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～

○河野善彌(現：Creation　Project)，　陳慧(現：国士舘大学)
高野英樹(現：日立製作所),森本祥一(現：産業技術大学院大学)

2007年　9月　6日
Koono@vesta.ocn.ne.jp

この発表は第26回ソフトウエア品質シンポジウムで経験発表として報告しました．発表報文集には概要A4　1枚と３枚に18slideを収めてます．これでは，詳細な内容が分かりません．

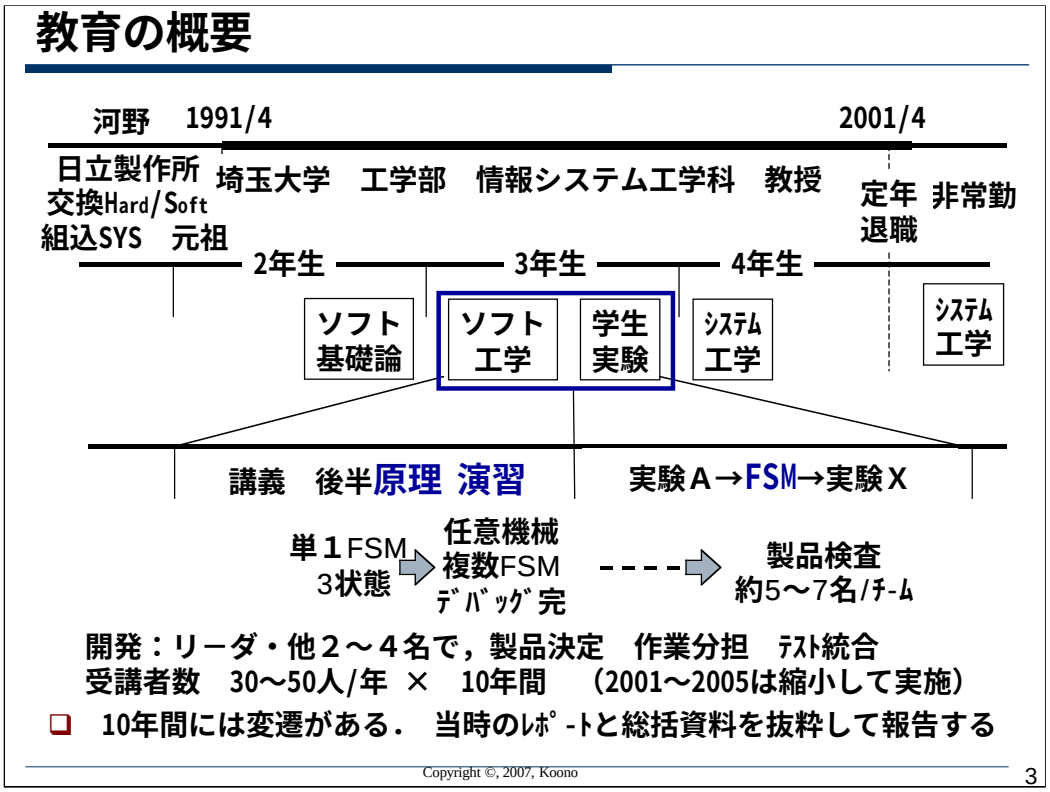
この資料は発表に用いたslideを使い，シンポジウムではお話できなかった詳細な説明も含め記しました．

教育のねらい　(人間育成から技術習得まで)

- 人間育成　　実作業の疑似経験
  - 触れ合わせ，協力させ，組織として働かせる
  - 役割を果たさせる　ﾁｰﾑ　ﾘｰﾀﾞ/OS担当/ ﾒﾝﾊﾞ
- 技術習得　　作業し実感を持って習得
  - チームを構成して分担/共同作業
  - ソフト作業法　図面/文書の意義～作成～ﾃﾞﾊﾞｯｸﾞ
  - 状態を持つ系を体験　(Finite State Machine, FSM)
    組合論理と順序論理とは作り方が違う

この教育は「ソフトウエア工学」の教科中の演習です．その狙いは，
第一に実作業の疑似経験を通じた人間育成です．
　チーム作業の場を作り，学生達に触合い協力する状況を作ります．
　メンバに役割を与えて，その役割を果たさせます．

第二は作業を通じての技術の習得です．
分担作業を通じて，各作業や文書の意義をわからせます．
次のポイントは，状態を持つ系を体験させることです．
論理には組合わせ論理と順序論理の２種があります．
　組合せ論理は，入力を論理的に変換した出力を出します．言換えると
　入力から出力への変換は論理で記述でき，組合せで実現できます．
　順序論理では入力から出力への変換関係が複数あり，固定できません．
　過去に与えた刺激の影響を受けるので，順序論理と名付けられて
　います．ほとんどのハードウエアでは，例えば”On”，“OFF”のよう
　な複数の状態を持っており，これをシーケンス性といいます．
このように両者の基本的な性質は違いますから，作り方も違います．
現在のソフトウエア工学では，順序論理について何も教育していません．
この演習は，その欠落部分の教育でもあります．



時間経過でご説明します．

私は1991年４月，埼玉大学情報システム工学科の教授に着任しました．その前は日立製作所通信機事業部戸塚工場で，交換ｼｽﾃﾑのR&Dに従事しました．電話交換システムは，最も早くにオンライン化された状態論理のシステムで，組込システムの元祖と言えます．私の技術的対象は，交換制御用コンピュータ（ハード）から交換システム（ソフトウエア）です．

1991年から2001年定年退官するまで10年間，それ以後は非常勤講師としてスケ-ルダウンして５年間，この教育を演習として行ないました．丸めて10年と銘打ち，そのエッセンスをお話します．

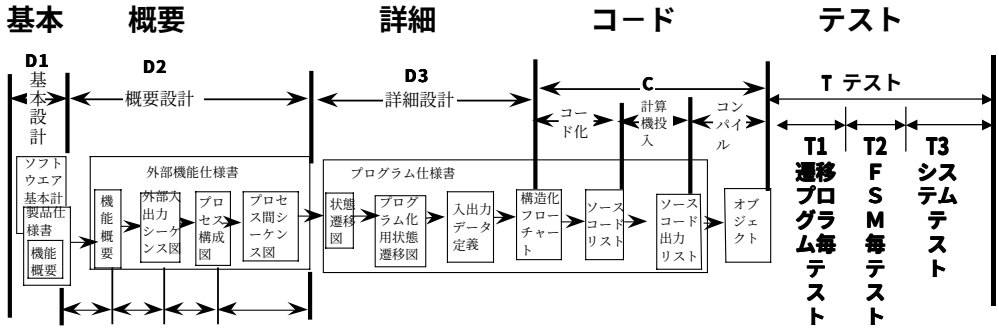
情報システム工学科学生の２年生後期には「ｿﾌﾄｳｴｱ基礎論」，次いで３年生前期で「ソフトウエア工学」を教えます．この教科で，演習の為に状態論理の原理と関連技術を教えます．対象は３状態で表した簡単な自動販売機／有限状態抽象機械/FSMであり，それを実際に動作させます．

演習は３～５名のチ-ムを作り，演習した自動販売のFSMを基礎にして，チーム毎に任意の自動販売機類を作り，テストして結果を発表します．

後期には必修の学生実験があり，約5～７名位のチ-ムで２週間毎に１テ-マの実験を行います．各教官の実験を回ります．私の実験では前期の演習で開発された自販機について「検査」を行います．

４年生は「システム工学」の教育です．

簡単な自動販売機　開発演習の工程



- チームリーダ（社長：作業禁止）,OS担当,各ﾒﾝﾊﾞ
- 勝てる製品・仕様～システムテスト完迄（実験で検査)
- 原形を販売機能の中心として，２～３FSM追加で実現
- 小さな進行毎の図面/文書を作成して入念にﾁｪｯｸ ZD

