仕様記述過程モデル化のための実験と分析

1S－1 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

1．はじめに

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．我々の目標は、仕様記述の観点から実験結果をもとに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することである．本稿では，このような実験に有効な分析方法の提案とそれによる分析結果を（記述スペースの関係で）図書館問題に絞って報告する。なお、実験と実験結果の詳細は参考文献[1]を参照されたい。

2．実験と分析の方法

(1)実験の方法

図書館問題について，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。仕様記述実験に適用した言語および方法論は、図1のとおりである。

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。

(2)分析方法

実際のソフトウェアの設計では、各設計プロセスごとにviewが設定され、そのviewを基づいて設計が行われる。従って、設計プロセスの事例を多く収集し、その中で設計のviewがどのように設定されたかを分析すれば、問題のタイプとそれに適した設計プロセスの在り方が解る筈である。そこで、実際の設計プロセスにおいて、viewがどのように設定されたかを被験者に書いて貰い、こ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事実と詳細を確認した。そして、設定されたviewとその設定順序によって実際の設計プロセスを分析した。

実験の中で被験者が設定した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。

①**data**（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。

②**process**（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。

③**function**（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。

④**state**（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。

⑤**event**（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 stateとの対応関係によって規定される。

⑥**behaviour**（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。

⑦**component**（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。

なお、設計のview設定に際しては、 STATEMATE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。我々の定義もこの原則に従っていることに注意されたい。

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．図書館問題の特性とその評価項目

3．1．　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。

An Analysis of Experimental Results on Specification Process for Process Modelling
by Seiichi KOMIYA(Information-technology Promotion Agency, Japan), et al.

(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）。

3．2　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？

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。

(3)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？）

(4)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？）

(5)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？

4．　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
```

(注)
DMC: Design method based on concepts
OOD: object oriented design
SORA: syntactic oriented requirements arrangement
EER: extended entity relationship model
=>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生成物がどのような操作とどのようなviewで得られたかに着目して表現されている。ここで、＞という記号は分解と読み、直前のステップで得られた生成物から、直前のステップで用いられた設計viewの中の一部を使って、その生成物が生成されていることを示している。→という記号は合成と読み、以前のステップで得られた複数の生成物から、そこで使われている設計viewの直和によって、その生成物が生成されていることを示している。=>という記号は変換と読み、直前のステップで得られた生成物から、＞(分解)や→(合成)とは別の操作で、その生成物が生成されていることを示している。…＞という記号は時間的順序関係と読み、生成に際してはその設計viewが使われているのみで、以前の生成物が利用されていない（即ち、時間的順序関係において、その設計viewがたまたまその位置を占めているのみである）ことを示している。

5．分析結果

図1で示された実際の設計プロセスから次の(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成果物を足し合わせるだけの作業である。＞(分解)は、直前のステップで使われた設計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)ではない操作はすべて=>(変換)となる。従って、view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)は＞(分解)であるから、viewをさらに絞り込むことによって、Cによる成果物をもとに D+Sによる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。

6．おわりに

設計プロセスを明らかにするには、設計の方法論自身を分析するよりも、実際の設計プロセスを分析するほうがよい。この立場に立って、実際の設計プロセスの事例を収集し、分析をした。分析に際しては、設計者が問題の糸口をどこに置くかというviewの設定とその順序に着目すべきである、というのが我々の主張である。より本質的な分析を実現するために、設計プロセスにおけるより多くの事例を収集し、より本質的な分析方法を検討して行きたい。

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

[参考文献]

[1]古宮誠一ほか: 仕様記述過程モデル化のための実験と分析, 情報処理学会研究報告, 89-SE-69, pp.1-8(1989).