Vol.2010-SE-168 No.8
Vol.2010-EMB-17 No.8
2010/6/1

有限状態機械モデルに準拠した設計方式の改善

河野善彌[†1]　近藤哲生[†2]　檜山邦夫[†3]
木村俊宏[†4]　陳慧[†5]

これは有限状態機械(FSM)モデルに準拠する組込みシステムの設計方法を数次にわたり改善した記録である．小状態数で，相互に直交的な階層的 FSM 群，この下位は拡張した構造化設計法により，相互に直交的な小プログラム群を Event driven OS で動かす方式が最良である．

Improvements of Design Method Based on Finite State Machine Model

Zenya Koono[†1], Tetsuo Kondo[†2], Kunio Hiyama[†3],
Toshihiro Kimura[†4] and Hui Chen[†5]

This paper reports repetitive improvements of embedded system design, modeled by Finite State Machine model. The best is groups of hierarchical FSM's, where each FSM is designed with small number of states, orthogonal with each other, and having lower level hierarchical programs, which is small and orthogonal with each other, and the entire system runs on an event driven OS.

1. はじめに

この論文は「組込みシステム」の１種である電話交換システムを中心に筆者等が数次の改善をした流れ,成熟過程の報告である．予備知識として 1950 年代末の技術から始める．当時，今の組込みシステムは順序論理であるとの認識が電子交換の研究者に広まった．これはハードウエアで論理ゲートにより容易に実現できた．しかし，ソフトウエア側で一般製品としてその技術が使えるようになったのは，計算機が 1 チップで作れ，その関連ソフトウエアが市場から利用可能になった時点以降であった．そこで，本論文ではこの時点以降の流れを中心に報告する．

2. 布線制御方式での実現

1959 年，当時の通信業界の巨人であったベル電話研究所(Bell Telephone Laboratories, BTL)は，先駆的研究モデル ESSEX (Experimental Solid State Exchange)[1] を公開した．これはデイジタル伝送を電話交換のスイッチに使う夢のシステムであり，日本では 1990 年代に普及した ISDN 方式の約 30 年も昔の祖先である．この論文には，このシステムの制御の概念を簡単な状態遷移図で示していた．

この時期の日本は BTL が 1940 年代に開発した電磁式のクロスバ交換機を 20 年遅れで（それも大変な苦労をしながら）日本化していた．その制御部（マーカー）は，多数の接点を持つリレーを数十個使うもので，最も設計が難しい装置であった．

「この図のとおりに作れば，難問題である交換機の制御が簡単にできる！」．BTL は当時ノーベル賞を続々と受賞していた．次々と新技術を導入して難問題を解決する BTL の優れた技術力は，電子交換を研究していた電電公社，交換機メーカ，大学の研究者に大衝撃を与えた．

故猪瀬博博士（当時東京大学助教授）は, BTL に留学中にデイジタル交換の新基本方式である Time Switch を発明された．その縁で BTL の委託を受け ESSEX を引継ぐデイジタル交換システム（CAMPUS)プロジェクトを開始した[2,3]．1950 年代，電子装置の開発は方式研究以前に，自分自身で基礎部品と基礎回路を開発せねばならなかった．当時の東京大学では，かって日本最初の真空管式電子計算機を目指して出発した TAC (Tokyo Automatic Calculator)の調整が最終段階であった．これを追掛け，パラメータ励振現象を用いる磁性部品論理用 パラメトロン のコンピュータが部品から出発して計算機の完成を目指していた．

河野，故高木幹男，上野三朗他のチームは磁歪遅延線路と周辺回路を開発し，続けて CAMPUS システムの遷移制御部を担当した．図 1 [4]の左はその状態遷移図で,１４状態であった．右は実現した制御部の論理構造図である．中央は遅延線路による 4 ビットの動的に循環する記憶で 20 多重(20 タイムスロット分)の記憶である.その上は状態遷移を行わせる「組合せ論理」,その下は記憶情報から状態毎の信号を作る「組合せ論理」である．簡単明解である！

†1 クリエーション プロジェクト，神奈川，http://www.creationproj.org
Creation Project, Kanagawa Japan, http://www.creationproj.org
†2 （有）ウイン アンド ウイン，神奈川，http://www.wwiinn.net/index.htm
Win and Win Inc., Kanagawa Japan, http://www.wwiinn.net/index.htm
†3 元日立製作所，神奈川.
Formerly with Hitachi, Ltd. Kanagawa, Japan.
†4 元日立製作所，神奈川.
Formerly with Hitachi, Ltd. Kanagawa, Japan.
†5 国士舘大学 情報科学センター，世田谷.
Center for Information Science, Kokushikan University, Setagaya, Japan.

Vol.2010-SE-168 No.8
Vol.2010-EMB-17 No.8
2010/6/1

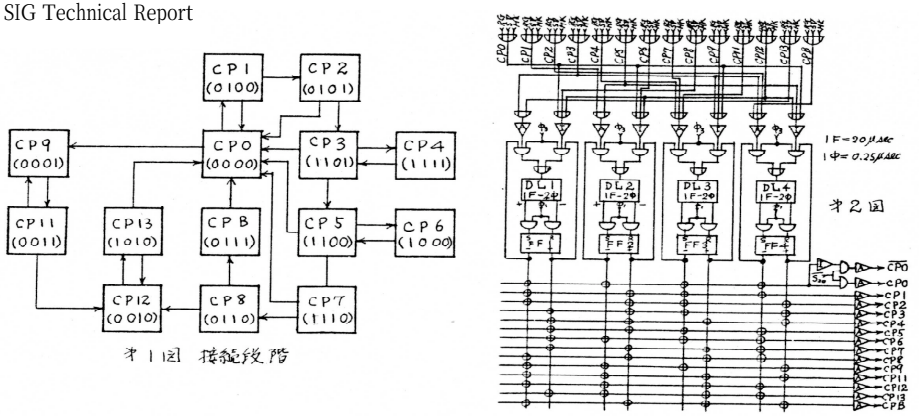


図１　CAMPUS 交換機の制御部の状態遷移図とその構造

磁歪遅延線路[1]は不安定で苦労したが，1961 年には制御部は動作状態に入った．確かに**簡単明解で，バグも少なく，完全な制御ができる優れた新技術**[1]**であった．**