開発工程はこの図のとおりです．概要設計，詳細設計，コード化があり，テストに入ります．テストは状態遷移のルート毎の単体テスト，次にFSM毎に行うテストがあり，最後に全体で動作させるシステムテストです．

各ドキュメントについては後でお話しします．

各チームのメンバには役割が決めてあります．

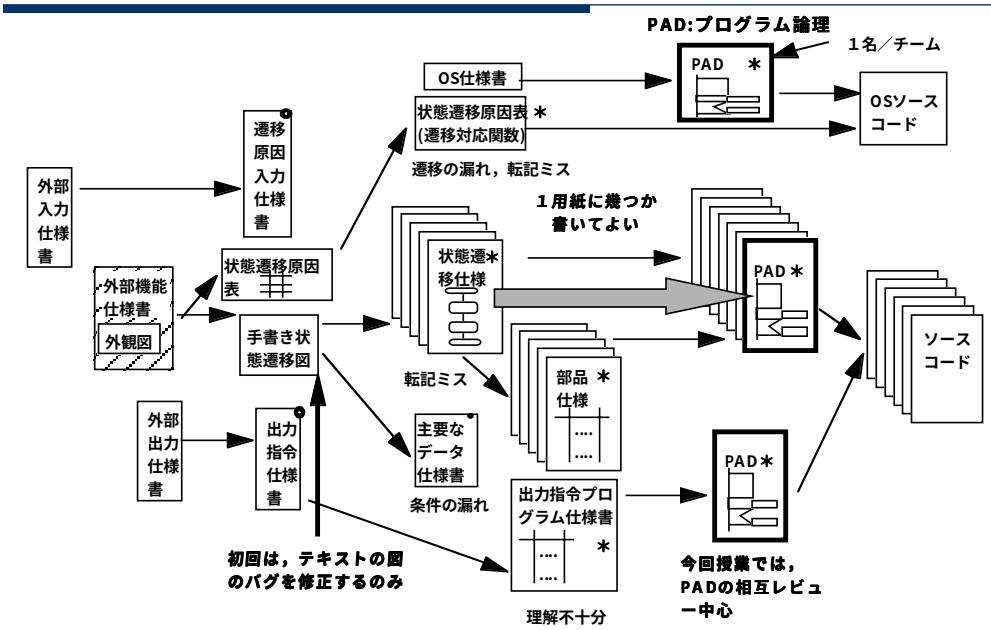
チームのリーダは「社長」と定義され，全メンバを率いて仕事をして貰うことが仕事です．従って設計等の実作業は原則として，やってはならない，と言い渡します．

チームリーダは，チーム全員の知恵を結集して「他社に勝てる(市場競争力のある)製品」の仕様を決定します．開発作業をして最後にはシステムテスト迄行います．

ここでは，設計文書を作ることとチェックが大きなポイントです．

小さな進行ステップ毎に文書を作り，厳密にチェックします．

簡単な自動販売機　標準文書体系



- 文書/図面には，一つずつに夫々の顔がある

5

この図は，設計ﾄﾞｷｭﾒﾝﾄの推移を示します．

左端の外部的なレベルからソフトウエアレベルに入ります．

　外部的レベルでは，10円＝直径　24　㎜，　厚さ＝1.5　㎜などと，物理的であり，

　内部に入ると　　10円の投入　　遷移原因：お金投入
　　伴う情報：　１０円

のように変わっていきます．

　中央には状態遷移図があり，これは振舞いを規定します．この中はフローチャートと同様ですが，概念レベルから具体的なレベルに移って行きます．

　これらを経て，最後には構造化チャートで詳細化してソースコードになり，計算機を動かします．大きな目で見ると，抽象的から具体的，概念的から実際的になり，最後には計算機での加減乗除などに変わって行きます．

簡単な自動販売機　指導文書類

文書構成

２．ソフトウエア関係文書の構成
　0．ソフトウエア基本計画書
　１．製品仕様書
　２．外部機能仕様書
　　外部機能及び内部の機能階層図，
　　データフロー図，シーケンス図，
　３．内部機能仕様書
　　状態（遷移）図，データ仕様書
　４．プログラム設計書
　　データ仕様，構造化チャート，
　５．インタフェイス仕様書
　　制御構造／機能部分間等
　６．試験ツール仕様書
　７．テスト設計書
　８．作業工程図
　９．進捗管理表

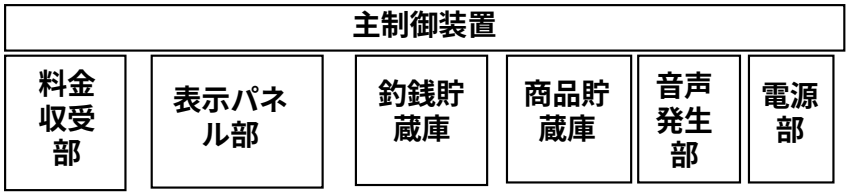
企業の中の開発作業は，企業幹部から多くの関係部署にかかわり，また，工程毎に各種のドキュメントが使われます．しかし，学生諸君に此処の詳細を教えても理解を越えるでしょう．主なものを捕らえ，流れを理解させます．また，演習に先立って各種の環境条件を理解させます．

これは，システム／ソフトウエアの文書類の項目です．

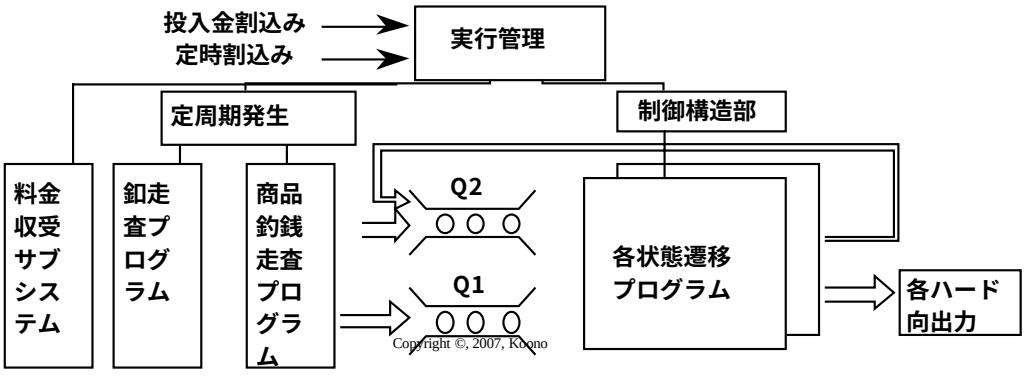
（学会発表では数行読んで次へ行きますが）先の設計文書の流れと併せて，全体の中での各文書の目的と概要を説明します．

