と等価な操作を抽出する．詳細は 5.4 節で説明する．

(4)　**副作用解消**：実装の代入を大域変数を参照する参照部と大域変数を変更する代入部に分割することで副作用を解消する．詳細は 5.5 節で説明する．

(5)　**証明責務最小化**：副作用解消で得られた細分化実装の操作に対して式 (11) が真になる制約条件を求める．証明責務最小化で得られた制約条件から細分化実装の制約条件と実装依存モデルの制約条件を得る．詳細は 5.6 節で説明する．

証明責務が真になる出力を得るため，本研究では入力された記述から証明責務を生成し，その証明責務が真になるよう証明責務を最小化し，その証明責務から細分化実装と実装依存モデルを生成することを考えた．これは，手順 (5) によって証明責務のゴールを真にするような仮定を得ることで行う．そのため，ゴールを構成する細分化実装の代入 V'_{ref}; V'_{def} を先に定める必要がある．モデル細分化の粒度は ‘1 操作における 1 変数の状態変化’ であるため，細分化モデルで変化する変数に対応する実装変数 y としたとき，細分化実装の代入は実装操作末尾における変数 y の状態を計算するのに必要なプログラムであることが分かる．これを抽出するのが手順 (1) と手順 (3) である．ただし，モデル細分化では細粒度化のために非決定的値生成の分離と SELECT 文の分割を行ったため，手順 (2) で実装操作についても非決定的値生成の分離を行うとともに，操作抽出時に細分化モデルの制御構造に合わせて実装の制御構造を抽出する．以下の節では実装抽出の各手順を説明する．

5.2 大域変数の対応付け

大域変数の対応付けはモデルの大域変数 x と実装を入力として実装の変数群 $Y = \{y_1, \ldots, y_k\}$ を出力する．ここで，実装の変数群 Y はモデル変数 x を実装する変数群である．B Method ではモデルの変数は抽象変数と呼ばれ実装で扱えないため，実装では実装変数を定義して実装変数と抽象変数の関係を不変条件に記述する．このような不変条件をリンク不変条件と呼ぶ．本稿では 3.4 節に示したようにリンク不変条件を ‘抽象変数 = (実装変数の式)’ の形に制限している．そのため，変数群 Y とプリミティブな値と型のみで構成された式を $E(Y)$ としたとき，抽象変数 x に関するリンク不変条件は $x = E(Y)$ と表され，これにより実装においてモデル変数 x の代わりに用いられる変数群 Y が決定する．以降では変数群 Y をモデル変数 x のリンク変数と呼ぶ．

5.3　非決定的値生成の分離

5.3.1　概　　要

実装における非決定的値生成の分離はモデル細分化における非決定的値生成操作の代入 $V_{AnyDefA}$ と非決定的値参照操作の代入 $V_{AnyRefA}$，および，実装の代入 V_I を入力として実装における非決定的値生成操作 $V_{AnyDefI}$ と非決定的値参照操作 $V_{AnyRefI}$ を出力する．ここで $V_{AnyDefI}$，$V_{AnyRefI}$ はそれぞれ $V_{AnyDefA}$，$V_{AnyRefA}$ に対応する実装の操作である．この分割ではモデルの ANY 文で生成される値を格納する局所変数 v に対応する式 E を実装の操作から特定し，その式 E の値を生成する操作 $V_{AnyDefI}$ と式 E を用いて代入を行う操作 $V_{AnyRefI}$ に分割する．モデルと実装間の局所変数の対応づけはモデルにおける局所変数の参照の仕方に合わせて実装の式を抽出することで行う．モデルの局所変数に対応する実装の式が特定できたら，非決定的値生成操作と非決定的値参照操作を構築する．非決定的値生成操作の構築にはプログラムスライシングを用いる．また，非決定的値参照操作では局所変数に対応する式の値を操作引数として受け取り，局所変数に対応する式を引数で置換する．

5.3.2　局所変数に対応する式の特定

細分化モデルにおいて生成される値が局所変数 v に格納されるとする．局所変数に対応する式の特定では局所変数 v に対応する実装における式 E を特定する．本手法では大域変数および戻値への代入文を手がかりに局所変数 v に対応する式 E を特定する．これは ANY 文から局所変数 v を用いた代入文を抽出し，それと同様の代入を行う代入文を実装から特定することで行う．

