モジュール間の整合性検査のための要求仕様記述の検証†

山田宏之†† 松下芳典†††* 井上 健†††† 手塚慶一††† 高松雄三††

本論文では，応用ソフトウェアを構成する部分ソフトウェア（モジュール）の接続関係の妥当性を確証するための，要求仕様に対する全体的枠組みを考察する．この枠組みは，モジュール間の整合性の検証を考慮して構成された要求仕様記述形式および検証系からなる．本要求仕様記述は，各モジュールのインタフェースと挙動の定義およびその接続関係の定義から構成される．モジュールに対する手続き呼出しの時系列(トレース)により挙動が定義され，モジュール間での処理の移動条件により接続関係が定義される．本記述形式では，モジュールをデータ抽象化されたものとみなし，その仕様をブラックボックス的に記述できるために，モジュール内のデータ構造等の実現レベルの情報を考慮する必要がない．本記述の検証は，要求仕様記述を一階述語論理形式に変換し，処理が移動するときにモジュールの挙動が移動条件を満足することを反駁導出により求めることで実現されている．従来のソフトウェア検証法である不変表明法に比べ，本手法は，検証に用いる仕様として，モジュールのインタフェースと挙動およびモジュール間の接続関係を定義するだけでよく，記述対象が明確である点で仕様の記述が容易であり，より実際的であると考えられる．また，実験検証システムによる検証実験を通して，提案した記述形式の有効性を明らかにしている．

1. はじめに

ソフトウェアの要求仕様作成段階で発生した誤りが早期に検出されない場合には，それ以降の開発作業が無効となるために，開発効率は著しく低下する[1]．

従来から，ソフトウェアの正しさを証明する方法として，ソフトウェアが満足すべき性質を記述（仕様記述）し，その記述を検証する方法の研究がなされている[2]．その代表的な方法として，不変表明法がある．しかしながら，不変表明の発見法が明確にされていないこと，ソフトウェアの具体的な実現方法が議論されていない要求仕様記述段階で不変表明を記述することが容易でないこと，などから実用的なものとなっていない．

また，要求仕様記述段階では，具体的な実現レベルの情報を隠ぺいして仕様が記述できることが望まれる．このような仕様記述法として，応用ソフトウェアを構成する部分ソフトウェア（モジュール）に対する手続きの呼出し系列（トレース）によりその挙動を記述する方法が Parnas らにより提案されている[3]．トレースによる仕様記述法は，モジュールに対する手続き呼出しとその応答について注目するものであり，モジュールを抽象データ型としてとらえ，その仕様を記述することができる．しかしながら，トレースで表現された仕様を利用した検証法について，具体的な考察が十分なされていない．

本論文では，要求仕様記述段階においてモジュールの接続関係の妥当性を確証するため，モジュール間の整合性の検証を考慮した，要求仕様に対する全体的枠組み（要求仕様記述形式および検証法）を考察する．本要求仕様記述は，各モジュールのインタフェースと挙動の定義およびモジュールの接続関係の定義から構成される．モジュールの挙動は，インタフェースにおけるトレースにより定義され，モジュールの接続関係は，モジュール間の処理の移動条件により定義される．

本記述形式では，モジュールをデータ抽象化されたものとみなし，モジュールの仕様がブラックボックス的に記述できるので，従来の仕様記述法に比べて，モジュール内のデータ構造等の実現レベルの詳細な情報を考慮する必要がない．さらに，本記述ではモジュールのインタフェースと挙動およびモジュール間の接続関係を定義するだけでよく，記述対象が明確であるために，仕様を記述することが容易である．

以下においては，ソフトウェアに対する要求仕様記

† A Verification Method of Requirements Specification for Checking Consistency between Modules by HIROYUKI YAMADA (Department of Computer Science, Faculty of Engineering, Ehime University), YOSHINORI MATSUSHITA (Department of Communication Engineering, Faculty of Engineering, Osaka University), TAKESHI INOUE (Faculty of Mercantile Marine, Kobe University of Mercantile Marine), YOSHIKAZU TEZUKA (Department of Communication Engineering, Faculty of Engineering, Osaka University) and YUZO TAKAMATSU (Department of Computer Science, Faculty of Engineering, Ehime University).
†† 愛媛大学工学部情報工学科
††† 大阪大学工学部通信工学科
†††† 神戸商船大学商船学部計測工学講座
* 現在 (株)野村総合研究所

