ソフトウェア工学 69－1
（1989. 11. 24）

仕様記述過程モデル化のための実験と分析

古宮誠一[1]，佐伯元司[2]，池克俊[2]，大蒔和仁[3]，白井豊[4]，本位田真一[5]，内平直志[5]，西村一彦[5]，
大槻繁[6]，蓬莱尚幸[7]，加藤潤三[8]，大林正晴[9]，松浦佐江子[9]，上林憲行[10]，荒谷徹[10]，
大木幹雄[11]，松田元彦[12]，村井進[13]

[1]情報処理振興事業協会，[2]東京工業大学，[3]電子技術総合研究所，[4]協同システム開発㈱，[5]㈱東芝，
[6]㈱日立製作所，[7]富士通㈱，[8]日本ユニシス㈱，[9]㈱管理工学研究所，[10]富士ゼロックス㈱，
[11]日本電子計算㈱，[12]住友金属工業㈱，[13]フリーランス・テクニカルライター

設計方法論と実際の設計プロセスは別物である．というのは，設計方法論は、誰が用いても同じプロセスで同じ成果が得られるほど，人間の作業を規定しているものではないからである．それ故，設計プロセスは設計方法論のインスタンスである．従って、設計方法論自身を分析するよりも，設計プロセスの事例を収集し，分析分類するほうがソフトウェアの設計プロセスを明らかにすることに繋がる．そこで、タイプの異なる問題として図書館問題とLift問題を選び，種々の仕様記述言語や設計方法論を用いて仕様記述を行なった．本稿では，この結果をもとに，仕様記述の観点から設計プロセス自身を分類するとともに，仕様記述言語や方法論を分類し，問題のタイプと設計プロセスの関係を明らかにしている．

An Analysis of Experimental Results on Specification Processes for Process Modeling

Seiichi Komiya[1], Motoshi Saeki[2], Katsutoshi Ike[2], Kazuhito Ohmaki[3], Yutaka Shirai[4], Shinichi Honiden[5], Naoshi Uchihira[5], Kazuhiko Nishimura[5], Shigeru Ohtsuki[6], Hisayuki Horai[7], Junzou Kato[8], Masaharu Ohbayashi[9], Saeko Matsuura[9], Noriyuki Kamibayashi[10], Tooru Aratani[10], Mikio Ohki[11], Motohiko Matsuda[12], Susumu Murai[13]

[1]Information-technology Promotion Agency Japn(IPA), [2]Tokyo Institute of Technology, [3]Electronical Laboratory Agency of Industrial Science and Technology(MITI), [4]Joint System Development Corp., [5]Toshiba Corpration, [6]Hitachi Ltd., [7]Fjutsu Ltd., [8]Nippon Unisys Ltd., [9]Kanri Kogaku Ltd., [10]Fuji Xerox Co. Ltd., [11]Japan Information Processing Service Co. Ltd., [12]Sumitomo Metal Industries Ltd., [13]Freelance tenhnical writer

[1]6F Shuwa shibakoen 3-chome BLG, 3-1-38 Shibakoen 3-chome, Minatoku, Tokyo 105, Japan

Actual specification process do not exactly follow specification methodologies. Generally, specification metodologies do not regulate human activities precisely enough for any person who performs the same precedure to generate the same results. A specification process can be regarded as an instance of a specificatin methodology. General characteristics of specification process can be clarified by gathering and analyzing actual instances of specification efforts rather than by studying specification methodologies. The Library problem and the Lift problem which represent different problem areas were specified with several specification languages and specification methodologies. From these results, this paper clearly addressses the relationships between specification processes and problem types by classifying specification processes as well as by cassifying specification languages and specification methodologies from specific viewpoints.