設計の前に，シーケンス回路とオートマトン理論を学んだ．要点は以下である．

１．順序論理は状態の記憶と組合せ論理で系統的に構成せよ．
２．これによらないと，論理が増えるのみならず論理構成が複雑だから，誤りが中々消せない．必ずこれによれ！
３．有限状態機械（FSM）は身辺の殆どの事物を表現できる．
４．複雑な論理は，相互に直交的な FSM に分割して実現せよ．

状態数 X の FSM_A とこれに直交する状態数 Y の FSM_B とを用いると，これは状態数 X･Y の FSM_{II} に相当するが，そのコストは X+Y に過ぎない．

当時は論理ゲート当り価格は大変に高価であったから，全論理をカルノーマップを用いて簡単化した．しかし，４項によれば，その程度の小幅な改良を事後に行うのではなく，大幅に経済的な方式を最初から設計できる．これは優れた技術！これらは，河野がプロセッサ設計者として働く時，重要な指針となった．

・現在の論理設計の教科書には 1 のみの記述が多い．具体例が書いてないから後の人々には伝わらなかったか？　１項はハードウエアでもソフトウエアでも共通である．組込みシステムである交換システムは，順序論理とされていた．　２項について，実際に悩んだ経験のあると思われる著者は，きつく戒められていた．これも同様に消えたのか？大学に移った時，オートマトンが大学で教えられて居らず，「形式言語」に置換えられていて，衝撃を受けた．　Object 指向設計の曖昧な状態定義を知った時にも，衝撃を受けた．ある学会誌論文のお手伝いをした時，査読者が状態や状態遷移を全く知らず，説明も受け付けない．全てに共通する「誤りの原因」がある．

・1970 年代から計算機分野で育った人々が交換分野に導入され始めた．交換の状態遷移図を扱う設計作業に従事して半年くらい経つと大体マスターできる．交換の仕事から，元職に復帰後，「状態の概念と方法を知った事は大変なプラス」と云う．以上は御参考まで.

この時期以降，技術は以下のように進歩した．1963 年，BTL は世界初の計算機制御の電子交換システム No. 1ESS を公表し，1965 年より年間数百万回線の量産と運用を開始した．日本は 1964 年に電電公社と交換４社で共同研究を開始し，1970 年に初号システム開局，以後は量産して運用を開始した．BTL の No.1 ESS のソフトウエアには，先に世界を驚かせた FSM の思想は遺伝していなかった．しかし，日本の交換業界には状態遷移図を使用した成功体験が残っていた．特に状態遷移記号中に接続図を記入する方式は視覚的で使い易く，FSM 技術の利用が仕様表記用に定着した．

これは後に国連下部の標準化機構 ITU で図式仕様記述言語（Specification Description Language, SDL)の CCITT 国際規格になる．その後 ISO で仕様記述の標準化が審議され，計算機側は LOTOS，通信側は SDL を標準化した．しかし，LOTOS は実用にならず衰退，SDL のみ交換や通信プロトコル記述の実使用が続いた．その後，ITU では旧来の図式表記に加えてテキスト表記を開発し，両用の入出力を持ち，複数の階層的な FSM 群用に拡張された仕様記述言語 SDL として[2]標準化された．これに従い C 言語, Ada 等への変換も開発され，検証しか役立たない他の形式言語より遥かに有用であり，諸外国の交換ソフトウエアの設計には SDL CASE ツールがかなり多く使われた．デイジタル回線交換の時代が終り，これら技術者達が表舞台から立去った後，現在の組込みシステム時代が訪れた．

1 磁磁歪遅延線は，Ni 細線に蚤位のコイルと磁石を組合せ作る．1 mill volt の桁の信号の測定結果とマイクロメータで計る機構的条件で研究を進める．見えない相手と闘うには定量測定は不可欠であり，それは唯一無二の支えでもある．「見えないから難しい」ことは理由にならない．弱電と強電を併合した電気の学生として，音響領域，超音波領域，機械工学領域に始めて触れて，すこし条件が各技術の適用領域を外れる無力になることを知り，「工学とは近似の学問」と知った．そして，（謎や神秘にも見える）未知に挑戦することが工学の任務であり，その中で効果が大きいが簡単な解をプラグマチックに見出すことが重要と思い知った．

2 後の 4 節で説明する技術は SDL 等の情報無しに日立内部で育った．筆者等は 1988 年 SDL Forum に発表して，始めて SDL の最新技術に触れた．それは日本や日立に育った結果と一致していた．しかし 3 種の相違点があった．第 1 は早くから SDL 系技術を使ってきた日本勢は他国と異なり「図面から作業を始めてテキストベースの情報を結果として得る」と考えた．SDL CASE ツールが普及すると各国は日本の意見に同意した．第２は図 7 に示した状態遷移表であり，日本では実務上必須の道具と訴えたが実らなかった．最後は SDL CASE ツールの早期公開であり，理論を更に充実したい希望あるので通らなかった．この結果，SDL は世界に普及する時期を逸して miner に留まり，Object 指向設計の中心になる意図から逆転して，SDL に Object 指向を取込む羽目に陥った．理論は科学に属するから，工学のように使用者の視点も無ければ成功できない．

