ソフトウェア工学の課題

山　野　紘　一

１．は　じ　め　に

今日，コンピュータは金融・証券などのシステムをはじめ，通信，製造，販売などあらゆる分野のシステムや自動車，航空機，列車などの機械を動かしている。そのソフトウェアが日常生活に果たす役割は益々増大してきているが，ソフトウェアの生産性，信頼性に関する課題はほとんど解決していない。ソフトウェア開発における最大の課題は，変化するユーザニーズやビジネスニーズへの柔軟で迅速な対応，生産性の向上，開発期間の短縮である。ソフトウェア以外の工業製品の場合は，開発工程が規格化，最適化され成果を上げているのに対し，現在もなおソフトウェアの欠陥や開発の遅延の問題が多数発生している。

こうした問題の原因は，仕様を正確に表現して記述，検証していない場合がほとんどである。さらに，ソフトウェア開発が一貫した思想，原理にもとずいた設計方針で実施されていないため，仕様記述の曖昧さを実装設計の段階で見つけた欠陥や抜けの修正でもって補ったりしているのが実情である。

このようなソフトウェアの設計自体の問題は，ソフトウェアの危機として，1965年ころ叫ばれて，当時以下の問題あることが指摘された。

⑴　ソフトウェア規模の増大
⑵　開発コストの増大
⑶　バックログの増大
⑷　生産性の低下
⑸　品質の低下
⑹　技術者の不足

これらの問題を解決し，信頼性の高いソフトウェアシステムの構築を目指して，ソフトウェア工学が提唱された。1968年10月に（西）ドイツのガルミッシュで開かれたNATO後援のソフトウェア工学会議［１］で，ソフトウェア工学の目的に，

⑴　ソフトウェア危機の解消
⑵　ソフトウェア開発の生産性向上
⑶　ソフトウェアの品質向上

が掲げられ，ソフトウェアの仕様作成，開発から保守に至るまでのすべての工程で，他の諸工学の分野のように，理論的基礎と実践的技術にもとづく方法論や技法を体系化し，それらを適用することによって，ソフトウェアの生産性と品質の向上を図ることを目標とし

た。

しかしながら，その後，ソフトウェア工学は上記の目標を果たすようには進展しなかった。そこで，ブルックス［２］は，1986年にソフトウェア工学に「銀の弾などない」という論文を発表して，技術と管理手法において，ここ十年間で生産性，信頼性，容易性で飛躍的な改善を約束できるものはないと指摘した。この状況は現在に至るまで依然として続いている。

先に触れたようにソフトウェア開発における本質的問題は，開発工程で作成する仕様，設計書の記述の曖昧性や不完全性によるものである。ソフトウェア開発工程は，要求定義，仕様，設計書，プログラムというような記述の積み上げである。

ジャクソン［３］は，1995年，「ソフトウェア博物誌――世界と機械の記述」において，ソフトウェア工学の本質を的確に把握し，その方向性を示唆した。この中で，「ソフトウェア開発の中心は記述である。」と位置づけた。このジャクソンの著書は，体系化されたテキストではなく，用語集的な記述形態であるのが残念ではあるが，内容は実世界の適用領域と機械，およびその関係をいかに記述するかについての画期的な考察から構成されている。

われわれは，この方向を発展させていけば，ソフトウェア工学の目標に近づき，他の工学と同じようにソフトウェア工学を厳密でかつ精密な学にすることができると考えるものである。本稿では，主として仕様記述に関して，ジャクソンの提唱を発展させるための課題を考察する。なお，ジャクソンの概念と同じくするものは同じ用語を使用する。

２．記述の工学

2.1　記号の機械

コンピュータは記号の機械である。ソフトウェア開発は記号を使って新たに機械を組み立てることである。開発された機械は，システムとして実世界に組み込まれ，利用者の道具に供される。利用者の要求や問題解決のために開発された機械が設置され作用する場を適用領域という。適用領域は，利用者がシステム構築を行う実世界の特定部分で，要求，問題，帳票，伝票，人，他の機械システムなどとの関連をもつ領域である。

