特　集　「知的プログラミングの基礎」

ソフトウェア再利用技術の動向

Trend of "Reusable Software Engineering" Tools

千吉良英毅* 小林　正和*
Eiki Chigira　Masakazu Kobayashi

*　(株)日立製作所システム開発研究所
Systems Development Laboratory, HITACHI Ltd.

1987年6月8日　受理

Keywords : program parts, reusable software engineering engine, software productivity improvement.

1. はじめに

ソフトウェアの生産性・信頼性を確保する技術の1つとして，ソフトウェア再利用技術がある．これは，ソフトウェア開発に当たり，再利用できる既開発ソフトウェアを最大限流用し，新たに開発するソフトウェアの量を最小限にしようというものであり，ソフトウェアの生産性と信頼性を高めるうえで，ソフトウェア生産現場では，実用上きわめて重要な技術となっている．

従来，ソフトウェア再利用は，最終生産物であるプログラムの再利用が主であった．それは，しばしば使われるソフトウェアの機能をプログラムレベルで部品化し，再利用していく方式である．

現在は，プログラム部品を想定しながらソフトウェア設計を行う方式が実用化され，生産性・信頼性向上に寄与している[24]．

ソフトウェアの製造工程の改良から始まった再利用技術は，設計工程の改良へと進み，現在，技術開発の重点は，要求分析，要求定義など仕様構築工程に移りつつある．

この変遷は，ソフトウェア生産現場で発生した過去のトラブルの変遷とも一致している．

本稿では，ソフトウェアの生産現場から観たソフトウェア再利用ツールについて述べる．次節では再利用ツールの基本モデルについて考察する．3節ではアメリカにおける技術開発動向の概況を記し，4節では仕様獲得を指向した再利用ツールの事例を簡単な例題を用いて紹介する．

2. ソフトウェア再利用支援ツールの具備機能

2・1　ソフトウェア生産ツールに対する要求

ソフトウェア生産は，対象業種別に分けて実施している．すなわち，ビジネスソフトウェア，交換ソフトウェア，端末機組込みソフトウェアなど，業種別に組織化し，生産している．さらに，ビジネスソフトウェアの中でも，金融，流通，製造，病院等々，端末機組込みソフトウェアでも，流通端末，金融端末，自動販売機など細分化してソフトウェア開発を行っているのが実状である．

このような方法は，対応した業種ごとに専門家（システム設計者，ソフトウェア設計者など）が育成でき，業種に対応した知識と経験が，それぞれの技術者に蓄積できるメリットがある．同時に類似ソフトウェア生産物も一箇所に蓄積できる．これがソフトウェア生産現場の現状である．

以上のような環境で，ソフトウェア生産性・信頼性を維持し，向上させていくために必要となるソフトウェア開発支援ツールは，各環境に最適なツールでなければならない．現場では，汎用性よりも最適化を重視する．業種に対応したソフトウェアを，同一業種について次々と生産した経験・知識から，標準的な設計技法を構築し，標準的な生産物を設定している．各現場に導入できるソフトウェア開発支援ツールは，各現場で構築した標準設計法に基づいてソフトウェアが生産できるものでなければならない．このことは，ツールが適応対象ごとに経験や知識を吸収する機能が必要

であり，また，適用対象ごとにソフトウェア生産物を再利用するための，支援機能を具備すべきことを示している．ソフトウェア生産現場から観ると，ソフトウェア再利用支援ツールは，きわめて実用性の高いツールであると考えられる．

2・2 “経験・知識”の標準化とツール化の過程

特定の業種に開発対象を限って，ソフトウェアを生産する場合，次のような手順で標準化・ツール化に至るケースが多い（図1参照）．

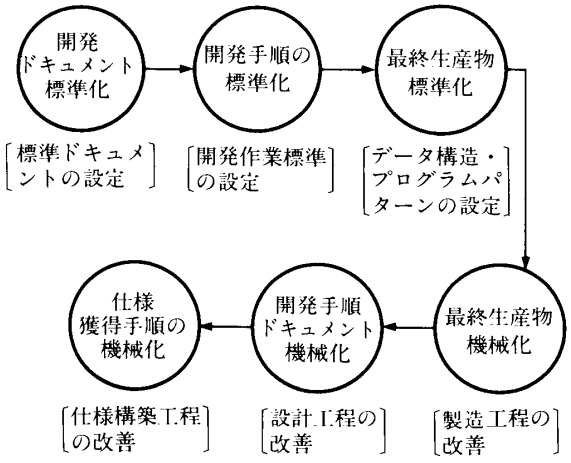

図1 ソフト生産現場における生産物・開発手順の標準化機械化過程

〔1〕 開発ドキュメントの整備

通常，世の中で普及しているドキュメントのフォーマットを利用し，設計・開発を開始するが，生産性を上げるために独自の記法，略号を用い，開発対象を指向したドキュメントに変更していく．やがて，1つの組織の中の標準ドキュメントとして定着する．