述形式を提案し，その記述形式に基づく要求仕様を解析してモジュール間の整合性を検証する手法を述べる．さらに，本検証法に基づく実験検証システムを構築し，待ち行列シミュレーションソフトウェアの要求仕様記述に含まれる構造的誤りの解析例からその有効性を明らかにする．

2. 要求仕様記述

本論文では，例題として，一つの独立した機能をもつモジュールを組み合わせることにより，モデル化が容易にできる待ち行列シミュレーションソフトウェアを取り上げ，その要求仕様記述について考察する．

2.1 待ち行列シミュレーションにおける要求仕様

設計者は，待ち行列シミュレーションソフトウェアを開発する場合に，対象世界を待ち行列モデル（ソフトウェアモデル）で表現する．まず，対象世界に現れる構成物（オブジェクト）を待ち行列モデルを構成する要素であるモジュールに割り当て，その機能を定義する．ここで，モジュールには，トランザクションを待ち行列モデルに入れるためのモジュール(ARRIVAL)，トランザクションを待ち行列モデルから出すためのモジュール (DEPARTURE)，サービスを提供するためのモジュール(SERVER)，待ち合わせ処理をするためのモジュール (QUEUE) がある．つぎに，待ち行列モデルのタスクの実現される処理の流れ（トランザクションフロー）を定義し，モジュール間の接続関係（接続箇所，処理の移動条件）を記述する．この一連の記述により得られるものが待ち行列モデルである．待ち行列モデルは待ち行列シミュレーションソフトウェアの要求仕様に相当する．

2.2 要求仕様記述の言語仕様

本論文で提案する要求仕様記述は，2.1 節で述べたモデル化の過程に応じて要求仕様が作成できるように，ソフトウェアモデルを構成する各ステップに対応したモジュール記述，トランザクション記述，接続記述から構成される[4]．本要求仕様記述において，モジュール記述により個々のモジュールの機能が定義され，トランザクション記述により処理の流れが定義され，接続記述により処理の接続関係が記述される．本要求仕様の記述形式を**図1**に示す．

モジュール記述（図1(a)）は，モジュール名(NAME)，モジュールに対するインタフェースを定義するインタフェース部 (INTERFACE)，モジュール中の手続きに対する挙動を定義する機能定義部

```
NAME 〈モジュール名〉: facility
INTERFACE
  〈呼出し名〉 {[〈引数の型〉]}
              : {〈値の型〉}
DEFINITION
        〈基準トレースの定義〉
        〈正当なトレースの定義〉
        〈等価変換規則の定義〉
        〈手続きが返す値の定義〉
```

(a) モジュール記述の形式

```
NAME 〈トランザクション名〉: transaction
        arrive 〈モジュール名1〉
        thru   〈モジュール名2〉
                 ⋮
        depart 〈モジュール名N〉
```

(b) トランザクション記述の形式

```
NAME 〈リンク名〉: link
        move 〈トランザクション名〉
        from 〈モジュール名1〉
        to   〈モジュール名2〉
CONDITION
        〈トランザクションの移動条件〉
ACTION
        〈移動にかかる動作記述〉
```

(c) 接続記述の形式

図1 要求仕様の記述形式
Fig. 1 Notation for requirements specification.

(DEFINITION) から構成される．

インタフェース部では，そのモジュールに対する呼出し名（手続き名），手続きの引数に対するデータ型および手続きの返す値のデータ型が記述される．

機能定義部では，以下に述べるようにトレースにより各手続きの挙動を定義する．

まず，モジュールの挙動を決定する上で基準となる基準トレースを定義する．つぎに，各手続きについて，モジュールが手続き呼出しを正しく受け入れる条件を示す正当なトレースを定義する．また，正当なトレースから基準トレースへの等価変換規則を定義する．このほか，手続き値を返す場合には，その値に対する条件を定義する．

実際の動作において，ある手続き呼出しがなされ，それが正しいトレースであるときには，正当なトレースは等価変換規則により基準トレースへ変換される．また，その手続きの返す値があれば，値は，変換された基準トレースに基づいて決定される．

