ソフトウェア工学 71−16
（1990. 2. 9）

アニメーションによるプロトタイピング
～制約を利用したその方式～

古宮 誠一
情報処理振興事業協会 技術センター

プロトタイピングを効果的に行うためには、対象とするソフトウェアの性質を分析し、その性質に適合したプロトタイピング手法を適用する必要がある。本稿では、対象ソフトウェアの性質とプロトタイピング技術を分析し分類して、プロトタイピング技術の各々が、どのようなソフトウェアに対してどのように有効かを明らかにしている。そして、その評価を踏まえ、事務処理ソフトウェアに代表される、 data-drivenな振舞いをするソフトウェア対しては、自動プログラミング・システムと結合したアニメーションによるプロトタイピングが有効であることを示している。さらに、アニメーションによる方式では制約を利用した方法が有効であることを示している。

Prototyping a Business application Software by Animation and its method using Constraints

Seiichi KOMIYA

Software Technology Center
Information-technology Promotion Agency, Japan

6F Shuwashibakoen 3-chome BLG.
1-38, Shibakoen 3-chome, Minatoku, Tokyo 105, Japan

Software designers should analyze the nature of target software and use a prototyping method suitable to its nature in order to make prototypes efficiently. In this paper, we analyze and classify the nature of target software and prototyping methods, and clarify that each method is effective for what kind of software in what manner. Moreover, this paper describes that the prototyping by animation provided with facilities for automatic programmming is efffective for data-driven softwares in business application, and a method using constraints is effective for prototyping by animation.

1．はじめに

事務処理ソフトウェアに代表される、data-divenな振舞いをするソフトウェアに対する開発方法の理想は、ソフトウェア開発が下記のステップ①～④で行われ、条件⑤を満足することである。

①ライフサイクルの早い段階に、プロトタイピングすることにより、与えられた要求仕様がユーザの意図どおりであることを確認する。

ここでのプロトタイピングとは、ユーザ確認によりデータ構造を決定し、システムの振舞いを決定することである。ここで特筆すべきことは、システムの振舞いの確認に先立ってデータ構造を決定するということである。data-divenな振舞いをするソフトウェアでは、データ構造を決定することによって初めてプログラム構造の設計が可能となるからである。システムの振舞いの確認には、アニメーションによるプロトタイピング手法が用いられる。ここでは、データ項目間やプロセス間の制約が利用される。

②与えられた要求仕様を解析して、要求仕様そのものが無矛盾性・（並行プロセス間の）妥当性・完全性（曖昧さや欠落情報のないこと）・到達可能性などを満足することを確認する。

①によって、機能面でのユーザの要求が確認された。しかし、プロトタイピングだけでは、要求仕様そのものに内在する矛盾などはチェックできない。これらのことをチェックするのがステップ②である。

③①と②で確認された要求仕様を満足するプログラムが得られる前に、そのプログラムの実行時における性能（レスポンスタイムやスループット）を計算し評価して、性能面でもユーザの了解を得る。

機能面でユーザの意図どおりであることを確認し、要求仕様そのものにも問題がないことが判明したこの段階になると、要求仕様を満足するプログラムを完全自動で生成することが可能となる。しかし、プログラムを自動生成する前に、自動生成するプログラムの性能を計算し評価して、性能面でもユーザの了解を得る必要がある。このためのステップが③である。

なお、自動生成されるプログラムの性能予測が可能であることは、要求仕様を満足するプログラムを完全自動で生成できることに起因している。

また、ＰＡＰＳで自動生成されるプログラムは、性能が最高となるように工夫されているので、ソフトウェアの改良だけではこれ以上性能が上がらない。このため、試算された値よりも上の性能を要求された場合には、より高性能なハードウェアを推奨する形でコンサルテーションを行う。システム価格が折り合わなかった時には、試算した性能と現行のハードウェア構成でユーザに我慢して戴くことになる。

④機能と性能の両面でユーザの要求仕様を満足するプログラムを完全自動で生成する。

機能と性能の両面でユーザの意図どおりであることを確認したこの段階になって、初めてユーザの要求する仕様を満足するプログラムを完全自動で生成する。

⑤自動生成されるプログラムは品質が保証されており、ユーザはデバッグする必要がない。

ソフトウェアの生産性が最も高いのは自動プログラミングである。自動プログラミングで得られたプログラムを目の前にうず高く詰まれ、バグがあるからデバッグしてくれと言われたら、いくらプログラム生成率が高くても誰でも逃げ出すであろう。それ故、自動プログラミングによって得られるプログラムの品質を保証することにより、ユーザをデバッグ作業から解放しなければならない。