〔2〕 開発手順の標準化

開発ドキュメントを整備すると同時に，ドキュメントの作成順を決めることが多い．これは，1つの組織の中での設計・開発手順を標準化したことにほかならない．

〔3〕 最終生産物の標準化

類似プログラムを多く開発してくると，限られた個数の処理を頻繁に使っていることに気がつく．ここから，処理に着目したプログラム部品を設定することが多い．また，設計開発時にデータに着目した，いわゆるデータ中心の設計を実施している場合は，データ構造の標準化を行うことになる．

〔4〕 最終生産物の操作支援

標準プログラム部品の修正・編集を，機械を用いて実施することから始まり，最終生産物の標準化と操作支援のための機械化は，同時に試行するものと考えられる．標準部品を機械で扱うためには，プログラム部品を管理し，必要な部品を検索し，結合したり修正するための支援環境を備えることが重要である．

〔5〕 開発手順・開発ドキュメントの操作支援

標準開発手順は，対象とする業種のソフトウェア開発に関する技術者の知識・経験に基づいて構築される．知識を再利用しながら設計作業を実行し，その結果，各種ドキュメントを作成している．この工程は，対象とする業種により知識の内容が異なることから，機械化に当たっては，強力な知識処理機能を考慮しなければならない．

〔6〕 仕様獲得手順の機械化

ソフトウェアの設計工程，製造工程が改善されると，残る大きな問題は，要求の漏れのない獲得である．これまでの過程は，どうやってソフトウェアを作るかということが問題であったが，ここでは何を作るかということが問題になる．限定した対象に対し，経験を積んだ技術者が，要求に基づく仕様定義法を標準化し機械化してゆくものと考えられる．

ソフトウェアの生産は，開発対象を限定し，その範囲内でソフトウェアの開発に関する経験・知識・生産物を蓄積する．その中から上述したような経過をたどって標準化・機械化が進められてきている．

2・3 ソフトウェア再利用エンジン

ソフトウェア生産は，非形式的な記述による要求仕様から，形式的に厳密な仕様プログラムへ，段階的に仕様を詳細化してゆく過程ととらえることができる．この変換過程は，さらに「何を開発するかを決定する」という仕様構築の段階と，「どう実現するか」という設計・製造の段階に分けて考えることができる．現在，実用的な再利用支援ツールとして実現しているのは後者を対象としたものである．

設計・製造工程は，各ソフトウェアの生産手順ごとに最適な中間工程を設定し，分割運用されている．各部分工程では，前工程の生産物を入力とし，これを加工し新たな中間生産物を，次工程へ出力する．

このような設計・製造工程で繰り返し再利用できるものは次の2項目である．

① 生産物（仕様書，プログラム，テストデータなど）

② 開発手順（対象業務の知識，開発経験・知識など）

ソフトウェア開発作業で蓄積してきた開発手順を再利用し，開発手順に含まれた知識を集約したものが生産物である．生産物は，それ自体知識の塊りと考えることができる．

ソフトウェアの生産で有効となるソフトウェア再利

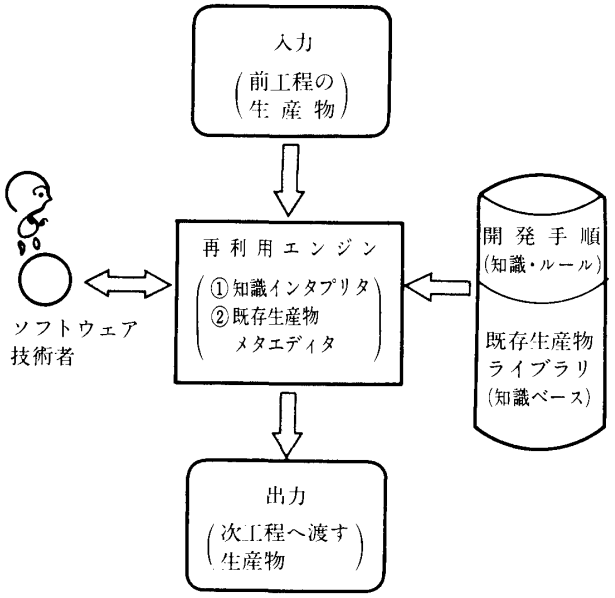

図 2 ソフトウェア再利用エンジンのモデル

用支援ツールは，各工程ごとに上記2項目の知識を一体化して使用できるものでなければならない．このような前提条件のもとで，ソフトウェア再利用支援ツールのモデルを考察する（図2参照）．