細分化元のモデルにおいて 1 大域変数への代入文が 1 操作に 1 つだけならばこの代入文の特定は容易である．しかし，実際にはモデルと実装は条件分岐によって 1 大域変数に対して複数の代入文が記述されるため，モデルの代入文に対応する実装の代入文を特定するためには実装の制御構造がそれぞれモデルのどの制御構造に対応するかを特定しなければならない．これは実装の条件式をリンク不変条件から抽象変数に書き換えたうえで，モデルの条件式と比較することで行う．特定されたモデルと実装の代入文はそれぞれモデルの大域変数への代入文とそのリンク変数への代入文となっている．ここで，この 2 つの代入文は等価な代入文であるため，代入文それぞれの右オペランドを等号で結んだ式が成り立つ．この式を変形してモデルの局所変数 v に対して $v = E$ の形に等式を変形することで局所変数 v の値に対応する式 E を特定する．

5.3.3　分割された操作の構築

モデルの局所変数の値に対応する式 E を特定したら，‘式 E を決定し，それを戻値として返す非決定的値生成操作’ と ‘式 E の値を引数として受け取り，それを用いて代入を行う非決定的値参照操作’ を構築する．以下に非決定的値生成操作の定義を示す．

戻値，引数：モデル細分化で構築した非決定的値生成操作と同一．

操作：局所変数の値を返す戻値の変数名を `v_res`，Weiser の手法で得られる式 E の値を決定するのに必要な代入文を V_E とする．このとき，代入文 V_E の末尾に `v_res:=`E という代入文を加えたものを操作とする．

また，以下に非決定的値参照操作の定義を示す．

戻値，引数：モデル細分化で構築した非決定的値参照操作と同一．

操作：局所変数の値を持つ引数の変数名を v とする．このとき，前節の ‘モデルの代入文に対応する実装の代入文’ において式 E を v で置き換えたものを操作とする．

この非決定的値生成操作は計算に寄与しない局所変数の値を決定する代入文を含むが，操作抽出で Weiser の手法を適用することで計算に寄与しない代入文は取り除かれる．

5.4　操 作 抽 出

操作抽出は細分化モデルの代入 V'_A と実装の代入 V を入力として V'_A と等価な実装の代入 V' を出力する．ここで，V'_A が非決定的値生成操作を細分化して得られる代入ならば V は実装の非決定的値生成操作 $V_{AnyDefI}$ であり，V'_A が非決定的値参照操作を細分化して得られる代入ならば V は実装の非決定的値参照操作 $V_{AnyRefI}$ である．この抽出にはプログラムスライシング手法である Weiser の手法を用いる．モデル細分化では条件分岐を分割したが，Weiser の手法は変数を参照する際にはつねに値が定まるようスライシングを行うため，モデル細分化では抽出しなかった遷移条件や代入が Weiser の手法の結果に含まれる．本手法ではこのような細分化モデルにあてはまらない実装を抽出しないようにするため，モデル細分化で抽出した遷移条件に対する実装の抽出（制御構造の抽出）とモデル細分化で抽出しなかった代入の特定（抽出しない文の特定）を行う．この結果に対してプログラムスライシングを適用することで細分化モデルの代入 V'_A に対する実装を V から抽出する．以下の項では ‘制御構造の抽出’，‘抽出しない文の特定’，‘プログラムスライシングの適用’ について説明する．

5.4.1　制御構造の抽出

制御構造の抽出は与えられた実装の IF 文と細分化モデルにおける遷移先が空でない遷移条件群 $P_1, \ldots, P_k$ を入力として，$P_1, \ldots, P_k$ に対応する遷移を実装した IF 文を出力する．これは，入力された IF 文の条件 $P_1, \ldots, P_k$ で実行されないブロックを空にすることで行う．入力された IF 文の条件式 P，THEN 部を条件式 P の制御ブロック，ELSE 部を条件

式 $\neg P$ の制御ブロックと呼ぶ．$(P_1 \vee \cdots \vee P_k) \Rightarrow \neg P$ であるならば IF 文の条件式を $\neg P$ にし，ELSE 部を THEN 部にして，ELSE 部を取り除いた IF 文を結果とする．これは細分化モデルで空でないブロックに到達する条件において到達不可能な制御ブロックを実装操作から取り除くことを意味する．