なお、自動生成されるプログラムの品質を保証できるのは、要求仕様を満足するプログラムを完全自動で生成できることに起因している。

ソフトウェア開発環境ＰＡＰＳは、上記の理想を実現するためのシステムであり、①のプロトタイピング機能、②の要求分析機能、③の性能を事前に評価するための機能、④と⑤の自動プログラミング機能から構成されている。本稿では、対象ソフトウェアとプロトタイピング技術を分析し分類して、各々のプロトタイピング技術が、対象ソフトウェアの各々に対してどのように有効であるかを評価する。そして、この評価を踏まえ、事務処理ソフトウェアに代表される、data-driven な振舞いをするソフトウェアに対しては、アニメーションによるプロトタイピングが有効であることを述べる。それも特に、制約を利用した方法が有効であることを述べる。

2．要求定義法から見た対象ソフトウェアの分類とそこで有効なプロトタイピング手法

(1)対象ソフトウェアの分類

対象ソフトウェアは、その振舞いをいかに規定するかということから次の４つに大別できる。

①reactive system

event drivenな振舞いをするソフトウェアのことで、システムの振舞いが、システムの内部状態とそれへの

入力信号によって決定するという特徴がある。

②transformational system

data drivenな振舞いをするソフトウェアのことで、データの入出力仕様と変換仕様を規定すれば、システムの振舞いも自ずから規定できるという特徴がある。

③temporal system

時系列的な振舞いをするソフトウェアのことで、システムの振舞いが時間経過によって規定されるという特徴がある。

④functional system

システムの対象が、①～③のいずれかに属する機能の集合と見なされるソフトウェアのことで、機能間に流れるデータがあるという観点から、システムの振舞いがデータフローやバブルチャートなどで規定されるという特徴がある。

(2)対象ソフトウェアとそこで有効なプロトタイピング手法

①reactive system

システムの内部状態とそれへの入力信号によって、システムがどのように振舞うかを規定することが要求定義であり、定義された要求仕様に合わせて、その振舞いをシミュレートすることがプロトタイピングである。従って、状態遷移記述を基にした方法が効果的である。

②transformational system

データの入出力仕様と変換仕様を規定することが要求定義であり、定義された要求仕様に合わせて、その振舞いをシミュレートすることがプロトタイピングである。従って、データの入出力仕様と変換仕様をビジュアル化して、その中でトランザクション・データがどのように変換されて行くかを示す方法が効果的である。

③temporal system

システムの振舞いを時間経過で規定することが要求定義であり、定義された要求仕様に合わせて、その振舞いをシミュレートするのがプロトタイピングである。従って、時間論理、時区間論理などの概念を含んだ論理型言語で記述する方法が効果的である。

④functional system

システム全体の機能については、その振舞いをデータフローやバブルチャート等によって規定することが要求定義であり、定義された要求仕様に合わせて、その振舞い（＝データの流れに沿って各機能が実行される様）をシミュレートすることがプロトタイピングである。個々の機能については、①～③のいずれかに分類できるので、その分類に従ってその振舞いを規定することがプロトタイピングである。

3．プロトタイピング技術の分析と有効性の評価

プロトタイピング技術は、その実現方式から次の3つに分類できる。

(1)プログラミング言語によるもの

既存のプログラミング言語でプログラミングすることにより、プロトタイプを作成する方法である。作成部分のすべてを新規に作成する方法と、既存のプログラムを再利用することにより、作成部分の一部またはすべてを作成する方法の2つがある。これらが短い工期で安く作るというプロトタイプ作成上の原則（本章で後述する5条件の1つ）を満足させるための手段となり得るか否か、という観点からこれらを吟味してみる必要がある。前者がこの原則を満足し得ないことは明白である。従って、前者はプロトタイピング技術とは言い難い。後者は、プロトタイプ作成のために再利用可能な既存のプログラムが、たまたま存在するときにはこの原則を満足できるが、常にこの原則を満足させるために、どのようなプログラム部品を用意しておくべきかということを明らかにしない限りにおいては、プロトタイピング技術とは言い難い。

(2)実行可能な仕様記述によるもの

要求仕様を記述し、これをコンパイルするだけで、プログラムとして実行可能であれば、ライフサイクルの早い段階にプログラムを短期間に低コストで作成することができる。従って、この様な仕様記述言語とその処理系があれば、プロトタイピング技術として利用できる。この様な技術を実行可能な仕様記述(executable specification)という。実行可能な仕様記述には、状態遷移記述を基にしたオペレーショナル・アプローチと自動プログラミング技術を基にしたブラックボックス・アプローチがある。