前工程の生産物（仕様書）を入力とし，次工程へ渡す生産物（仕様書，プログラムなど）を出力する再利用エンジンには，たとえば，プロダクション・ルール表現などで記述した開発手順知識を実行解釈し，入力を出力に変換する機能（知識インタプリタ）と，各ソフトウェアの生産で現実に使用している生産物を編集する機能を提供する，メタエディタの機能が最低限必要である．

ソフトウェアの生産にソフトウェア再利用エンジンを導入する場合，以下のような問題に対処できることが必要である．

① ソフトウェア生産現場ごとに異なる中間生産物の形式

② 顧客仕様の差異による生産物の構造の変更

ソフトウェア生産現場では，対象業種ごとに生産物および開発手順の標準化を行っている．たとえば，交換ソフトウェアの開発では，状態遷移図を中心とした標準化が行われているが，ビジネス系の開発は，データダイアグラムを中心とした標準化が進んでいる．

ソフトウェア再利用エンジンのメタエディタは，中間生産物も含め，すべての生産物を，1枚ごとの図面と見なす．図面上で使用する表記法（シンボル，シンボル使用規則，図面構成文法など）の定義と図面間の関係の定義およびエディタの操作法の定義を与えることにより，各ソフトウェアの生産に合致した生産物エディタが実現できる．

顧客仕様による差異は，さらに2つの問題に分けて考えることができる．

一般に顧客が異なれば，要求仕様は異なることを見込んで開発手順を標準化してある場合と，標準化した開発手順を越えるような要求仕様に対処しなければならない場合である．

前者は，各現場ごとの開発手順をプロダクション・ルールやフレームなどの知識表現を用いて記述し，再利用エンジンにある知識インタプリタを用い，開発手順を再利用できる．入力となる前工程の生産物から，次工程へ渡す生産物への変換のためのルールが完備している場合，ルールをアルゴリズム化し，再利用エンジンに埋め込み，ジェネレータとして，開発手順を効率よく再利用することも可能である．

標準化した，開発手順を越えるような要求仕様に対処するケース，または，標準化した開発手順の一部分を知識化してあるようなケースでは，知識化の範囲内の作業を再利用エンジンに対応させる．残る試行錯誤的要素の多い作業は，ソフトウェア技術者が主体となり，既存のソフトウェアの事例を参照しながら，対話型変換作業を実行できることが必要である．

ソフトウェア再利用エンジンは，ソフトウェア生産の状況に合うよう，知識処理，編集処理，対話環境，類似例表示機能を整えておくことが必要である．

3. アメリカにおけるソフトウェア再利用技術の概要

生産物と開発手順を再利用の対象とすることはアメリカでも同じである．

プログラム部品を数多く（数百～数千個）備えたツールでは，プログラム部品を操作する対話型環境を充実させている[7]．一方，開発手順を再利用するシステムでは，技術開発の当初からAI技術の導入を図っているものが多い[11]-[23]．

AI技術を用いることにより，プログラミング知識の再利用[18]-[20]，対象業務を限定し，共通的なプログラム部品を適用する知識の再利用[22]などが行われている．さらに，AI技術を核技術として導入し，ツール化を図ったものも見られる．要求定義を自然語で実施し，プログラムまで変換する知識を備えたもの[14][15]，超高級言語で要求仕様定義ができ，プログラムまで変換する知識を備えたツール[11]-[13]などが注目される．

限定した対象業務に対する再利用支援ツールは，実用化の段階に入り，さらに汎用性を備え，かつ対象業務ごとに専用化できるソフトウェア生産支援ツールの

研究開発へ進んでいる．

ソフトウェア生産支援ツールの中核にAI技術を設定することにより，仕様獲得の段階へも研究が進んでいる[12]-[15][21]-[23]

4．システム仕様書の再利用によるソフトウェアの開発技法

現在，国内では，システムの設計を支援するツールが実用化されている[24]-[26]．ここでは，システム仕様書を再利用することを指向し，仕様の獲得から設計工程への橋渡しをスムーズに行うことを支援する技法REUSEを紹介する[27]-[30]．

既存ソフトウェアの仕様書を再利用する技術（REUSE）では，新たに開発しようとしているシステムに似ている仕様を備えたシステムの仕様書を，検索することが重要な課題である．これには要求を解読することと，要求の漏れを意味的に補足することが必要である．要求の漏れを推論するために，業務上の概念依存関係をあらかじめ登録することが必要である．

REUSEは，要求がユーザの言葉で定義できること，最大限再利用できる既存システムの仕様書が得られること，などの特徴があり，業務用語で書いた仕様書の迅速な作成や，類似仕様書の再利用による効率的なシステム設計が実現できる．

