ソフトウェア工学 103－13
（1995．3．10）

プログラム理解とその支援ツール設計に於けるドメイン・モデリング

秋山 義博
金沢工業大学情報工学科

ドメイン分析とモデリング手法は，ソフトウエア成果物の再利用過程を示すものとして注目されている。この論文では，システム設計情報の再利用の適切過程がどのようなものか，プログラム理解支援システム，ＯＳ，ソフトウエアアーキテクチャ，システムアーキテクチャ等を参考にして述べる。

新規に或は機能追加拡張を目的にシステム開発を行なう場合，設計及びプログラムコード情報の再利用は重要な課題である。この場合，設計情報はプログラムコードの中に埋もれてしまう為にプログラム理解が必要になり，プログラム機能或は設計情報を得ながらプログラムコード再利用を図る事が最も現実的であると考えられている。これは現在のソフトウエア開発技術の不完全性によるもので，この特殊事情を取り去ることは暫くの間，出来そうにない。この仮定を注意深く扱いながら，プログラム理解支援ツール開発におけるドメイン分析手法(DAM, Domain Analysis and Modeling)の提示する方法論の適用限界とその改良版について述べる。

On the Domain Analysis and Modeling Approach to Develop Tools for Program Understanding

Yoshihiro Akiyama
Information and Computer Engineering
Kanazawa Institute of Technology
7-1 Ohgiga-oka Nonoichi-machi Ishikawa 921 Japan
E-mail: akiyama@infor.kanazawa-it.ac.jp

The domain analysis and modeling (DAM) approach proposes a global view of how software can be reused. According to the characteristics of application domains of program understanding tools, OS, and software and systems architectures, this paper describes an idea on the optimal path over the processes of how system design information and program code can be reused.

Reusing design and code of existing software is helpful in developing new products or functional enhanced products. However, the design information has been mostly included in the program source code. This leads that the reuse of the program code is considered to be practical and must be helped by the program understanding to capture the design information. This may be because of premature of the current software technology and it seems not to be resolved soon. With carefully handling this situation, this paper discusses the application limit of the DAM and proposes an improved model of DAM.

はじめに

ソフトウエアの新規又は機能変更/拡張の開発を行なう場合,機能とそのプログラムコードが再利用出来ることが判れば,その後の開発ワークロードは大変違って来ることは時々経験する。又，これが，再利用する人の業務知識やそのシステム開発上で得た経験に依存することも経験する所である。従って，再利用方法は千差万別になることから,再利用管理が重要であると言う主張/報告と，こうした特別な事情が無い場合のソフトウエア（実際には，プログラム）再利用の割合は経験的に，３分の１～４分の１程度であると報告されている。

最近提案されているドメイン分析とモデリング手法（DAM: Domain Analysis and Modeling）は，この２つの問題:再利用の工程を簡単にすることと再利用度を改善するものと考えられている。この方法論のモデルは，「Triadic Domain Model」と呼ばれ，その特徴は，個別の記述をドメインとして明確化することにより,問題ドメインモデル，その問題に対するプロダクトドメインモデル，問題からプロダクトを導出するプロセスドメインモデルの３つのモデルを構築し，これらのモデルを参照しながら再利用を促進することを目的としている。しかしながら，この問題ドメインモデル，プロセスドメインモデル，プロダクドメイントモデルを通して,どのように再利用するかはこれからの研究を待たなければならない。

この論文では，このDAMについて，ソフトウエア事例を参考にしながらその適用範囲と再利用可能性を高めるDAM内経路を検討すること，その適用制限を除く一つの改良モデル，そして,プログラム理解又はそのツールをアプリケーションドメインとした場合の改良モデルアプローチを述べる。

Triadic Doamin Model(３元ドメインモデル)