ソフトウェア開発でもっとも基本的なことは「何を」と「いかに」を分けることであると従来から主張されてきた。このことは，対象世界を大きく実世界と機械世界との二つに分けることの対応する。実世界では，構築する機械が何をするかを記述すし，機械世界では機械がいかに処理するかを設計する。

仕様とは，適用領域と機械との相互作用の記述である。ソフトウェアの開発工程（図１）の概略は以下のものである。利用者の要求や，問題を分析して仕様を作成する。次に，仕様にもとづきプログラムを作成する。

プログラムは，記号で名前付けられ，機械に作用するように順序付けられた命令語の列である。プログラムを作成する工程の記述は，機械世界の概念，意味，作用を枠組みにし

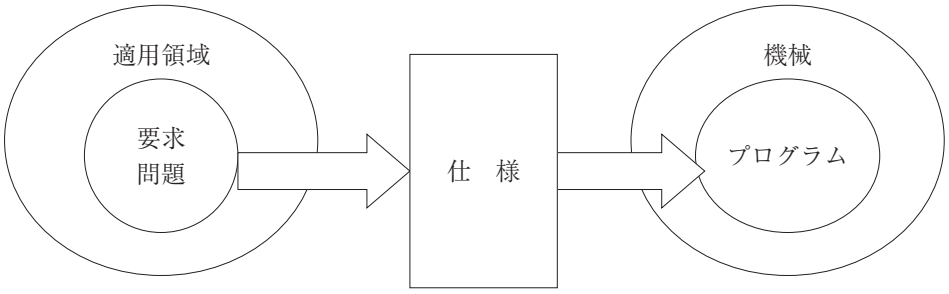

図1　仕様

て展開される。

仕様は一般的には，自然語が主体で，図式，表や数式などを交えて，適応領域と機械との相互作用の状態を適応領域の操作や振る舞いで厳密に記述したものである。

以上のようにソフトウェア開発は記述が中心であり，ソフトウェア工学は，記述の体系を中心課題としている。

2.2　記号

仕様記述の自然語で使われる言語記号や機械世界の記号の特性を考える。工学における仕様や設計に用いる言語は，事象，機能，構造，データ，イベント，メッセージなどの名前（記号）に意味や概念を担わした記述ができ，それらの定義や関係を厳密に伝える役割を果たしている。

ソシュール［4］は，記号は図2に示すように，意味するもの（聴覚イメージ）すなわち，記号表現と，意味されるもの（概念）すなわち，その記号表現が指し示している記号内容から成り立っているとした。

このような記号の特性として，ソシュールは，記号の原理を二つ挙げている。

一つは，言語記号の恣意性の原理である。

「第一の原理，または基本的な真理として，言語記号は恣意的である。」（p 154）

すなわち，概念と与えられた聴覚イメージを結び付けるものは，根本的に恣意的である。記号内容と文字や図形，知覚できる音である記号表現には必然的な結びつきはないということである。このことを対応の恣意性という。

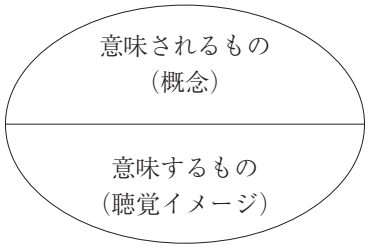

図2　記号

また，ソシュールによれば，ある記号の記号内容と記号表現は，同時発生で不可分離性を持っている。記号は，記号内容と記号表現の確定とともに生まれる。互いの存在があって初めて存在するので，相互依存的である。すなわち，記号が生まれる以前に，既成の概念は存在しないということである。

もう一つは，線状性の原理である。

「第二の原理，または第二の基本的な真理。言語記号は，ある延長を持っていて，この延長は，一つの次元の中だけで展開します。」（p 157）

これは，記号は一列に並ぶということ，すなわち，一列に配列される。よく知られているように，テキストの場合，文字は空間的に一列に配置されている。

仕様やプログラムの構成は空間的に配置され，平面的で一列になっているので，それを読む人や機械は線形的に走査する。このような線状性は言語の宿命的なものである。特に，プログラムの場合，線状性による名前の未定義使用などのチェックを容易にするために，プログラミング言語は，名前の宣言の前置を規則としている。

