ドメインモデルに基づくソフトウェア再利用に関する一実験

橋本正明† 廣田豊彦† 横田和久††

ソフトウェアの生産性向上や信頼性向上は現在大きな課題となっており，その対策の一つとしてソフトウェアの再利用が注目されている．本研究では，適用ドメインの概念に着目したドメインモデルを蓄積し，そこから仕様を抽出し，さらにプログラムを自動生成するためのSoftReuseシステムを試作した．SoftReuseシステムでは，ドメインモデルの記述に，ERモデルをベースとした非手続き型の仕様記述言語PSDLを用い，大規模ソフトウェアへの対応として階層的なサブジェクトを組み合わせている．利用者は画面に図的に表示されたドメインモデルを見ながら，対話的に作成対象プログラムの仕様を抽出することができる．そして，抽出されたプログラム仕様はコンパイラによって，CまたはC++プログラムへ変換される．本論文では，このSoftReuseシステムの概要について述べるとともに，実際の社会保険システムを対象としたソフトウェア再利用実験について報告する．実験の結果，適切な支援のもとで，ドメインモデルの利用がソフトウェアの生産性を改善する効果があることが明らかになった．また，本方式の再利用プロセスでは，ドメインモデルの理解にもっとも時間がかかることが明らかになった．したがって，より一層の生産性向上のためには，理解性のすぐれたドメインモデルを記述すること，またドメインモデル理解を支援する手法やツールを開発することが重要であることがわかった．

An Experiment on Reusing Software Based on Domain Model

Masaaki Hashimoto,† Toyohiko Hirota † and Kazuhisa Yokota ††

Software productivity and reliability are the main concerns in present software engineering. Software reusability seems to improve productivity and reliability. In this study, the authors have prototyped the SoftReuse System which stores domain models that describe a universe of discourse of a domain, and extracts program specifications, and automatically generates programs. In the SoftReuse System, domain models are described with the program specification description language PSDL which is non-procedural language based on ER model, combining with hierarchical subjects to be applied to large scale software. The users can interactively extract the program specification to be obtained, consulting a domain model displayed graphically on the screen. Then the extracted program specifications are transformed into C and/or C++ programs by the PSDL compiler. The purposes of this paper are to describe the outline of the SoftReuse System and also the experiment of reusing software in the actual social insurance domain. The experiment estimated that, with appropriate supports, domain models are effective in improving productivity. On the other hand, in this type of software reusing process, it has been shown that understanding domain models requires considerable efforts. Therefore, for improving productivity further, it is important to develop the methods and tools which support understanding domain models.

1. はじめに

ソフトウェアの生産性向上や信頼性向上は現在大きな課題となっており，その対策の一つとしてソフトウェア再利用が着目されている[1]．実際，COBOLのような手続き型プログラムの再利用は，生産性向上や信頼性向上に寄与してきた．一方，プログラムコードの再利用には様々な限界があることも指摘されている．そこで，最近はソフトウェア開発に現れる知識の再利用，すなわちドメインの知識や，開発経験，仕様などの再利用が研究されている[2]．

ところで，仕様を再利用するにはその仕様を理解しなければならない．具体的な個々のシステムの仕様を理解するのは，システムの特殊性が現れるので，必ずしも容易ではない．このため抽象的なモデルの再利用

† 九州工業大学情報工学部知能情報工学科
Department of Artificial Intelligence, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology
†† 横河・ヒューレット・パッカード株式会社アジアパシフィックプロダクト開発本部システム環境開発部
Asian System Laboratory, Asia Pacific Product Operation, Yokogawa-Hewlett-Packard, Ltd.

が望まれている．そこで著者らは，ドメインに現れる概念に着目したドメインモデルを蓄積し，再利用することを試みた．このドメインモデルをカスタマイズして種々のプログラム仕様を抽出することによって，新たに仕様を記述するよりも効率的に仕様を作成できることが期待される．

ドメインモデルを記述する言語には，特に理解性と拡張性が要求される．著者らは従来から，ER モデルをベースとした非手続き型の仕様記述言語 **PSDL**(Program Specification Description Language)[3] を研究してきた．そこで本研究では，PSDL に階層的サブジェクト機構を追加してドメインモデルの記述を行うこととした．PSDL では，仕様からプログラムを自動生成するコンパイラを研究中なので，ドメインモデルから個別のプログラム仕様へとカスタマイズすれば，プログラムを生成することができる．

以上で述べたようなドメインモデルに基づくソフトウェアの再利用プロセスを支援するために，ドメインモデルの蓄積や，モデル理解を支援する機能を備えた **SoftReuse システム**[4] を試作し，社会保険システムを対象として，ソフトウェアの再利用実験を行った．従来の研究では，再利用の単位として，小規模の部品を対象としていることが多く，本研究のような規模の大きなシステムを再利用の対象とした研究の報告は見当たらない．また，再利用のベースとしてドメインモデルを用いることは，近年注目を集めているが，実際に再利用実験まで行ったという報告は見当たらない．

本論文では，ドメインモデルに基づく再利用支援システム SoftReuse の概要と，再利用実験の結果，ならびにソフトウェア再利用に関する考察を述べる．第 2 章に関連研究を述べる．第 3 章で SoftReuse システムを説明し，第 4 章に実験を述べ，第 5 章で考察する．

2. 関連研究

ソフトウェアの再利用に関する研究のうち，特に本論文の SoftReuse システムと関連が深い研究について述べる．

（1） プログラム再利用

Lanergan ら[5] は COBOL で書かれた事務処理プログラムを分析して，7 種の制御構造パターンを抽出することによって，60%以上の再利用率を達成した．しかし，まとまった処理を再利用する場合，その内容を理解するのも容易ではないが，さらに再利用部のインタフェースが重要な問題となる．Novak ら[6] は，インタフェースプログラムの自動生成や再利用プログラムの自動変換を用いるアプローチを提案し，LISP 系の言語で実現した．しかし，このような手法を手続き型言語に適用するのは容易ではない．