5.4.2　抽出しない文の特定

細分化モデルの代入 V'_A と V'_A に対応する細分化実装 V'_I を考える．V'_A で変化する変数群を X_A，V'_I における X_A のリンク変数を X_I とする．また，V'_A で変化しない変数群を Y_A，V'_I における Y_A のリンク変数を Y_I とする．このとき，細分化モデルの代入 V'_A と細分化実装の代入 V'_I が対応するためには，V'_A における X_A の変化と同様に V'_I において X_I が変化することが重要であるが，V'_A において Y_A が変化しないのと同様に V'_I において Y_I が変化しないことも重要である．このため，Y_I は‘変化してはならない大域変数（代入禁止実装大域変数）’であり，Y_I を変化させないよう細分化実装の操作 V'_I を生成しなければならない．これは代入禁止実装大域変数群 Y_I への代入文 f を 5.4.3 項のプログラムスライシングで抽出しないことで実現する．また，代入文 f が操作呼び出しである場合，1 代入文で複数の値が変更する場合がある．このように 1 代入文で代入禁止実装大域変数 Y_I とその他の変数 v が同時に変化するとき，f の抽出を抑制すると変数 v が未定義になる．そのため，このような未定義の変数を参照する文もプログラムスライシングにおける抽出を抑制する必要がある．代入文 f において値が未定義である変数 $U(f)$，およびプログラムスライシングにおいて抽出しない文の集合 L としたとき，それらは以下のように計算される．

抽出しない文の集合 L：代入文 f の直前が条件分岐である場合には直前に行われる代入は複数ありうるが，これらの集合を $prev(f)$ とする．代入文 f が代入禁止実装変数群 Y_I を定義する場合，または f が未定義な変数群 $\bigcap_{g \in prev(f)} U(g)$ を参照する場合は f を L に追加する．ただし，f が変数群 X_I に対する代入である場合は追加しない．これにより，細分化モデルで変化する変数が細分化実装においても変化する．

未定義変数群 $U(f)$：$U(f)$ は $\bigcap_{g \in prev(f)} U(g)$ を引き継ぐ．これにより，条件分岐のいずれかによって値が定まるならば未定義変数群から取り除かれる．f が L に含まれるならば，f で定義される変数群を $U(f)$ に加える．そうでないならば f で定義される変数群を $U(f)$ から取り除く．

5.4.3　プログラムスライシングの適用

プログラムスライシングとして Weiser の手法を用いる．ただし，本手法では未定義大域変数を考慮して Weiser の手法を適用する必要があることを 5.4.2 項に示した．そのため，Weiser の手法に対して‘5.4.2 項で求めた抽出しない文の集合 L に含まれる文は抽出される文 S_C に加えない’という拡張をする．拡張した Weiser の手法への入力は Weiser の手法における注目する文を実装操作末尾の代入文，Weiser の手法における注目する変数群を細分化モデルで変化する変数群のリンク変数とする．これにより細分化実装の操作が得られる．

5.5　副作用解消

副作用解消は代入 V を入力として，V の参照部 V_{ref} と V の代入部 V_{def} を出力する．ここで，V の代入結果と V_{ref}；V_{def} の代入結果は等しい．3.1.1 項の細分化実装の定義で述べたように，細分化実装の代入は副作用を解消するために参照部と代入部に分割される．副作用解消手順を下記の箇条書きに示す．与えられた代入を参照部と代入部に分割するためには大域変数への代入文での大域変数の参照を回避しなければならない．そのため，手順 (2) のように大域変数 v に格納される値 E を局所変数 X に代入し，v には X を代入する．ここで，局所変数 X の計算が参照部であり，X を大域変数 v に代入するのが代入部である．モジュールの操作呼び出しにより大域変数が変化する場合には大域変数に代入される値を直接知ることができないため，手順 (3) のように参照部で操作呼び出しのパラメータを局所変数群 $X_1, \ldots, X_n$ に代入し，代入部ではそれを用いてモジュールの操作を呼び出す．参照部は入力された代入の制御構造をそのまま維持するが，この制御構造において大域変数 v への代入が生じる条件 p と生じない条件 $\neg p$ がある場合，条件 $\neg p$ に対して参照部は局所変数 X に値を代入しないため，代入部において X の値が未定義になる．そのため，手順 (1) のように代入部に参照部と同様の制御構造を作り，未定義な局所変数を参照することを回避する．

(1)　IF 文 f の条件式それぞれに対して新たな局所変数群を宣言し，これに条件式の値を代入する．この局所変数群を参照することで条件分岐を代入部に複製する．すなわち，条件式が p で，THEN 部 V_t，ELSE 部 V_e である IF 文 f に対して局所変数 $X_p := p$ を参照部で定義し，条件式が X_p，THEN 部と ELSE 部が空である IF 文 f' を作る．ただし，IF 文 f の THEN 部 V_t，ELSE 部 V_e が IF 文を持つ場合，それらに対して再帰的に IF 文の複製を行い，複製された IF 文を f' の THEN 部，ELSE 部に格納する．

(2)　大域変数 v へ代入を行う文が条件式 p の制御ブロックに格納された等価代入文 $v := E$ であるならば，新たな局所変数（ここでは X とする）を宣言し，大域変数への代入文を $X := E$ に置き換える．置き換えて得られた操作を参照部 V_{ref} とする．また，代入部 V_{def} の条件式 p の制御ブロックに代入文 $v := X$ を格納する．代入文 $v := X$