記号が指し示すものは，対象世界から切り取ったものである。対象世界を切り分けるためには，何か名前を付ける必要がある。名前の記号は，ある視点から対象世界の事象を切り取り，その事象が異なっているか，同一であるかを意味付ける働きを持つ。この働きは分節と呼ばれる。記号の分節の働きには，同一性と差異性とを併せてもつ相補的な働きがある。ここで重要なのは，この分節の働きも恣意的であるということである。

上に述べたように，記号の恣意性には，記号内容（意味されるもの）と記号表現（意味するもの）との対応の恣意性と記号の分節の恣意性とがある。

2.3　構造の恣意性

仕様においても，プログラムにおいても内容は，記号（概念）と記号（概念）との関係を記述している。仕様やプログラムにおける記述の構造とは，記号と記号との関係のことである。従って，記号の関係から構成されるこの構造も必然的に恣意性をもつ。

このような恣意性自体は悪いことでなく，恣意性こそが，独創の源になるケースも多い。恣意性は，人間の創造力でもあり，思考や感性の高度化を促し，精神を豊かなものにしてきたともいわれている。

ソフトウェア工学においてこれまで提案されてきた多くの設計方法論は，記号や構造の恣意性を考慮していない。

３．形式と構造

コンピュータは，計算の理論をひとつの基盤としているので，数学との関係が深い。計算の理論は，チューリング機械という記号機械や帰納的関数，λ 計算などを基礎としている。数学における形式化や構造化の概念は，ソフトウェア工学に多くの影響を与えている。そこで，数学の形式と構造とはどのようなものかを概観しておく。

3.1 数学の形式

数学は厳密性を目指して発展し，機械的な規則で証明を実行できるように形式化を達成してきた。数学でいう形式系とは，機械的な言語と規則をもった体系である。機械的とはチューリング機械で処理できるという意味で，コンピュータで処理できる体系と同義である。

数学的な概念や対象は，数学的思考の精神活動から捉えられたものである。数学の本質は自由に考えることのできるものなら，そのすべてを対象にし，物理や化学が対象とする自然現象とは異なり，現実世界に存在するしないにかかわらず思考の対象とすることができる。このような自由性により，数学はあらゆる事象に応用できる。

ソフトウェアが対象とする世界は，最終的には機械に処理できるものに写像できるものであるが，その概念や思考の形式は，数学の世界の延長にある。

数学の形式化は，ヒルベルトによって提案された「ヒルベルトのプログラム」にその源がある。

ヒルベルトのプログラムに関して，竹内［5］（p 91）は，次のように述べている。

⑴ 現在用いられている論理の体系を形式化する。すなわちいかなる推論方式が用いられているか，いかなる論理的な公理が用いられているかを形式的な体系としてはっきり記述する。

⑵ 数学的体系を形式化する。すなわち上の論理の体系の上にどのような数学的公理を付け加えると現在の数学の体系ができるかを形式的な体系として記述する。

⑶ 上の公理的な体系から矛盾が出ないことをその形式的な体系についての推論として証明する。

ヒルベルトのプログラムは，形式化において，集合論や自然数論などを含む数学全体を対象として，その無矛盾性を証明するもので，あらゆる数学理論を公理化し，公理や定理をすべて記号で表現しようとするものである。

形式系ではその対象は意味をもたない記号，対象記号や関数記号などの記号，それから作られる項や論理式の記号列であらわされる。形式系においては，記号も推論規則もそれ自体に意味はない。このような形式的記号体系において無矛盾性を証明するには，記号を対象とする数学の分野をヒルベルトは，超数学と呼び，その方法論として有限の数学の立場を提唱した。そこでは，有限個の記号の対象とそれらに対する有限の操作という方法に基礎をおき，数学の厳密性を保障している。

このように数学おける形式系では，記号も推論規則も意味をもたないのである。

3.2 数学の構造

ブルバキは，統一された思想の下で，数学の世界を体系化するために「数学のアーキテクチャー」（L’architecture des mathematiques）［6］を提唱した。ブルバキが数学原論を執筆するうえで，その構築哲学あるいは数学観を記述したものである。