1．はじめに

設計方法論(methodology)と実際の設計プロセス(specification processes)は別物である．というのは，設計方法論は、誰が用いても同じプロセスで同じ成果が得られるほど，人間の作業を規定しているものではないからである．それ故、設計プロセスは設計方法論のインスタンスである．従って、設計方法論自身を分析するよりも，設計プロセスの事例を収集し，分析分類するほうがソフトウェアの設計プロセスを明らかにすることに繋がる．

仕様記述言語や設計方法論は，設計者の思考を制限する．特に，設計者が問題の切口をどこに置くかというviewの設定とその設定順序を規定する．その規定は強いものもあれば設計者の裁量に任されているものもある．与えられた問題によって，適したviewとその設定順序は異なる．設計の初心者にとっては，規定の強い方が良いように思われるが，問題に適したview設定とその規定が一致していなければ意味がない．エキスパートにとっても問題に適したview設定を規定してくれる言語や方法論を選択することが，設計作業の能率化に繋がる．

本稿では，文献[３]の共通問題から，タイプの異なる問題として図書館問題とlift問題を選び，種々の仕様記述言語，設計方法論を用いて記述を行なう．その際に得られた設計プロセス自身をviewの設定とその順序から分析し分類する．また，実験に使用した共通問題を同じ観点から分類する。記述結果をもとに問題に適したviewの設定とその設定順序を考察する．実験に使用した言語，方法論が、タイプの異なる問題である図書館問題とlift問題のそれぞれに対して適切なview設定を行なえるかどうかを議論する．我々の目標は，設計プロセス自身を分類するとともに，同じ観点から記述言語や方法論を分類し，問題のタイプと設計プロセスの関係を明らかにすることである．

他にも記述実験を行なったり，設計法を比較した研究[2]があるが，assessmentを与えているだけで，我々のような観点に立って，分析しておらず，支援系に反映させることが不可能である．

2．実験と分析方法

(１)実験方法 -- 目的と内容

設計プロセスは、実験対象としての問題に依存するので，性質の異なる問題を２題（図書館とlift）を選んだ．各問題について，被験者が経験している言語と方法論を使用して仕様記述を行い、そのときの履歴を整理してもらった。

(２)使用した言語と方法論

仕様記述実験に適用した言語および方法論は、表１のとおりである。

表１　実験に使用した仕様記述言語と方法論

言語	方法論
LOTOS[8),9)]	JSD[14),15)]
PAISLey[12)]	OOD[13)]
STATEMATE[11)]	FOREST[16)]
MENDEL[1),2)]	DMC[18)]
	SORA[7)]
	EER[17)]
	TELL[4),5),6)]
	Algeraic Approach + SA[10)]

（注）

DMC: Design method based on concepts

OOD: object oriented design

SORA:syntactic oriented requirements arrangement

EER: extended entity relationship model

設計方法論を使用した被験者にはその方法論に準拠した方法で仕様記述実験をして貰い、仕様記述言語を使用した被験者にはその言語仕様に準拠した方法で仕様記述実験をしてして貰った。しかし、前述したように、設計方法論は、誰が用いても同じプロセスで同じ成果が得られるほど，人間の作業を規定したものではないので、方法論で規定されていない部分については被験者の裁量に任せた。また、一般的には、仕様記述言語のほうが方法論よりもview設定に関する規定が弱いので、view設定の方法など設計の多くの部分を被験者の裁量に任せた。

(３)分析方法

設計プロセスの履歴をもとに，設計プロセスを分析し分類するためのviewを設定する。そのために、各設計プロセスにおいて、設計のviewをどのように設定したかを被験者に書いて貰った仕様記述過程の履歴とそれをもとにした事後のインタビュー調査によって確認する。設定したviewとその設定順序によって設計プロセスを分析する。実験の中で被験者が設定した設計のviewのすべてを以下に列挙し、併せて各々に対する本稿での定義を明らかにする。

設計のview設定に際しては、STATEMATE[11]におけるfunctional viewとbehavioural viewの定義においてHarelが指摘したように、functionとbehaviourが直交する（共通部分を持たない）ように定義できる。我々の定義もそれに従っていることに注意されたい。