簡単な自動販売機　指導文書類

ハード構造



ソフト構造



（学会発表では数秒間で飛ばしました）

これは，システム／ソフトウエアの文書類の項目です．

上の図はハ-ド構成を示し，下の図はソフトの機能構成です．

外部レベルから始めます．

料金収受があり，

その入力(I)　は「投入金」で紙幣と硬貨であり，

その出力(O)　は「返戻金」で紙幣と硬貨である

料金収受の為に，

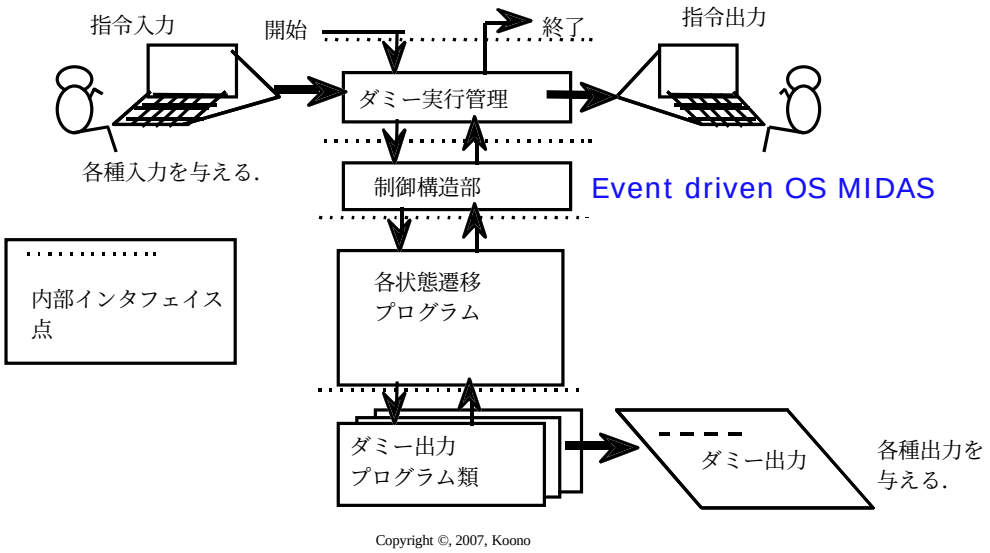
定時刻（数　ms)毎に定時割込が入って釦走査プログラムが全釦の0/1を並列に調べる．過去の走査結果と照合すると，「お金投入(10円)」のメッセージ（遷移原因）が得られる．

これは○印の待ち合せ行列につながれる．

実行管理はある状態遷移プログラムが終わる度に，待ち合せを調べて○があれば取出し，状態遷移プログラムを実行させる．場合によりこの中で外部ハードに対する出力等が出ます．

簡単な自動販売機　指導文書類

試験用構成



この図は教育に用いる単一FSMと演習する複数FSMの自動販売機のデバッグとテストの仕掛けです．

中央の大きなブロックは状態遷移図です．

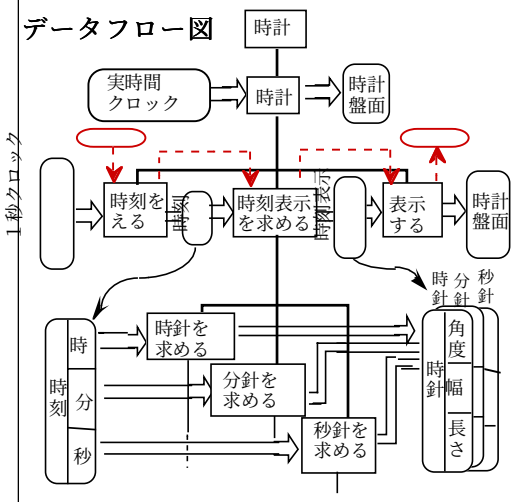
その上の制御構造部とあるのは，イベントドリブンOSである　MIDASです．

MIDASはギリシア・ローマ神話のマイダス王です．彼は触るものを金に変える力を持っていました．このOSを使うと，状態を持つ系が簡単に作れます．そこでMidasに因んだ名を与えました．

これは河野が電子交換機の自動機能試験機の中核として1985年頃に設計したもので，最高3万2千の並行プロセス/FSMの状態を制御します．現在のものは，サブセットでコンパイラ言語でできています．

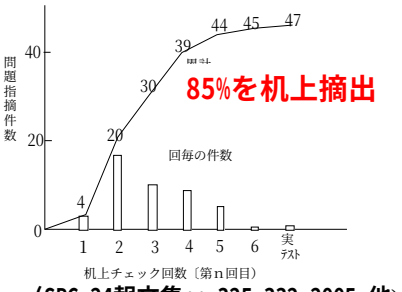
最上部の実行管理は試験の実行管理を司り，更に上部のUNIX，（後にはWindowsに交代）につながります．

基礎技術１　(2年 ソフト基礎論で教育)



- 設計: 概念を表す単位DFDを階層展開して詳細化DFDにする．これは子単位DFDの群である．
- 机上チェック： 小さな進行毎に厳密にチェック

（全体をM区分すると残留誤りは1/Mに低下する）



(SPC-24報文集pp.325-332,2005.他)

9

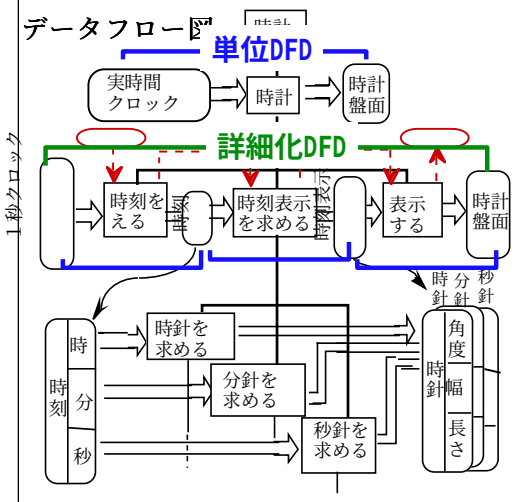
これから，設計の基本となる幾つかの基礎技術を説明します．

初めは設計の基礎です．左のデータフロー図の最上部には，仕様である「時計」があります．これに入力と出力のデータを加え，全体を「単位データフロー(DFD)」にします．

この単位DFDを親として階層展開すると子である詳細化DFDになります．この詳細化DFDをよく見ると，下位の単位DFD群で構成されています．

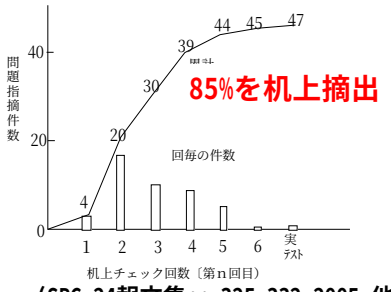
以下，次頁へ．

基礎技術１　(2年 ソフト基礎論で教育)



- 設計: 概念を表す単位DFDを階層展開して詳細化DFDにする．これは子単位DFDの群である．
- 机上チェック： 小さな進行毎に厳密にチェック

  (全体をM区分すると残留誤りは1/Mに低下する)



(SPC-24報文集pp.325-332,2005.他)

10

前頁より

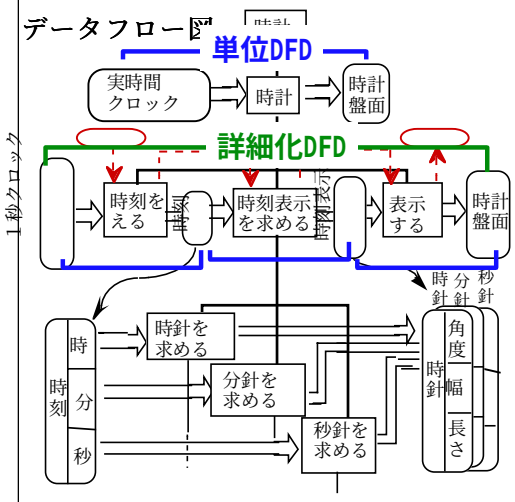
これら単位DFD群は，親である単位DFDの子であります．詳細化DFDは子である単位DFDをつなぎ合わせたものです．親である単位DFDは親の概念を表し，子であるDFD群は，親概念を展開した子概念群です．

赤破線で小さなフローチャーが示してあります．これは詳細化データフローを構成するデータの流れが決まった後に決めた制御の流れです．プログラムを意識する段階では，データフローと制御のフローの両者で厳密に規定できます．

上のDFD群を上から下に追って見てください．このように設計とは，単位DFDで表した概念の階層展開を繰返し,展開の度に明確化，具体化，詳細化して行きます．この繰り返しで，概念を微細化し，各機能が単位的な演算で代替できる時，機能を実現手段であるプログラム言語を使って表記します．

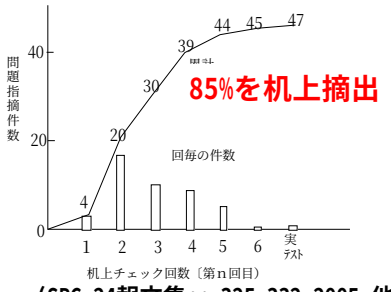
以下次ページへ．

基礎技術１　(2年 ソフト基礎論で教育)