4・1 設計思想

システム設計では，ユーザの要求を正確に把握し，要求を過不足のないシステム機能で満たすことが目標となる．おおむね次のような作業手順を踏む．

(1) ユーザの要求を，フリーハンドで描いた絵，図，表，情報の流れ図，文章などを用い，各業務ごとに記述（非定型的記述）する．

(2) システム設計者は，ユーザの要求をシステムの仕様として見直し，冗長な部分，欠落した部分を要求元（ユーザ）に確認しながら補い，ユーザと

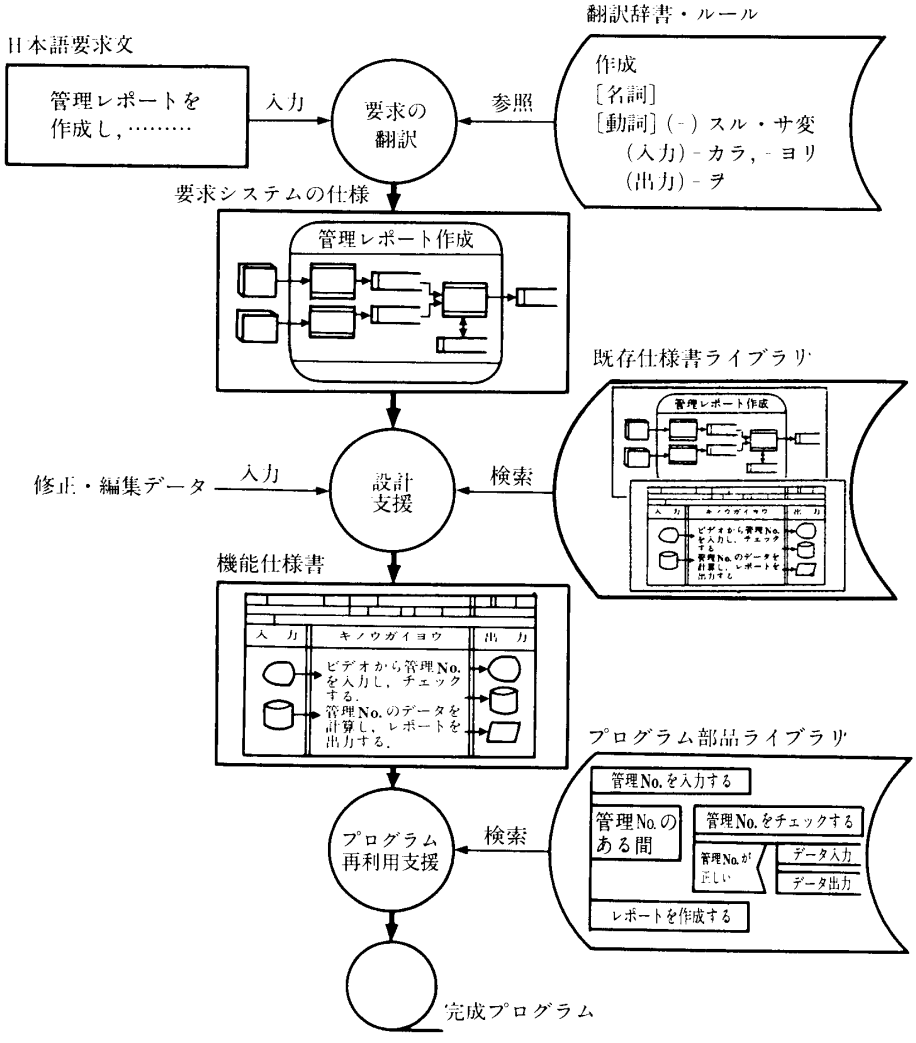

図 3 REUSEの機能の流れ

合意のとれたシステムの仕様書を作成する.

(3) 作成したシステム仕様書に基づいて詳細設計を行う.

このような手順を踏まえ，次のようなREUSEの開発基本方針を設定した.

(1) あいまいな要求から，定型的なシステム仕様書の記述へ変換する過程を支援する機能を導入する(要求の翻訳).

変換過程の支援機能は，次のようなサブ機能で構成する.

① あいまいな要求を定型的な記述の仕様書に変換する機能

② 変換・生成したシステム仕様書から冗長性を削除する機能

③ 変換・生成したシステム仕様書から欠落している情報を付加する機能

(2) 既開発システムの仕様の中から，要求機能を含む仕様を抽出し，システム詳細設計を支援する機能を導入する(設計情報の再利用).

以上の基本方針に基づいたREUSEの操作と機能のフローを図3に示す.

4・2 REUSEで実現した基本機能

〔1〕 要求をシステム仕様に翻訳する機能

翻訳機能は，要求を解読しシステム仕様書に変換する機能と，システム仕様書としての冗長性を削除する機能，欠落情報を付加する機能から構成される.

〔2〕 要求記述手段およびシステム仕様書への変換機能

要求を記述する手段は多数ある．記述する規則が厳密なものほど仕様としての完備性が保証される．しかし，利用者にとっては記述規則を忠実に守らねばならないため，使いにくい場合もある.