①data（以後、Dと略記されることもある。）
計算機による処理対象となるものをdataと呼ぶ。

②process（以後、Pと略記されることもある。）
dataに直接働いてそのdataを変化させるものの実体をprocessと呼び、その機能は変化する前のdata（=input data）と変化した後のdata(=output data)との対応関係によって規定される。

③function（以後、Fと略記されることもある。）
process, input data, output dataの３つ組またはその系列をfunctionという。

④state（以後、Sと略記されることもある。）
計算機によってシステム化される対象を、有限個の事象間の遷移としてモデル化したとき、遷移の対象となる事象をstateという。

⑤event（以後、Eと略記されることもある。）
stateを変化させる要因となる、外部からシステムへの入力をeventといい、その振舞いは変化する前のstateと変化した後のeventとの対応関係によって規定される。

⑥behaviour（以後、Bと略記されることもある。）
event, 遷移する前のstate, 遷移した後のstateの３つ組またはその系列をbehaviourという。

⑦component（以後、Cと略記されることもある。）
functionまたはbehaviourとなり得るもの（実体）でprocess, data, event, stateなどの概念がまだ顕在化されていないものをcomponentという。

上記の定義より、次式が導かれる。

$F = D + P$

$B = S + E$

$C = F / B = (D + P) / (S + E)$

但し、＋は「直和」を意味し、／は「論理和」を意味する。

3. 設計プロセスの分類

本章では、図書館問題やlift問題が持つソフトウェア・システムとしての特性や問題の与えられ方の特徴が、実際の設計プロセスにどのように影響を与えているかを明らかにするために、問題固有の特性を踏まえた評価項目と、実験で行われた実際の設計プロセスを示す。

次に、問題固有の評価項目を評価するための項目として、問題によらない一般的な評価項目を以下に列挙する。

(1)最初の切口として設定したviewによって、問題文に容易に取り掛かれるか（必要な情報を漏れなく抽出できるか？）
(2)設計プロセスは自然か（前の作業で得られた情報を利用できているか？）
(3)設計上の手戻りはどの程度か？
手戻りは前作業での抽出漏れが原因か？
手戻りの回数は？
手戻りするとき、どこまで戻るのか？
修正コストとそれに伴う影響はどのくらいか？

3．1 図書館問題

本節では、図書館問題の特性を明らかにし、この特性を踏まえた評価項目を設定する。しかる後に、仕様記述実験で行われた実際の設計プロセスを明らにする。

3．1．1 図書館問題の特性

図書館問題の特性は次のとおりである。
(1)図書館はdata-drivenな振舞いをするシステムの典型的な例である。従って、data構造の設計法を主体にした方法が有利だと思われる。
(2)問題の与えられ方として、システムに対する外界からの作用（eventやprocess）であるトランザクションが予め与えられている。
従って、eventやprocessが抽出し易い。
(3)トランザクションは、図書館の状態についての問い合わせに限られている。
(4)図書館の状態に関する制約が、予め明示的に与えられている。
(5)システム化の対象としての範囲限定（外界と内界の区別）が容易である。
(6)並行プロセスである。しかし、並行プロセス間での同期問題や排他制御の問題も仕様化の段階では起こらない（インプリメントの段階では起こる）。

3．1．2 評価項目

図書館問題固有の評価項目は以下のとおりである。
(1)図書館問題は、設計viewとしてdata構造の設計から着手するのが有利だと思われる。このため、これ

らの設計view主導型の手法では設計しやすく、他の設計view主導型の手法では設計し難いと予想されるが、実際にそうなっているか？またこのとき、それをどのようにカバーしているか？