Vol.2010-SE-168 No.8
Vol.2010-EMB-17 No.8
2010/6/1

３．初期の FSM の利用

初期の交換システムは計算機の能力を最大限に活かす為，アセンブラ言語を使用したが，これはプロセッサー毎に違う専用言語である．従って，時代の最新技術をサポートして貰えるのは，世界シエヤの高いメーカのプロセッサーしかない．しかし，LSI の１チップ プロセッサー以前の計算機は極めて高価で，組込みシステムには使えない．結局より安価なプロセッサーを使うから，必然的に支援は極めて弱かった．

また，構造化設計の product，process である文書体系，これら両側面の技術も未知であった．許されるメモリ量は小さく，しかもぎりぎりの処理能力を求められる．「無イ無イ」ずくしが押寄せて，所謂スパゲッテイプログラムに陥った．アセンブラだからレジスタレベルの動きを記述する．設計者は全てを手順に落とさねばならないが，他の人が読むと本質が何か判り難い．判らせる為に設計文書には色々と沢山書いたが，予備知識が無い人にはますます判らなくなる．加えて，ハード側の云うことは，別世界だから中々わからない．かくして，技術レベルは中々上がらない．熟練者やハードウエア側からみれば，滑稽な位何も判らない人人に見える．

新技術の発展初期には新しいプロジェクトが次々と始まるが，既存プロジェクト自体が格闘しつつ新技術を構築する途上である．当時良く読まれた James Martin の著書[5]には「全てのプロジェクトはごく小数のエキスパートと大部分の新人で始まる..」と喝破した．洋の東西を問わず悪環境は同じ．でも客観的には技術未熟な人々が原因に見える．それは本当なのだが.... 右をご参照頂きたい．

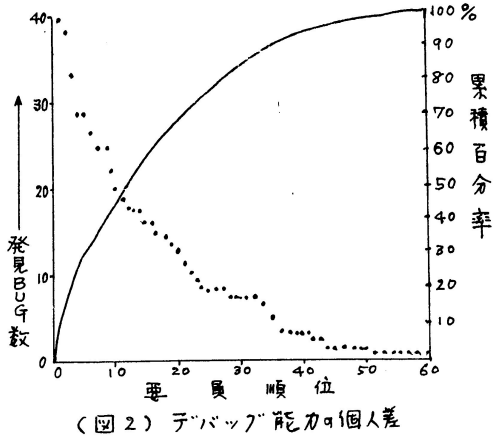
FSM は本質的にはイベント駆動であるが，OS は単なるスタートストップしかサポートしていない．現存するが「基本機能不足な OS」で如何に対処するかが課題になる．標準的な方式は，遷移原因を受取ると各種フラグを調べ分析して遷移先を決定する．しかし，このフラグは種々の混乱を招き，殆どのプロジェクトでの難物であり，FSM 方式の短所の一つと云われた．

状態遷移図を使い実用に供し始める．システムは年間に約 10%の技術的～社会的理由による機能増大が必要になる．始めは，状態遷移図中心に修正を行った．しかし，数年経つと状態数は大幅に増え持て余す．一般に「状態の爆発」と呼ばれる．爆発現象の故に「状態遷移を使う方式は駄目」との議論すら見かけた．

往事の記憶を取り戻して複数 FSM 方式を手掛けた所も幾つかあったが，どれも余り成功しなかったようである．不成功は成功の鍵，河野は各種分析を続けた．

かくして，FSM を使う技術進歩は次の時期に持越された．しかし， 不成功は成功の鍵，河野は自分が責任者であったプロジェクトを中心に各種の定量的および定性的分析を行い続けた．最初は闇の中であった．いつしか，すこしづつ判り始め，主に IEEE COMSOC の国際会議に報告した．これらの結果は，後続するプロジェクトに活かされた．

図２[6]はこの時期の問題を定量的に示した図である．これは交換ソフトウエアが巨大化し始めた 1972 年に報告された．アセンブラ時代には，バグを机上で摘出する方法が最も効率的と当時のリーダは的確に理解していた．そこで，机上チェックを集団作業で強力に推進した．図の横軸は机上チェックでのバグ摘出記録に摘出者を記録しておき，作業終了後に，累計摘出数の順に設計者をソートし累計も求めた．図の点は個人毎の合計摘出数，上に凸なグラフは最多摘出者から始めた累計数を示す．



（図２）デバッグ能力の個人差

図 2 debug 能力と累計摘出数

図の「左端近くの人達は効率的にバグを摘出するが，右端近くの人達は殆ど摘出しない」ことを示す．更に実は右端近くの人達は，「多量にバグを作り込んだ？」嫌疑もかかる．左端の人人は「自分達のように取れとは云わないが，せめてもう少し何とかならないか？」と嘆きが募る．

全体構造とその機能配備等を必死に考え抜いたリーダなどは，直ちに「これは XX がおかしいぞ！」と言い当てる．熟練者なら範囲外で可能性のある所を調べて，誤りを発見する．でも，自己担当部分しか知らない初心者はひたすらバグを作り見逃すのみ．これは「バグを取ろうとして注視している所を見ても，そこではその誤りが見えない」構造的問題である．

構造化設計，構造化文書やモジュール化が進むと上の問題は解決して行った．構造化設計は，一応機能毎に分割するから，ある程度データフローを機能毎に分断する形になり，これに対応する構造化文書は各設計結果を記すから，分断したインタフェイスが陽に出てくる．関わり合う論理やデータは局所毎に集中し，機能～処理の構造と文書構造が一致して，判り易くなり，構造問題は解決に向かった．