- 設計: 概念を表す単位DFDを階層展開して詳細化DFDにする．これは子単位DFDの群である．
- 机上チェック： 小さな進行毎に厳密にチェック

（全体をM区分すると残留誤りは1/Mに低下する）



(SPC-24報文集pp.325-332,2005.他)

11

前頁より．

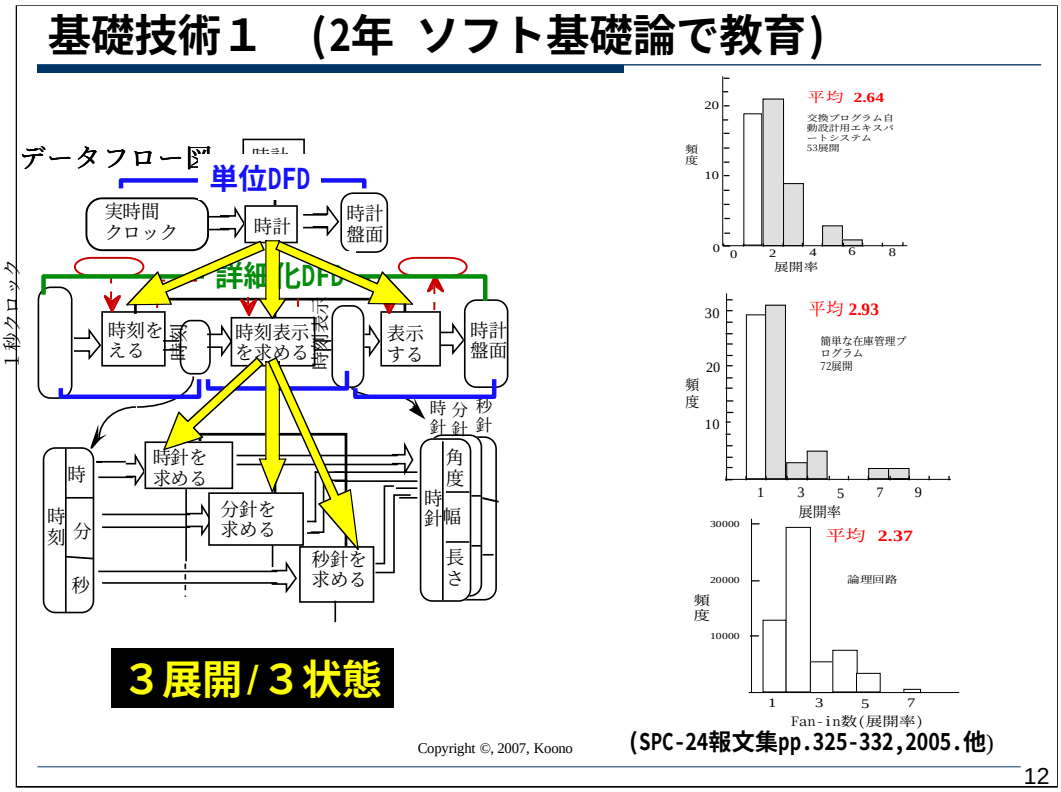
各展開をする度に，厳密に概念として正しく展開しているか，チェックします．あるプログラムを作った後でチェックして正しさをチェックすることは容易ではありません．しかし，その設計の中間の各過程でチェックするなら，チェックは容易になります．ここで説明した単位的なデータフローの階層展開毎であれば，最も容易にチェックできます．

ｓlideの右下のグラフを見てください．これは200行弱の小さなプログラムを念入りに各展開毎に繰り返しチェックした記録です．データフロー展開しつつチェックして，原図を訂正したら，また念の為の再チェックをして確認しながら先に進むことを繰返します．ｋの時には，最後にバグは無くなったと思ってコード化してテストをしたら，残念ながら２件のコーディングミスがあり，残念ながら作込欠陥率では95%の事前摘出になりました．

従って，**小さなステップ毎の階層展開**の度に，まず**正確に表現し記録**することが重要です．そして，展開毎に**厳密に確認して誤りを摘出**します．これにより品質が向上します．これは皆さんが中学や高校で数学や物理の試験の答案で**「小さなステップ毎にキチンと書け，念入りにチェックせよ」**と教育されたことと同じです．

ハードの製造でも同じことで，小さなステップ毎にいちいちチェックすることで，製造不良の少ない製品ができます．問題を見たらすぐさまコードを書く，しかもチェックは必ずしもせず，テストで誤りを取る積りである．そんな根性で誤りの少ない信頼度の高い製品ができるはずがありません．

以下次頁へ．



前頁より．

図を見ましょう．最初の親である時計の単位DFDは，下の子である単位DFDに3展開しています．真中の子である時刻表示を求める単位DFDの下の3単位DFDに階層展開されています．その次も同様に3展開されています．

機能だけではありません．データも同様です．時刻は時刻の時，分，秒に3展開されています．時刻表示を見ると，省略された形ですが，時針，分針，秒針に3展開されています．各針は，更に3展開されて（針の）角度，太さ，長さになります．

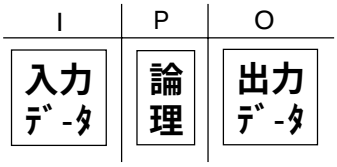
このように単位的な段階毎に展開すると，展開率は約３弱です．Slideの右の上２例はソフトの実績例です．平均は約3弱です．色々と調べると，理論的には多分e=2.7182..と思われます．この図は組合せ論理の場合です．展開の上位でも下位でも約3展開ですから，更に下ると同傾向が続きます．右の最下部はハード論理設計の展開で，1論理ゲートの入力数（fan-inと云います）も同傾向です．

この例は，組合わせ論理の場合でした．組込システムの場合には，一つの単位となる抽象機械/Finite State Machine/FSMは，単位的階層展開すると，平均３個のFSMの群に階層展開します．また一つのＦSMの内部を展開すると，平均3の状態群が現れます．これは後に実例をお見せします．

これらは意味の領域で考えると，親概念から子概念群への階層展開ですから，同傾向が現れるのは，（ヒトの脳の中では全て同様に見えているから）約3展開になるのはあたり前なのでしょう．

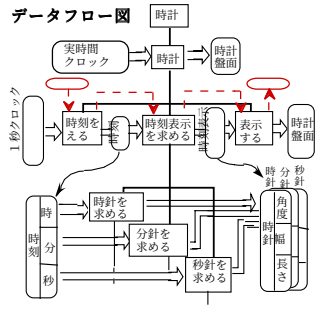
基礎技術２　ｲﾍﾞﾝﾄ駆動OS (MIDAS)

ソフトの仕様

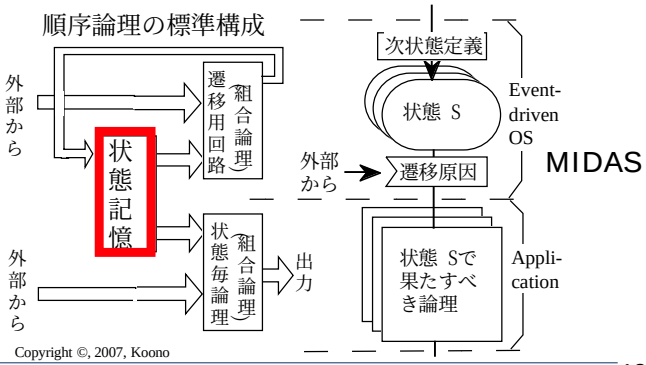


- 組込システムは通常の組合論理と異なる順序論理である
- 状態記憶と状態毎の組合論理で表す
- Event-driven OSで状態毎・原因毎のﾌﾟﾛｸﾞﾗﾑを起動させれば良い