は大域変数を参照しないため，制御ブロック内であれば他の大域変数への代入文との代入順序は自由である．

(3)　条件式 p の制御ブロックでモジュールの操作 $v.op(arg_1, \ldots, arg_n)$ を呼び出すことで大域変数への代入を行う場合，新たな局所変数（ここでは $X_1, \ldots, X_n$ とする）を宣言し，操作呼び出しを $X_1 := arg_1; \ldots; X_n := arg_n$ に置き換える．置き換えて得られた操作を参照部とする．また，代入部の条件式 p の制御ブロックに $v.op(X_1, \ldots, X_n)$ を格納する．

5.6　証明責務最小化

証明責務最小化はモデル，実装，細分化モデル，および，それらから 5.5 節の副作用解消で得られた操作 $V_I' = V_{ref}'; V_{def}'$ を入力として実装依存モデルの制約 C_D'，P_D'，I_D'，Q_D' と細分化実装の制約 P_I'，I_I' を出力する．出力において C_D'，P_D'，I_D'，Q_D' は細分化モデルの制約 C_A'，P_A'，I_A'，Q_A' に実装依存の制約を論理積で追加したものであり，実装依存モデルと細分化実装から得られる式 (5) の証明責務は真になる．このような出力を得るため，証明責務最小化では入力を証明責務に変換したうえで，その証明責務が真であることを保つように証明責務を最小化する．これを実装依存モデルと細分化実装における証明責務とし，証明責務から実装依存モデルと細分化実装を生成することで，出力において証明責務が真になることを保証する．証明責務の最小化は式 (11) の細分化実装の証明責務のゴールを定め，そのゴールが真になるように仮定を削減することで行う．このため，ゴールにおいて未知である細分化実装の不変条件を仮に I_I'' と定め，細分化実装の証明責務のゴールと元の実装の証明責務の仮定を持った式 (18) のような命題をつくり，その命題において命題の証明に寄与しない仮定を削減することで実装依存モデルと細分化実装における証明責務の要素を得る．

$$(C_A \wedge P_A \wedge I_A \wedge Q_A) \wedge (P_I \wedge I_I) \Rightarrow [V_I']\neg[V_A']\neg I_I'' \tag{18}$$

証明責務最小化手順を以下に示す．

(1)　細分化実装の操作 V_I' に現れる変数，細分化モデルに現れる変数，および，プリミティブな値や型のみで構成された実装の不変条件を仮の不変条件 I_I'' とする．

(2)　式 (11) に対して仮定削減を行う．仮定削減後の仮定が含む不変条件を I' とする．

(3)　I_I'' を $I' \wedge I_I''$ で置き換え，I' が変化しなくなるまで手順 (2)，手順 (3) を繰り返す．

(4)　収束した仮定削減結果の仮定の C_A，P_A，I_A，Q_A と細分化モデルの制約の論理積から実装依存モデルのパラメタ制約 C_D'，プロパティ制約 P_D'，不変条件 I_D'，事前条件 Q_D' を得る．また，同様に仮定削減結果の P_I から細分化実装のプロパティ制約 P_I'，I_I'' から不変条件 I_I' を得る．

上記手順において，仮定削減は定理証明の過程で証明に寄与しない仮定を削減することで行う．そのため，仮定削減は機械証明の補助を受けて人手によって行われる．この定理証明で対象とする命題は細分化元ソフトウェアにおける証明責務に似た命題であり，細分化元ソフトウェアで証明を行った作業者であれば容易に証明することができる．

6.　手法適用例

本章では例として簡単なグループ管理を行うソフトウェアに対して提案手法を適用する．このグループ管理ソフトウェアはグループ（group）と利用者（person）および，どの利用者がどのグループに属するか（belong）という情報を保持する．利用者を登録する操作 addPerson のモデルを図 1 (a) に，実装を図 2 (a) に示す．操作 addPerson は未登録の利用者 ID を新規に発行し，その利用者を登録と同時に与えられたグループに所属させる．本章ではこの操作を変数 belong について細分化して ‘利用者に対して所属グループを設定する部品’ を生成する．