Vol.2010-SE-168 No.8
Vol.2010-EMB-17 No.8
2010/6/1

４．分散型状態遷移方式

技術が進展し下記が重なった時点から，

・大規模集積回路のマイクロコンピュータ
・コンパイラ言語
・構造化設計，構造化プログラム，構造化文書体系

世界中で成功が伝えられた．この時期を目指して先行して技術開発してきた[檜山7,8,9]はFSMの直交分割を徹底させ，FSMの下位のプログラム群に構造化設計方式を適用して成功した．基本は以下のとおり．

a．**全てを図面に陽に表し，図面とおりに正確に設計する．**

b． 各メカニズムに1FSMを割当て閉じた系とする．共用には厳禁．
全体のメカニズムを相互に独立なFSM群に，直交的かつ階層的に分割し位置づけする．必然的に各プログラムは単位概念毎に，かつ小さくなる．
上位はFSMの階層的な群，下に各FSMに固有な遷移ルート群ができる．これらは構造化設計により階層化し，また部品は全体に共通化できる．
上記の構造に対応した設計文書体系ができ，全てが陽に文書化できる．

c．CCITT SDLに準拠する．

図3[8,9]の最上部のビヤ樽状記号は各状態，下の楔状記号は遷移原因である．（これで過去のフラグの悩みが消えた．）この{状態S_i・遷移原因E_j}に対応する遷移ルート上を制御が流れるが，これはフローチャートと同様に書く．その中の三角形記号は分岐，長方形記号はFSM外への出力である．遷移ルートの最末尾の長方形記号は次状態定義で，図7の最上部にある「状態用記憶」を次状態に更新する．このようにすると，遷移ルートはプログラムとして独立に取扱える．

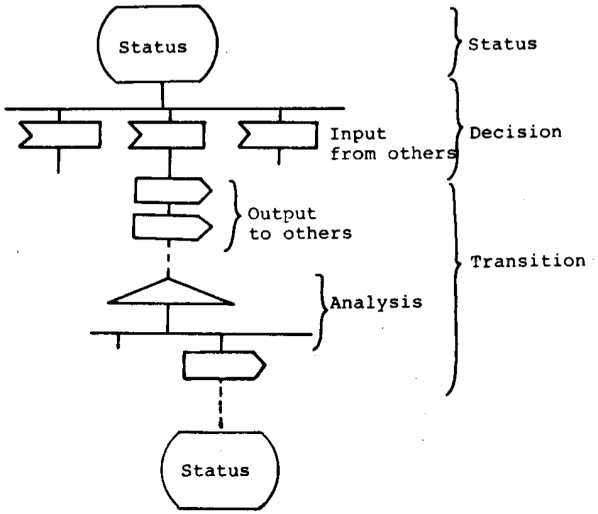


図3　SDL記法の状態遷移図

図4[8,9]はFSMを８角形で表した標準階層構造である．ソフトウエア規模を減らす為，システムを相互独立なFSM群に展開する．

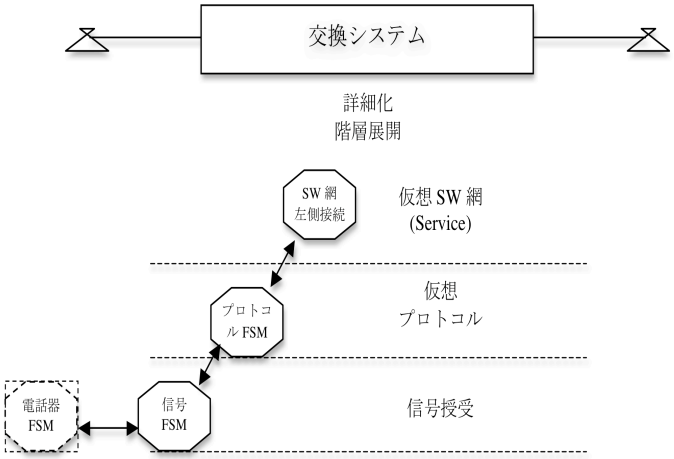


図４　FSMのなりたち

各サービスをFSM化するが，発信側と着信側（図の左/右側）に分けて実現する．その下位では，電話線からスイッチ網端子につながる加入者回路を信号プロトコルFSMと下位の信号FSMに分けた．各層毎に別機能で順次抽象度を下げ/具体化/詳細化し，相互に直交的にして，規模最小化を狙う．

各FSMの機能は，旧来の複合機能のFSMあるいは陽に表示されなかった場合よりも，単純化し個別化し明示され小さいから作り易い．あるサービス/機能名群を詳細に見ると若干の差異がある．これらは１例を作り，変更すれば，これらは標準版とオプション群に整理できる．

（FSMに全て明示する増分を許したが）ソフトウエアの全体規模は，図５[8,9]のように，旧方式の55%に小型化した．巨大で蜘蛛の巣状に入り組んだ交換ソフトウエアは，各個片の群の集合になり，ソフトウエア規模も圧縮でき，構造問題も文書問題も階層化で全て片付いた．

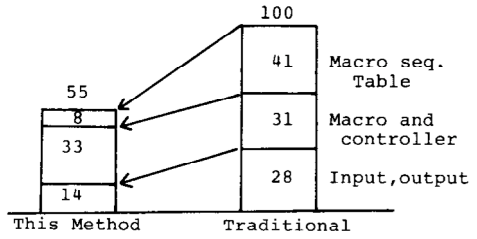


図５ ソフトウエア規模の減少

Vol.2010-SE-168 No.8
Vol.2010-EMB-17 No.8
2010/6/1

研究側母胎が略出来ると，国内 PBX，次に外国 PBX[10], 更に局設備にと，技術移転し製品化が驚く程に早く進み，個々の版毎の生産性も高い．これは

・システムが，合理的かつ多種の階層毎に，概念毎の単位機能毎に分割され，
・更に各プログラムは構造化設計で機能毎に階層化されて，全て小さく，
・それぞれに設計文書があり，全て骨組み同一，作業方式同一，文書はすぐ判る，
・規模が小さく流用度が高く，各部が小さく独立的で生産性が高く，品質も良い，