順序論理でない
組合論理の場合



組合論理でない順序論理の場合



13

次は第２の基礎技術です．

初めに記したように，組込システムは通常の組合せ論理とは異なる順序論理です．

slideの右の図は順序論理の構成を示す図です．順序論理はシーケンス性とも云います．ハードウエア論理設計では，早い段階でシーケンス回路の教育をします．

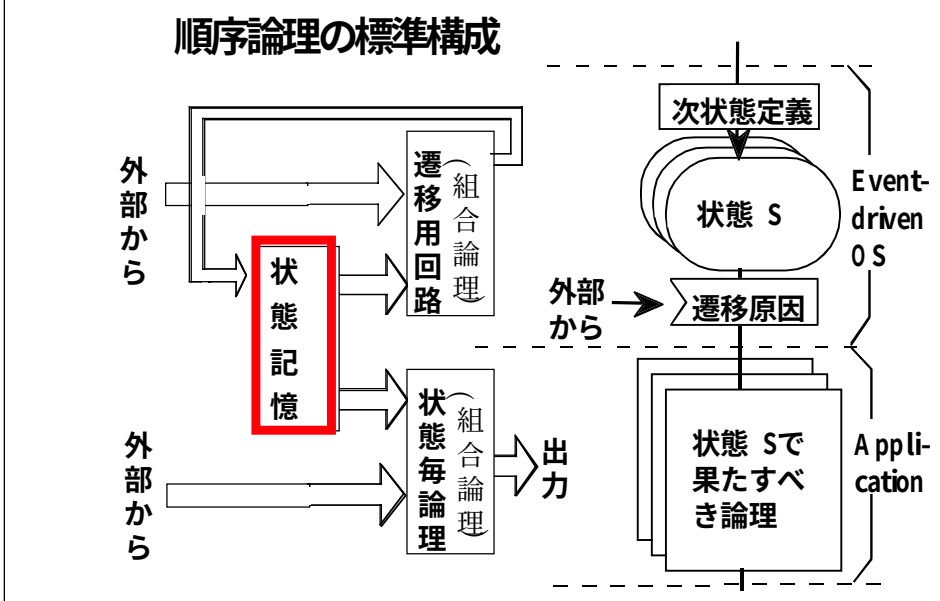
それはシーケンス回路は赤枠で囲った状態の記憶と組合せ論理で表記されます．

シーケンス回路は必ず状態記憶を使え！従わないと，回路がゴタゴタしてコスト高で，しかも完全な回路にならない．必ず状態記憶を使え

と教育します．順序論理は通常の組合せ論理と同様には扱ってはなりません．

以下次頁へ．

基礎技術２　ｲﾍﾞﾝﾄ駆動OS (MIDAS)



順序論理では，赤線で囲った状態を表す記憶が必要です．

左上の論理構成部分は現在の状態に新しく外部から入力があって，状態を更新する機能部分で組合せ論理です．

Slideの右の図は，これに対応するソフトの表記手段です．現在の状態Ｓで，外部からある遷移原因/イベントが到来すると，この状態でこのイベントに対応する状態遷移ルートが選ばれます．

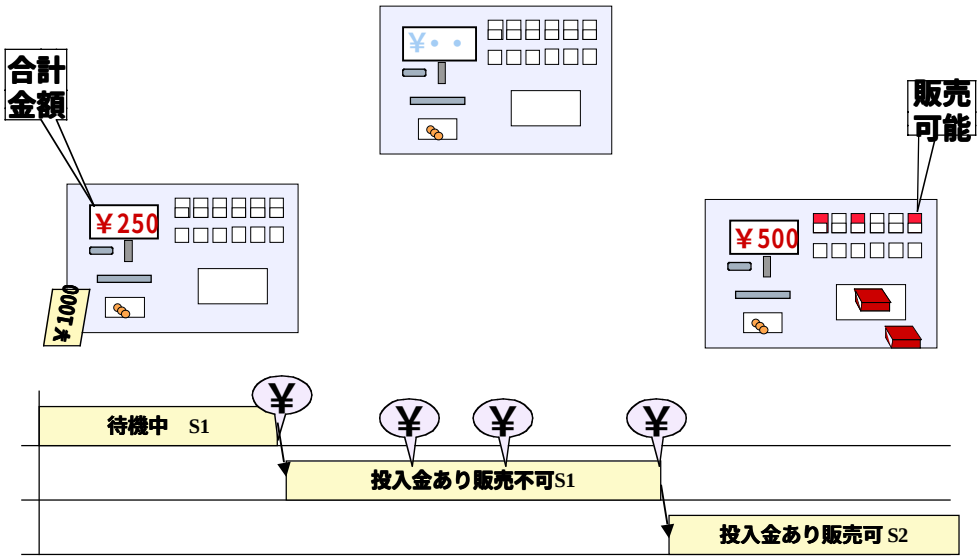
左下の論理構成部分は，状態毎に入力から出力への変換を示す部分です．右下のソフトの表記では，外部からきた入力を如何に処理するか，を記録します．ここではこの状態でこのイベントに対応する論理を記せば良い訳です．

このように，シーケンス性のシステムでは状態遷移図が仕様を表記します．状態遷移図の上部の部分がイベントドリブンOSに相当し，下のフローチャート的な部分が，ある状態であるイベントが来たときの状態遷移のルートを示します．

簡単な自動販売機　有限状態機械ﾓﾃﾞﾙ　SDL準拠

- 状態の概念の教育

国連下部機構　ITU制定
国際標準　仕様記述言語



15

このスライドで，自動販売機を取り上げて，その状態遷移を説明します．空色の箱は自動販売機です．実際の自動販売機を観察して特徴的な契機とその状態を求めます．

最上部は，お客様が現れる前の「待機中」状態の自動販売機です．最下部のタイムチャートでは，信号は定常的に”1”です．ある状態が継続します．

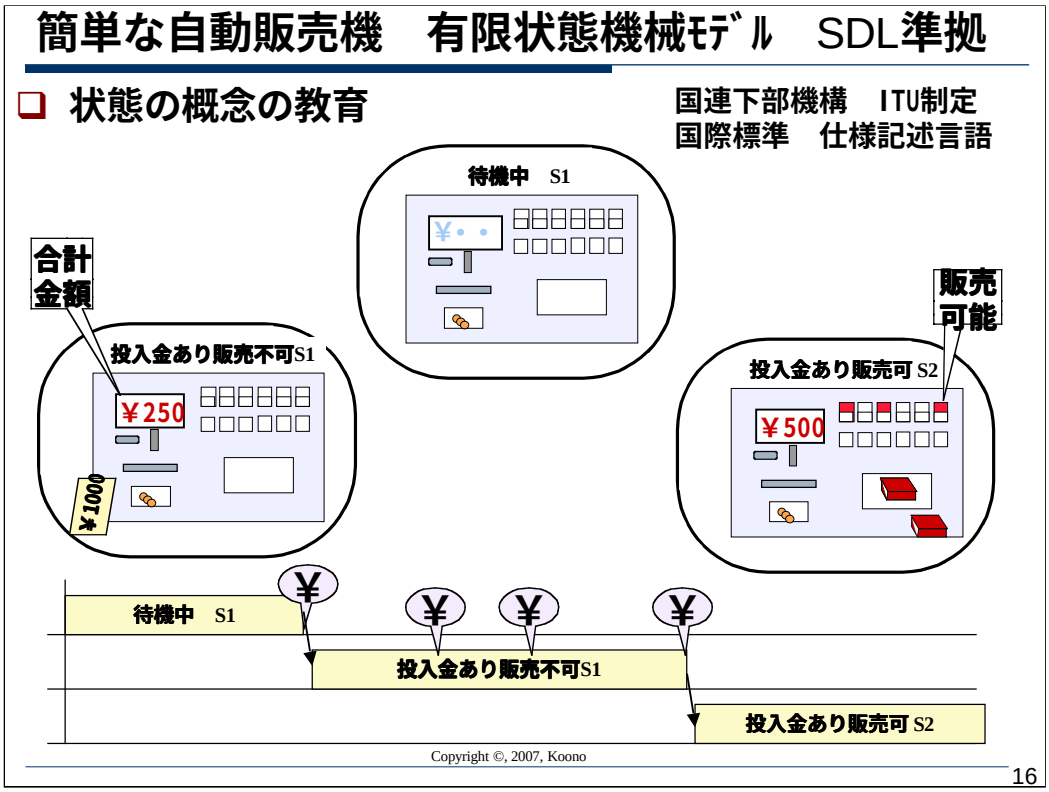
下の左は，お金を初めて投入して，左の金額表示が動作し始めた所です．最下部のタイムチャートでは今までの状態に対応した信号は”0”であり，新しい状態に対応するタイムチャートが立ち上がります．続けてお金を投入しても，金額の値が変わるのみです．これに対応するタイムチャートは”1”を表示し続けます．これは「投入金あり，販売不可」な状態です．