本章の例では非決定的値参照操作についてのみ細分化を行い，非決定的値生成操作に関する細分化は割愛する．また，実際の部品自動生成では他の変数についても同様に細分化を行う．操作 addPerson を非決定的値生成操作と非決定的値参照操作に分割し，belong 以外の変数に関しても細分化すると全部で 4 つの部品が得られる．これらは ‘利用者に対して所属グループを設定する部品’，‘利用者を登録する部品’，‘未使用の利用者 ID を返す部品’，‘与えられた値を返す部品’ である．操作をモデル細分化して得られる細分化モデルの証明責務が真になるか否かは 4.2 節の制約条件展開に依存するが，この例では表 2 の推論規則によってすべての部品において証明責務が真である細分化モデルが得られた．また，証明責務最小化で得られる制約はいずれの部品でもモデルにおける *fileOpen* に関する事前条件と実装変数のリンク不変条件のみであり，証明責務最小化は容易である．これにより得られる部品は図書の貸し出しシステムなどに再利用することができる．この場合，person が蔵書の集合を表し，group が図書館利用者を表す．このとき，belong は蔵書から図書館利用者への部分関数となり，図書の貸し出しを表す．

6.1　モデル細分化

4.2 節に示した制約条件展開を適用することで，図 1 の矢印 1 が示すように暗黙的な不変条件が明示される．図 1 (a) の操作に 4.3 節で示した非決定的値生成の分離を適用すると非決定的値参照操作として図 1 (b1) が，非決定的値生成操作として図 1 (b2) が得られる．

大域変数 belong と等価な変数はこのモデルに存在しないため，図 1 (b1) に対して 4.4 節の操作分割を適用すると図 1 (c) の操作のように代入文 `belong(person):=group` が得られる．この操作に登場する変数は belong，person，group の 3 つなので，この 3 変数とプリミティブな値や型のみで構成された事前条件と不変条件を制約条件抽出で抽出する．この結果，図 1 (c) のような細分化モデルが得られる．

6.2 実装抽出

大域変数の対応付けによって persons に `persons_I.val` が，belong に `belong_I.value` が対応付けられる．非決定的値生成の分離を図 2 (a) の実装操作と図 1 (b1) のモデルにおける非決定的値参照操作に適用すると図 2 (b) の非決定的値参照操作が得られる．実装操作で呼び出される `belong_I.setVal(xx, group)` は輸入されるモジュール PFunc において `belong_I.value(xx):=group` と定義されるため，モデルの操作 `belong(person):=group` に対応する．この対応から，実装の局所変数 xx がモデルの局所変数 person の値を持つことが分かる．非決定的値参照操作を作るため，モデルの局所変数 person を引数に加え，大域変数への代入において xx を person に置き換える．次に操作抽出を適用して図 2 (b) の操作から belong のリンク変数について操作抽出を行う．モデルの大域変数 persons は注目する細分化モデルに含まれないため，persons のリンク変数に対する代入文を抽出しないよう，5.4.2 項に示したように抽出しない文を特定する．この例では図 2 (b) に L で示された文が抽出しない文である．ここで `belong_I.value` について 5.4.3 項のように拡張した Weiser の手法を適用すると図 2 (c) の操作のように `belong_I.setVal` の 1 行のみが抽出される．この例では大域変数への代入のみが行われるため，副作用解消の結果は代入部のみが存在する操作となる．証明責務最小化を行うために仮の不変条件と操作の事前条件から仮のゴールを定める．仮の不変条件は操作で用いる変数とプリミティブな値や型で構成された不変条件であるが，ここでは `belong = belong_I.value` がそれにあたる．また，実装操作で呼び出す `belong_I.setVal` は輸入されるモジュール PFunc において事前条件として `belong_I.file = TRUE` を持つ．このゴールが真になるためには図 2 (c) の I_I' の実装の不変条件と図 1 (a) の Q_A' のモデルの事前条件が必要である．以上により，図 2 (c) のように細分化実装が得られる．また，図 2 (d) のように命題 `fileOpen=TRUE` を図 1 (c) の細分化モデルの事前条件に論理積で加えた実装依存モデルが得られる．

7. 考　　察

本章では提案手法により生成される高信頼ソフトウェア部品の再利用性と信頼性を定性的に評価し，その有用性を考察する．

7.1 再利用性

本節ではソフトウェア部品の再利用性について部品の粒度と機能性に注目して定性的に評価する．本手法では 7.1.1 項のように部品に形式的仕様を付加することで部品の粒度が細かいほど再利用性が向上するようにし，また，7.1.2 項のように部品の粒度がなるべく細かくなるよう細分化手法を工夫した．これにより，本手法で得られる高信頼部品は高い再利用性を持つ．