（2） 仕様再利用

プログラムの再利用には前述のような限界があることから，仕様の再利用が研究されている．千吉良ら[7] は，自然言語で記述した要求をデータフローモデルで表された仕様に変換し，類似の仕様を検索する手法を用いた．しかし，そのような類似検索は，再利用においてそれほど重要な要因ではないとの指摘もある[8]．また，仕様の再利用においては，再利用すべき仕様の理解が重要であるが，具体的なシステムの仕様の理解よりも，抽象的な仕様，すなわちドメインモデルの理解の方が容易であることが指摘されている[9]．

（3） ドメインモデル

近年，再利用のベースとしての**ドメインモデル**が注目を集めている．Prieto-Diaz はドメイン分析の方法として図書分類法に類似した手法を提案し[10]，ドメインモデルの記述としてはデータフローモデルを用いている[11]．しかし，ドメインモデルを構築するためのドメイン分析や，ドメインモデルの記述法について，数多くの課題が残されている[12]．

（4） 理解性と拡張性

ドメインモデルの再利用性を上げるには，モデルの理解性と拡張性が重要である．そのためには従来は**データフローモデル**[13] などが用いられてきたが，近年は**オブジェクト指向モデル**[14] が注目を集めている．SoftReuse システムの PSDL に適用した **ER モデル** (entity-relationship model)[15] は，対象世界を自然に表すという点でオブジェクト指向モデルと共通しており，理解性に優れている．一方，オブジェクト指向モデルでは，処理内容は代入可能な属性と，手続き的な意味を持つメソッドによって記述されるが，PSDL では，単一代入型の属性と，非手続き的な制約によって記述される．このため，PSDL は拡張性にも優れている．

（5） プログラム生成

オブジェクト指向手法や CASE ツール[16] ではオブジェクト指向言語やアクションダイアグラムなどを用いて，プログラムの制御フローを意識した記述を行っている．一方，PSDL はプログラムの制御フローを意識せずに非手続き的に記述でき，MODEL 言語[17] と同じように，コンパイラが入出力データの構造不一致[18] を自動的に検出し解決して，プログラムを生成する．

3. ドメインモデルに基づく再利用支援システム

データベースを共有する複数の応用プログラムは，そのドメインに現れる概念を共有していると考えられる．そこでそれらのプログラム仕様を共通化ならびに統合化することによって，ドメインモデルを構築することができる．そのようなドメインモデルから個別のプログラム仕様を抽出し，プログラム生成を支援するシステムとしてSoftReuseシステムを開発した．

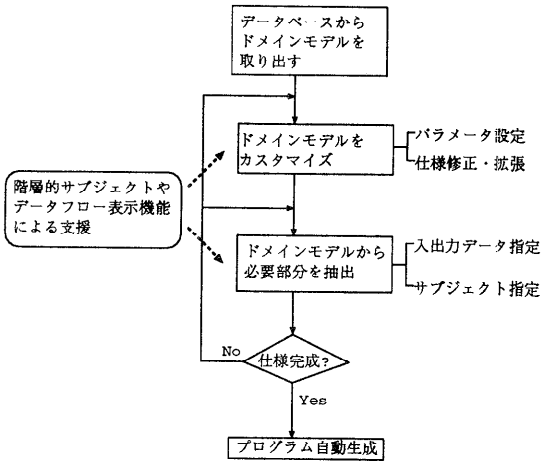


図1 ソフトウェア再利用手順
Fig. 1 Software reusing procedure.

SoftReuseの主な機能は，1) ドメインモデルやプログラム仕様を記述する言語PSDL，2) ドメインモデルの蓄積機能，3) プログラム仕様の抽出機能，4) プログラム生成機能，である．SoftReuseを利用したソフトウェア再利用の手順を図1に示す．

3.1 仕様記述言語 PSDL

簡単な販売管理プログラムをPSDLで記述した仕様を図2に，その一部を図的に表現したものを図3に示す．

PSDLで記述された仕様は情報層，データ層，アクセス層から構成され，それぞれ，INFORMATION文，DATA文，ACCESS文で表される．

情報層では，対象世界の情報構造を**ERモデル**で記述する．E文がエンティティ型を，R文が関連型を表す．また，エンティティの属性値を計算するための**従属性制約**を＝で記述する．図2の例では，product, sale, customerの三つのエンティティ型と，sold, buyの二つの関連型がある．そしてsaleの属性amountと，customerの属性totalに従属性制約が記述されている．

データ層では，入出力データのデータ構造を記述する．この層のデータと情報層との関係は**情報制約**として，＝で記述する．図2の例では，product_data, sale_data, account_dataの三つのデータ構造があり，それぞれのフィールドに対して情報制約が記述されてい

```
INFORMATION
  E product
      EN -50
    A name  STR
     K
    A price NUM
  R sold
    C .product
      RN M
    C .sale
      RN 1
  E sale
      EN -100
    A number   NUM
     K
    A quantity NUM
    A amount   NUM
      = quantity * .sold..product.price
  R buy
    C .customer
      RN M
    C .sale
      RN 1
  E customer
      EN -50
    A name  STR
     K
    A total NUM
      = ASUM(.buy..sale.amount)
DATA
  I product_data
       IX product_id
     G product_record ON EndOfFile(product_data)
    O product_record
        %12s product_name
          = product.name
        %8d  product_price
          = product.price
  I sale_data
       IX sale_id
     G sale_record ON EndOfFile(sale_data)
    O sale_record
      %4d  sale_number
        = sold..sale.number
        = buy..sale.number
      %16s sale_customer
        = buy..customer.name
      %12s sale_product
        = sold..product.name
      %4d  sale_quantity
        = sale.quantity
  I account_data
       IX account_id
     G account_record ON PEntityNumber(sale)
    O account_record
      %4d  sale_number
        = sale.number
        = buy..sale.number
      %8d sale_amount
        = sale.amount
      %16s sale_customer
        = buy..customer.name
      %10d customer_total
        = buy..customer.total
ACCESS
  D product_file INPUT  20 product_data
  D sale_file    INPUT  36 sale_data
  D account_file OUTPUT 38 account_data
```