(2)図書館問題は、eventが抽出しやすいような形で問題が与えられているが、実際の仕様記述においてそれが有利に働いているか？

これらの評価項目の評価に際しては、本章の冒頭に掲げた「問題によらない一般的な評価項目」を拠り所にして評価する。

3．1．3　設計プロセスの実際

図書館問題における実際の設計プロセスを、実験において実際に設定された設計のviewとその適用順序という観点からまとめたものを図1に示す。

なお、アンダーラインのあるものは、それが方法論ではなく仕様記述言語であることを示す。

```
DMC#1:    C=>(C+P+D)→F
EER:      C＞(D+S)          →(F+B)
                  …＞(P+E)→(F+B)
JSD:      (P+E+C)＞C＞P   =>D→F
          (P+E+C)       ＞E=>S→B
LOTOS#1: F＞(D+P)
OOD#1:    C＞F=>B=>P
SORA:     F＞S=>F=>(F+B)＞F
```

=>変換　＞分解　→合成　…＞時間的順序関係

図1　図書館問題における
実際の設計プロセス（その1）

図1は仕様記述の過程で得られた中間生成物に着目して表現されている。これらの表現をどのように読むかを例を挙げて説明しよう。

例えば、DMC法という方法論のプロセス#1では、最初に設計viewのCで生成されたものを中間生成物とし、次のステップでは設計viewのCで生成されたものと、設計viewのPで生成されたものと、設計viewのDで生成されたものの3つを組み合わせたものを中間生成物としている。そして、最後のステップでは、設計viewのCで生成されたものを中間生成物としていると読むのである。ここで、＞という記号は分解と読み、その生成物が、以前のステップで得られた中間生成物をもとに、以前のステップで用いられた設計viewの中の一部を使って生成されていることを示している。次のステップでは、(C+P+D)の設計viewによる中間生成物をもとに、設計viewのFによる中間生成物を得る。

また、方法論 EERでは、最初に設計viewのCで生成されたものを中間生成物とし、次のステップでは設計viewのDで生成されたものと、設計viewのSで生成されたものの2つを組み合わせたものを中間生成物としている。次のステップでは、設計viewのPで生成されたものと、設計viewのDで生成されたものの2つを組み合わせたものを中間生成物としている。このとき、(P+E)の前にある…＞という印は、時間的順序関係と読み、中間生成物の生成のために使われているのは設計viewの(P+E)のみで、以前の中間生成物が利用されていないことを示している。そして、次のステップでは、(D+S)と(P+E)の設計viewで生成されたそれぞれの中間生成物から、(F+B)の設計viewで生成された中間生成物を作り出していることを示している。このとき、→という記号は合成と読み、設計viewのDとPとの合成（直和）により新しい設計viewF、また、設計viewのSとEとの合成（直和）により新しい設計viewBを、それぞれ産み出していることを示している。

なお、=>という記号は変換と読み、分解＞や合成以外のview切り換えによって、以前に得られた中間生成物から新しい中間生成物を作り出すことを示している。

設計viewの中で基礎的なものは、data, process, state, eventの4つである。これらの4つだけに着目して、その適用順序がどうなっているかを調べ、図2に示す。

```
DMC#1:    PとDが同時
EER:      D → P；S → E
JSD:      P → D；E → S
LOTOS#1: DとPが同時
OOD#1:    DとPおよびSとEがそれぞれ同時
SORA:     S → E
```

(注)→は設計view適用の順序関係を示す。

図2　図書館問題における
実際の設計プロセス（その2）

3．2　lift問題

本節では、lift問題の特性を明らかにし、この特性を踏まえた評価項目を設定する。しかる後に、仕様記述実験で行われた実際の設計プロセスを明らにする。

3．2．1　lift問題の特性

lift問題の特性は次のとおりである。

(1)lift問題は、 event-drivenな振舞いをするソフトウェアの典型的な例である。従って、与えられた問題文から eventや stateが容易に抽出できる。