(3)アニメーションによるもの

対象とするシステムの振舞いをグラフや図表により動的に表現できれば、プロトタイピング技術として利用できる。要求仕様はインタラクティブに与えられるのが普通で、その実行にはコンパイルを必要としないという特徴がある。この様な方法をアニメーションによるプロトタイピングという。

アニメーションによるプロトタイピングの実現方法は、次の2つに分類できる。

①WYSIWYG(= what you see is what you get)方式

ユーザが画面を通じて確認したもの（＝プロトタイプ）が、そのままユーザが得られるもの（＝最終システム）になるという方法である。このような方法をWYSIWYG方式によるプロトタイピングと呼ぶ。対象システムの叩き台として、これ以上確実なプロトタイピングはない。WYSIWYGが可能なのは、対象システムの中でビジュアルなもの（＝対象システムの入出力物）を動作させることが、そのまま対象システムの振舞いを確認することになる場合である。このような入出力物は、事務処理プログラムにおける画面や帳票などに限られる。従って、WYSIWYG方式によるプロトタイピングとは、対象システムで使用する画面や帳票などをワークステーションや端末の画面上に作成し、それを動作させることによって、その形式や使い方を確認する方法である。

②等価回路方式

システムの振舞いを規定する構成要素と理論的に等価な図形やグラフを使い、それを動作させることによって、対象システムの振舞いを確認する方法である。このような方法を等価回路方式によるプロトタイピングと呼ぶ。このような方法が可能なのは、対象システムの振舞いが状態遷移で規定される場合である。

ソフトウェア生産ツールが、プロトタイピング支援ツールとして有効であるための条件は

①与えられた仕様を忠実に表現する動的なモデルを、短い工期かつ少ない工数で実現できること。

②一度作成したモデルの修正が容易であること。

③対象とするプログラムの範囲が充分広く、様々な要求を仕様として表現できること。④要求仕様の与え方が容易であること。

⑤要求仕様の与え方そのものの習得が容易であること。

⑥作成済みのプロトタイプから、最終システムを速やかに作成する手段を持つこと。

の６つである。上記の分類による３つの方式が、２章の分類による対象ソフトウェアの各々に対して、プロトタイピング支援ツールとしてどのように有効であるかを、これら６つの条件と照らし合わせて分析して評価した結果を表１に示す。

プロトタイピング技術の実用性の評価

～ツールがプロトタイピングに有効であるための条件からの評価～

区分	プロトタイピングを実現するための技術	対象ソフトウェアの種別	条件1	条件2	条件3	条件4	条件5	条件6	備考
方式1	プログラミング言語を使うもの	Event Driven	×	×	◎	△	△	△	
		Data Driven							
		time Driven							
方式2	実行可能な仕様記述	Event Driven	◎	◎	○	×	×	○	4GLがこれに該当する
		Data Driven	○	○	○	○	○	○	
		time Driven	◎	◎	○	×	×	○	
方式3	アニメーションによるもの	Event Driven	◎	◎	○	○	○	○	等価回路方式に該当
		Data Driven	◎	◎	○	◎	◎	×	WYSIWYG 方式に該当
		time Driven	×	×	×	×	×	×	
結論	実用的なプロトタイピング技術を開発するための戦略	Data Driven	方式3の採用	方式3の採用	方式3の採用	方式3の採用	方式3の採用	APSとの結合	

ツールがプロトタイピングに有効であるための条件

1．与えられた仕様を忠実に表現する動的なモデルを、短い工期かつ少ない工数で実現できること。

2．一度作成したモデルの修正も容易であること。

3．対象とするプログラムの範囲が充分広く、様々な要求を仕様として表現できること。

4．要求仕様の与え方が容易なこと。

5．要求仕様の与え方そのものの習得が容易なこと。

6．作成済みのプロトタイプから、最終システムを速やかに作成する手段を持つこと。

－以上－

(注) 4GLは実行可能な仕様記述を指向したプログラミング言語である。
APS：自動プログラミングシステムの略

4．アニメーションによるプロトタイピング方式

ＰＡＰＳは、事務処理ソフトウェアに代表される、data-drivenな振舞いをするソフトウェアを対象とするソフトウェア開発環境である。３章での表１の結論は、data-drivenな振舞いをするソフトウェアに対しては、方式３のアニメーション（それも特にWYSIWYG方式）によるプロトタイピングと自動プログラミング・システムを結合する方式が優れていることを示している。従って、ＰＡＰＳのプロトタイピング機能はこの方式を用いる。この方式は、方式３で唯一×印にな