ことであり，更に全体を統合する下の標準化

・一般ユーザ向 PBX と電話会社向局設備，国内と国外，規模は小/中/大

目標から出発したことによる．これらから下記も重要な経験であった．

・母胎開発は早期に開始し，母胎の生産性よりも流用/拡張で評価すべきである．

5. 制御の高度化

４項の後，各種の試行をした．その中で最高度の実績例[11]を報告する．これは電話交換システムの機能を自動的に並行試験する．PBX 交換の試験の基本形は，発信，着信と交換手台の相互でダイヤルして接続等の動作と通話を試験する．また，システムに掛かる（各種サービスの）呼びの数を増やす負荷試験がある．各種試験はテストスクリプトに従い，多重に並行動作させる．

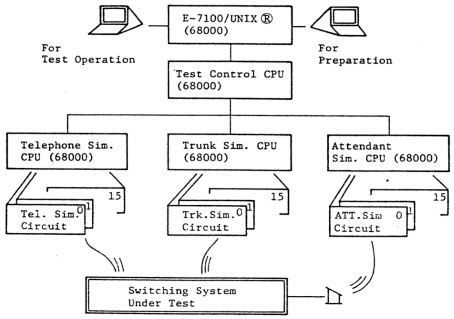


図 6　試験システムの構成図

図 6[11]はその構成を示す．最下部の被試験システムの試験用に

・電話回線　　・他電話局向け中継線　　・交換台

に接続し夫々の疑似動作を行う 3 種の疑似回路群がある．各群の 68000MPU（３台）は複数回路を並行制御する．最上位の E-7100PC は 68000MPU が UNIX 上で働き，試験の制御とその入出力およびテストスクリプトのコンパイラとして働く．これらの中間のテスト制御の 68000MPU は前記の上下間の一切をおこなうが, 試験グループ毎に，テストプログラムを配布して結果を逐次上位に上げ，テストする対毎に管理する．

これは，交換システムと同様な基本構造と下位の接続機能を持ち，通常の交換の場合とは逆に動作する．従って内部ハードウエアの回路は殆ど同じで，基本プログラムの構造等も既知である．短期開発の為，ブレークダウンして文書化するが，以下を狙った．

・(小さくすると品質が良くなる傾向がある)徹底的に小さく細切れにする．

・OS 系を高度化する．

高度化の中心は Midas OS[3]で，イベント駆動関係を浄化した．図 7 は中心部分の構造を示す．図の左端は遷移原因，中央に FSM の遷移ルートがあり，右端に外部出力がある．到来する遷移原因と外部出力は同一構造パケットに標準化した．これは遷移原因/対外指令とこれに伴う補足情報から成立ち，パケットの封筒には宛先 FSM No，送元 FSM No.を記してある．

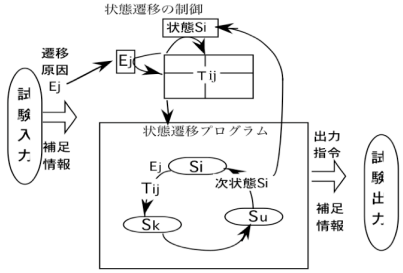


図 7　FSM の制御

図 7 の左でパケットを開いて FSM No.から中央の指定 FSM を知り，これに対応する TCB から状態情報 S_i を取出す．この S_i と，パケットから取出した遷移原因 E_j から，上部の２次元の状態遷移原因表を読む．その交点欄から遷移ルートプログラムの情報 Tij を得て状態遷移を実行する．実行の最末端で状態表示を新状態に書換えて終わる．簡単な構造で，アセンブラで作り, 大幅に高速化した．

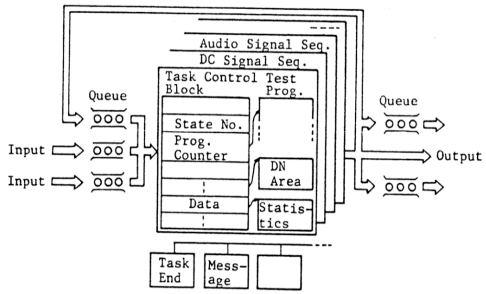


図 8　疑似プロセッサ

状態遷移は単純で定型的な処理だから，全てマクロ命令で記述し interpretive に実行する．図 8[11]はその TCB 等の構造を示す. 上部は状態遷移ルート表示があり, TCB の program counter はルートのマクロ中の現在位置を示す．

マクロ命令は Load, Store，Jump 等の単純な命令群で処理を行なう．外部への出力は原則としてキュー経由で送り出す. 他の FSM に対するものは，左側の高位の入力キューに繋ぐ．外部からのパケットは低位のキューに繋ぐ．実行管理はまず高位キューを見て，若し有れば指定の FSM を用いて処理させ，全ての FSM 間のパケットの処理が終わると, 低位のキューを調べ, もしあればそれを実行する. この仕掛けで複数 FSM 化が処置できて，最高 65K の TCB に対処できる．

[3] 使う人にご利益を齎すから，触るものを金に変えたギリシア神話の Midas の名を用いた．OS と銘打つが，状態遷移プログラムの full event driven なプログラムの実行制御に限る．

Vol.2010-SE-168 No.8
Vol.2010-EMB-17 No.8
2010/6/1

このプロジェクトでは方式要員が少ないから，方式構成の基本形を階層的に，かつデータフロー中心に記述した．受けて作る側では大変なご苦労をされたが，短期間で開発を終えることができた．