ドメインとは，類似している個別要素の集合である。問題ドメインは類似問題の記述群であり，その問題をどのように解くかを記述したものの集合をアプリケーションドメインと呼ぶ事にする。ここで，再利用対象とするドメインは時間的に安定的でなければならない。その理由は，他へ再利用する間の変更があると，それ以降の再利用が不可能になるからである。ドメイン分析とは，ドメインに属する要素群から考えられるオブジェクト，操作，関係付け等ドメインの特徴的構成要素とそれらの間の関係を求めるプロセスである。そして，類似問題ドメインに対して，それぞれの問題を解くアプリケーションも類似していることから，問題からアプリケーションを導く類似プロセス群がある。問題領域モデル(Domain Problem Model)，問題から解を求めるプロセス領域モデル(Domain Process Model)，及びアプリケーション領域モデル(Domain Product Model)を与えることが出来て，新しい問題が与えられた場合には，この問題領域モデル内に似ている問題を探すことが出来れば，それに対応するプロセスとアプリケーションを選び可能な限りその情報を再利用する。伊藤等は,この体系をTriadic Domain Model（３元ドメインモデル)と呼び，図－１に示すようなモデルで表し提案している。

各ドメインの記述を「ひな型」（クラス記述の様なもの）と言って良いかは注意が要る。これらは，すべてドメインに対して或はドメイン要素に対して数多くある見方の一つを表して，ドメインの記述になるかどうかは必ずしも自明ではない。ドメインを本質的に表す（必要分だけのビューを同時に与える）事が，ドメインモデリングにおいては，大切になる。これが，上で”問題領域モデル”のように呼んでクラスと区別した由である。ドメイン工学(Domain Engineering)は，ドメインの構造を求める技術を提供するものと定義され，その成果物は，ドメイン固有のソフトウエアアーキテクチャ等である。プロセス領域モデルは,個々のプロセスを集めたプロセス群に対してドメイン工学を適用することにより得られるものであり,他のドメインに対しても同様にドメインモデルを得ることが出来て,図-1に示すように,それらが３元ドメインモデルとして接続される。

３元ドメインモデルの改良モデル

３元ドメインモデルについて,コンピュータハードウエア,ネットワーク制御プログラム（両方とも長期に渡って前技術を再利用しながら発展してきた）,及び物理法則による記述方法を当てはめて,その妥当性を考える。

ビジネス計算用コンピュータのドメインを考える。個々のハードウエアとソフトウエアに関連した問題をまとめ，それぞれのシステムの問題の組みから(それらを解決している)ビジネス計算用コンピュータの設計（アーキテクチャ）を考えることが出来，この設計群もドメインを形成する。又，それぞれの設計開発プロセスも多少異なるが，大きくは違わないであろう。しかしながら，与えられた問題に対して得られる開発プロセス，コンピュータ設計等には，設計者の考えや判断がふんだんに入るので，これらは自明ではなく，これが，再利用の対象となる。更に，コンピュータアーキテクチャのような一般的な設計を考えて，その情報を利用して，そのアーキテクチャを持つ各種モデルの設計を実現するような事が出来る。IBM S/360システム開発に於いては，機械語命令セットとI/Oチャネル系等をを不変にして各モデルの設計(実現技術，マイクロコードを含めて）の仕様を個別に与えたが普遍部分を実現するところで設計情報と一部ハードウエアの再利用を行なった。

ネットワークプログラムと端末についても同様なことが言える。個々の問題と個々の製品をそれぞれまとめて一般化するよりも，個々の問題を，より基本的な観点から再分析し，現存する個々の製品を可能な限り含むような，新しいネットワークアーキテクチャを考えて，機能層と各機能層の目的と機能の仕様を規定する。そして，その範疇において，個別製品，プログラムを再利用することを考える。その時のシステムに対して多少の修正が求められても将来的にジャステファイ出来れば良い。

もうひとつの例は，物理法則である。個々の類似系(これは，無数にある)の細かい現象を観測してそれぞれ規則を見つけたとする。しかしながら，それらは，ある特定の条件下での振る舞いであって，見つけたそのルールを物理法則と呼ぶには不十分である事が多々ある。更に，現象的に類似していない場合でも本来同じ現象と考える場合もある。このように数多くの現象を記述するアーキテクチャのような役割をする運動法則を見つけ，次にこの運動法則から，求めたい系に個有な条件(初期条件又は境界条件とも言う)を与えてシミュレーションを行なうことが出来ることは良く知られている。現象的には，始めの数多くの現象の中のある現象の知識を再利用しているようにも見える。