ここで，トレースはモジュールの呼び出された手続き名の並びにより表現する．すなわち，あるモジュールが T1，T2，T3 の順で呼び出されるとき，トレー

スTは T=T1. T2. T3 と表現する．また，トレースTが正当であることは，L(T)，基準トレースであることは，N(T) と記述する．トレースT内の最後の手続き呼出しにより値aを返すことは，V(T)=aと記述する．トレースTの長さは，内部関数 length(T) により与えられる．さらに，機能定義部の各々の定義は，上述したトレースの表記およびトレースの表記を等価記号「≡」，等値記号「=」で結合した項から構成されるホーン節形式で表現する．

トランザクション記述（図1(b)）は，処理要求を表すトランザクション名（NAME），処理要求を実現するために必要なモジュールの並びを表す到着モジュール名（arrive），通過モジュール名（thru）および退出モジュール名（depart）から構成される．

また，モジュール間には，処理が移動するためのリンクが存在するものとし，リンクが主体となって処理の移動が実現するものとする．接続記述（図1(c)）は，リンク名（NAME），リンクの接続関係（move, from, to），条件部（CONDITION）および動作記述部（ACTION）から構成される．リンクの接続関係には，モジュール1からモジュール2へ処理が移動するリンクの存在することを記述する．条件部には，処理の移動条件（起点のモジュール1が処理を渡すときに満足するべき出力条件および終点のモジュール2が処理を受け入れるときに満足するべき入力条件）を記述する．動作記述部には，移動条件が満足されたときに，実行する動作（処理を移動させるためにリンクが実行するモジュールへの手続き呼出し）を記述する．

本要求仕様記述は，対象世界をモデル化するステップに対応して系統的に作成される．さらに，同一モデルを複数の視点から表現するために，設計者がモデルに対してもつ意図を的確に反映できる．

2.3 要求仕様の記述例

トランザクションが1種類（Tr1）のみ存在する単一待ち行列モデルの一例を**図2**に示す．この待ち行列モデルには，モジュールとして ARRIVAL (A1), QUEUE(Q1), SERVER(S1) および DEPARTURE (D1) がそれぞれ一つある．処理要求は，A1 で発生したトランザクションが待ち行列Q1で待ち合わせた後，サービスを S1 で受け，D1 でモデルから退去するトランザクション Tr1 で表現される．ここで，Q1 のバッファ容量は無限であるとし，S1 の窓口は一つであるとする．この例の要求仕様記述を**図3**に示す．

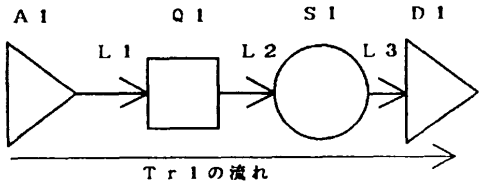


図2 シミュレーションモデルの例
Fig. 2 An example of a simulation model.

Q1 のモジュール記述（図3．A(b)）において，そのモジュールに対するインタフェースとしては，in, out および size があることが定義されている．また，モジュールの機能定義として，in の呼出しが任意個続くトレースが基準トレースであることが定義されている．さらに，個々の手続きに対して，その挙動が定義されている．たとえば，out の挙動は次のように定義されている．

呼出し out がモジュールに正しく受け入れられるのは，トレースTが基準トレースであり，かつ，トレースTの長さが0より大きいときである．また，トレース T. out は，基準トレース T (T=in(_).T1) の先頭の呼出し (in(_)) を除いたトレース T1 と等価である．さらに，out が返す値は基準トレースTに含まれる先頭の呼出し (in (A)) の引数である．

トランザクション記述(図3．B)には，トランザクション Tr1 が A1 からモデルに入り，Q1, S1, D1 をへてモデルから退去することが記述されている．

L1 の接続記述（図3．C(a)）では，A1 が out を正常に受け取ることができ，Q1 が基準状態であるならば，A1 からトランザクションを取り出し，Q1 へ渡すことが記述されている．