っている部分を◎印に変えることのできる方式である。具体的には、『作成済みのプロトタイプから、最終システムを速やかに作成する手段を持たない』というこの方式の欠点を補うために、自動プログラミングシステムと結合する方式を採用する。この方式を図1に示す。ここで、一重の線で囲まれた長方形は作業を表わし、二重線で囲まれた長方形がコンピュータで自動的に行う処理を表わしている。

本章では、アニメーション（それも特に WYSIWYG方式）によるプロトタイピングとはどのようなものか、ということを具体的に明らかにする。そして、そこでは制約に着目した方法が有効であることを示す。

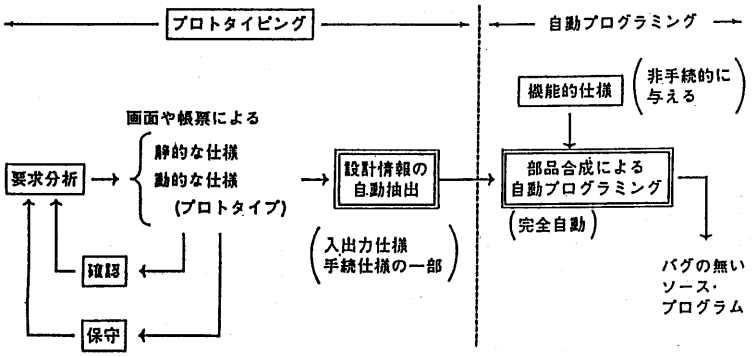

図1　PAPSによるソフトウェア開発

［例題］

《営業日誌》

9月1日　現金 ¥200,000, 商品 ¥100,000 (A商品 400個 @¥250) を出資して営業を始める。

3日　商品陳列用ケース¥40,000を買い入れ，代金は現金で支払う。

7日　名古屋商店から次のとおり商品を仕入れ,代金は掛とする。
B商品　500個　@¥ 360　¥180,000

12日　広島商店に次のとおり商品を売り渡し，代金は現金で受け取る。
A商品　100個　@¥ 300　¥ 30,000
B商品　100〃　〃〃 400　〃 40,000

16日　1株の額面 ¥500のA株式会社株式 100株を@¥ 650で買い入れ，代金は現金で支払う。

21日　福岡商店に次のとおり商品を売り渡し,代金は掛とする。
A商品　100個　@¥ 350　¥ 35,000
B商品　300〃　〃〃 450　〃135,000

25日　本月分給料 ¥15,000を現金で支払う。

ＰＡＰＳでは、データの入出力仕様が定義されていれば、図1の設計情報の自動抽出機能によって、その情報が自動プログラミング・システムに渡されるので、ユーザが与えた要求仕様を満足するプログラムをいつでも完全自動で生成することができる。しかし、対象システムの振舞いが完全にユーザの意図どおりであることが確認された後でなければ、プログラムを生成する意味がない。そこで、ＰＡＰＳでは、自動プログラミングに先立って、対象システムの振舞いが完全にユーザの意図どおりであることを確認する。このための機能がプロトタイピングである。

ここでは、ＰＡＰＳのプロトタイピング機能を下記のような事例で説明しよう。

4．1　形式やレイアウトに関するプロトタイピング

画面・帳票・ファイルなどの形式やレイアウトを確認しながら自分で設計できるようにする機能のことである。例題のような取引事例に対して、会計処理では最初に図2のような仕訳帳を作成する。このとき、この帳票のフォーマットや画面上でのレイアウトをユーザが納得するまで試行錯誤することによって、ユーザの意に叶ったレイアウト設計を行う。これは図1における静的なプロトタイピングに相当する。ＰＡＰＳでは、これをビットマップ・ディスプレイを用いた高機能ワークステーションの機能を使って実現している。また、この機能を使ってデータエントリ用の画面や帳票などにおける入力データ項目の桁数・入力コードの有効範囲・編集方式・その他の属性を定義しておけば、入力データのチェック機能の仕様を指定したことになる。これにより、入力データのチェック処理プログラムについてのプロトタイピングや自動生成も可能となる。

4．2　システムの振舞いに関するプロトタイピング機能

(1)データ項目間制約に従って動作するシステムの振舞いを確認する機能

データ項目間に成立すべき条件を制約として与えたとき、任意の入力データに対して、システムの振舞いがこの制約を満足していることを確認するための機能である。この機能は、事務処理プログラムに代表される、data drivenな振舞いをするソフトウェアでは、その振舞いを確認するのに特に有効である。