ブルバキは，数学の形式系としての公理主義的方法に抽象化された構造を導入して，統

一のとれた数学の体系を構築しようとした。公理主義的方法は，原始的な概念の関係を公理系で定め，あとは，形式的な推論規則から性質を導くことである。

ブルバキは，この公理主義的方法の意義を次のように述べている。「公理主義的方法は，その発見の深い理由を探り出し，初めの理論のおのおのに特有な細目のために，その外衣の下に隠されている共通の本質を見つけ出し，それらの本質を抜き出してそれを白日の下に置く，こうしたものを教えてくれるのである。」（p 35）

このように，公理的方法はものごとの本質を発見する手段として有用であると主張している。さらに，ブルバキは，抽象的な構造の基本として，群，環，イデアル，体などの代数構造，集合の要素の順序付けるなどの順序構造，近傍，極限，連続などの概念を規定する位相構造を母構造と呼び，これらから，一般的な数学体系を組み立てることとした。

抽象化した構造の作り方は，「一つの数学的構造を定義するためには，それらの要素を結び合わせる一つあるいは二つ以上の関係を与えればよい。次に，この与えられた関係が，いくつかの条件（それらは列挙される）を満足させることを要請する。これが考えている構造の公理である。」（p 39）とすればよいとした。このような抽象化は，その要素のもつ固有の性質を捨てることである。この原則は，単純なものから複雑なものへ，一般から特殊へなどと構造の階層化をもたらす。その結果，基本的な構造の数を増やしたり，新しい構造の発見が数学を発展させるとしている。

ブルバキの思想をまとめると，「公理主義的方法は，１つの思考の経済を少なからず実現できるという点である。構造とは，数学者にとってはいわば道具のようなものである。」（p 41）に集約できる。

3.3　ソフトウェアの形式化

形式的技法がソフトウェア工学の一分野として研究されている。形式的技法の正確な定義は難しくいろいろと幅広く用いられているが，その基本的方向は，数学や記号論理学のような厳密性を与える記述形式を求めている。大きく分ければ，ソフトウェアの仕様を形式的な言語で記述する形式的仕様記述の分野とプログラムの性質を形式的理論で検証する分野がある。

ソフトウェアの形式的仕様記述には，数学のような公理的方法を用いる手法と，特定の視点からソフトウェアの内部構造を構成的に定義するモデルを作成する手法（モデル化技法）とがある。

形式的仕様記述には，形式的言語を使うので，自然言語で記述したものより精密になるとされ，仕様記述にはコストがかかるが初期の段階でバグの発見に結びつき，ソフトウェアの品質の向上になるとされている。また，仕様の正しさの検査や分析を自動化することができるといわれている。いずれの場合も記号論理，代数，集合などの数学を使う。

たとえば，公理的方法は，仕様を抽象データ型などの代数で記述しているが，その内容は実装の世界での記述であり，先に述べたような仕様の概念，適用領域と機械との相互作用の記述にはなっていない。

これまでのソフトウェアの形式化の試みは，ソフトウェアの部分に対してであり，全体的で包括的な方法はまだ提案されていない。

数学の形式化や構造化のもっとも重要なことは，数学における記号や規則は意味をもたないものであり，構造は数学的事象の発見の道具であるということである。ソフトウェアの記述で使用される記号は，実世界の適用領域の意味を担ったものであり，構造は適用領域での振る舞いをあらわすものである。このような相違を十分に考慮したアプローチが要請される。

4．仕様記述

4.1　同一性

ソフトウェアという人工物の設計において記号（概念）は非常に重要な役割を果たしている。思考対象としている世界を分節して，思考の結果を主張したり，他と区別したりするための名前付け，すなわち，命名という行為，は設計の基本である。要求定義書，仕様，設計書，プログラムなどの記述物に使われる記号や構造は，先に述べたように，その設計者が設定したもので恣意的である。