図2 PSDLプログラム仕様記述
Fig. 2 PSDL program specification description.

る．

アクセス層では，データ層の入出力データとファイルの関係を記述する．図2の例では，二つの入力ファイル product_file，sale_file と一つの出力ファイル account_file が用いられる．

3.2 ドメインモデル蓄積

3.2.1 階層的サブジェクト

一つのドメインを記述したドメインモデルは記述量が非常に多くなる．たとえば，社会保険システムの主要部を PSDL で記述するのに約2,000行を要した．そのため，再利用時にその内容を理解するのが困難になると予想される．そこで，仕様のマクロな理解を支援するため，いくつかのエンティティ型や，属性，関連型，制約をまとめて名前づけができる**サブジェクト**[19]を導入した．

サブジェクトは，階層的な理解を支援するため，階

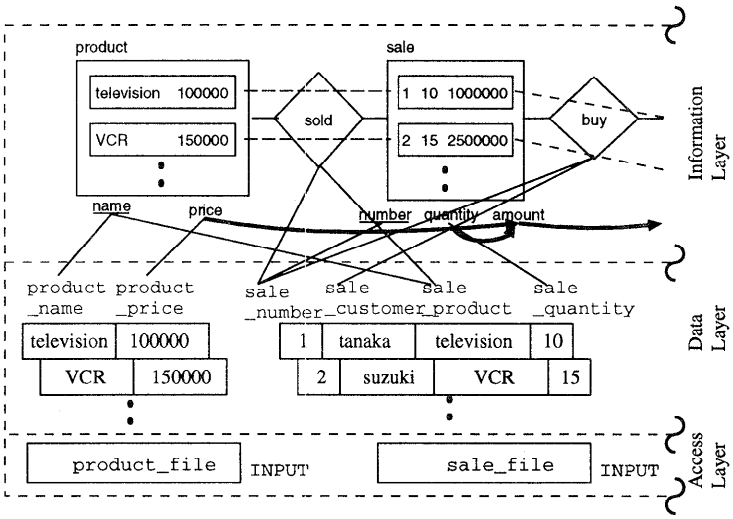


図3 PSDL プログラム仕様図
Fig. 3 PSDL program specification illustration.

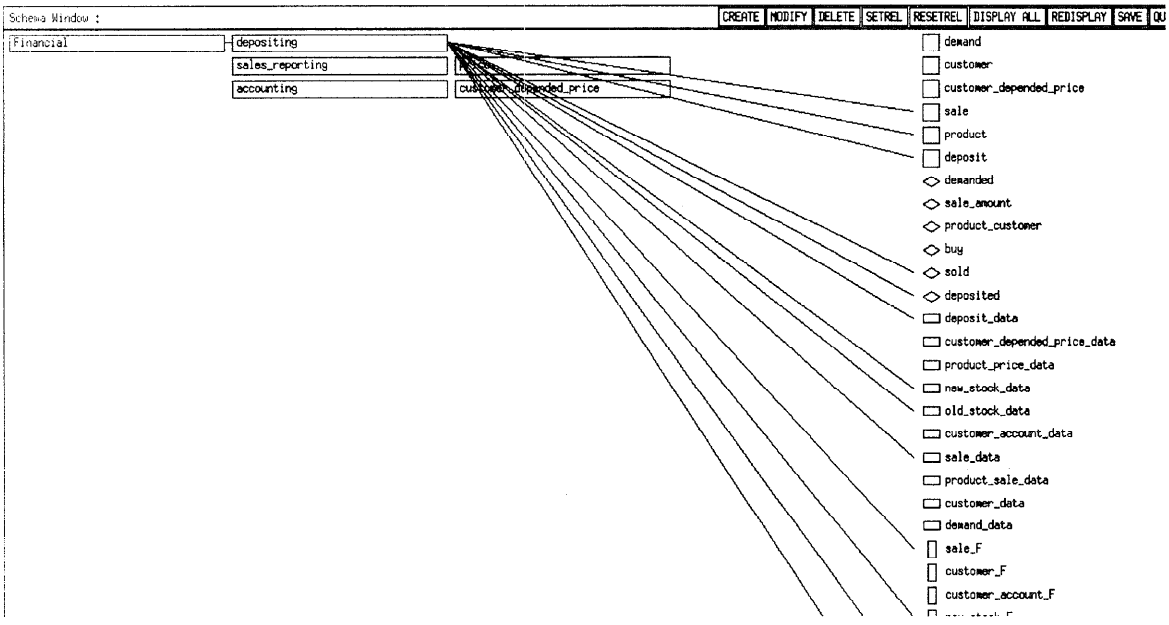


図4 階層的サブジェクト
Fig. 4 Hierarchical subject window.