1

昭和×年		摘　　要	元丁	借　方	貸　方
9	1	諸　口　(資本金)	9		300,000
		(現　金)	1	200,000	
		(繰越商品)	4	100,000	
		現金および商品の出資により開業			
	3	(備　品)	5	40,000	
		(現　金)	1		40,000
		商品陳列用ケースを現金で買入			
	7	(仕　入)	11	180,000	
		(買掛金)	6		180,000
		商品を掛で仕入			
	12	(現　金)	1	70,000	
		(売　上)	10		70,000
		商品を現金で売渡			
	16	(有価証券)	3	65,000	
		(現　金)	1		65,000
		A株式会社株式を現金で買入			
	21	(売掛金)	2	170,000	
		(売　上)	10		170,000
		商品を掛で売渡			
	25	(給　料)	12	15,000	
		(現　金)	1		15,000
		本月分給料を現金で支払			
				840,000	840,000

図2　仕訳帳

典型的な事例は会計処理プログラムである。上記のような取引事例に対して、会計処理では最初に図2のような仕訳帳を作成する。ここでは、『仕訳帳の借方の合計と貸方の合計が、取引ごとおよび取引の累計で一致する』という会計処理の原則が仕訳帳の記帳に利用されている。この原則は、コンピュータを知らなくても会計処理を知っている人なら誰でも利用できる知識である。仕訳帳における借方の累計欄の値と貸方の累計欄の値が一致するという形で、この知識を仕訳帳記入の直前にユーザが与える。このように定義しておくと、この知識がプロトタイピングに利用できる。例えば、9月1日の諸口の項目で、貸方に資本金として300,000円、借方に現金と繰越商品としてそれぞれ200,000円と100,000円を記入したとき、この取引の合計として借方の累計欄と貸方の累計欄がそれぞれ300,000円となり、この項目のデータが正しく記帳されたことが判り、システムの振舞いがチェックできる。

同様の事例は、貸借対照表における借方の合計欄と貸方の合計欄が、また損益計算書における費用(＝借方)の合計欄と収益(＝貸方)の合計欄とがそれぞれ一致するなど随所に見られる。

現　金　1

9/1	資本金	200,000	9/3	備　品	40,000
12	売　上	70,000	16	有価証券	65,000
			25	給　料	15,000

売掛金　2

9/21	売　上	170,000			

有価証券　3

9/16	現　金	65,000			

繰越商品　4

9/1	資本金	100,000			

備　品　5

9/3	現　金	40,000			

買掛金　6

			9/7	仕　入	180,000

資本金　9

			9/1	諸　口	300,000

売　上　10

			9/12	現　金	70,000
			21	売掛金	170,000

仕　入　11

9/7	買掛金	180,000			

給　料　12

9/25	現　金	15,000			

図3　各種元帳

次に、仕訳帳(図2)をもとに各種元帳(図3)に記帳される。例えば、9月1日の諸口に対しては、資本金の300,000円が資本金元帳(元帳番号9)に、現金の200,000円が現金元帳(元帳番号1)に、繰越商品の100,000円が繰越商品元帳(元帳番号4)にそれぞれ転記される。ここでも『帳票の転記では、転記元と転記先の数値が一致する』というデータ間制約が働いている。従って、転記元と転記先において両者の数値が一致すると定義しておけば、システムが帳票の転記作業を代行してくれることになる。

『或る帳票の複数欄の間の計算結果が他の帳票の或

る欄の値に一致する』というような制約もある。例えば、各元帳（図３）における借方の合計値と貸方の合計値が、合計残高試算表（図４）の借方の合計欄と貸方の合計欄における元帳別の値にそれぞれ一致する、という例がある。

このように事務処理プログラムに代表される、data drivenな振舞いをするソフトウェアでは、システムの振舞いがデータ項目間制約に従う事例が随所に見られるので、データ項目間制約によるプロトタイピングが有効である。

昭和×年９月30日

借方		元丁	勘定科目	貸方	
残高	合計			合計	残高
150,000	270,000	1	現金	120,000	
170,000	170,000	2	売掛金		
65,000	65,000	3	有価証券		
100,000	100,000	4	繰越商品		
40,000	40,000	5	備品		
		6	買掛金	180,000	180,000
		9	資本金	300,000	300,000
		10	売上	240,000	240,000
180,000	180,000	11	仕入		
15,000	15,000	12	給料		
720,000	840,000			840,000	720,000

図４　合計残高試算表

(２)画面遷移のシミュレータ機能

システムの動的な振舞いをシミュレートすることにより、ユーザの要求する仕様を確認する機能である。具体的には、入力データとシステムの内部状態によって画面が切り替わり、その処理結果が想定された画面の想定された位置に表示される様を本番と同じデータを使ってシミュレートし確認する機能である。例えば、図２の仕訳帳の諸口において9/1 現金 200,000円が記入されると、画面が図３の資本金元帳に切り替わり、その借方の欄に9/1 資本金 200,000円が表示される。次いで、画面が図４の合計残高試算表に切り替わり、その借方の合計欄と残高欄がともに 200,000円と表示される。この一連の画面切り替えと処理結果の表示を制御し支援するのが画面遷移シミュレータである。