REUSEでは，要求記述手段として日本語文を用いた．利用者の使いやすさを考慮し，使用する日本語文に次のような記述内容の制約と構文上の制約を設けた.

(1) 記述内容の制約

① システムの機能を記述すること.

② システムの機能をトップダウン的に記述すること.

(2) 記述構文の制約

① 単文または単文を結合した重文を原則とする.

② システム機能の起動条件を記述する場合，定められた用語(14種)による複文を用いることができる.

③ 前後する2つの文が，互いの機能の出力・入力の関係で意味的に結ばれている場合，指示代名詞を用いた係り受けのある文章を用いることができる.

以上のような制約のもとで，要求を実際に日本語文で記述すると次のような問題が生じた.

① 業務固有の用語が多用されること.

② 業務上常識に当たる内容は，記述されないことが多い.

③ 業務上のデータに関する記述漏れが多い.

④ 業務処理の上位・下位に関する記述漏れが多い.

REUSEにおける日本語処理では，システムの機能を表す動詞の格を定めた(10種類の格を設定)．各格に，物・データ・情報・方法および条件という5種のデータフロー変換属性を設定した．この属性のデータは，用語辞書に登録する.

REUSEは，要求を記述した日本語文(要求文)を，システム機能とデータとの関係図(データフロー図)へ変換する．変換時データフロー変換属性データを利用する(図4参照).

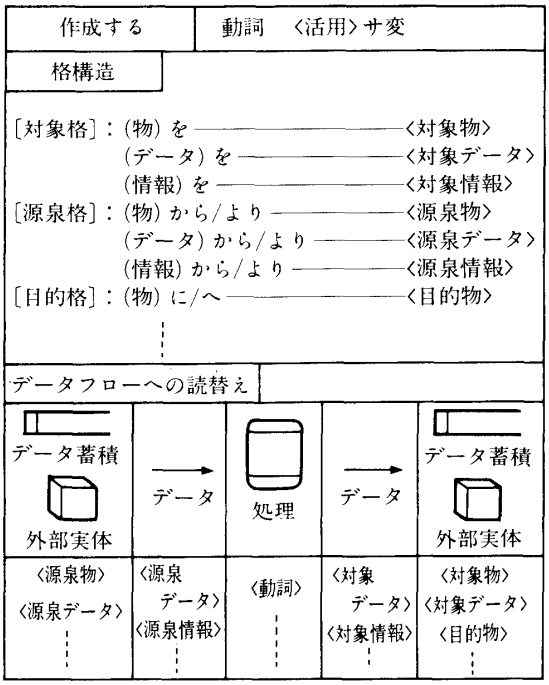

図4 REUSE用語辞書

〔3〕 システム仕様書の冗長性を削除する機能

システムの仕様を記述する方法も多種多用であるが，記述しやすく読みやすいデータフロー図を利用する.

要求文は，複数の文章で構成される．変換機能により各文ごとに1個のデータフロー図を生成し，これらを統合してシステム仕様書を作成する.

設計者は，数個のデータフロー図を指定して統合することも可能である．また，データフロー図の構成要素(システム機能，データ)を指定して統合することも認められる.

本機能により設計者は，機械的にデータフロー図を統合し，冗長性を削除できること，および望むとおりの編集を行うことができる．

〔4〕 **要求文に書かれていない情報の付加機能**

日本語文章の持つ冗長性を考慮すると，要求文にすべての要求が書かれているという保証はない．むしろ記述漏れがある場合が多い．要求文の解析結果に基づいて意味上の補足を行い，隠れた要求を推論する必要がある．

意味付加を目的とした推論を実行するために，一定の規範に基づいて構築した知識体系が重要である．REUSE は，業務固有の概念・物・ノウハウの定義とそれぞれの関係を，フレームと意味ネットワークとを用いて知識化（概念依存関係）し，要求文からの欠落情報を推論（意味付加推論）する機能を備えている．

システム仕様やソフトウェア仕様は，機能中心にまとめる場合が多いため，ある業務の処理手順やノウハウを，処理や機能を中心に体系化することも，よくとられる方法である．しかし，処理や機能は，まとめる人によってとらえ方が異なることがあり，人による差異のない体系を確立することは困難なことが多い．これに対し，業務で実際に使用している物は，特定できるため差は生じにくい．したがって，業務で実際に使われている物を中心に，付随する概念をも含めて体系化を図った．

このような対象物および概念に関し，対象指向モデルを適用し，対象物または概念と業務処理を組にして1つの知識の単位とした．さらに，対象物または概念を分割あるいは抽象化し，知識を構造化した．これらの知識は，分割の場合は“part of”，抽象化の場合は抽象化の尺度名を付け，“is a”の名称で関係づけた(図5参照)．