7.1.1 部品の粒度と再利用性

一般的に部品の粒度が細かいほど低機能だがより多くのソフトウェア開発に利用できる．また，部品の粒度が粗いほど高機能だが部品を適用できる場面が限られる．本手法では部品の粒度をできる限り細かくすることで部品の適用性を向上させているが，部品単体の機能が低いため 1 つの部品を再利用して得られるメリットは小さい．本手法と同様に細粒度リポジトリを扱う Sapid ではソフトウェア部品に細分化元ソフトウェアの仕様を関連付けることで部品の組合せによる高機能の実現を容易にした．しかし，Sapid は C 言語と自然言語による仕様を対象とした手法であったため計算機による仕様の解釈が難しく，部品の組合せは人間が行わなければならなかった．Koala などの形式的仕様を持つソフトウェア部品は計算機による仕様解釈が容易であるため機械的な部品の組合せが可能である．しかし，これらの手法も機能の表現に部品名や機能名などの固有名詞を用いるため，どの機能をどのように組み合わせれば要求する機能を実現するかを人間が与えなければならない．これに対して本手法ではソフトウェアと部品の仕様を固有名詞によらずに数学的に表現することで，要求仕様を実現する部品の組合せを機械的に定めることができる．これにより，部品単体が低機能であっても部品合成により要求する機能を容易に実現できるため提案した高信頼部品の再利用性は高い．

7.1.2 粒度を小さくする工夫

本手法では部品を細粒度化するために 4.3 節の非決定的値生成の分離や，4.4 節の操作分割に示すような分割を行っている．非決定的値生成の分離では操作から非決定的な値を生成する部分を分離することで大域変数に関する操作分割だけでは得られない‘ある値を生成する機能’を部品化している．また，4.4 節の操作分割では可能であれば条件分岐の各条件ごとに部品を生成する．

このようにモデルを細分化する際に問題となるのは‘モデルに対して行ったのと同様の細分化が実装に対しても行えるか’である．本手法では非決定的値生成の分離と同様の分割を

行うために，‘実装操作において大域変数への代入文で参照される式は局所変数に格納可能’という制約を与えている．この制約によって状態の変更と値の取得を同時に行う操作呼び出しが制限されるが，状態の変更，値の取得を別々に行う操作呼び出しであれば，操作引数と戻値が必ず局所変数に格納可能であるため制約による不都合は小さい．実装の大域変数への代入文で参照される式のいずれかがモデルの ANY 文で生成される値に対応するため，このような制約を与えることで ANY 文で生成される値に対応する式を戻値とする非決定的値生成操作を構築できる．また，モデルの条件分岐では排他的でない条件群に対して非決定的な分岐を記述可能である．これに対して，実装の条件分岐では決定的な分岐のみが可能なため，モデルの条件分岐それぞれをすべて部品化できる保証はない．そこで，4.4 節に示したように本手法では条件式が排他的か否かで実装を条件式ごとに分割可能か否かを判定し，可能であるならばモデルの条件分岐それぞれを部品化している．このような粒度を細かくする工夫により，1 つの大域変数に注目して部品生成を行ったときに 1 操作から複数の部品が得られるまで細粒度化した．

7.2 信　頼　性

3.1.1 項に示したように，本手法では B Method の枠組みに則って部品の信頼性を定義している．信頼性を証明責務という数学的な命題に基づいて定義することでその定義に基づいた手法自体の信頼性評価が可能になる．本節ではモデル細分化，実装抽出の信頼性を評価する．

7.2.1 モデル細分化の信頼性

4.6 節では B Method の証明責務に基づいて ‘初期化に関する証明責務’ と ‘操作に関する証明責務’ の 2 種類を細分化モデルの証明責務として定義し，さらに，モデル細分化で得られる細分化モデルにおいてこれらの証明責務がつねに真であることの証明を試みた．この結果，初期化に関する証明責務はつねに真になることを証明することに成功した．また，操作に関する証明責務は制約条件展開が任意の変数間の暗黙的な関係をすべて推論できるならば真になることを証明することに成功した．ただし，証明 2 では理想的な推論器を仮定しているが，現実には理想的な推論器は定理証明と同様に停止性を保証できないため，操作の整合性は完全には保証されない．このため，提案手法ではモデル細分化で得られた細分化モデルに対して人手による証明責務の証明を行う．細分化モデルの証明責務は B Method のモデルの証明責務と同じであるため，この作業には AtelierB[5] などの証明支援環境が利用できる．さらに，細分化モデルの証明責務は細分化元モデルの証明責務のゴールと仮定を小さくしたものであるため，細分化元で証明作業を行った作業者にとっては容易な作業となる．この証明責務が偽になる場合，推論できなかった変数間関係が存在するため，これを推論したうえで制約条件に加える必要がある．細分化元モデルの証明責務が真になるためにはこの変数間関係が推論されていなければならないため，細分化元モデルの証明作業者はこの追加を容易に行うことができる．また，上述の証明 2 では制約条件展開の推論能力が高いほど，信頼性も向上することが示されている．このことから，証明責務が偽になる細分化モデルが生じる場合には制約条件展開を修正することでモデル細分化手法の信頼性を向上させることができる．