(３)入力データの確認表示機能

データ間制約や画面遷移のシミュレータ機能などを使ってシステムの振舞いを確認する折りに、入力されたデータが正しく入力されたか否かを確認するために、入力データを表示する機能である。

(４)プロセス間制約に従って動作するシステムの振舞いを確認する機能

シングル・プロセスで動作するソフトウェアを対象とする場合には、与えられた仕様を満足するプログラムを自動プログラミング機能によってそのまま自動生成すればよい。しかし、並列または並行で動作するソフトウェアを対象とする場合には、プロセス（またはタスク）間で成立する条件（＝プロセス間制約）を考慮しなければならない。プロセス間制約には２種類ある。第一は、同一処理内容のプロセスが同時に幾つまで動作してもよいか、という最大並列（または並行）プロセス数に関するもので、プログラムを実現する上での制約となる。第二は、並列または並行して動作する複数プロセス間における実行順序と同期に関する制約である。この制約は、これまでは処理結果の無矛盾性をいかに保証するか、という問題として扱われてきた。従って、この問題の本質は、システムの振舞いが制約に従っていることをいかにして確認（＝プロトタイピング）するかということよりも、プロセス間に成立する条件を制約として与えることにより、処理結果が無矛盾となるようなプログラムをいかにして実現するか、ということにある（制約の解消は自動プログラミングの問題である）。それ故、プロセス間制約に関するプロトタイピングでは、プロセス間制約の情報が、自動プログラミングのための情報として正しく与えられたか否かを確認するだけでよい。

先に掲げた例題の仕様を満足するオンライン・リアルタイム処理プログラムのプロトタイピングと自動生成を考える。

４．１節の形式やレイアウトに関するプロトタイピングと４．２節のデータ間制約に従うシステムの振舞いの確認が終わったので、シングル・プロセスで動作するプログラムの自動生成を行ってもよい状況となった。そこで、ユーザは、シングル・プロセスで動作するプログラムの自動生成に必要な情報（＝プログラムの機能に対する要求仕様）を図５～図６の手順で与える。オンライン・リアルタイム処理では、図５に示したほぼ９種類の骨組み部品を用意しておけば、それらのカスタマイズ（自動修正）機能と自動組合せ機能により、オンライン・リアルタイム処理のどのようなプログラム制御構造をも作り出せることが既に判明している13)。従って、図５では、それらの中の１つ「データ・エントリ処理（確認画面を表示するもの）」を選択すればよい。すると図６のような、その詳細仕様についての問い合わせ画面が呈示されるので図６のように応えればよい。その結果、ＰＡＰＳは図７のようなシステムの流れ図を生成する。

```
オンラインリアルタイム処理一覧

・目標プログラム名称：  TEST01
・処理選択→_____
  (以下の処理から選択する処理の番号を全て処理順に入力して下さい)
  1. データ・エントリ処理(確認画面を表示するもの)
  2. データ・エントリ処理(確認画面を表示しないもの)
  3. 問い合わせ処理(検索編集処理)
  4. 更新処理(更新する前に更新情報の確認画面を表示するもの)
  5. 更新処理(更新する前に更新情報の確認画面を表示しないもの)
  6. 照合更新処理(更新する前に更新情報の確認画面を表示するもの)
  7. 照合(更新)処理(更新する前に更新情報の確認画面を表示しないもの)
  8. 業務振り分け処理(起動した業務プログラムの画面を呈示するもの)
  9. 業務振り分け処理(起動した業務プログラムの画面を呈示しないもの)
  0. システム終了
```

図5　オンライン・リアルタイム処理での処理選択

```
データエントリ入力(確認画面出力あり)の
仕様を入力して下さい。

入力ＤＢの数はいくつですか。        0
出力ＤＢの数はいくつですか。        10
出力ＤＢの名称を入力して下さい。
  1. SHIWAKE-CHO
  2. GENKIN-MOTOCHO
  3. URIAGE-MOTOCHO
  4. SHOKEN-MOTOCHO
  5. KURIAGE-MOTOCHO
  6. BUHIN-MOTOCHO
  7. SIHONKIN-MOTOCHO
  8. URIAGE-MOTOCHO
  9. SHIIRE-MOTOCHO
 10. KYURYO-MOTOCHO
```