REUSE は，以上のような知識と推論を実行するルールを用い，要求文に記述されていない隠れた要求を，設計者と対話しながら抽出し，データフロー図へ変換する．

〔5〕 **仕様情報の修正を支援する機能**

既存仕様書を再利用し，システム設計・ソフトウェア設計を支援するためには，既存仕様書の検索機能と仕様書の編集機能が必要である．

REUSE では仕様書ライブラリに既存仕様書を登録し，これを検索し，ワークステーション上で修正編集を行う方式を採用している．

再利用するシステムのデータフロー図を決定すると，関連する仕様書はすべて検索可能となる．したがって，設計者が必要とする仕様書を検索し，再利用エディタ

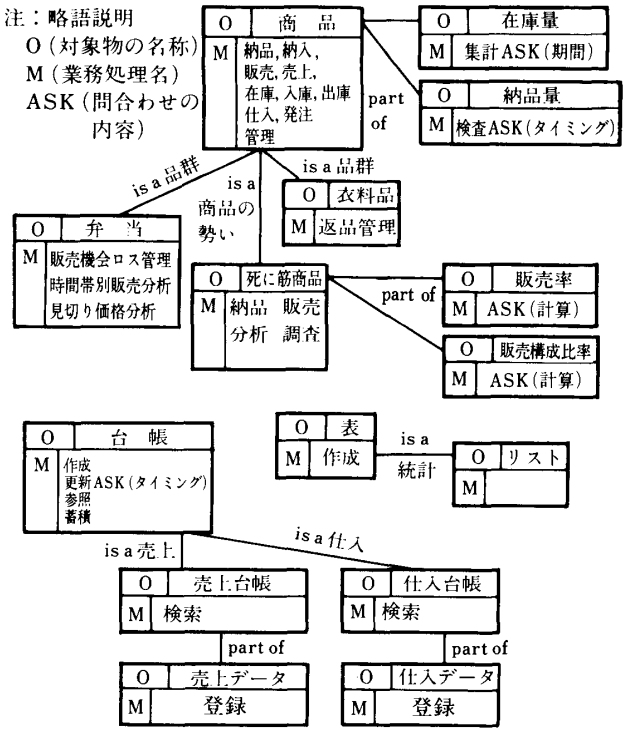

図 5 REUSE 知識体系

で修正編集し，最終的にソフトウェア仕様書を完成できる．

4・3 適　用　例

要求文の解読から，既存仕様書を検索するためのデータフロー図の作成までの手順を，「死に筋商品（売れていない商品）リスト出力」の例を用いて説明する(図6参照)．

要求文「売上データを登録し，死に筋商品リストを作成する」を入力する(図6(a)参照)．日本語解析を行い，「売上データ登録」と「死に筋商品リスト作成」の2種の未完成データフロー図を生成する(図6(b))．未完成のまま両者を統合する(図6(c))．売上データを対象物の中から探し(図5)，「売上データを登録し，売上台帳を作成する」を描出する(作成するには上位の台帳の作成業務からインヘリットする)．死に筋商品を対象物の中から探し(図5)，さらにリストを探し(図5)，両者を合成した新しい知識単位「死に筋商品リスト」を，「リスト」と“is a”(リンク名は死に筋商品リスト)，「死に筋商品」と“part of”で結ぶ．これにより「死に筋商品を分析し，死に筋商品リストを作成する」を描出する．

次に「死に筋商品」と「売上台帳」を合成し，「売上台帳を検索し，死に筋商品を分析する」を得る．図5の範囲内の知識では，品群の選定が残っており，これは問合わせとなる(図6(d))．図6(e)は，設計者が「販