６．学生演習

河野の埼玉 1991-2001 年の在勤中とその後 5 年間，3 年生学生の約 100 チームで自販機の開発演習[12] を行った．ソフトウエア工学の授業で事前教育の後，３状態の基本的な自販機の状態遷移図を Midas OS で動かす．その後，各チームで話し合って「他社に勝る任意の自販機」を決めて，共同作業で作る．但し，Product は平均 3 状態[4]の小 FSM を使う複数 FSM 方式，Process は別途定めた作業方式で実現し，最後に各チームの全員がクラスで報告する．

図 9 は FSM の状態遷移を起こす遷移原因（例：お金の投入）と補足情報（例：500 円）がある場合を示す．出力指令も{(商品 S)を(排出せよ)}のように両者を持つから，標準化パケットは両者を持つ．この他，外部入力が直接 FSM に入ることも，あるいは外部出力を直接出すこともできる．

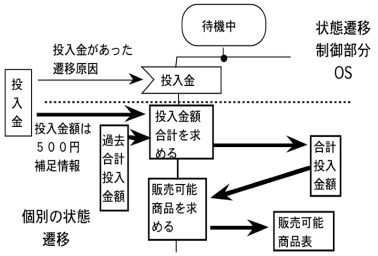


図 9 遷移原因とデータ

教育は報告済みなので，成果状況を表[12]で簡単に報告する．これは 1995 年の 8 チームの成績で，T1/2 は遷移ルートと FSM 単体テストの小計，T3 はシステムテスト，QA は別部隊が動作させて摘出した欠陥数，最右欄は総欠陥密度である．終了時の感想は「品質が良い」，「文書のお蔭だ」等．

表

1995年　自販機演習および実験　結果総括

チーム	規模（行）	T1/2 Test（件）	T3 Test（件）	QA Test（件）	総欠陥密度（件/K行）
A	283	0	0	1	3.53
B	438	4	4	1	20.5
C	486	3	0	2	10.0
D	357	2	2	1	14.0
E	858	3	0	7	11.7
F	495	1	7	2	20.0
G	442	4	1	1	13.6
H	463	3	0	0	6.5

3 年生の欠陥作り込み率は約 100 件/K 行である．彼等は，課題が出ると即刻に端末に向かいソースコードを叩込み，設計の 10 倍の時間をデバッグに掛ける．従ってこれらの総欠陥密度は驚きであり，通常と全く異なる文書と机上チェックの効果と，すぐに理解できる．

他に勝る製品に必死になったチーム，「リーダは直接作業には手を出すな！全員をその気にさせることが仕事だ！」とハッパを掛けられて，目の色が変って頑張ったリーダ達，「ソフトの仕事とは，これは全く人間相手の仕事」と悟った人など，教育効果は大きかった．

7. 今後の課題

大学に在勤中，「人に倣ったソフトウエア自動設計」[16]を研究した．これは大学院終了後に産業界で働くうちに，製造作業，ハードウエア，ソフト..設計職，管理職，経営等を経験し，全てに共通なヒトの知を確信した．多くの方々の協力を経て，これは目処がつき米国[5]，中国特許を得た．この設計の中核を付録 A に収めた．付録 A は時計プログラムの設計をデータフローの階層展開で説明する．そこでは単位データフローを 1 概念と見て，概念が定率展開網状のモデルがヒトの知を表わす．

図 10 は自動販売機の機能を小 FSM 単位に順次追加し詳細度を上げた．FSM もまた概念で，メカニズム/振舞いを持つ object であり，定率展開網状の知になる．FSM の下位の拡張した構造化設計は，図 10[13]では誌面に垂直方向に機能中心の(時計の場合のデータフロー群のような)定率展開網状の知識がある．

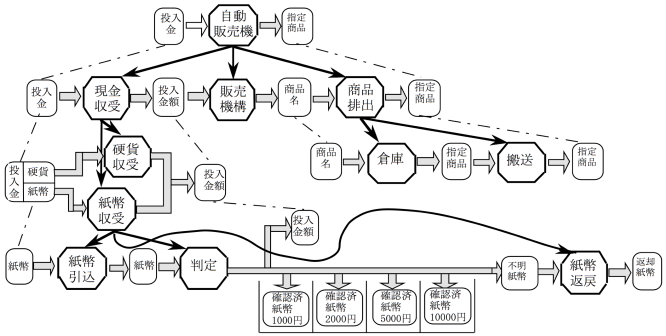


図 10 階層的 FSM 群

このように，実行手段であるプログラムは Fully Event Driven な OS があり，ある FSM は図 9 で説明したように，組合せ論理でもあるから，該 OS は縮退モードで働かせる．これは前記のように小さい部品プログラム群であるから，その生産は筆者らの自動設計を使えば，現在の労働集約的なソフトウエアは消去る．

そこで今後の課題は，上記の Fully Event から単純な start/stop までを一本化する新しい OS の概念から実現迄の仕事，本論文で説明した方式設計を各種に適用して使用するノウハウを積上げる仕事と普及，データフロー図と構造化チャート CASE ツールを用いる自動設計システムの継続的な強化と普及，今回説明を省略したが合理的定量的科学的なプロセスの普及と教育，など全てが労力的作業ではなくて知的創造的な仕事になる．これらの構築は知の集積を積上げるもので，これらが次世代の知の社会を築くであろう．これらで明るい将来の道筋が見える．