図-2に，以上に述べた再利用方法を示す。これは，図-1に示すような直接的再利用ではない。一度，一般化を行なって，対象系の設計目的のエッセンスをその一般化形式の中に綴じこめる所にこのアプローチの特徴があり，その情報量が多ければ多いほど効果的になる。従って，この一般形は，それまでの事例の合計値や平均値ではなく，設計や実現方式について新しい設計が入っていて，時間的に余り変化することはなく(普遍性が高い!)長期間に渡って利用出来る情報から出来ている。

このように，基本設計情報はすべて アーキテクチャ仕様に納められていて，個別に対応する必要のある部分は，後日最適に決める事が出来るようになっている，と言うアプローチを"アーキテクチャ型再利用アプローチ"と呼ぶ事にする。

プログラム理解

プログラム理解に関して，一般化した問題の幾つかの例を次に挙げる：a)プログラムの全体設計について分からない，b) 依存関係，構造について分からない，c) 業務処理の特定のケースについて分からない，d) 業務処理の全体について分からない，等々。これらは，個別プログラム理解によらず，どの場合にも現れる問題である。これらの問題全体が，問題領域モデルを形成する。このような状況が発生すると，担当プログラマーは，次のステップの順に沿ってプログラムを進めるであろう：1) プログラムのシンタックスレベルでの理解，2) プログラム構造把握，3) プログラム機能レベルの理解，4) プログラム設計レベルの理解。通常，テキストエディターを用いてプログラムソースをディスプレイ画面に表示して数行ステートメント毎に理解を進める。従って，これらの理解作業の大半は，エディター操作と低レベルの記号関係の把握というマニュアルワークとメンタルワークである。この理解の可能な進め方の全体がアプリケーション領域モデルとなる。プログラム理解支援ツールの設計目標は，これらの作業をコンピュータ・エイデットにして無くすることである。しかしながら，プログラム理解全体をコンピュータエイデットにすることは完全には出来ないので，図-3に示すような支援範囲になるであろう。ここでは，プログラム理解の具体的な検討についてはこの論文の目的から外れるのでこれ以上述べない。

プログラム理解ツールとその開発

プログラム理解問題を取扱う場合，言語別，システム別，業務別等が考えられる。COBOL，C，C++，PL/I，アセンブラー等数多くの言語に対して，DOS，UNIX，OS/2，MVS，VM等の各種OS，ビジネス業務，科学計算等の数多くのアプリケーションシステム，及びこれらのバージョンの組み合わせがこのドメインの大きさである。従って，個別対応は不可能に近いが，問題領域モデル，プロセス領域モデル，アプリケーション領域モデルのドメイン形式で，共通情報を再利用出来れば状況は改善される。そして，アプリケーションにも依るが，これらのドメインの時間的安定性は大体良いと考えられる。プログラムサイズは，プログラム理解とは無関係に思えるが，実は，プログラム理解プロセス，プログラム理解支援システム全体のインターフェスや内部実現方法に関係する重要なパラメータである。これは，プログラムやツール個別のパラメータであるというよりも，ドメインを特徴付けるパラメータと考えた方が良い。プログラム固有のパラメータとしては，上記のパラメータの他に作成時の設計/プログラミング方法論がある。長期間保守されてきているプログラムについては，複数の方法論(主に表現形式)が使われている可能性が高いが，この場合，使用された方法論の全てに共通な方法論（表現形式）を用いることにより解決する。

図-4にプログラム支援ツールを求める為の３元モデルを示す。上半分が現在のプログラム理解の問題とそのアプリケーション(プログラム理解のプロセス)を表している。これに対して，DAMプロセスを通して，それぞれのドメインモデルを求めると(実際には難しい作業であるが)，問題領域とアプリケーション領域の個々の記述又は共通部分の記述が得られ，マニュアルワーク，メンタルワーク中心の記述又はモデルになる。これは，モデリングにおいて”人の介在”を排除出来ないことによる。更に，このモデルを再利用して個別のプログラム理解ツールを生成してもやはりマニュアルワーク，メンタルワーク中心のツールが出来るに違いないと思われる。従って，モデル情報の再利用は，余り進まない可能性がある。これは，再利用情報の品質が余りよくないことを示しており，本来目指したい目的ではない。