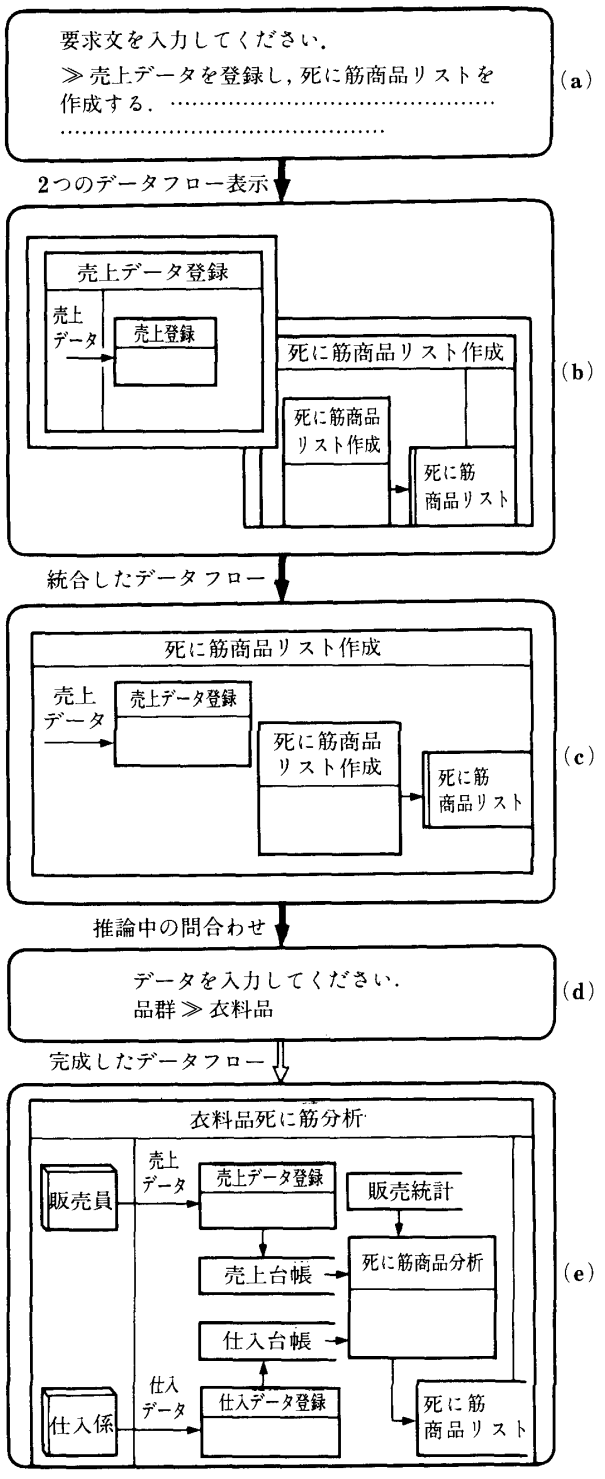

図 6 REUSEデータフロー生成までの画面の流れ

売員，仕入係，販売統計」を編集入力したものである．

4・4 効　　果

〔1〕 システム要求仕様設定の迅速化

粗い表現の要求から，類似システムの外部仕様を修正して顧客に提示できるようになり，従来の人手による方式と比べて，要求仕様設定のための工数，要求定義の漏れによる修正工数を大幅に改善できる．

〔2〕 システム設計工数の低減

段階的な設計詳細化を，類似例を用いて実施できるため，従来方式に比べ端末操作数，端末作業時間，仕様記述と修正工数を大幅に改善できる．

〔3〕 仕様書再利用の促進によるソフトウェア資源の活用

システム開発のために作成された仕様書が，プログラムを含めて再利用の対象となる．すでに開発したソフトウェアの仕様は，ソフトウェアを新たに開発する場合の資源とすることができる．既開発ソフトウェアのすべてがソフトウェア開発の財産となるならば，効果は図りしれない．

5. むすび

これまでソフトウェアの生産に視点を置き，現場のニーズから必要となる技術を中心に述べた．

今日のソフトウェア生産を一般的に述べることは，もとより不可能なことであるが，各業種ごとに設計の専門家が，自分自身に蓄積した経験や知識を使って，顧客仕様をプログラムへ変換してゆく過程と，その機械化の過程を考察した．

「再利用」そのものは，本来，現場では常套的手段として人手で運用されてきたもので，機械化に当たっては，従来の人手で採られていた手順を吸収できること，効率が人手より数段向上することなどが必要である．

人工知能技術を応用した，ソフトウェア生産支援ツールの技術開発が，遠からず，多くのソフトウェア生産に受け入れられる実用的なツールを生み出すことを期待する．

◇参 考 文 献◇

(1) Frenkel, K. A. : Toward automating the software-development cycle, CACM, Vol. 28, No. 6, pp. 578-589, ACM (1985).

(2) Yeh, R. T., *et al.* : Current Trends in Programming Methodology, Vol. 1, Prentice-Hall, Englewood Cliffs, N. J (1977).

(3) Horowitz, E., *et al.* : Practical Strategies for Developing Large Software Systems, Addison-Wesly, Reading Mass (1975).