層的に定義できる．たとえば，上記の社会保険の事例では3階層を用い，その第1階層は，たとえば年金裁定などの業務を表し，第2階層はその業務を分解した処理要素を表した．第3階層はその処理要素の中の複雑な計算を表した．特に複雑な計算についてはさらに階層を深くした．なお，サブジェクトで仕様の代替案を表すため，サブジェクトの間に類似性の関係も定義できる．

3.2.2 ドメインモデル作成

ドメインモデルを作成するには，適用ドメインの対象世界を分析して，最初に**図4**に示すサブジェクト名やその階層を投入する．同図は，第2階層に示す在庫管理や，販売管理，請求書作成の業務を持った販売在庫管理システムの簡単な例を示している．その第3階層の二つのサブジェクトは，製品一律の価格と，顧客別の価格の代替案を表す例である．

次に一つのサブジェクトを指定して，**図5**に示すエンティティ関連図で，エンティティ型と関連型を投入する．属性や従属性制約などの詳細はサブウィンドウから投入する．それらのエンティティ型や，関連型，属性，制約は，前もって指定されているサブジェクトに含まれる．サブジェクトを指定する前に既にエンティティ型や，属性，関連型，制約を投入していれば，マウスで拾ってサブジェクトに含めることができる．

3.3 プログラム仕様抽出

プログラム仕様は以下の手順で抽出する．

（1） ドメインモデル理解

業務概要書で概要を把握してから，階層的なサブジェクトにそってドメインモデルを理解する．その理解を支援するため，1）サブジェクト間のデータフローを，PSDL仕様から自動的に編集して表示する**マクロなデータフロー表示機能**と，2）PSDLの従属性制約を前向き，または後向きに自動的に追跡して表示する**ミクロなデータフロー表示機能**，を付加した．

（2） ドメインモデルのカスタマイズ

ドメインモデル上で，サブジェクトを用いて定義された仕様代替案の中から一つを選択し，さらに修正や拡張を直接加える．そのカスタマイズされた仕様を，

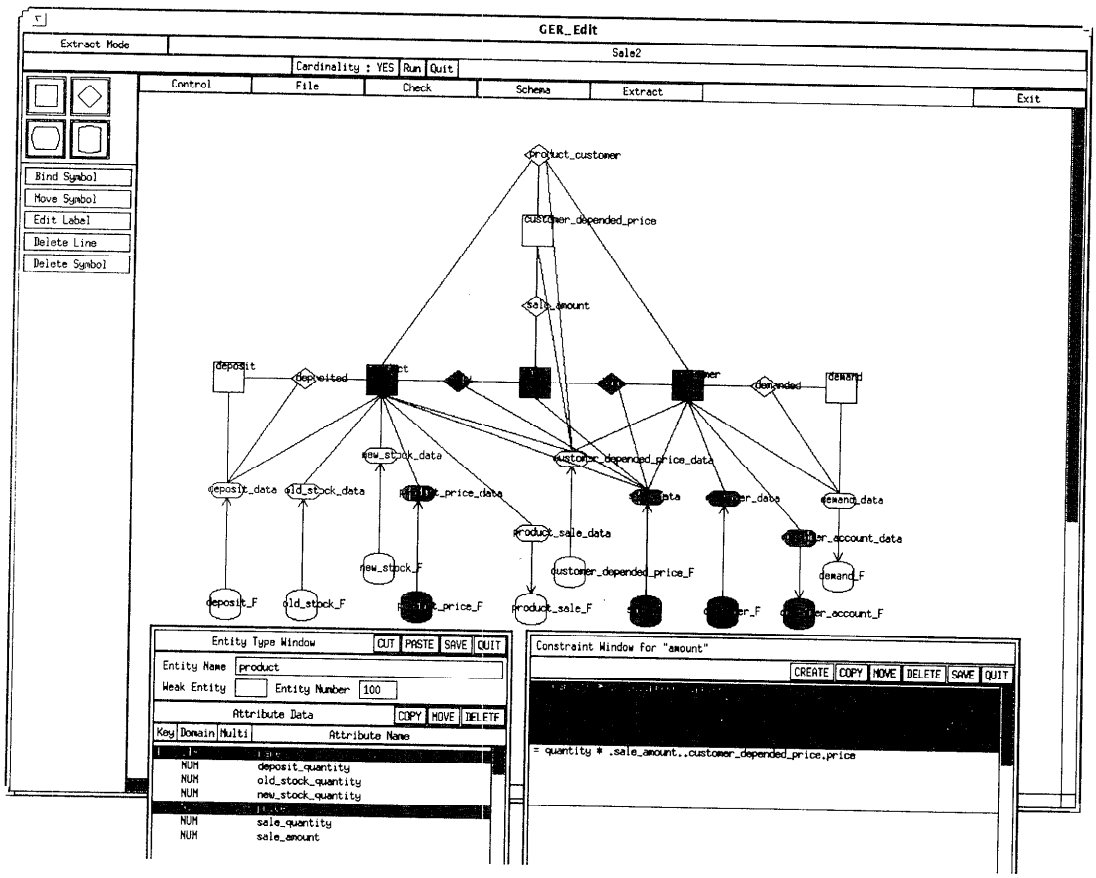


図5 ドメインモデルとプログラム仕様抽出
Fig. 5 Domain model and specification extraction.