図6　データエントリ処理(確認画面出力あり)の仕様入力

この図7はシングル・プロセスで動作するソフトウェアのシステムの流れ図(＝システム・フローチャート)であり、並列または並行して動作するプログラムを自動生成するには、複数のプロセス間で成立する制約条件をさらに与えなければならない。そして、このときプロセス間制約を規定するための情報が正しく与えられたことを、要求仕様を満足するプログラムを自動生成する前に確認できなければならない。

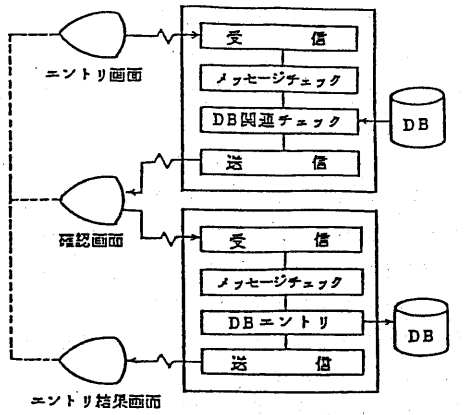

図7　データエントリ処理(確認画面出力あり)のシステムフローチャート

システムの振舞いがプロセス間制約に従うことの確認は、プロセスの数だけ用意されたトークン(◎印や●印など)が、システムの流れ図の上を制約に従って遷移する様をアニメートすることによって行う(図8)。

ここで、プロセス間制約の情報は、システムの流れ図の上を遷移する、これらのトークンの振舞いに対する制約として与えられる。

システムの振舞いがプロセス間制約に従うことが確認された段階で、自動生成を指示すると、ＰＡＰＳはユーザの要求仕様を満足する、並列または並行で動作するプログラムを完全自動で生成する。

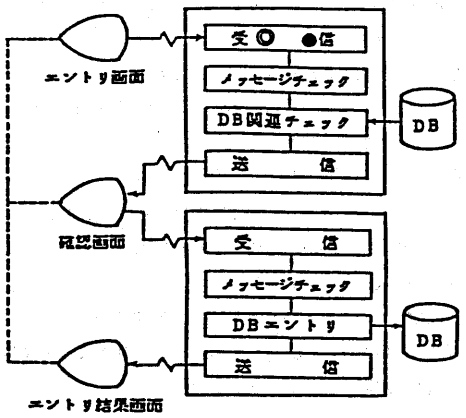

図８　データエントリ処理（確認画面出力あり）のシステムフローチャート＜トークン付き＞

５．性能を事前評価するための機能

ＰＡＰＳでは、対象とするプログラムを自動生成する前に、自動生成されるプログラムの性能（レスポンスタイムやスループット）を予測し評価する機能を持っている（詳しくは文献14を参照されたい）。ここで、レスポンスタイムとは、１件のトランザクションが入力されてから、その処理結果が出力されるまでに要する時間である。スループットとは、単位時間に処理されるトランザクションの件数である。対象とするプログラムを自動生成する前に、そのプログラムの性能評価が可能となる理由は、ＰＡＰＳが完全自動によるプログラム生成を実現している[12]からである。即ち、プログラム自動生成のためにユーザが与える情報は要求仕様だけであり、オウン・コーディングは一切必要ないからである。プログラム自動生成過程において、生成されるプログラムにユーザが一切手を加えていないということは、自動生成されるプログラムのソースコードが、与えられた要求仕様のみによってユニークに決定するということであり、それによってプログラム実行時の走行ステップ数（＝ダイナミック・ステップ数）も決定するということである。従って、ＣＰＵのMIPS（＝Mega Instruction PerSecond）値やDISCへのアクセスタイムなどをデータベースに登録しておけば、単位時間当りのトランザクションの発生件数を与えるだけで、待ち行列の理論によりレスポンスタイムやスループットを計算することができる。こうして計算されたレスポンスタイムを画面遷移シミュレータの画面切り替え時間としてセットすれば、ロギングされていた入力データがデータ入力操作のシミュレート機能によって自動的に読み出されるので、ユーザが指定した条件に従って画面が正しく切り替って行く様を予測された時間間隔でシミュレートすることができる。これによって、ユーザの要求するレスポンス時間を実際の時間で示すことができるので、性能要求を達成できるか否かを、ユーザがレスポンスタイムに対して抱いていた時間的感覚に照らし合わせて確認することができる。