4 付録の図 A2 を参照．

5 US Patent 7,480,642 B2, Jan. 20, 2009. Priority date: May 28, 2001.(JP).

7．結論

組込みシステムについて，ハードウエアによる先駆時期から 21 世紀初めの現在まで，筆者等の改善と向上のステップを報告した．

組込みシステムは，平均 3 状態程度で，相互に直交的な小 FSM 群により，階層的に編成したシステムの構造をとり，（付録に示すように）各 FSM の下位で，拡張した構造化設計で，階層的に編成したプログラム群とで組込みシステムを構築し，完全イベント駆動の OS と組合せ用いるのが最善である．これは，システムが割に干渉が少ない仕掛けでバラバラの階層的部品群で作られるから，全てが単純かつ簡単で作り易く（生産性が高く），全体規模も小さく，誤りが少なくテストし易いから信頼度も高く，かつ機能の更新が容易である．

将来は完全イベント駆動の OS が本流になり，状態遷移構造を持つシステムが普遍化し，それは縮退動作時に現在のアプリケーションを実行するようになるであろう．

謝辞

共にバグと闘った仲間達，演習や研究に取組んだ埼玉大学の学生諸君，種々ご協力を賜った同僚各位，ご指導頂いた上長のかたがたと恩師の先生達に深く感謝いたします．

参考文献

[1] H. E. Vaughan, Research Model for Time Separation Integrated Communication, B. S. T. J., Vol. 138, pp. 909-932, July 1959.

[2] H. Inose, Time Slot Interchange in Time-Division Electronic Exchanges, J. of The Faculty of Engineering, University of Tokyo (B) Vol. XXVIII, No. 2. (1965)

[3] 河野善彌, CAMPUS 交換機とディジタル交換ネットワーク，電気・電子情報学術振興財団 第 14 回ワークショップ，講演資料集 pp.25-34，電子情報学術振興財団, 2002 年 3 月．

[4] 猪瀬博，高木幹雄，河野善弥，上野三朗，接続段階記憶装置，昭和 37 年電気通信学会全国大会，548, pp682, 1961.

[5] 北原安定（訳）James Martin（著），**電子計算機リアルタイム−システムの設計・管理・運用，日本経営出版会，1968.**

[6] 緒方秀夫，広川幸三，石野福彌，巨大プログラムのデバッグ，昭和 44 年電子通信学会全国大会 895, 1969.

[7] 檜山邦夫，水原登，草間正彦，河野善弥，電子交換機の機能分散ソフトウエア方式，日立評論 Vol. 64, No. 3, pp. 175-180, 1982 年 3 月．

[8] Hiyama K., Mizuhara N., Mochizuki K. and Koono Z., A software system for electronic switching system using distributed state transition method, IEEE International Conf. on Communications 1982, pp. 5G3. 1-5

[9] 檜山邦夫，水原登，草間正彦，河野善弥，分散型遷移方式に基づく交換プログラムの構成法，子信学会論文誌　Vol. J65-B, No. 6, pp.785-792, 1982 年 6 月．

[10] 花田吾郎，星徹，薄井彬弘，デイジタル PBX システム「DX シリーズ」，日立評論 Vol. 64, No. 3, pp. 158-162, 1982 年 3 月．

[11] Koono, Z., Kimura T., Iwamoto M. and Soga M., A stored program controlled environmental function tester based on FMM/SDL design, International Switching Symposium 1987, pp. B. 9. 5. 1-7

[12] 河野善彌,陳慧,高野英樹,森本祥一，学生チームによる組込システムの開発〜10 年間の教育から〜，第 26 回ソフトウエア品質シンポジウム報文集，一般 3-3, p. 171，日科技連, 2007.9.

[13] Zenya Koono and Hui Chen, The Ultimate Systems Development Method Based on Finite State Machine, SoMeT 08 2008, pp. 126-145, IOS Press, 2008.

[14,15] 河野善彌，陳慧，人の設計知識構造と定量評価（1/2~2/2),信学技報 KBSE2003-57-58, pp. 67-78, 2004.

[16] Koono, Z., Abolhassani, H. and Hui Chen, A new way of automatic design of software (Simulating human intentional activity), SoMeT 06, pp. pp.407-420, 2006.

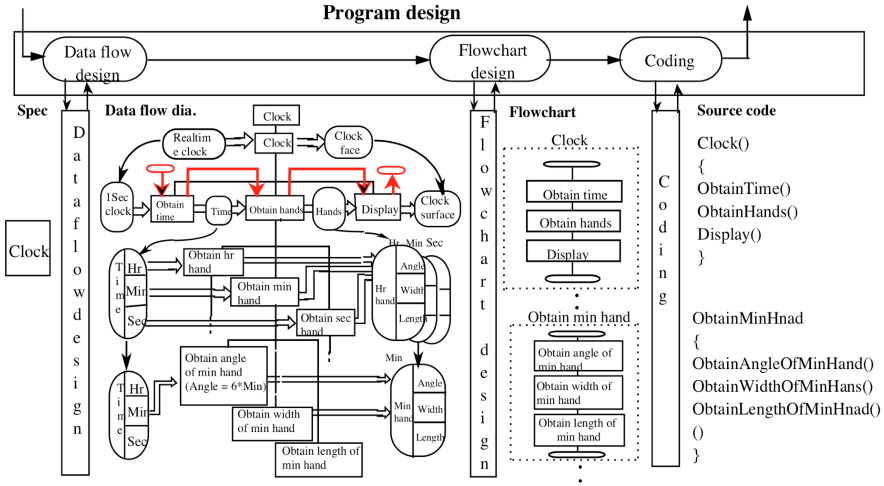


Figure A1 Design of a clock program

Vol.2010-SE-168 No.8
Vol.2010-EMB-17 No.8
2010/6/1

A. 設計の原理

図 A1 は時計プログラムの設計の軌跡を示す．報告[14, 15, 16]済みだから，要点のみを記す．中央左のデータフロー図群が設計の方向を示す．仕様「時計」に出力次に入力の各データを加え明確化し，概念「時計」の「単位データフロー」にする．第３段のデータフローは親概念「時計」を階層展開し詳細化した子概念群である．これは Myers の STS 分割[Myers]によるもので{「時を得る」,「時刻表示を求める」および「表示する」}の各概念（「単位データフロー」）の３直列系になり，これら３概念（機能）の実行順序のフローチャートも得られる．