(2)event間の関係の記述が必要となってくる。

(3)並行プロセスである。

(4)外界との境界が比較的明白である。

(5)身近な問題であるため、システム構成を直観的に作成することができる。

3．2．2　評価項目

lift問題固有の評価項目を以下に列挙する。

(1)lift問題は、設計Viewとして eventや stateが抽出しやすい。このため、これらの設計view主導型の手法では設計しやすく、他の設計view主導型の手法では設計し難いと予想されるが、実際にそうなっているか？。またこのとき、それをどのようにカバーしているか？

(2) eventの並列性をいつ考えるのか？。最後まで考える必要がないのか、あるいは考えないのか？。

これらの評価項目の評価に際しては、本章の冒頭に掲げた「問題によらない一般的な評価項目」を拠り所にして評価する。

3．2．3　設計プロセスの実際

Algebra+SA:	(C+E)=>F
DMC#2:	C=>(C+P+D+E+S)＞(F+B)
FOREST:	C=>(C+E)=>(D+S)=>B
LOTOS#2:	F=>(D+P+E)=>(F+B)
MENDEL:	C＞E=>B＞(C+E)=>B
OOD#2:	C＞F=>B=>P
PAISLey:	C＞S=>E→(C+E)=>S=>B
STATEMATE:	C=>(C+E)=>F=>B
TELL:	C＞(F+B)=>(D+S)=>(F+B)

=>変換　＞分解　→合成　…＞時間的順序関係

図3　lift問題における
実際の設計プロセス（その1）

lift問題における実際の設計プロセスを、実験において実際に設定された設計のviewとその適用順序という観点からまとめたものを図3と図4に示す。

Algebra+SA:	E → S
DMC#2:	DとPおよびSとEがそれぞれ同時
FOREST:	E → S；P → D
LOTOS#2:	E → S；PとDは同時だがPに重点
MENDEL:	E → S；P→D （CはPの意味あいが強いのでP→D）
OOD#2:	DとPおよびSとEがそれぞれ同時
PAISLey:	S → E；P→D （CはPの意味あいが強いのでP→D）
STATEMATE:	E → S
TELL:	D → P；S → E

(注)→は設計view適用の順序関係を示す。

図4　lift問題における
実際の設計プロセス（その2）

4．議論

4．1　図書館問題について

図1で示された実際の設計プロセスからを次の(1)と(2)が言える。

(1)図書館問題のような data-drivenな振舞いをするソフトウェア・システムに対しては、 EER法のようにdata設計から着手する方法論が効果的である。

→合成は、以前のステップで得られた中間成果物を足し合わせるだけの作業である。＞分解は、以前のステップよりもviewをさらに絞り込む作業なので、以前のステップで得られた成果物を抽出する作業である。→合成や＞分解以外のview切り換えはすべて=>変換となる。従って、viewの切り換えが被験者に与える負担は＞分解や→合成では少なく、=>変換では大きくなる。

なお、…＞時間的順序関係は、以前のステップで得られた中間生成物を全く利用しないので、それが問題文のみから得られるときには、被験者の負担が最も少ない。逆に、それが問題文のみから得られないときには、以前のステップで得られた中間生成物が利用できないので、最も大きな負担となる。図書館問題の場合には、トランザクションが与えられており、 eventや process が問題文のみから抽出できるので、最も容易な場合に当たる。

一方、最初に着手する抽出作業で最も楽なのはCである。これは、FやBとなり得る候補を抽出するのみで、完全な設計viewとして分化されなくても（＝曖昧さが残っていても）よいからである。また、C＞(D+S)については、＞分解であるから、viewをさらに絞り込むことによって、Cによる中間成果物もとに D+Sによる中間成果物を生成するのは容易である。