仕様は，顧客，設計者，開発者で，その内容の解釈，認識は，同一である必要がある。ソフトウェア開発における曖昧さや不完全性の要因は，このような同一性が保証されていないことに起因しているので，仕様の同一性をいかに保証するかは，仕様記述の最大の問題である。

特に，現在のソフトウェアの大きな問題の一つに，顧客の要求と実現されたシステムの機能との不一致がある。顧客の要求は，顧客がシステム関して求めている事項であり，顧客の要求は機械に対するものではなく，適用領域に関するものであり，また，要求が正確なものであるとは限らないから，設計者は文脈を理解して正確な記述を仕様に反映させなければならない。顧客も得られた仕様については理解して，要求の実現を確認できなければならない。

ソフトウェアには，図１が示すように，実世界における適応領域における仕様の記述と機械世界のプログラムの記述との世界がある。それぞれの世界での記述の概要を表１にまとめておく。

仕様の実現性は，機械世界の記号で正確に記述できることである。それには，前提条件として，仕様には記述内容の無矛盾性と記述漏れがないという完全性が満たされなくてはならない。

ソシュール［４］は，コトバの同一性，すなわち，話し手から聞き手に同じ意味を伝えることができる仕組みとしてラングという概念を提案した。このラングの共有こそが，伝達する意味の同一性を実現するものであるとした。ラングは，そのコトバを使う人々が構成している社会での規則である。

この考えに従えば，ソフトウェア工学に必要なのは，仕様などの記述に関する規則，すなわち，同一性を保証するような記述体系の設定である。確かに，機械工学や電気工学な

表１　仕様とプログラムの記述特性

	仕　様	プログラム
対象	実世界	機械世界
記述	自然語，図式，表 仕様記述言語	プログラミング言語
意味の基盤	記述内容	機械語
名前の恣意性	機能名，データ名	変数名，データ名，関数名
構造の恣意性	機能の分割，データ構造	モジュール，クラス，関数
厳密性	なし	プログラミング言語
完全性の検証	不可	コンパイラ
同一性の確認	不可	プログラム

どの伝統的な工学では，いろいろの設計も詳細な分野に細分化され，それぞれ専門化された精確な用語ならびに術語が設定されている。ソフトウェアの対象とする分野が，ほとんどすべての工学や科学を対象とするため，記述に使う用語や術語はユニークに定義されていない。まずは，このような共通了解の制度としての記述体系を設定することが必要である。

4.2　アーキテクチャー

ソフトウェア開発は問題解決からプログラム作成までの設計過程で持続的な思考活動を要する。設計は創造と理解の相互作用である。この相互作用を包括する指針をアーキテクチャーと呼ぶことにする。自然科学は対象世界の現象を観測して，個体には記号（概念）を付与し，個体と個体の関係には構造を組み立て，普遍の形式を構築し，段階的に客観的にしていく行為である。

ソフトウェアの場合は，適応領域で設定された概念を通して記号の機械を構築していくことである。記号に担わせる概念の役目が大きく，実世界が概念によって分節されたら，その概念は，設計者の認識様式と思考様式を制約する枠組みとなる。アーキテクチャーの記述は，図３に示すように，このような主要な概念を定義し，問題から実装までの一貫した設計指針を明確にし，仕様の実現性を保証する情報を与えるものでなければならない。

コンピュータの世界で，最初にアーキテクチャーというコトバを使ったのは，Amdahl［７］たちである。Amdahl たちは，アーキテクチャーとは，コンピュータの設計哲学と考えた。詳細には，「アーキテクチャーとは，プログラマからみたシステムの属性，すなわち，データフローと制御，論理設計，物理的実装の機構とは明確に区別した概念構造と機能的な振る舞いのことである。」と定義した。この内容は，コンピュータの機械語セット，すなわち，プロセッサの概念的な構造と機能的な振る舞い情報を提案したものでプログラ