7.2.2 実装抽出の信頼性

実装抽出の信頼性の定義は細分化実装と実装依存モデルの証明責務が真になることであるが，実装抽出では 5.6 節に示した証明責務最小化により証明責務が真であることを保つよう証明責務を最小化する．このため，実装抽出の信頼性は原理的に保証されている．ただし，証明責務最小化は 5.6 節に示したように定理証明を完全に自動化することができないために作業者の負担軽減が問題となる．経験的に仮定削減作業の定理証明は証明責務のゴールが大きいほど難しくなる．このゴールの大きさは仮の不変条件と操作抽出で得られた操作の事前条件で決まる．そのため，操作抽出で得られる操作が小さく，その操作が含む大域変数が少ないほど，仮定削減の定理証明は容易になる．本手法では部品の粒度を小さくしたことで部品群が大量に生成されるが，仮定削減の定理証明が容易であるため自動定理証明が期待できる．自動定理証明が失敗した部品に対しては作業者が証明を行うが，5.6 節に示したように細分化元ソフトウェアで証明を行った者が証明を行うことで作業負担を軽減できる．以上のことから提案した実装抽出は信頼性に対して作業者負担が小さい．

7.3 今後の展望

本稿で提案した高信頼ソフトウェア部品とその自動生成手法により，高信頼で再利用性の高いソフトウェア部品リポジトリを容易に整備することが可能になる．我々はこのようの性質を持った高信頼ソフトウェア部品を用いることで低コストで高信頼な自動コード合成を行う枠組みを提案してきた[19]．

図 3 に示したように，本手法で提案した高信頼ソフトウェア部品は細分化モデルを検索キーとした健全性の高い再利用が可能である．また，要求モデルからの細分化モデル生成は本稿で提案したモデル細分化で自動化される．これにより，要求モデルを構築するのに必要な部品を自動で収集，再利用できるため，要求モデルに対してソフトウェアを自動合成することが可能となる．また，整合性を保証された高信頼部品を合成して作られたソフトウェアの整合性を保証するためには部品間の整合性のみに注目すればよく，信頼性保証に要するコ

ストも削減できる．現状のソフトウェア自動生成手法では自動生成に用いるソフトウェア部品リポジトリがドメインに依存するため，新たな問題領域に自動生成手法を適用するためにはその問題領域の過去成果物などを分析して部品リポジトリを整備する必要がある．これに対して，本稿で提案した部品自動生成を用いることで過去に構築したソフトウェアから自動でソフトウェア部品を整備することができ，新たな問題領域に対してソフトウェア自動生成を迅速に適用できる．また，提案した高信頼部品は部品名や機能名によらず数学的な意味から部品を検索することでドメインを横断した部品検索もできるため，リポジトリのドメインへの依存性も低減できる．以上のように本稿で提案した高信頼ソフトウェア部品と高信頼部品自動生成を用いることで自動コード生成の適用性と信頼性の改善が期待できる．

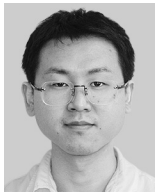
8. おわりに

本稿では B Method の枠組みをソフトウェア部品に応用し，部品において実装が仕様を満たすことを保証できる高信頼部品を提案した．さらに，この高信頼部品を B Method で構築された既存のソフトウェアから自動生成する高信頼部品自動生成を提案した．高信頼部品自動生成手法の構築では入力ソフトウェアが B Method の整合性を満たすならば，そこから得られる部品群も整合性を満たすことを目標とし，部品が整合性を保つこととその条件を証明によって示した．この証明により細分化モデルの操作の整合性を保証するためにはモデル細分化の制約条件展開の推論能力を向上させればよいことを示した．また，実装抽出では人手によって仮定削減を行わなければならないが，この作業量が十分に小さいことを示した．本手法で得られる高信頼部品は粒度がきわめて細かいため，部品を再利用できる場面は増えるが部品単体が提供する機能は小さく，高機能を実現するためには多くの部品を適切に組み合わせなければならない．この組合せを判断する材料として細分化モデルを持つため，提案した高信頼部品は高い再利用性を持つ．今後はこの性質を応用して要求仕様に対してそれを満たすソフトウェアを自動生成する自動コード生成に取り組みたい．