また、(D+S)→(F+B)と(P+E)→(F+B)は、(D+P)をFに(S+E)をBとすればよいので、この作業は簡単である。

以上の理由により、EERが図書館問題に効果的であることが示された。

(2)次に容易なのは LOTOSで採用された方法である。

LOTOSで採用された方法はF＞(D+P)であり、分解だけなので図書館問題には効果的である。但し、LOTOSは仕様記述言語であり方法論ではないので、今回の実験で採用された方法を支援する方法論と LOTOSとを組み合わせれば、図書館問題のような data-drivenな振舞いをするソフトウェアに適した支援系を実現することができる。

図2に示された実際の設計プロセスから、次の(3)と(4)が言える。

(3)state 対 event

図書館問題では、もともとeventやprocessがトランザクションとして明示的に与えられている。このため、外見上、state 対 eventという対比で、図書館問題に適する設計法か否かを分類することは難しい。勿論、stateとeventは概念的に相補的であり、図書館問題のような情報管理システムでは、多くの場合両者の記述は互いに翻訳可能である。例えば、Statemate, JSD, LOTOSでは、eventを構造化したものを中間仕様として用いており、これからプロセスの構造やその内部状態を導出している。これに対して、EERでは、Event構造の代わりにデータベース仕様を用いている。DMCやSORAでは、「概念」や項の構造によってそれを表している。

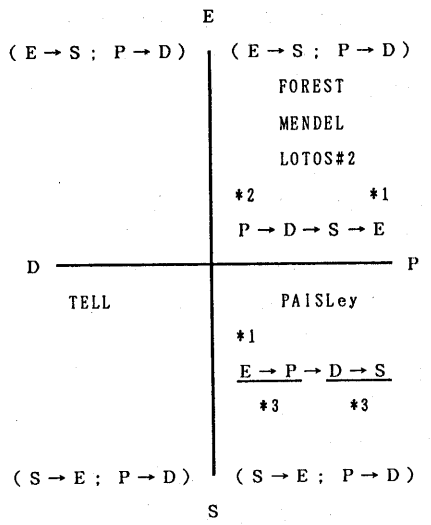
いずれの設計法においても、最終仕様の中には、幾つかに分割された動的な仕様の構成要素としてのプロセスがあり、そのプロセスの構造は、何等かの形でのeventの構造を反映させたものになっている。しかも、一般的には、Eventもプロセスの内部状態として仕様化されている。

(4)process 対 data

dataの位置づけ方法は、processに対して、その外部要素とする方法と内部要素とする方法の2通りが考えられる。前者は、DMCではdataを管理する「概念」と呼ばれるもの、OODではdata object、EERではデータベース、LOTOSではADT(abstract data type)という形で仕様化される。processの外部要素としてのdataを区別しているこれらの設計法では、図書館問題に対しては、processよりもdataの抽出に重点を置いて設計を行っている。設計順序から見ると、EERだけがprocessの抽出に先立ってdataの抽出を行っており、他の方法も少なくともprocessとdataを同時に行っている。processを優先させている方法では、dataの抽出に重点を置いた設計は困難であろう。dataの抽出に重点を置いた方法の中でも EERと他の方法との違いは、前述したように EERが data-drivenな振舞いをするソフトウェアに重点を置いた方法論であるのに対して、他の方法はそこまで徹底していない。逆に、event-drivenな振舞いをするソフトウェアへの対応も徹底していない分だけ可能となっている。区別していない設計法では、外部のデータは、実現(implementation)の概念であるとして、その意思決定を先に延ばしている。

後者は、processの内部状態として仕様化されている。すべての設計技法で、この内部データの概念は用いられている。

4．2　Lift問題について



(注)→は設計view適用の順序関係を示す。

図5　設計プロセスの分類

図３と図４で示した設計プロセスを３．２．２の評価項目で考察するために、各設計法では４つの設計Viewがどの順序で適用されているかに注目して分類し、図５を作成した。最初にどのviewで問題文に取り掛かり、その後どのようなviewで設計を進めて行くかという順序が分類ごとに異なっているからである。Lift問題を記述する上で、それらの適用順序の相違が、各設計プロセスにどのような与えるかを考察する。