続けてお金を投入し続ける内に，突如状態が変わり，販売可能な商品を示す表示が現れます．下の右がこれで「投入金あり，販売可能」な状態です．

ここで販売可能な商品を指定すれば，商品が転がり出て，自動販売機は当初の「待機中」に戻ります．

自動販売機を観察して，最も特徴的な変化/遷移原因と状態をとると，この自動販売機は３状態の有限状態機械でモデル化できます．

以下次ページへ．

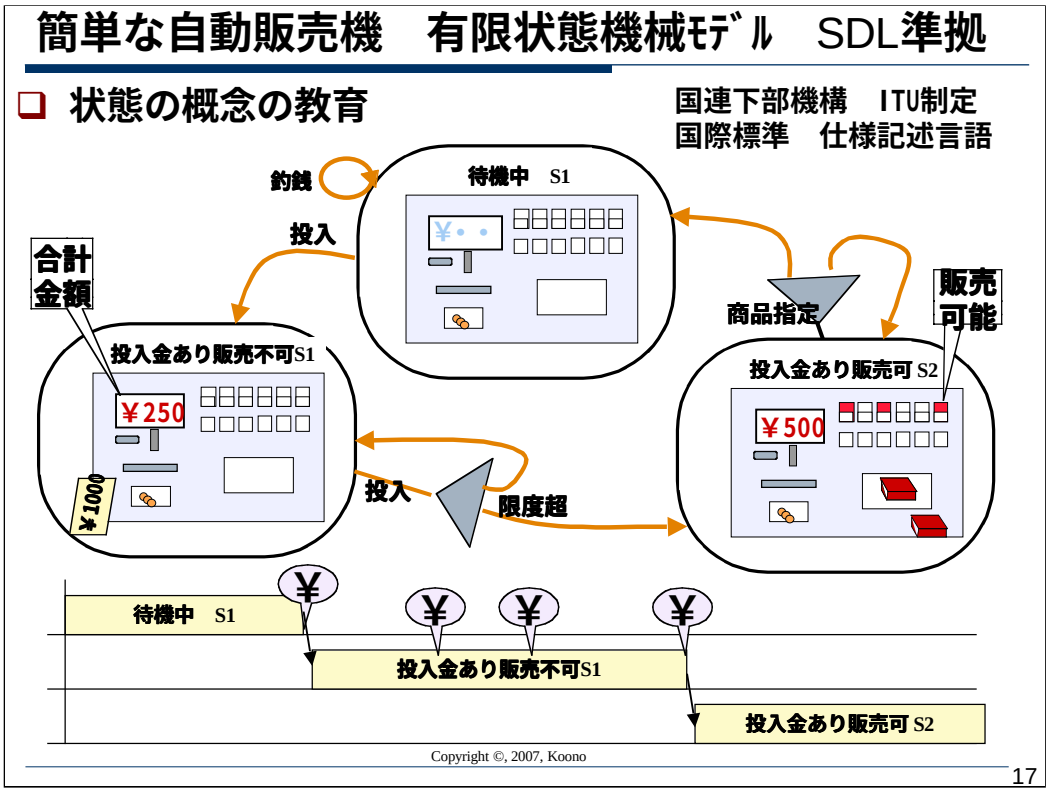


前頁から．

３状態の有限状態機械でモデル化することを図式的に表示するために，状態を示すシンボルとしてビヤ樽状の図形記号を用います．ここでは各状態毎の番号を付けました．「待機中」：S1,　「投入金あり販売不可」：S1,　投入金あり販売可能」：Ｓ2．

この分野の理論はオートマトン理論ですが，そこではビヤ樽状の記号が使われます．したがって極力ビヤ樽状の記号を用いることで，この分野の先達に敬意を表することが後進である我々の当然の責務と，私は思います．

次頁へ．



前頁より．

ある契機で状態から状態に移り変わる契機を状態遷移原因と呼びます．これはタイムチャートで，ある状態から他の状態に移る原因です．

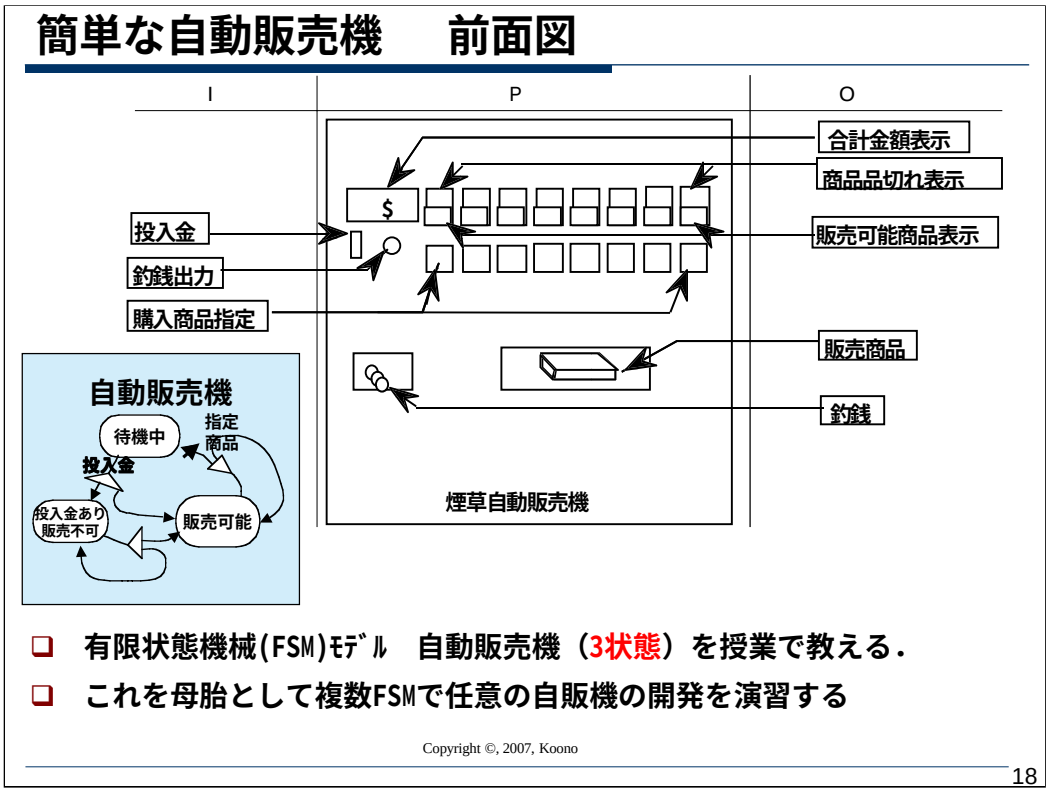
ある状態である遷移原因○○で始まって他の状態に移行するルートを状態遷移（ルート）と云います．この中で条件により分岐することができます．slideでは△印で示してあります．(中には，ある状態から出発してまた元の状態に戻る場合もあります．)同様に流れの中に機能（ある入力に何かの処理をする）を挿入することもできます．これはフローチャートと同じです．