3. 要求仕様記述の検証

3.1 モデル検証

本要求仕様記述に対する検証（モデル検証）は，上述の三つの記述を相互に参照して，各モジュール間の整合性を調べることにより行う．すなわち，トランザクション記述に基づいて，モジュール間を処理が移動するときに，モジュール記述で定義されたモジュールの挙動が接続記述で定義された移動条件を満足することを検証することにより，モジュール間で正しく処理が移動できることを証明する．本検証は，与えられた要求仕様記述を一階述語論理式で表現し，処理を移動させるための動作記述をゴールとして反駁導出することにより実現する．反駁に成功する場合には，処理はモジュール間を正しく移動することを意味しており，反駁に失敗する場合には，要求仕様内に誤りが含まれ

```
NAME A1: facility
INTERFACE
      out: transaction
DEFINITION
      ; 基準トレースの定義
      N(T)→N(T. out)
      ; out の挙動
      N(T)→L(T. out)
      N(T)→V(T. out)=Tr1
```

(a) ARRIVAL のモジュール記述

```
NAME Q1: facility
INTERFACE
      in [transaction]:
      out: transaction
      size: integer
DEFINITION
      ; 基準トレースの定義
      N(T)→N(T. in(_))
      ; in の挙動
      N(T)→L(T. in(_))
      ; out の挙動
      N(T) & (length(T)>0)→L(T. out)
      N(T) & (T=in(_). T1)→T. out≡T1
      N(T) & (T=in(A). T1)→V(T. out)=A
      ; size の挙動
      N(T)→L(T. size)
      N(T)→T. size≡T
      N(T)→V(T. size)=length(T)
```

(b) QUEUE のモジュール記述

```
NAME S1: facility
INTERFACE
      in [transaction]:
      out: transaction
      exist: boolean
      empty: boolean
DEFINITION
      ; 基準トレースの定義
      →N(φ)
      →N(in(_))
      ; in の挙動
      N(T) & (length(T)=0)→L(T. in(_))
      ; out の挙動
      N(T) & (length(T)>0)→L(T. out)
      N(T) & (T=in(_))→T. out≡φ
      N(T) & (T=in(A))→V(T. out)=A
      ; exist の挙動
      N(T)→L(T. exist)
      N(T)→T. exist≡T
      N(T)→V(T. exist)=(length(T)>0)
      ; empty の挙動
      N(T)→L(T. empty)
      N(T)→T. empty≡T
      N(T)→V(T. empty)=(length(T)=0)
```

(c) SERVER のモジュール記述

```
NAME D1: facility
INTERFACE
      in [transaction]:
DEFINITION
      ; 基準トレースの定義
      N(T)→N(T. in(_))
      ; in の挙動
      N(T)→L(T. in(_))
```

(d) DEPARTURE のモジュール記述

A. モジュール記述

```
NAME Tr1: transaction
      arrive A1
      thru   Q1
      thru   S1
      depart D1
```

B. トランザクション記述

```
NAME L1: link
      move Tr1 from A1 to Q1
CONDITION
      A1 [L(T. out)] & Q1 [N(T)]
ACTION
      Tr=A1(out)
      Q1(in(Tr))
```

(a) L1 の接続記述

```
NAME L2: link
      move Tr1 from Q1 to S1
CONDITION
      Q1 [L(T. out)] & S1 [N(T) & V(T. empty)]
ACTION
      Tr=Q1(out)
      S1(in(Tr))
```

(b) L2 の接続記述

```
NAME L3: link
      move Tr1 from S1 to D1
CONDITION
      S1 [N(T) & L(T. out)] & D1 [N(T)]
ACTION
      Tr=S1(out)
      D1(in(Tr))
```

(c) L3 の接続記述

C. 接 続 記 述

図 3 要求仕様記述
Fig. 3 Requirements specification.

る可能性を示唆している．このように，本モデル検証から要求仕様内の誤り（ソフトウェアのモデルに内在する構造的欠陥等）が発見されうることは，設計者がソフトウェアモデルを構築する時点において，その妥当性を確認する有効な一手段となりうることを示している．この点については，次節の後半でさらに検討を加える．

3.2 要求仕様記述の検証例

本節では，単一待ち行列モデルを具体例として取り上げ，実際の検証例を示すことにより，3.1 節で述べた検証法の有効性を示す．

（1） 図2のモデルの検証

まず，Tr1 のトランザクション記述（図3．B）から Tr1 は A1から Q1へ移動することが得られる．図3．Cに示される接続記述から，A1 から Q1へ Tr1 を移動させるリンクは L1である．L1の動作記述から検証内容は，