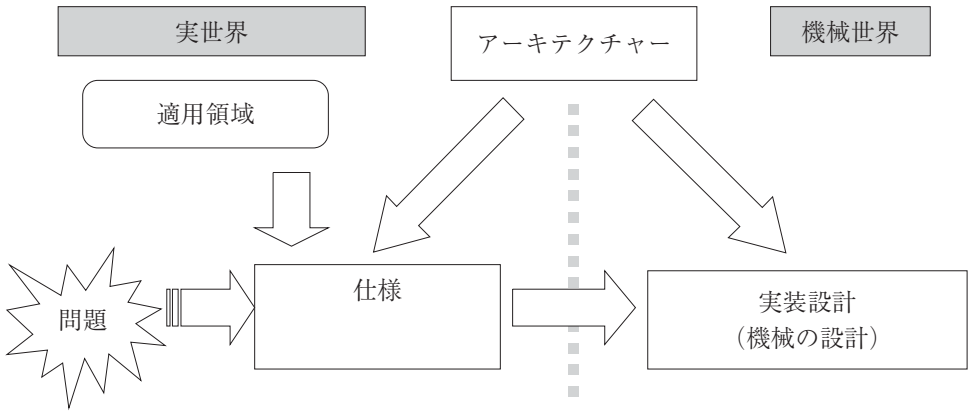

図3　アーキテクチャー

マにとっての指針の提供であった。

さらに，マイヤーズ［8］は，「アーキテクチャーとは，計算システムのあるレベル，あるいはある境界において機能を分割することおよびその境界を正確に定義すること」（p 2）と述べている。このようにするには，計算システムの全体像を与えるための設計方針なり思想がアーキテクトに求められている。言い換えると，マイヤーズは，コンピュータ・アーキテクチャーとは，システムの機能をハードウェアとソフトウェアにどのように分担させるかを決定し，それらの境界のインターフェースを定義することであるとしている。

また，ブルックス［2］は，ソフトウェア開発において，「コンセプトの完全性こそ，システムデザインにおいてもっとも重要な考慮点だと言いたい。一つの設計思想を反映していれば，統一性のない機能や改善点などは省いたシステムの方が，優れていてもそれぞれ独立していて調和のとれないアイデアいっぱいのシステムよりましである。」（p 36）と述べている。ここで，コンセプトの完全性とは，デザインの一貫性であることを主張している。

このように，アーキテクチャーというコトバは，いろいろの場面で多様な使い方がされているが，われわれは，ソフトウェア開発において，構築する機械の全体的構想であり，機械を構築するための思想，原理の一貫性を与えるものを明確に記述したものとしたい。そうすることによって，アーキテクチャーは，先に述べた仕様記述の同一性を保証するための有効な手段のひとつになると考える。

4.3　複雑性

ソフトウェアには，規模の大きいものや本質的に複雑なものなどが存在する。規模が大きくても互いに依存関係が少なく構成要素が多いものは，小さい単位に分解して取り扱えばよいが，そうでないものは，要素の増大に対して，その要素は互いに非線形で依存し合うので，複雑性の増加は膨大なものとなる。

ブルックス［2］は，ソフトウェアの複雑性について，「ソフトウェアの複雑性は本質的

な性質であって，偶有的なものではない。したがって，複雑性を取り去ったソフトウェア実体の記述は，しばしばその本質も取り去ることになる。数学や物理学は，複雑な現象を単純化したモデルを構成し，そのモデルからある性質を引き出し，実験的にその性質を証明することで，三世紀にわたって偉大な進歩を遂げた。この方法でうまくいったのは，モデルで無視された複雑性が現象の本質的な性質ではなかったからだ。複雑性が本質である場合にはこの方法は使えない。」（p 169）と述べている。

ここには，二つの重要なことが主張されている。

ひとつは，複雑性が本質的なものは，単純化できない。すなわち，複雑なものは複雑にしか取り扱うことができないということである。

もうひとつは，数学や物理学のような単純化したモデル化のアプローチは適応できないということである。

しばしばソフトウェアの開発においてモデル化技法が使われているが，その多くは上記の指摘に反した使い方をしており，間違ったアプローチのものもある。

モデルとは，実世界の対象をある視点から抽象化し，それを一定の記述形式あるいは言語のもとで表現したものである。抽象化は情報を取捨選択し，情報を圧縮することである。情報が圧縮されることによってモデルの不確かさが生じるので，実世界の情報を完全に網羅するにはモデルの集合を必要とする。対象を一つのモデルで表現するのではなく，可能な視点からのすべてのモデルを作成する。このようなモデルの集合を作成することで，情報の不完全さをなくせる場合もあるかもしれない。しかし，この方法では，いったん対象を要素に分解し，要素をモデル化してそれらを組み合わせて全体を構成することで，もとの複雑性の要因である本質が失われているかもしれない。