このように状態から他の状態への遷移を全て表記することができます．状態，状態遷移原因，分岐や機能を使えば，この系の仕様を完全に表記できます．

交換機/交換システムはこの方法で仕様を表記できます．国連下部機関の電気通信連合(ITU)では，状態記号の中にシステムの状態を表記するシンボルを記入した図式で仕様記述方式を標準化しました．この時期には状態表記を含めてLOTOS, ESTELLEなど数種の仕様記述言語が標準化されました．これらの中で交換系のみ使用者が居るので永続しました．1980年代末には図式表記/テキスト表記の両者が表記できるようになり，SDL (Specification Decription Language)として標準化されました．この最終版はUMLにも移植され，UML2.0の中心になりました．

このような仕様を表記する図面は，この振舞いをする系，ある抽象機械(有限状態機械，Finite State Machine, FSM)の仕様を表します．この遷移ルートをプログラム化して，イベント駆動すれば，このFSMの仕様に合致したプログラムが得られます．

以上で，第２の基礎技術を説明を終えます．



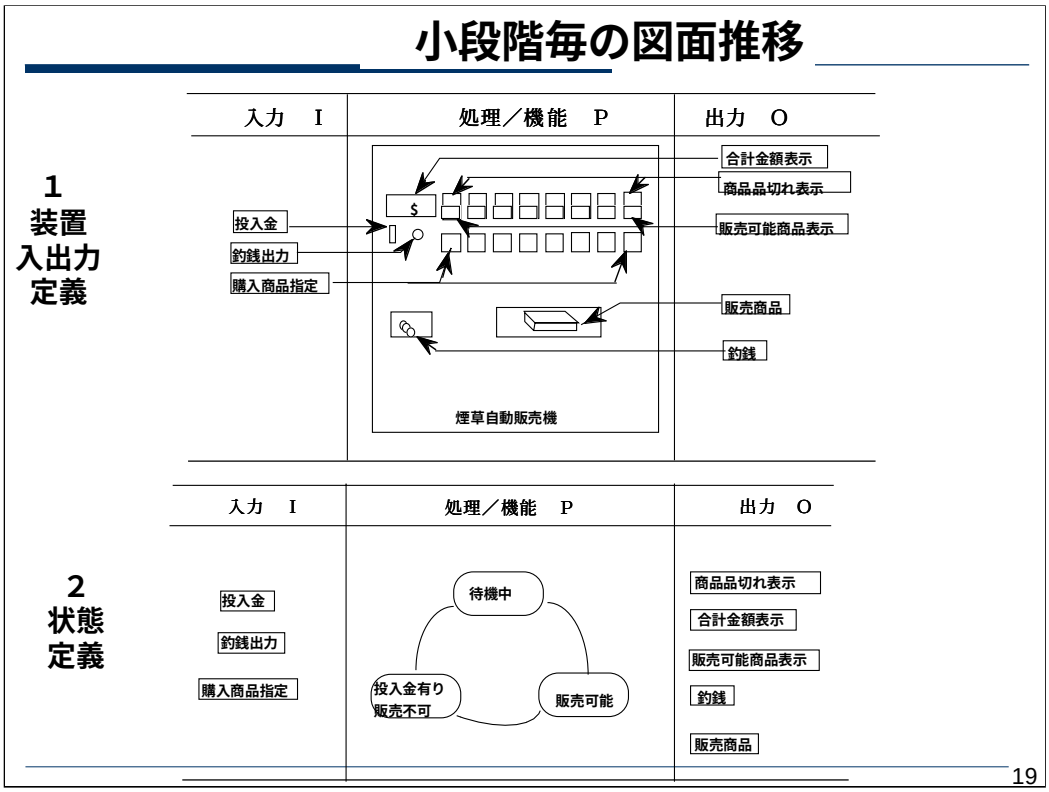
これから簡単な自動販売機の設計過程を説明します．左下はその状態遷移図です．

slideの中央の図枠はIBMが創始したIPO図の様式です．これは左に入力 Input I，中央に機能/処理 Process P，右に出力Output Oを記します．略してこれはIPO図と呼びます．

以後，IPO図の様式を使い，データフローを中心として，時にはフローチャートを併用して，説明します．ソフトは入力から出力への情報変換の機構です．また，計算機の命令やプログラム言語も入力から出力への変換です．**最も基本的な図面はデータフロー図**です．

IPO図の中のP欄に自動販売機を書き，左のI欄に入力を列挙し，右のO欄に出力を列挙しています．（なぜ自販機の略図を書くかと云うと，常に入力と出力を意識させ，以後の設計で参照する為です．）

授業では，以下３状態の簡単な自動販売機の設計を追い，実際に動作させ，次に複数FSMの自販機類を設計します．

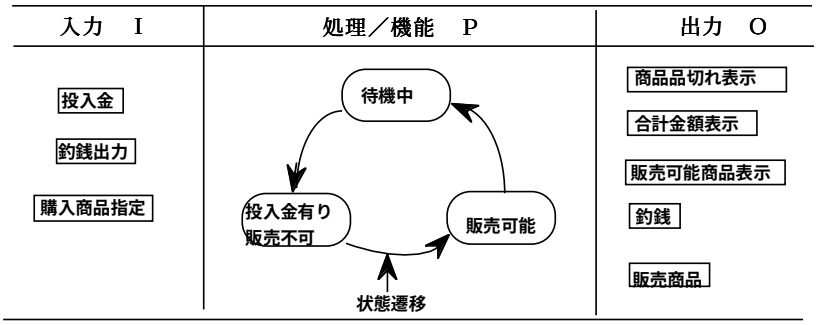


上の図は装置と入出力を定義するもので．これを最終目的として，小さな段階毎に詳細化のための意思決定をして図面の上に記録し，段階毎に厳密にチェックして行きます．

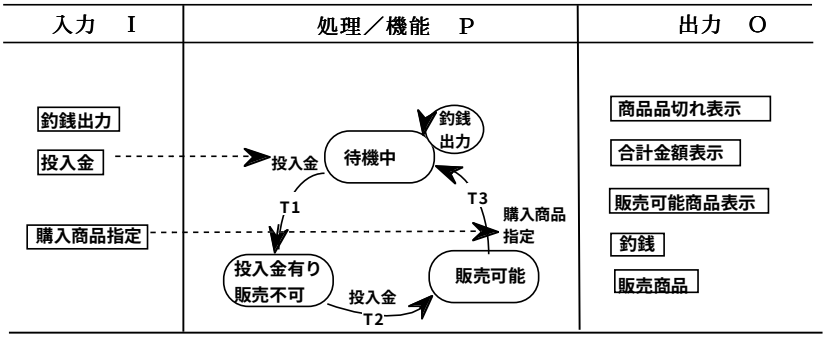
下の図は，次のステップとして先に求めた３状態を示します．

小段階毎の図面推移

3 状態遷移



4 状態遷移原因



上の図は，ある状態から他の状態への遷移を示します．

下の図です．ある状態である外部事象を契機としてある状態遷移が起こります．この原因となる事象を状態遷移原因，あるいは遷移原因，イベントと呼びます．遷移原因は遷移ルートの始点近くに記しました．