開発対象のシステム仕様とする．なお，プログラム仕様抽出後にカスタマイズすることもできる．

（3） プログラム仕様抽出

以下の方法のいずれかで，システム仕様からプログラム仕様を抽出する．1) **サブジェクト指定**—作成対象プログラムをカバーするサブジェクトを指定すれば，その中の仕様を抽出できる．2) **入出力データ指定**—作成対象プログラムの入出力データを指定すれば，入力データから出力データへ至るデータフローパスが自動的に検出され，そのパス上の仕様を抽出できる．

2)では冗長なパスも合わせて検出されるので，1回の抽出で目的のプログラム仕様が得られるわけではない．抽出された仕様は図5のように画面上で反転されるので，冗長なパスはマウスで再反転して除去し，2)の抽出を繰り返す．関連型のカーディナリティ記述によって，ある一つのエンティティにいくつのエンティティ（1個か，n個か，あるいは高々1個）がつながるかが判明するので，それに基づいて冗長なパスが存在しうる箇所を警告する．図5の反転部は図2のプログラム仕様を抽出した例である．なお，1)で抽出された仕様に対して，さらに2)を適用することもできる．

3.4 プログラム生成

プログラム仕様から実行効率の良いプログラムを生成するため，**入出力データの構造不一致**を自動的に検出して，その構造不一致を解決するプログラム構造を自動的に設計して，Cプログラムを生成する．構造不一致を検出するには，プログラム仕様から得られた有向グラフ上で，局所的かつ大局的にデータフローの同期性を解析する．また，構造不一致を解決するには，そのグラフを連結部分グラフに分割する．その結果，スカラ変数とテーブルを使い分けたデータ構造と，そのデータ構造を処理する手続き構造を設計できる．プログラム仕様からC++プログラムを生成するコンパイラについても現在研究中である．

4. 実　　験

4.1 ドメインモデルの生成

実験対象として，実際の社会保険システムを選び，マスターファイルを更新するシステム主要部，すなわち異動記録と，年金裁定，年金調整，年金支給，の業務を実験対象とした．今回の実験テーマは再利用であるので，業務を直接分析するのではなく，個別の業務プログラムのPSDL仕様を統合することによって，ドメインモデルを作成した．ドメインモデル作成にあたって，企業によって異なる部分はパラメータ化などを行ったが，普遍的なデータ形式などは具体的な形式のまま残し，抽象化は行っていない．

4.2 仕様再利用の実験

（1） 実験手順

プログラム仕様抽出の要求条件は以下の三つを設定した．1) 既存のプログラムと同じもの．2) 既存のプログラムを分割したり，合成したもの．3) 掛金の計算法と年金支給頻度を変更するため，仕様を拡張したもの．実験には，PSDLの経験がなく，社会保険の詳細に慣れていない2人の被験者が参加した．その一方はCOBOLの経験が20年あり，他方は3年であった．

被験者は以下の作業を行った．1) PSDL習熟—言語仕様書を学習し，不明な点はPSDLの熟練者が指導した．また，習熟のためにサンプルプログラムを作成した．2) 業務概要理解—ドメインモデルの理解に先立って，業務概要書を理解した．不明な点は業務概要書の作成者が指導した．3) ドメインモデル理解—画面や，そのハードコピー，PSDLソーステキストを用いて，ドメインモデルを理解した．4) プログラム仕様抽出—まずドメインモデルをカスタマイズし，上記のプログラム仕様を，3.3節（3）の入出力データ指定による方法で抽出した．抽出されたプログラム仕様の検証は，コンパイラがまだ機能不足なので，PSDLの熟練者が人手で行った．

（2） 実験データ

3.3節（1）で述べたマクロおよびミクロなデータフ

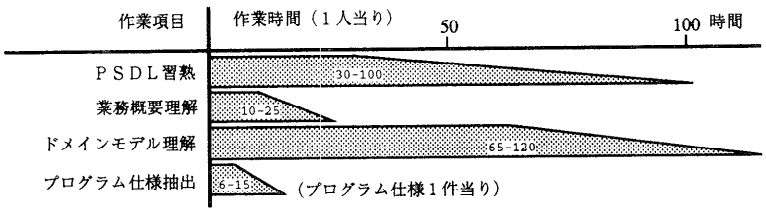


図6 仕様再利用の作業時間
Fig. 6 Man-hours for reusing.

ロー表示機能を付加していない段階で実験を行った．仕様再利用の各作業に要した時間は**図6**に示す．なお，入出力データ指定による抽出方法では冗長な仕様が抽出されるが，第1回目には情報層で平均1.8倍冗長な仕様が得られ，5回程度の抽出を繰り返して目的のプログラム仕様が得られた．ドメインモデルの規模はPSDLで2,020行あり，その源の四つの業務処理プログラム仕様の規模は合計すると，同一概念が重複して現れるので，ドメインモデルより23%大きかった．

ドメインモデルのカスタマイズの具体的な作業としては，パラメータの設定と個別に現れる特殊な処理の記述があった．たとえば，年金の保険料率のうちで企業年金分は各企業によって異なるため，ドメインモデルではパラメータ化し，個別に設定した．また，合併企業の場合，合併の前後の期間を通算するにあたってそれぞれ個別の特例があるため，新たに処理を記述して追加しなければならなかった．

4.3 プログラム生成の実験

PSDLプログラム仕様からCプログラムを生成するコンパイラ[20]は30本強のプログラムから構成され，それらはファイルを介して接続されている．なお，生成時に属性へ割り当てるデータ形式は，一律に数値の場合は単精度浮動小数点数とし，文字列の場合は64文字とした．実験の結果，コンパイラが生成したCプログラムの実行効率は，手書きのCプログラムと比較して1.5倍から2.5倍悪い範囲に収まった．

PSDLからC++へのコンパイラ[21]はまだ十分な大きさの仕様を扱えないため，ここでの実験に用いることはできなかったが，別の小規模の例題でプログラム生成の実験を行った．手書きのC++プログラムと比較して，プログラムサイズで6.3～6.8倍，CPU時間で1.6～3.7倍という結果が得られた．ここで用いたコンパイラは，Cプログラムへのコンパイラと比較して最適化が十分ではなく，今後改良の余地がある．

5. 考　　察

以下，ドメインモデルに基づくソフトウェアの再利用について考察する．

（1） 生　産　性

図6から，1) ERモデルの経験がないCOBOL経験者がPSDLを習熟するには時間がかかる，2) 業務概要者は通常のドキュメントなので，その理解に時間はかからない．3) ドメインモデルは量が多く，記述が詳細なので，その理解に時間がかかる，4) カスタマイズのともなうプログラム仕様抽出に時間はかからない，ことがわかった．

表1 再利用のための想定作業時間
Table 1 Supposed man-hours for reusing.