A1 [L(T. out)] (1)

Q1 [L(T. in(_))] (2)

である．(1)式は L1 の移動条件より常に充足される．また，(2)式は QUEUE のモジュール記述(図3．A(b)) における in の挙動より

Q1 [N(T)] (3)

となる．(3)式は L1の移動条件より反駁され，トランザクション Tr1 は常に正しく A1から Q1へ移動する．

さらに，Tr1 は Q1から S1へ移動する（図3．B参照)．接続記述から，Tr1 を Q1から S1へ移動させるリンクは L2である．L2の動作記述から検証内容は，

Q1 [L(T. out)] (4)

S1 [L(T. in(_))] (5)

である．(4)式は L2 の移動条件より常に充足される．また，(5)式は SERVER のモジュール記述(図3．A(c)) における in の挙動より

S1 [N(T) & (length (T)=0)] (6)

となる．

ここで，L2 の移動条件は，

S1 [N(T) & V(T. empty)] (7)

であり，SERVER のモジュール記述における empty の挙動から，

V(T. empty)=(length (T)=0) (8)

であるので，(6)式は(7)式，(8)式より反駁され，トランザクション Tr1 は常に正しく Q1から S1へ移動する．

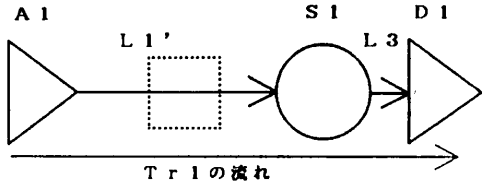


図 4 構造的な誤りの例
Fig. 4 An example with a structural error.

同様にして，S1 から D1 へのトランザクションの移動についても検証される．

（2） 誤りを含むモデル

つぎに，図4に示すように，S1 の待ち合わせ処理を行う Q1がモデル化の過程で欠落したとする．このとき，トランザクション Tr1 は A1を出た後，直接 S1 に入る．その結果，Tr1 のトランザクション記述は，以下のようになる．

```
NAME Tr1 : transaction
  arrive  A1
  thru    S1
  depart  D1
```

また，リンクL1′の接続記述は，以下のようになる．

```
NAME L1' : link
  move Tr1 from A1 to S1
CONDITION
  A1 [L(T. out)] & S1 [N(T)]
ACTION
  Tr= A1(out)
  S1(in(Tr))
```

ここでは，Tr1 が A1から S1への移動に関する検証について述べる．L1′ の動作記述から検証内容は，

A1 [L(T. out)] (9)

S1 [L(T. in(_))] (10)

である．(9)式は L1′ の移動条件より常に充足される．また，(10)式は SERVER のモジュール記述(図3．A(c)) における in の挙動より

S1 [N(T) & (length (T)=0)] (11)

となる．さらに，(11)式は L1′ の移動条件より

S1 [length (T)=0] (12)

となるが，(12)式を反駁するための記述が存在しないために，導出はここで停止する．(12)式は，S1 が空であればトランザクションは正しく S1に移動することを意味している．換言すると，L1′ には S1の内部状態によって A1の出力を制御する記述が存在しな

いので，S1 は空でない限りトランザクションを受け取れない．これはモデルに内在する構造的欠陥と考えられる．たとえば，S1 の手前に緩衝用としての待ち行列を入れる，A1 の直後にフロー制御用としての待ち行列を入れるなどにより，このような制約的条件は解消される．