図-５に”アーキテクチャ型再利用アプローチ”を示す。問題群を睨みながら，プログラム理解の基本的入力(プログラム情報)，プロセス（自動分析），分析結果の自動表示及びユーザによる対話検索等を与える基本設計（アーキテクチャ)を先ず実現する。この”問題をまとめアーキテクチャを与える過程”が前述のケースには含まれていない。この過程では，問題の”形式的分析/記述”に限らず，プログラマーのプログラム理解プロセスの全体像を工学的に分析して，そのプロセスの単純化と自動処理が出来る部分，プログラマーによる必要な手助け等を明らかにする。この”アーキテクチャ”は，問題領域に対する解決にはなっているが，その問題領域にまだ含まれていない問題についても解決能力がある。これが，将来に向けてソフトウエア情報を再利用出来る根拠である。具体的に個々のプログラム理解ツールを開発する為には，このアーキテクチャ記述に用いられているパラメータをその関連する問題とプログラムやアプリケーションの特性に合わせて設定して，設計実現を行なう。

プログラム理解ツールのコンポーネント機能/構成とユーザインターフェスを実現する為に，先ず，一般的に，そのシステム環境管理，プログラムブラウジング及びプログラム分析が生成/参照/表示する情報の基本構造を，言語，システム，環境の特殊パラメータに依らずに定義することを考える。これは，例えば，プログラムの始まり，データ定義，等が，定義文自体は言語やシステム依存であるが，プログラミング上での意味はそれらに依存しないで(ツールに対して)定義出来る。このような設計を設けることにより，現在使用中の言語やマニュアルワークやメンタルワークに依存度を置かないで，プログラム理解のエッセンスを記述出来て，そのような基本設計から，個別ツールを生成開発することは，容易になるであろうことは，期待出来る。

まとめ

ドメイン分析とモデリング手法，特にTriadic domain modelの特徴を調べ，その適用範囲について述べた。プログラム理解支援ツールの設計は，対象とするプログラムのバラエティーと現存するプログラム理解の方法と支援ツール技術がそれぞれこれから発展すると期待されることから，DAMのアプローチを改良した”アーキテクチャ型ドメインモデリング”アプローチが有効と考えられる。

参考文献：

1. 田村恭久，伊藤潔，杵嶋修三，ドメイン分析・モデリング技術の現状と課題，情報処理，Vol. 35，No. 10, pp952-961, 1994
2. 伊藤潔，杵嶋修三，リアルタイムシステムに於けるネット指向開発技術の適用，情報処理，Vol. 34, No. 6, pp747-760, 1993
3. 伊藤潔，杵嶋修三，リアクテブシステム分析・設計向きドメインモデル:Asdreas STD Triad, 情報処理学会誌, Vol. 34, No. 9, pp2025-2036, 1993
4. 秋山義博, 大規模複雑プログラム理解のためのプログラム分析と可視化技術, ソフトウエア工学 102-6, pp31-36, 1995
5. C. W. Krueger, Software Reuse, ACM Computing Surveys, Vol. 24, No. 2, 1992