この性能予測機能におけるもう１つの利点は、平行に動作するプロセス数（＝タスク数）を変更することにより、最適なタスク数を決定することができることである。

一方、ＰＡＰＳで生成されるプログラムは綿密に設計されており、アクセス特性に基づくデータベースのチューニングを除けば、ソフトウェアだけでは改造によってこれ以上実行効率がよくなることはない。従って、データベースのチューニングをしても要求性能を達成できないときには、上位のハードウェアを使用することになる。このとき、どのクラスのハードウェアを使用すればどのような性能を確保できるかを、推奨するハードウェア構成とそのときの性能値という形で呈示することも容易である。

６．おわりに

事務処理ソフトウェアに代表される、data-drivenな振舞いをするソフトウェアに対する開発方法の理想は、ツールの利用によってソフトウェア開発が①～④のステップで行い、⑤の条件を満足させることである。

①ライフサイクルの早い段階に、プロトタイピングすることにより、与えられた要求仕様がユーザの意図どおりであることを確認する。

②与えられた要求仕様を解析して、要求仕様そのものが無矛盾性・（並行プロセス間の）妥当性・完全性（曖昧さや欠落情報のないこと）・到達可能性など

を満足することを確認する。

③自動生成される前に、自動生成されるプログラムの性能（レスポンスタイムやスループット）を計算し評価して、性能面でもユーザの要求を満足するようにする。

④確認された要求仕様を満足するプログラムを完全自動で生成する。このとき、

⑤生成されたプログラムの品質が保証されており、ユーザはデバッグする必要がない。

ＰＡＰＳは、上記の理想を実現するためのシステムである。本稿では、このうちの①プロトタイピング機能、②要求分析機能の一部（並行プロセス間の妥当性の確認機能）、③性能を事前に評価するための機能について明らかにした。プロトタイピングを効果的に行うために、対象とするソフトウェアの性質を分析し分類して、プロトタイピング技術の各々が、どのようなソフトウェアに対してどのように有効かを明らかにした。そして、その評価を踏まえ、事務処理ソフトウェアに代表される、 data-drivenな振舞いをするソフトウェアでは、自動プログラミング・システムと結合したアニメーション（それも特に WYSIWYG方式）によるプロトタイピングが有効であることを示した。さらに、アニメーションによる方式では制約を利用した方法が有効であることを示した。

[参考文献]

[1]古宮誠一：「プログラム・シンセシス法の分析と実用化への方向付け」,第４回技術発表会論文集,情報処理振興事業協会,pp.71-85(1985).

[2]古宮誠一：「部品合成によるプログラム自動合成システム　～部品表現のモデルとその意味記述について～」,第５回技術発表会論文集,情報処理振興事業協会,pp.151-154(1986).

[3]古宮誠一：「知識ベースを利用したプログラム自動合成システム」第５回技術発表会論文集,情報処理振興事業協会,pp.17-25(1986).

[4]古宮誠一：「部品合成によるプログラム自動合成システムＰＡＰＳ　～知識工学的アプローチを用いたその実現方式」, 情報処理学会研究報告, 87-SF-21, Vol.87,No.40,pp.5-12(1987).

[5]古宮誠一,原田 実：解説「部品合成による自動プログラミング」,情報処理, Vol.28, No.10, pp.1329-1345(1987).

[6]古宮誠一：「部品合成によるプログラム自動合成システムＰＡＰＳ　～部品追加に強い部品の管理・検索方式」,第35回全国大会講演論文集, pp.1062-1062(1987).

[7]古宮誠一：「ソフトウェアの再利用における類似部品の扱い方について」,電子情報通信学会春季全国大会講演論文集」,D-374(374).

[8]古宮誠一：「類似性の概念と類推の正しい適用方法について～部品合成による自動プログラミングへの応用を考える」,ソフトウェア・シンポジウム」, pp.69-78(1988).

[9]古宮誠一：「部品合成による自動プログラミング・システムの実現方式について」,情報処理学会研究報告,88-SF-24(1988).

[10]部品合成による自動プログラミング・システムＰＡＰＳ～プログラミングの知識とその使い方について」,電子情報通信学会,技術研究報告,AI88-34, pp.19-26(Sept.,1988).

[11]古宮誠一：「プロトタイピング技術の分析と実用化への方向付け」,第７回技術発表会論文集,情報処理振興事業協会,pp.37-465(1988).

[12]古宮誠一：「部品合成による自動プログラミングシステムで生成されるプログラムの品質保証方法」電子情報通信学会,技術研究報告SS89-7(July,1989)

[13]古宮誠一：「ソフトウェア再利用支援ツールの構築に必要な部品とその完備性の保証方法」、日本ソフトウェア科学会全国大会論文集、pp.377-380(1989).

[14]古宮誠一：「性能予測機能を持つ自動プログラミングシステム」,ソフトウェア・ツール・シンポジウム, 情報サービス産業協会,pp.67-84(1990).