現在使用されているモデル化技法の多くは，このような問題点の検証を十分していない。

4.4　相互作用

仕様は，適用領域と機械との相互作用の記述である。従来の仕様記述は，ジャクソン［３］が指摘しているように，ほとんどが機械世界の用語，概念，モデルなどを用いて，適用領域の振る舞いなどのシミュレーションを記述していることが多い。その結果，顧客が仕様を正確に理解することを困難にしている。仕様は実世界の適用領域の用語，概念，事象，モデルでもって記述することにしなければならない。しかし，現在この分野の系統的な記述方法論や言語体系が存在しないので，それらを新たに開発していかなければならない。

さらに，コンピュータは益々多様な分野で利用されることになるし，コンピュータ自身が新しい分野を創造していくので，適用領域はどんどん広がっていく。このような状況にいかに対応していくかも課題である。

また，相互作用の形式的なアプローチには，最近，計算の概念としての相互作用を提唱している Milner［９］の π-calculus の適用が有効であると考える。

5．ま と め

ソフトウェア開発を正確で安定した過程にするには，仕様を厳密かつ精密にすることであることを述べた。今後，以下のことを考慮して仕様記述の体系を発展させていかなければならない。

⑴ 仕様は，顧客，設計者，実装者の間での解釈の同一性が確保されなければならない。それには，記述言語の記号の恣意性と構造の恣意性を考慮した記述体系を開発する。

⑵ ソフトウェア開発全体の方針を構想するアーキテキチャーの記述が必要である。つまり，人工物を構築する設計思想を明確にする。

⑶ 本質的に複雑なソフトウェアの開発には，複雑なものをそのまま扱う技術を開発する必要がある。いわゆる生命系，物理系などの複雑系の成果を利用する。

⑷ 仕様は，適用領域と機械との相互作用である。相互作用の形式的記述には，Milner［9］の π-calculus の適用などを考える。

ソフトウェアは日々進化している。ソフトウェアは記号による人工機械であるので，容易に変化させることができるし，簡単に新しいものを生み出すこともできる。このようなソフトウェアの特徴を発展させるためにもソフトウェア工学を厳密かつ精密な学にすることが望まれる。

【参 考 文 献】

［1］ Naur, P., and B. Randell「SOFTWARE ENGINEERING」Report on a conference sponsored by the NATO SCIENCE COMMITTEE Germany, October 1968

［2］ ブルックス，フレデリック・**P, Jr.**「人月の神話　狼人間を撃つ銀の弾はない――」滝沢徹・牧野祐子・富澤昇訳　アジソン・ウェスレイ・パブリッシャーズ・ジャパン(株) 1996年4月

［3］ ジャクソン，マイケル「ソフトウェア博物誌――世界と機械の記述」玉井哲雄，酒匂寛訳　トッパン　1997年4月

［4］ ソシュール，フェルナンド・ド「一般言語学第三回講義――コンスタンタンによる講義記録」相原奈津江，秋津伶訳　エディツト・バルク　2003年4月

［5］ 竹内外史「集合とはなにか」講談社ブルーバックス　昭和51年9月

［6］ ブルバキ，ニコラ「数学の建築術」pp 31-48，ル・リヨネ編「数学の流れ　1」村田全訳　東京図書　1974年1月

［7］ Amdahl, G. M., G. A. Blaauw, and F. P. Brooks, jr.,「Architecture of the IBM System/360」, IBM J. Res. Develop., Vol. 8, No. 2, 1964

［8］ マイヤーズ，G. J.「コンピュータ・アーキテクチャの設計」渡辺勝正訳，共立出版，昭和56年1月

［9］ Milner, R.「Communicating and Mobile Systems: π-calculus」Cambridge university press, 1999