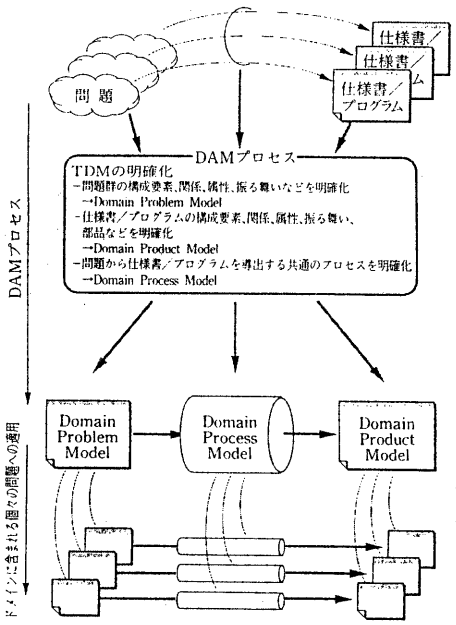


図-1 Triadic Domain Model の概念

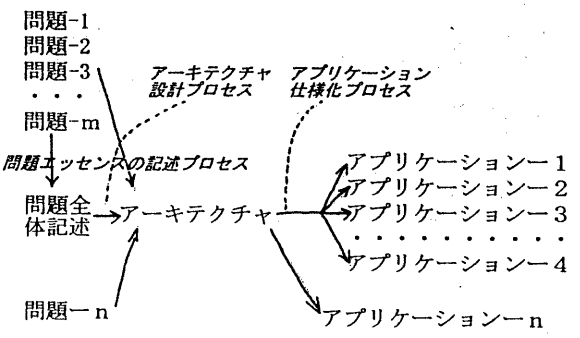
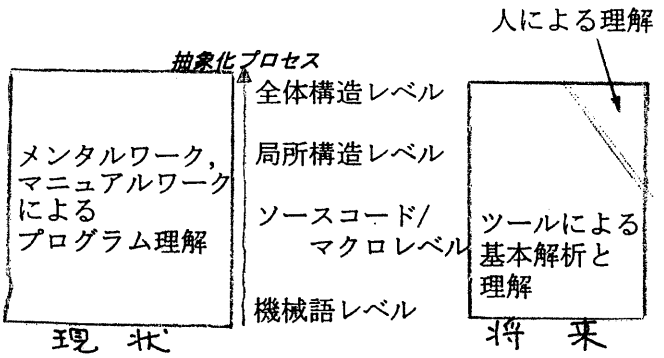
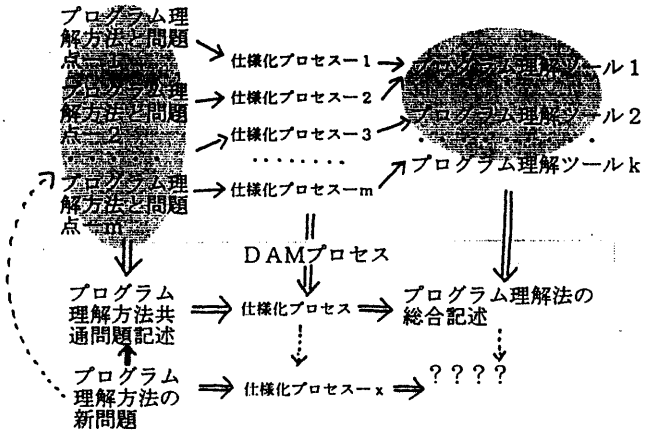


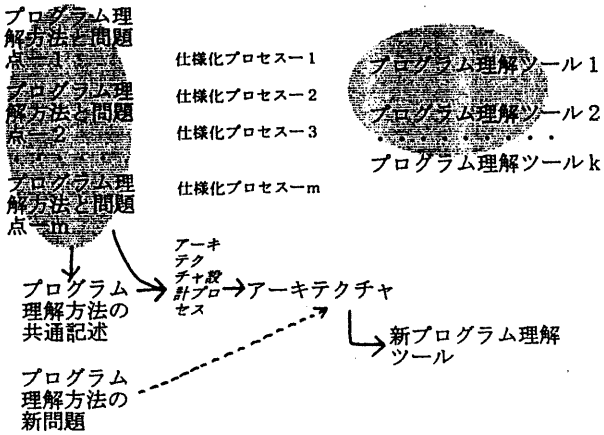
Fig. 2 アーキテクチャ型再利用アプローチ



Ｆｉｇ．３　プログラム理解支援アプローチの変化



Ｆｉｇ．４　ＤＡＭに依るプログラム理解情報の再利用方法



Ｆｉｇ．５　ＤＡＭに依るプログラム理解情報の再利用方法