参考文献

1) Abrial, J.: *The B-Book*, Cambridge University Press (1996).
2) ARTech: GeneXus, available from ⟨http://www.genexus.com/⟩ (accessed 2010-12).
3) Boehm, B.: Software Engineering, *IEEE Trans. Comput.*, Vol.25, No.12, pp.1226–1241 (1976).
4) CEA: Papyrus UML, available from ⟨http://www.papyrusuml.org/⟩ (accessed 2010-12).
5) CLEARSY: Atelier B, available from ⟨http://www.atelierb.eu⟩ (accessed 2010-07).
6) JML, available from ⟨http://www.eecs.ucf.edu/~leavens/JML/⟩ (accessed 2011-05).
7) Gannod, G., Chen, Y. and Cheng, B.: An automated approach for supporting software reuse via reverse engineering, *Proc. 13th IEEE International Conference on Automated Software Engineering*, pp.94–103 (1998).
8) Jeng, J. and Cheng, B.: Using formal methods to construct a software component library, *LNCS 717*, Vol.717, pp.397–417 (1993).
9) Jones, C.: *Systematic software development using VDM*, Prentice Hall International (1986).
10) Krueger, C.: Software reuse, *ACM Computing Surveys* (*CSUR*), Vol.24, No.2, pp.131–183 (1992).
11) Lau, K. and Wang, Z.: Software component models, *IEEE Trans. Softw. Eng.*, Vol.33, No.10, pp.709–724 (2007).
12) Manna, Z. and Waldinger, R.: Toward automatic program synthesis, *Comm. ACM*, Vol.14, No.3, pp.151–165 (1971).
13) Matinlassi, M.: Comparison of software product line architecture design methods: COPA, FAST, FORM, KobrA and QADA, *26th International Conference on Software Engineering* (*ICSE'04*), IEEE Computer Society (2004).
14) Meyer, B.: The grand challenge of trusted components, *Proc. 25th International Conference on, Software Engineering*, pp.660–667, IEEE (2003).
15) Weiser, M.: Program Slicing, *ICSE*, pp.439–449 (1981).
16) 吉田　一，山本晋一郎，阿草清滋：XML を用いた汎用的な細粒度ソフトウェアリポジトリの実装，情報処理学会論文誌，Vol.44, No.6, pp.1509–1516 (2003).
17) 橋本　靖，山本晋一郎，阿草清滋：Program slicing を利用したプログラムカスタマイザ，電子情報通信学会技術研究報告，Vol.SS94, No.10, pp.73–80 (1994).
18) 古山将佳寿，山本晋一郎，阿草清滋：ドキュメントを含むソフトウェアモデルの提案，日本ソフトウェア科学会 FOSE，Vol.99, pp.100–107 (1999).
19) 中村丈洋，織田　健：B Method における自動コード合成フレームワークの提案，情報処理学会研究報告，Vol.SE170, No.18, pp.1–8 (2010).
20) 日本経済新聞：2010/12/13-15, 特集：企業向けアプリケーション，available from ⟨http://www.nikkei.com/tech/business/⟩ (accessed 2010-12).
21) 福安直樹，山本晋一郎，阿草清滋：細粒度リポジトリに基づいた CASE ツール・プラットフォーム Sapid，情報処理学会論文誌，Vol.39, No.6, pp.1990–1998 (1998).

(平成 23 年 3 月 8 日受付)
(平成 23 年 9 月 12 日採録)



中村　丈洋（学生会員）

昭和 58 年生．平成 20 年電気通信大学大学院電気通信学専攻博士前期課程修了．現在，同大学院電気通信学専攻博士後期課程に在学中．形式手法と再利用の研究に従事．

織田　　健（正会員）

昭和 39 年生．平成 5 年東京工業大学大学院情報工学専攻博士課程修了，同年電気通信大学電子情報学科助手，平成 11 年同情報通信工学科助手，平成 22 年同大学院総合情報学専攻助教．ソフトウェア設計手法，形式手法，再利用の研究に従事．日本ソフトウェア科学会会員．

西野　哲朗（正会員）

昭和 34 年生．昭和 57 年早稲田大学理工学部数学科卒業．昭和 59 年同大学院理工学研究科博士前期課程修了．同年日本アイ・ビー・エム（株）入社．昭和 62 年東京電機大学理工学部情報科学科助手．平成 4 年北陸先端科学技術大学院大学助教授．平成 6 年電気通信大学電気通信学部助教授．平成 18 年同教授．平成 22 年同大学大学院情報理工学研究科総合情報学専攻教授．現在に至る．理学博士．回路計算量理論，量子計算量理論，計算論的学習理論等の研究に従事．平成 7 年情報処理学会 Best Author 賞，平成 10 年人工知能学会研究奨励賞，平成 14 年電子情報通信学会ソサイエティ論文賞，平成 15 年船井情報科学振興賞，平成 20 年 IBM Faculty Award，平成 22 年文部科学大臣表彰科学技術賞各受賞．電子情報通信学会，日本ソフトウェア科学会，人工知能学会，日本数学会各会員．