(１)event抽出のタイミング（*1の位置）について

Ｄ－Ｓモデル（TELLなど）は、設計プロセスの後段でStateを洗いだし、その後でそのstateを変化させるeventを抽出している。これに対してＥ－Ｐモデル（FORESTなど）では、設計プロセスの前段で eventを抽出している（このとき、 stateの存在をexplicitには意識していない）。この２つのモデルでは、他の設計プロセス（Ｃ→(Ｄ+Ｓ)）が両者とも同じなので、eventの抽出タイミングの相違による手戻りが考察の対象となる。Ｅ－Ｐモデルでの手戻りは、主にデータとして解釈すべき部分まで動作主（object）として考えてしまった（例えば、ボタン）ために起こる同期関係の記述や process間の機能分担の変更というシステム構成に係わる作業であり、 eventの抽出漏れおよびその影響によるものではない。一方、Ｄ－Ｓモデルでの手戻りは、最初のviewでは抽出できなかった情報の漏れが主な原因である。

Lift問題は、問題文から eventが容易に列挙でき、processやdataに比較して漏れなく抽出できる対象であるから、取り掛かり易さからは、 eventを最初に抽出するほうが有利である。

ところで、Ｅ－Ｐモデルでは、 event抽出が最初であるため、多様な eventをいかに分かり易く分類して表現するかがポイントとなる。このため、設計手法としてもこれをきちんとサポートする必要がある。逆に、Ｄ－Ｓモデル（TELL）では、前段の作業で抽出した情報から eventが浮かび上がってくるので、その種のサポートはあまり重要ではない。

(２)Ｅ→ＰとＤ→Ｓの前後関係（*3の位置）

Ｅ－Ｐモデル（LOTOSなど）では、前段で動作の抽出（Ｅ→Ｐ）が行われ、後段でdataの抽出（Ｄ→Ｓ）が行われる。一方、Ｄ－Ｓモデル（TELLなど）では、この順序は全く逆になる。即ち、Ｄ－Ｓモデルでは、動作抽出時におけるobjectやdataの抽出漏れが手戻りの殆どである。この手戻りは、動作抽出よりもdata抽出のほうが困難な問題の場合に多くなる。逆に、Ｅ－Ｐモデルでは、data抽出よりも動作抽出のほうが困難な場合に手戻りが多くなる。

Lift問題では、動作抽出のほうが容易なのでＥ－Ｐモデルのほうがｄ－Ｓモデルよりも手戻りが少ない。

５．おわりに

問題に対する適切な設計プロセスのタイプは、図書館問題はD-S型であり，Lift問題はP-E型である。

各記述言語や方法論がD-S型，P-E型を規定しているか，積極的に支援しているかどうかは下表の通りである。

表２　仕様記述言語や方法論と問題のタイプとの関係

言語または方法論	D-S 型（図書館）	P-E型（Lift）
LOTOS	△	△
STATEMATE	△	△
PAISLey	×	○
TELL	○	○

上表は，２つの問題に対しての適合性だけであるので，総合的な評価ではない．

他のタイプの問題に対しては，当然異なる．

適切な設計プロセスの型に従って，問題のタイプも４つに分類できるようである．D-S型のようにある型に対してはもう１つ別の分類基準が必要かもしれないが，大枠はよさそうである．

問題のタイプを見究め，その型を積極的に支援している言語，方法論を選ぶべきである．また，その型を規定していない言語や方法論に対しては，その型になるように支援していくべきである．例えば，P-E型の問題が与えられたとき，P-EのView設定順序を規定していないSTATEMATEなどについては，先にProcessを抽出するようにガイドを行なう支援をすべきである．

今後の活動としては、問題の種類を増やして実験を行なうことにより、多角的な結論を導きたい．それによって、問題の特性と設計プロセスおよび設計法との関係を明らかにしたい。また、分類の軸や分類基準(Process-Data)をもっと検討すべきだと考えている。それによって、設計プロセスや設計法を評価・分類に最適な基準を明らかにしたい。これらの成果をもとに、