プロセス	想定作業時間比 PSDL：COBOL	プロセス相対時間比
理　解	1：6	5
仕様化	1：3	3
修　正	1：2	2

実験工数の都合のため，今回は手続き型プログラムの再利用と生産性比較はしなかった．そこで，1本のプログラムを再利用するプロセスを想定し，被験者とSoftReuse開発者のCOBOLとSoftReuse使用経験に基づいて机上評価を行った．**表1**にそのデータを示す．表中の「プロセス相対時間比」は再利用プロセス全体を10としたときの，理解，仕様化，修正の各プロセスの相対的な作業時間を表している．この表から，PSDLとCOBOLとで再利用のための作業時間比は10：43となり，約4倍良いと推定される．その根拠として，PSDLは概念に着目して記述するので，手続きの制御フローや中間データを考える必要がなく，また個人の思考癖がCOBOLよりも現れにくい傾向にあることが指摘された．

（2） 理　解　性

PSDLは概念に着目して記述するので，手続き型プログラムよりは理解性が良かった．しかし，前述のようにドメインモデル理解に時間がかかっており，仕様の理解性についてはまだ研究が必要である．その課題を以下にあげる．

a) 情報処理の目的と理由，方法―PSDLで記述したドメインモデルは，情報処理の方法しか表していない．このため，その内容を理解するのに，情報処理の目的と理由を述べた業務概要書が必須であった．目的と理由もモデル化し[22]，ドメインモデルと関係づける研究が必要である．

b) 宣言的仕様の手続き的理解―再利用時に仕様を切り貼りするのに，PSDLのような宣言的な仕様記述は必須である．一方，ドメインモデルの全体を理解したり，その正しさを判断するのに，たとえばPSDLではデータフローに沿って仕様を手続き的に理解することも必要であった．このため，マクロおよびミクロなデータフロー表示機能を付加して，その有効性を確認した．宣言的仕様と手続き的理解の関係は今後の研究課題でもある．

c) 型とインスタンス―PSDLの情報層はエンティティ型や関連型は記述するが，エンティティ自体や関連自体，すなわち型のインスタンスは記述しない．し

かし，被験者は，再利用対象仕様に記述された型から，1）典型的なインスタンスを想像し，2）そのインスタンスを新たな要求条件に合わせて修正し，3）修正されたインスタンスに基づいて型を修正する，という手順で仕様を拡張することが多かった．ドメインモデルにおける型とインスタンスの関係について研究が必要である．

d）階層的サブジェクト—ドメインモデル中で関連型はあらゆる方向に伸びているので，モデルを理解する際，関連型にそった思考は発散しやすい．その発散を阻止するのに，サブジェクトは有効であった．一方，実験では最下層のサブジェクトがまだ大きすぎるとの指摘があり，サブジェクトの最適規模について検討が必要である．また，仕様の内容を的確に表すサブジェクト名の付与方法も重要である．これに対しては，データベースにおける項目名の標準化に関する研究[23]が参考になると考えている．

e）関連型の理解—エンティティ型の意味を理解するには，その属性をすべて理解すればほとんど理解できる．関連型の意味を理解するにも，その関連型を経由する従属性制約をすべて理解すればよいが，属性を理解するよりは困難である．このため，エンティティ型よりは関連型の理解の方が困難であった．一方，関連型は，たとえば格関係[24]や，時間順序[25]など，その意味を整理できるものが多い．このため，関連型の意味の整理について研究が必要である．

f）対象世界の構造的な理解と動的な理解　ERモデルに基づいたPSDLは，対象世界の概念のつながりに着目した構造的な理解に向いている．一方，データのライフサイクル解析のように，時間の流れに着目した動的な理解も必要である．動的に理解するには，その理解に向いた仕様記述言語，たとえば状態遷移図[26),27)]やペトリネット[28]などへ変換することが必要である．

（3） 拡 張 性

PSDLは概念に着目した非手続き型言語なので，手続き型プログラムよりは拡張性が良かった．実際，プログラム仕様の抽出実験では，4.2節(1)で述べたように仕様を拡張するものと，拡張しないものの両者を対象にした．しかし，図6で示すようにプログラム仕様抽出の作業時間自体が短くなることもあって，被験者の印象では両者の間に大きな差異は認められなかった．

（4） 再 利 用 性

PSDLで記述された仕様の再利用性は手続き型プログラムよりは良かった．具体例として，PSDLで記述された同一の属性から，プログラム仕様の他の部分が異なれば，スカラ変数が生成されることもあれば，テーブルが生成されることもある．一方，手続き型プログラムではスカラ変数とテーブルの間に完全な再利用性はない．

ところで，PSDLの従属性制約は計算方向を一方向に定めている．一方，制約論理[29]では条件式を制約として記述し，条件式は両方向の計算を表すことができるので，再利用性が優れている．PSDLの従属性制約から一方向性を除く[30]ことが今後の課題である．