4. 検証システムの構成

本論文で提案した要求仕様記述を受理し，モデル検証を行う要求仕様検証システムの構成を**図5**に示す．入出力部では，設計者(ユーザ)から検証要求を仕様記述検証部に渡し，得られた結果をユーザに報告する．さらに，入出力部において，要求仕様記述は構文解析され，一階述語論理形式に変換し，検証システム内の該当するファイルに格納される．モジュール記述ベースには，節形式で表現されたモジュール記述が格納される．トランザクション記述ベースには，各モジュールを表すシンボルを処理の順に並べたリストをもつ節形式で表現されたトランザクション記述が格納される．接続記述ベースは，移動条件と動作記述を対とした形式で表現された接続記述をもつ．仕様記述検証部は，要求仕様記述を検証する．検証要求が与えられると，トランザクション記述に従い，接続記述ベースからトランザクションの移動条件を得る．移動条件は，一時的な宣言としてモジュール記述ベースに格納され，接続記述の動作内容をゴールとして導出法により検証する．検証が終了すると，検証結果を入出力部に渡す．本検証システムは，ホーン節の導出によりゴール節を変形するが，等価記号「≡」ならびに等値記号「＝」を含む項は，記号の左右の述語を両方含む項がゴール節に現れない限り導出することができない．そこで，線形導出部と等価導出部に分離して検証器を構成し，等値記号・等価記号を含む述語に対応できるように拡張した反駁導出により，変換された仕様記述を検証している．等価導出部では，トランザクションの移動条件の仮定等によりヘッドが等価記号（等値記号）を述語名とする項が導かれるとき，等価（等値）関係を新たに宣言する．この等価（等値）関係により，ゴール節は線形に導出できる．本検証システムによる図2および図4のモデルに対する実際の検証の様子を**図6**に示す．

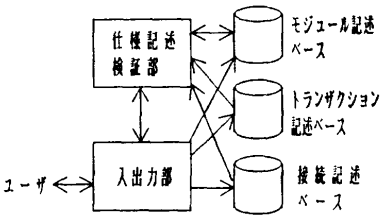


図5 システム構成図
Fig. 5 System organization.

```
Transactions are [Tr1]

Verify between [A1, Q1]
  get link L1
  verify link(L1, ask, A1, L([T|out]))
  try A1: [L([T|out])]
    success A1: [L([T|out])]
  verify link(L1, ask, Q1, L([T|in(_)]))
  try Q1: [L([T|in(_)])]
    success Q1: [L([T|in(_)])]

Verify between [Q1, S1]
  get link L2
  verify link(L2, ask, Q1, L([T|out]))
  try Q1: [L([T|out])]
    success Q1: [L([T|out])]
  verify link(L2, ask, S1, L([T|in(_)]))
  try S1: [L([T|in(_)])]
  try S1: [length(T)=0, N(T)]
    using equivalence
          V([T|empty])=length(T)=0
    success S1: [length(T)=0]
  try S1: [N(T)]
    success S1: [N(T)]
    success S1: [L([T|in(_)])]
```

(a) 図2の検証の一部

```
Transactions are [Tr1]

Verify between [A1, S1]
  get link L1
  verify link(L1, ask, A1, L([T|out]))
  try A1: [L([T|out])]
    success A1: [L([T|out])]
  verify link(L1, ask, S1, L([T|in(_)]))
  try S1: [L([T|in(_)])]
  try S1: [length(T)=0, N(T)]
    fail S1: [length(T)=0]
  try S1: [N(T)]
    success S1: [N(T)]
    fail S1: [L([T|in(_)])]

Verify between [S1, D1]
  get link L3
  verify link(L3, ask, S1, L([T|out]))
  try S1: [L([T|out])]
    success S1: [L([T|out])]
  verify link(L3, ask, D1, L([T|in(_)]))
  try D1: [L([T|in(_)])]
    success D1: [L([T|in(_)])]
```

(b) 図4の検証

図6 検証結果
Fig. 6 Results of the system.

5. おわりに

本論文では，ソフトウェアに対する要求仕様記述法とその要求仕様記述からモジュールの整合性を検証する検証法について述べた．

本要求仕様記述において，モジュールの挙動はモジュールに対する呼出し系列により定義され，モジュールの接続関係はモジュール間で処理が受け渡されるときの移動条件により定義される．したがって，本要求仕様記述法は，モジュールの具体的な実現方法が明確に定まっていない要求仕様記述段階において，不変表明の発見法が明確でない不変表明法と比べると，記述すべき対象が明確であるために，記述が容易であり，より実際的である．

本要求仕様記述は，モジュールの性質を記述するモジュール記述，処理の流れを記述するトランザクション記述，モジュール間の接続関係を記述する接続記述から構成されており，要求仕様は対象世界をモデル化するステップに対応して系統的に記述される．さらに，要求を表現するモデルは複数の視点から記述されるため，設計者のモデルに対する意図を的確に反映することができる．