言語や方法論を統合した，メタな支援系を開発したいと考えている。

この研究は、情報処理振興事業協会（ＩＰＡ）技術センターにおける研究プロジェクト「ソフトウェア・プロトタイピング技術の調査研究」の研究活動の一環として行われたものである。

［参考文献］

[1]N.Uchihira, H.Kawata, and S.Honiden, A concurrent Program Synthesis using Petri Net and Temporal Logic in MENDELES ZONE, ICOT technical Report TR-449, 1989.

[2]S.Honiden, N.Uchida, K.Matsumoto, K. Matsumura, and M.Arai, An Application of Structual Modeling and Automated Reasoning to Concurrent Program Design Proc. of 22th Annual Hawaii International Conference on System Science (HICSS22), Jan., 1989.

[3]Problem Set for the 4th International Workshop on Software Specification and Design, 1987.

[4]H.Enomoto, N.Yonezaki, M.Saeki, K.Chiba, T.Takizawa, and T.Yokoi, Natural Language Based Software Development System TELL, Proc. of 6th ECAI, pp.721-731(1984).

[5]M.Saeki, H.Horai, K.Toyama, N. Uematsu, and H.Enomoto, Specification Framework Based on Natural Language, Proc. of 4th IWSSD, pp.87-94(1987).

[6]M.Saeki, H.Horai, and Enomoto, Software Development Process from Natural Language Specification, Proc. of 11th ICSE, pp.64-73 (1989).

[7]M.Beppu, "ASPELA: Algebraic Specification Language － Experimental Design of Line Editor with ASPELA －, Joint System Development Corp., Tchnical Report No.1 "FASET: Formal Approach to Sotware Environment Technology", pp.6-11(1987).

[8]ISO/IS 8807, Information processing system － Open systems interconnection － LOTOS － A formal description technique based on the temporal behaviour, 1889.

[9]K.Ohmaki and K.Futatsugi, Specification of Library and Lift Problems in LOTOS, IPSJ, Technical Report, SE64-12, 1989.

[10]T.Furukawa, J.Tsuda, and S.honiden, A Complemental Role of Algebraic Specification and Graphical Specification, 1989 Info. Sci Sympo., IPSJ(1989).

[11]D.Harel, H.Lachover, A.Naamad, A. Pnueli, M.Politi, R.Sherman, and A.Shutul-Traurling, "STATEMATE: A Working Environment for the Development of Complex Reactive System, In Proc. of the tenth IEEE International comference on Software Engineering (Singapore, Apr. 13-15), IEEE Press, New York, 1988.

[12]P.Zave, An Overview of the PAISLey Project － 1984, ACM SIGFOFT SOFTWARE ENGINEERING NOTES, Vol.9, No.4,5, 1984.

[13]G. Booch, Sogtware Components with Ada, Benjamin Cummings, 1987.

[14]J.R.Cameron, JSP and JSD: The Jackson Approach to Software Development, IEEE Computer Society Press, Los Angels, 1983.

[15]M.Jackson, System Development, Prentice-Hall, 1983.

[16]A.Finkelstein and C.Potts, Building Formal Specification using "Structured Common Sense", Proc. of 4th IWWSSD, pp.108-113 (1987).

[17]S.Otsuki, A Specification Method Based on the Entity-Relationship Data Model, IPSJ Technical Report, SE 57-1, 1987 (In Japanese).

[18]M.Obayashi, An Example of Inventory Management System Described in the Declarative Specification Language, Joint System Development Corp., Tchnical Report No.1 "FASET: Formal Approach to Sotware Environment Technology", pp.20-28(1987).

[19]J.M.Wing, A Study of 12 Specifications of the Library Problem, IEEE Software, pp.66-76 (July, 1988).