（5） プログラム自動生成

プログラムの自動生成が理想的に実現されれば，ターゲット言語はどんなものでもよいであろうが，現実には必ずしもそうではないため，生成されたプログラムに対しても理解性や**追跡可能性**が要求される．現在はおもにCを対象としているが，4.3節でも述べたように，PSDLからC++へのコンパイラを研究中である．C++はCよりも理解性がよいだけではなく，さらに，C++がオブジェクト指向であることから，PSDLのERモデルとの親和性が高く，追跡可能性の点ですぐれている．

（6） ドメインモデルの記述

本実験で対象とした社会保険システムの処理は，ほとんどが静的な計算式であり，PSDLの従属性制約で容易に記述できた．ただし，対象とした四つの業務の相互の時間的な関係の記述は十分ではなかった．この点については，別の研究で，一部の機能を拡張することにより，実時間処理にも対応できることを示している[25]．

しかし，それぞれの適用ドメインには，そのドメイン独自の概念や記法が存在し，ドメインエキスパートにとってはそのほうが馴染みがある．ドメインモデルはそもそもドメインエキスパートが中心になって構築されるべきものであることから，ドメインモデルを記述するのに，汎用言語よりもむしろ，ドメインに特化した言語のほうがふさわしいという考え方もある．著者らはその点を明らかにするために，現在，建築設計の分野を対象にして，ドメインモデルを記述するための専用言語に関する研究[31]をすすめている．

（7） ドメイン分析

本研究では，実験ということで，各業務プログラムの仕様を統合する形でドメインモデルを構築したが，実務運用のためのドメインモデルを構築するには精密なドメイン分析が必要である．長澤[32]は，機械設計，プラント設計，医療支援システムなどの体験から，1）情報処理のエキスパートが適用ドメインについて学び，

ドメイン分析を支援するツールのプロトタイプを開発する，2) ドメインエキスパートと密接に協力してプロトタイプツールを洗練する，3) ドメインエキスパートが完成したツールを用いてドメイン分析を行う，というような手順が必要であると主張している．著者らも，建築設計を対象として，このようなドメイン分析の手法を確立するための研究をすすめている．

6. おわりに

ソフトウェアの仕様を抽象化してドメインモデルとして蓄積し，再利用するためのSoftReuseシステムと，その実験について述べた．本研究で用いた仕様記述言語PSDLは，ERモデルに基づいて対象世界の概念を非手続き的に記述するので，従来から効果を発揮してきたプログラム再利用に用いられるCOBOLなどの手続き型言語と比較して，ソフトウェア再利用に不可欠な理解性，拡張性，再利用性を備えている．

PSDLで適用ドメインを記述した仕様を階層的サブジェクトと組み合わせてドメインモデルとして蓄積したものは，類似のソフトウェアの開発に有効であることが実験によって確かめられた．そして机上実験の結果から，COBOLプログラムコードの再利用と比較して4倍程度の効率が期待できることがわかった．

SoftReuseシステムでは，仕様を生成するだけではなく，さらに最終製品であるプログラムを自動生成することができる．ターゲット言語をC++とすることで，自動生成されたプログラムについても，システムの保守に不可欠な理解性や追跡可能性を備えている．ただし，実用化のためのコンパイラの最適化は今後の課題である．

今後，ドメインの知識に基づくソフトウェアの再利用支援が重要になると考えられる．そのためにはドメイン分析の手法やドメインモデルの記述法についての研究をすすめる必要がある．

謝辞 本論文はATR通信システム研究所からの受託研究「ソフトウェアモデリングに関する研究」で完成したものであり，同研究所の諸氏に深謝いたします．また，SoftReuseシステムの試作と実験にご協力いただいた日本電子計算株式会社の諸氏に深謝いたします．

参考文献

1) Brooks, F. P.: No Silver Bullet: Essence and Accidents of Software Engineering, *IEEE Computer,* Vol. 20, No. 4, pp. 10-19 (1987).
2) Biggerstaff, T. J. and Perlis, A. J. eds.: *Software Reusability,* Vol. I, ACM Press (1989).
3) Hashimoto, M. and Okamoto, K.: A Set and Mapping-based Detection and Solution Method for Structure Clash between Program Input and Output Data, *Proc. IEEE Computer Software and Application Conference,* pp. 629-638 (1990).
4) Yokota, K., Hashimoto, M. and Sato, M.: An Experiment on Reusing Program Specifications Described with Conceptual Data Model- and Dependency Constraint-based Language, *Proc. Int. Conf. Computing and Information,* pp. 324-328 (1992).
5) Lanergan, R. G. and Grasso, C. A.: Software Engineering with Reusable Designs and Code, *IEEE Trans. Softw. Eng.,* Vol. SE-10, No. 5, pp. 498-501 (1984).
6) Novak, G. S., Hill, F. N., Wan, M.-L. and Sayrs, B. G.: Negotiated Interfaces for Software Reuse, *IEEE Trans. Softw. Eng.,* Vol. 18, No. 7, pp. 646-653 (1992).
7) 千吉良英毅，永松祐嗣，小林正和：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(ICASE-REUSE)，日立評論，Vol. 69, No. 3, pp. 47-52 (1987).
8) 磯田定宏：ソフトウェア再利用の管理的側面，情報処理学会論文誌，Vol. 34, No. 5, pp. 1117-1124 (1993).
9) Maiden, N. A. and Sutcliffe, A. G.: Exploiting Reusable Specifications through Analogy, *Comm. ACM,* Vol. 35, No. 4, pp. 55-64 (1992).
10) Prieto-Diaz, R.: Implementing Faseted Classification for Software Reuse, *Comm. ACM,* Vol. 34, No. 5, pp. 88-97 (1991).
11) Prieto-Diaz, R.: Domain Analysis for Reusability, *Proc. IEEE Computer Software and Application Conference,* pp. 23 29 (1987).
12) Iscoe, N., Williams, G. B. and Arango, G.: Domain Modeling for Software Engineering, *Proc. 13th Int. Conf. Softw. Eng.,* pp. 340-343 (1991).
13) DeMarco, T.: *Structured Analysis and System Specification,* Yourdon Inc. (1979).
14) Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W.: *Object-Oriented Modeling and Design,* Prentice Hall (1991).
15) Chen, P. P.: The Entity-Relationship Model—Toward a Unified View of Data, *ACM Trans. Database Syst.,* Vol. 1, No. 1, pp. 9-36 (1976).
16) Martin, J.: *Information Engineering,* Prentice-Hall (1989).
17) Prywes, N. S. and Pneuli, A.: Compilation of Nonprocedural Specifications into Computer Programs, *IEEE Trans. Softw. Eng.,* Vol. SE-9, No. 3, pp. 267-279 (1983).