次は，これら子概念を順次に１つずつ取出し，これを親として階層展開することを繰返す．第４段は中央の「時刻表示を得る」の階層展開を示す．ここでは入力と出力データの両方を階層展開したデータ群を用い，単位データフロー群の３並列系になる．（これは Jackson 法[Jackson]の特徴的なパターンである）．

最下段で単純化した「分針の角度を得る」の展開後は充分に詳細だから，この次には詳細化ではなく，対応するソースコード群に置換する．

設計の中心的機能は，科学技術用語を含む自然言語で表記された概念の階層展開で，ソースコードは最終出力でコンピュータを動かす為の実現手段に過ぎない．

人はある意図を持つと，その意図／目的を達成する実現手段群を考え出す．それらの実現手段群の各１を新しい目的として，これを実現するための方法を考え，階層展開を繰返す．展開の度に下位概念になり，明確化，具体化，詳細化する．

これを「ヒトの意図的行動」と名付ける．最高位の例は軍事科学の「目的の階層性」である．これは戦争の最終目的から戦略，戦術と階層展開しつつ具体化する．最下位は身体的行動である．例えば「写真を撮る」は，「カメラを構える」「ピントを合わせる」「シャッターを押す」に展開される．続けて階層展開を繰返して最後は筋肉を作動させる．設計は両者の中間で，ソフトウエアで云えば，段階的詳細化から構造化設計に進化して，階層的な機能構成にたどり着いた．

しかし，機能のみを階層展開するのは容易ではなく，下手な設計では不要な機能を作りだす．そこで，データフローを併用する「データ指向」が現れた．本設計法は単位的展開と云う枠を設け，必ず入力と出力のデータを書かせて，言葉を用いて正しい設計に導く，より厳密な方法である．これを拡張した構造化設計と名付けよう．ハードウエア論理設計の場合，このデータフロー図は論理設計に使える．ソフトウエアもハードウエアも，全く同様に設計できる．

図 A1 のデータフロー図の階層展開の率を見ると，全て 3 である．図 A2[6]に示すように平均値は約 3 弱（証明未完だが e=2.71828..と思われる）で皆同様な分布である．認知科学の知見では高速な場合には想起できる平均対象数は小さくなると云う．一方展開した子の数が増えると，明らかに頭脳の処理速度が落ちる．この２条件から最適平均値が生じると考えられる．

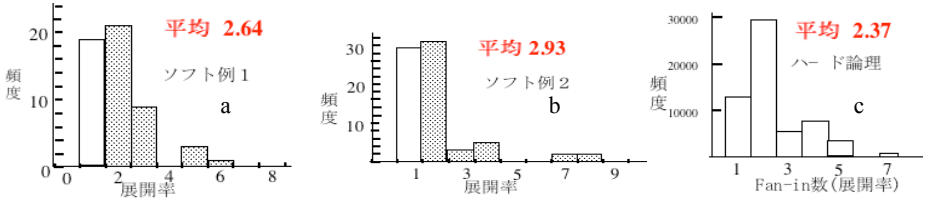


図 A2 階層展開率の分布

展開するときは，正確で明解な日本語でデータを記し，中間には処理ではなくて，処理で果たされる機能を表し，展開の深さは単位的に限り，展開率は平均３弱であることに留意する．最初のデータから機能→データと順に読み上げて違和感が無いように抽象度，視点を揃える．展開毎に厳しく厳密にデータ定義や機能定義等と照合して正邪を調べる．（小間隔だから）誤りの 80%は除去でき，驚く位品質が向上する．

設計は図 A1 で見たように，親から子への階層展開の繰返しである．親概念を与えると子概念群が得られる知識ベースを使うと，ソフトウエアの自動設計が出来る．筆者等はこれを 1998 年に Intelligent CASE Tool として作った．2001 年には．親概念を与えて，子概念群をルールベースで作る，高度化したソフトウエア自動設計 Integrated Intelligent CASE Tool[16]に強化し，2007 年中国，2009 年に米国特許を取得した．今迄ソフトウエア工学領域で何十年も成功しなかったのは何故か？それはソフトウエア工学の仕様を変換すればプログラムになると云う簡単な定義のみ考えたからであろう．なぜ何故できて特許なるのか？言語による概念展開は誰にも納得できる．

本モデルは更に発展できる．図 A１のデータフロー図群は，定率展開する網状の知的処理であり，単位知的作業あたり微小工数を消費すると考えれば工数∝作業規模)[1]，またその処理が微小確率で誤ると考えれば欠陥数∝(作業規模)[1]が導ける．系は線形性だから，展開や統合ができ，生産性や品質の改善ができる．階層展開連鎖であるデータフロー群は知の集積であり，ヒトの組織の原理や対数習熟効果も証明できる．

これら諸関係式は，製造作業での工数や誤り率等の経験式である．ハードウエア製造の定量的管理は合理的定量的科学的な Industrial Engineering, IE（品質管理等も含む）で技術が確立された．今日のハードウエアの隆盛は IE×各領域別工学の力で成遂げられた．その基礎は前記の諸経験式である．これらは使われ始めてから今迄約 100 年間，理論的な証明がなされていなかった．本モデルでは上のように証明できる．このモデルの応用は更に広がるであろう．

6 Zenya Koono, Hui Chen and Hassan Abolhassani, An Introduction to the Quantitative, Rational and Scientific Process of Software Development (Part 1 and 2), SoMeT 7 pp. 361-371 and 372-390, 2007.