本要求仕様記述形式および検証法は，要求仕様記述を一階述語論理形式で表現し，処理がモジュール間を移動するときに，その挙動が移動条件を満足することを反駁導出により検証するものである．したがって，本枠組みは，本論文で述べた待ち行列シミュレーションソフトウェアのみに限定されるものではなく，互いに独立なモジュールを組み合わせて構成されるソフトウェアの静的な解析に有効であると考えられる．

また，本検証法を実現した実験検証システムにより要求仕様に含まれる構造的欠陥が指摘できることを示した．本検証システムは，モジュール間の整合性が検証できるので，モジュールを部品としたソフトウェア合成における部品間の整合性検査に利用できると考えられ，部品合成によるソフトウェア開発を支援する有力なツールとなりうる．

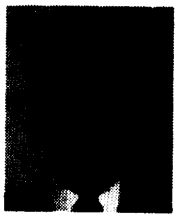
ただし，現段階では，検証過程において反駁に失敗した場合の原因の究明はシステム化されていないため，設計者が検証結果から原因を予測している．反駁証明に失敗した場合の結果を解釈する仕組みについては，モデル検証の支援能力を高めるためにも解決しなければならない今後の課題である．

謝辞 日頃，筆者が御激励頂く愛媛大学相原恒博教授に感謝の意を表する．

参考文献

1) Agusa, K., Ohnishi, A. and Ohno, Y : A Verification Method for Formal Requirements Description, *J. Inf. Process.*, Vol. 7, No. 4, pp. 223-229 (1984).
2) 斉藤，米崎：ソフトウェアの検証，ソフトウェア工学ハンドブック，pp. 177-221，オーム社 (1986).
3) Bartussek, W. and Parnas, D. L. : Using Assertions about Traces to Write Abstract Specifications for Software Modules, *Software Specification Techniques*, pp. 111-130, Addison-Wesley (1986).
4) 松下，山田，井上，手塚：トレース法による待ち行列網モデルの仕様記述とその妥当性の検証，信学技報，COMP 87-65 (1988).

(平成元年 3 月 29 日受付)
(平成元年 10 月 11 日採録)



山田 宏之（正会員）

昭和 36 年生．昭和 58 年大阪大学工学部通信工学科卒業．昭和 63 年同大学院博士課程修了．同年愛媛大学工学部情報工学科助手．プログラム理解，知識工学を応用したソフトウェア開発支援に関する研究に従事．工学博士．電子情報通信学会，人工知能学会，IEEE 各会員．

松下 芳典

昭和 38 年生．昭和 61 年大阪大学工学部通信工学科卒業．昭和 63 年同大学院修士課程修了．同年(株)野村総合研究所入社．在学中はソフトウェア開発支援システムの構成に関する研究に従事．現在はオンラインシステムにおけるベースシステムの開発に従事．

井上 健（正会員）

昭和 28 年生．昭和 52 年大阪大学工学部通信工学科卒業．昭和 57 年同大学院博士課程修了．同年豊橋技術科学大学工学部助手．昭和 60 年大阪大学工学部助手．昭和 63 年神戸商船大学商船学部助教授．この間，計算機網，通信プロトコルの性能解析に関する研究を行い，現在は，イメージパターンの記述・生成とその支援の研究を行っている．工学博士．

手塚 慶一（正会員）

昭和3年生．昭和26年大阪大学工学部通信工学科卒業．同大学院特別研究生となる．昭和29年愛媛大学助教授，以後山口大学，大阪大学助教授を経て，現在大阪大学工学部教授．オートマトンと言語，パターン認識，コンピュータネットワーク，データ伝送の研究に従事．工学博士．著書「電子計算機基礎論」，「待ち行列システム理論」（監訳）など．

高松 雄三（正会員）

昭和18年生．昭和41年愛媛大学工学部電気工学科卒業．佐賀大学理工学部助手，同講師，同電子工学科助教授を経て昭和62年より愛媛大学工学部情報工学科教授．論理回路の故障診断，高信頼化に関する研究に従事．工学博士．著書「電子計算機と情報科学」（共立出版），「論理設計入門」（日新出版）（いずれも共著）．電子情報通信学会，IEEE 各会員．