(4) Kant, E., *et al.* : The Refinement Paradigm ; The

interaction of cording and efficiency knowledge in program synthesis, IEEE Trans. on Softw. Eng., Vol. SE-7, No. 5 (1981).
(5) Osterweil, L. : Software Processes are Software Too, Proc. of 9th ICSE, pp. 2-103 (1987).
(6) Appleton, D. S. : Data Driven Prototyping, Datamation, pp. 259-268, Chaners Pub. Company (1983).
(7) 上谷晃弘編著：統合化プログラミング環境 －Interlisp-D と Smalltalk-80－，ワークステーションシリーズ，丸善株式会社 (1987).
(8) Gilb, T. : Evolutionary delivery versus the "Waterfall Model", ACM SIGSOFT SEN, Vol. 10, No. 3, pp. 49-61, ACM (1985).
(9) Boyle, J. M., *et al.* : Program reusability through program transformation, IEEE Trans. on Softw. Eng., Vol. SE-10, No. 5 (1984).
(10) Shankar, K. S. : Data structures, Types, and Abstractions, IEEE Compu., Vol. 13, No. 4, pp. 67-77, IEEE (1980).
(11) Cheng, T. T., *et al.* : Use of very high level languages and program generation by management professionals, IEEE Trans. on Softw. Eng., Vol. SE-10, No. 5, pp. 552-563, IEEE (1984).
(12) Green, C. : The Design of the PSI program synthesis system, Proc. of 2nd ICSE, pp. 4-18, IEEE (1976).
(13) Smith, D. R., *et al.* : Research on knowledge-based software environments at Kestrel Institute, IEEE Trans. of Softw. Eng., Vol. SE-11, No. 11, pp. 1278-1295, IEEE (1985).
(14) Balzer, R., *et al.* : Informality in program specification, IEEE Trans. on Softw. Eng., Vol. SE-4, No.2, pp. 94-103, IEEE (1978).
(15) Balzer, R., *et al.* : Operational specification at the basis for rapid prototyping, ACM SIGSOFT SEN, Vol. 7, No. 5, pp. 3-16, ACM (1982).
(16) Prywes, N. S. : Automatic Generation of Computer Programs, in Advances in Computers, Vol. 16, Rubinoff, M., *et al.*, pp. 57-125, Academic Press (1977).
(17) Neighbors, J. M. : The Draco approach to constructing software from reusable components, IEEE Trans. on Softw. Eng., Vol. SE-10, No. 5, pp. 564-574, IEEE (1984).
(18) Waters, R. C. : The Programmer's Apprentice ; A Session with KBEmacs, IEEE Trans. on Softw. Eng., Vol. SE-11, No. 11, pp. 1296-1320, IEEE (1985).
(19) Rich, C., *et al.* : Initialreport on a LISP programmer's apprentice, IEEE Trans. on Softw. Eng., Vol. SE-4, No. 6, pp. 456-467, IEEE (1978).
(20) Rich, C. : A formal representation for plans in the programmer's apprentice, Proc. of IJCAI-7, Vancouver (1981).
(21) Wasserman, A. I., *et al.* : Extending state transition diagrams for the specification of human-computer interaction, IEEE Trans. on Softw. Eng., Vol. SE-11, No. 8, pp. 699-713, IEEE (1985).
(22) Ramanathan, J., *et al.* : Use of annotated schemes for developing prototype programs, ACM SIGSOFT SEN, Vol. 7, No. 5, pp. 141-149, ACM (1982).
(23) 山本 裕 (訳) : ISDOS の PSL，「bit」臨時増刊号，Vol. 10, No. 10, pp. 1192-1216, 共立出版 (1978).
(24) 角谷一郎，ほか：システム・フローを自動生成する開発支援ソフト EAGLE 2，日経コンピュータ，Vol. 7, No. 7, pp. 121-136 (1986).
(25) 原田 実：ソフトウェア生産性向上ツール (上)，日経コンピュータ，Vol. 3, No. 5, pp. 145-160 (1984).
(26) 原田 実：ソフトウェア生産性向上ツール (下)，日経コンピュータ，Vol. 3, No. 19, pp. 175-199 (1984).
(27) 千吉良英毅，ほか：システム仕様書の再利用によるソフトウェアの開発技法 (ICAS-REUSE)，日立評論，Vol. 69, No. 3 (1987).
(28) 千吉良英毅，ほか：EAGLE におけるプログラム部品合成の概要，情報処理学会第 30 回全国大会予稿集 (1985).
(29) 千吉良英毅，ほか：EAGLE におけるプログラム部品合成実現方式，情報処理学会第 31 回全国大会予稿集，p. 459 (1985).
(30) 内藤一郎，ほか：EAGLE における仕様情報再利用方式，情報処理学会第 31 回全国大会，p. 473 (1985).

著 者 紹 介

千吉良英毅

昭和 44 年東京大学農学部農業生物卒業．昭和 46 年同大学院修士課程修了．同年，(株) 日立製作所入社，現在システム開発研究所第 2 部研究員．病院自動化 (HA) システム，知識処理システム，ソフトウェア生産システムの研究に従事．情報処理学会，AAAI 学会各会員．

小林 正和 (正会員)

昭和 41 年東京大学理学部数学科卒業．同年 (株) 日立製作所中央研究所に入社．同研究所において，計算機言語用の対話型エディタ，高信頼化二重系計算機システムの切替え制御方式などの研究開発に従事．昭和 48 年システム開発研究所設立に伴い，同研究所に転じ，以来ソフトウェア生産技術の研究に従事．ソフトウェア一貫生産システム (ICAS ; Integrated Computer Aided Software Engineering) の開発研究を続けている．昭和 61 年度大河内記念技術賞受賞．現在，第 2 部主任研究員．情報処理学会，電子情報通信学会，IEEE, ACM 各会員．