18) Jackson, M. A.: *Principles of Program Design,* Academic Press (1975).
19) Coad, P. and Yourdon, E.: *Object-Oriented Analysis,* 2nd edition, Prentice-Hall (1991).
20) 岡本克巳,橋本正明：入出力データの構造不一致検出解決法に関する研究，情報処理学会論文誌，Vol. 33, No. 5, pp. 707-716 (1992).
21) 山崎充彦，廣田豊彦，橋本正明：仕様記述言語PSDLからオブジェクト指向言語への変換，第48回情報処理学会全国大会論文集，1G-10 (1994).
22) 浜田雅樹,安達久人：設計プロセス情報を利用したソフトウェア修正支援方式，情報処理学会論文誌，Vol. 35, No. 5, pp. 887-896 (1994).
23) 黒川 清，中川 優，関根 純：複合語解析技術を用いたデータ項目名称の標準化手法，情報処理学会論文誌，Vol. 34, No. 3, pp. 447-456 (1993).
24) 大西 淳，阿草清滋，大野 豊：要求フレームに基づいてソフトウェア要求仕様化技法，情報処理学会論文誌，Vol. 31, No. 2, pp. 175-181 (1990).
25) 岡本克巳,橋本正明：概念データモデルに基づくプログラム仕様記述言語の実時間処理への拡張，情報処理学会論文誌，Vol. 34, No. 9, pp. 2002 2012 (1993).
26) Coleman, D., Hayes, F. and Bear, S.: Introducing Objectcharts or How to Use State-charts in Object-Oriented Designs, *IEEE Trans. Softw. Eng.,* Vol. 18, No. 1, pp. 9-18 (1992).
27) 川染 誉，加来田裕和，石畑佐代美，廣田豊彦，橋本正明：オブジェクト指向分析支援ツールの作成—動的モデルのシミュレーション，電子情報通信学会技術研究報告(知能ソフトウェア工学)，Vol. 93, No. 408, pp. 25-32 (1994).
28) Felder, M., Mandrioli, D. and Morzenti, A.: Proving Properties of Real-Time Systems Through Logical Specifications and Petri Net Models, *IEEE Trans. Softw. Eng.,* Vol. 20, No. 2, pp. 127-141 (1994).
29) Leler, W.: *Constraint Programming Languages: Their Specification and Generation,* Addison-Wesley (1988).
30) 佐藤正和，橋本正明，寺島信義：E-Rモデルを用いた視覚的プログラミング言語：PSDL-GRとその一実現法，情報処理学会論文誌，Vol. 35, No. 1, pp. 115-126 (1994).
31) 廣田豊彦，千原博司，佐藤俊孝，橋本正明：要求分析のための概念モデル記述言語に関する一考察,電子情報通信学会技術研究報告(知能ソフトウェア工学)，Vol. 94, No. 130, pp. 1-8 (1994).
32) 長澤 勲：ポスト・マスプロダクション・パラダイムとIMS，精密工学会知識工学とCAD専門委員会第23回例会 (1993年6月).

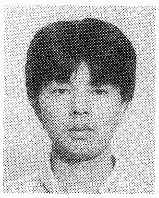
(平成6年 7月 4日受付)
(平成6年10月13日採録)

橋本 正明 (正会員)

昭和21年生．昭和43年九州大学工学部電子工学科卒業．昭和45年同大学院修士課程修了．同年日本電信電話公社電気通信研究所勤務．昭和63年ATR通信システム研究所出向．平成5年九州工業大学情報工学部教授．従来オペレーティングシステムや，データベース設計，仕様記述言語，ソフトウェア自動作成等の研究実用化に従事．著書「データ中心のプログラム仕様記述法」(井上書院)．工学博士．電子情報通信学会，IEEE等各会員．

廣田 豊彦 (正会員)

1954年生．1976年京都大学工学部電気工学第二学科卒業．1981年同大学大学院工学研究科博士後期課程研究指導認定退学．工学博士．1981年京都大学情報処理教育センター助手．1988年九州工業大学情報科学センター助教授．1989年同大学情報工学部知能情報工学科助教授，現在に至る．プログラム開発環境，プログラムテスト支援，CADシステムなどの研究に従事．著書「Cプログラミングの基礎」(培風館)ほか．日本ソフトウェア科学会，人工知能学会各会員．



横田 和久 (正会員)

1964年生．1988年電気通信大学電気通信学部計算機科学科卒業．1988年横河・ヒューレット・パッカード入社．1990年エイ・ティー・アール通信システム研究所に出向．1994年横河・ヒューレット・パッカードへ復帰，現在に至る．プログラム仕様記述，言語処理系などの研究に従事．