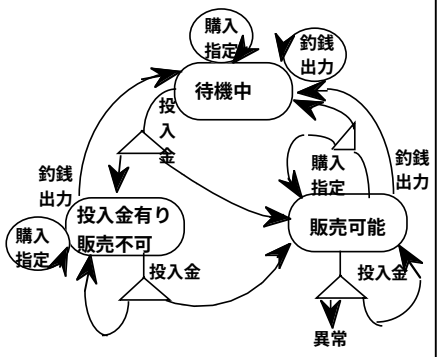
ここで考えられる遷移原因をすべて洗い出します．それは実物で考えた入力と対応せねばなりません．システムテストをしたら「返戻のレバーが抜けた！」という滑稽な失敗があって，以後は必ず実物の概略図から出発することにしました．

小段階毎の図面推移

状態遷移原因表

状態 / 遷移原因		S0 S	Si	
	Evj		Tij	
	Evn			

状態遷移原因表の欄には遷移ﾙｰﾄ名称/ﾌﾟﾛｸﾞﾗﾑ名称を書く．
終着状態は原理的に複数ある．
終着状態を1に限る方式は不可．

入力　I	処理／機能　P	出力　O
5 全ての 状態遷移 釣銭出力 投入金 購入商品指定		商品品切れ表示 合計金額表示 販売可能商品表示 釣銭 販売商品

全状態と全遷移原因を決定しました．次に全状態遷移ルートを決定します．

左上の表は状態遷移原因表といいます．これは全状態と全遷移原因を２次元の表に並べたものです．表の各欄には，表のある状態とある遷移原因の交点の欄には，（赤字のTが見えています）遷移ルートの名称（当該遷移ルートのプログラム名称）を書きます．

状態毎に，一つづつ各遷移原因について，何が起こるか考えます．ある遷移ルートに考えたら図面にルートを書きます．何も無い時には図のように自状態から出て自状態に戻るルートをかきます．遷移する途中で分岐を考え，必要の都度分岐を設け，対応する枝を書きます．仕様記述ですから，如何なる条件（販売可能など）で分岐するのかを明確化し，その実現（>￥120なら販売可能など）には触れません．

（例：左下の遷移ルートは投入金が大きい時右に分岐し，それ以外は元状態に戻ります．）

１遷移ルートの中で分岐が生じると，多くの場合に複数の終着状態が生じます．表の交点は（状態ではなく）遷移ルートの名称を書きます．

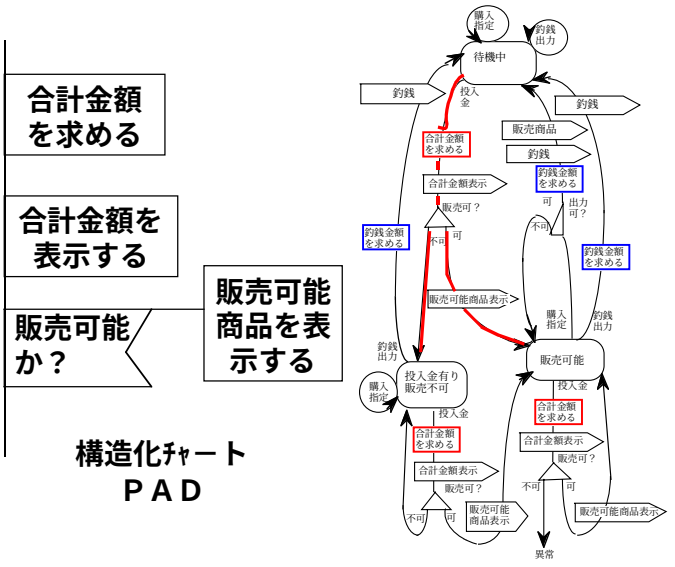
遷移ルートは最終的に，対応する遷移ルートプログラムになります．

状態遷移原因表の交点欄に終着状態を記入する方式があります．一般に１ルート当たり約60%位分岐の必要が起こります．従って，この方式は実用には不適です．

また，遷移ルートは中断や中間停止無しで頭から終わり迄，走り続けます．（中断や中間停止はバグの原因になります．）プログラムが並行して走る為のプログラミング的な手法は無視します．良い状態と状態遷移を考える人にはプログラム的な能力は必ずしも必要ありません．

簡単な自動販売機の状態遷移図

- ３状態なら遷移ルートは，最大3×2　=6ﾙｰﾄでかんたんに設計できる



６
遷移ルート中に出力を挿入する
７
必要なタスクを挿入する
８
頭脳で全動作をシミュレーションする
９
ルート中の共通部品括りだし
１０
構造化チャートで詳細化しコード化する

22

3状態しかありませんから，本質的なルートは状態当たり他の２状態に行く2ルート×３状態で計6ルートです．実際は遷移原因対応ですからこれより増えますが，それにしてもごく少数です．ここでバグを全て机上チェックで摘出しましょう．以下，slideの右の手順に対応します．

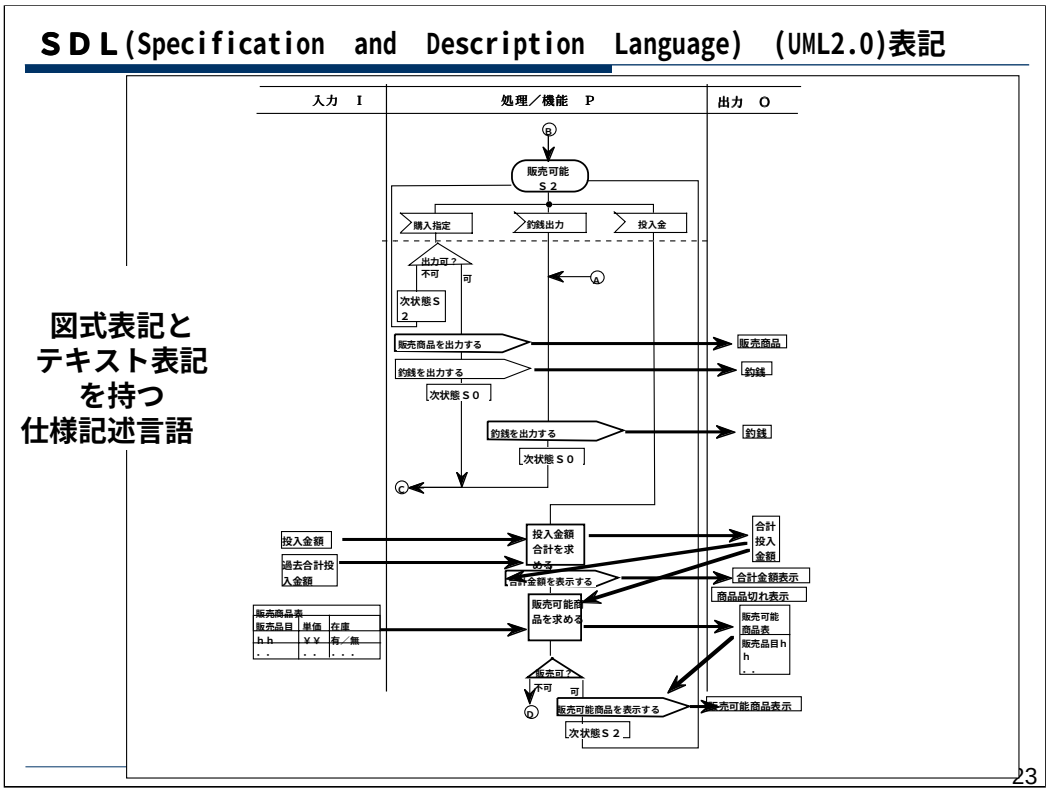
（６）先に決めた遷移ルートは単純な線のみですから，各遷移ルートに楔形の出力記号を挿入して行きます．

（７）各遷移ルートに機能箱とその記述（タスクと云います）を追加して，分岐，出力，機能が揃った正しい概要フローチャートになるようにします．（llideの左はPADと名付けられた構造化チャートです．たとえば合計金額を表示するには，先立って合計金額の計算が必要です．但し，詳細化は後で行うので，ある機能の存在だけを表示する文言に留めます．）

（８）この段階の状態遷移図は**必ずA4またはA3の１枚の図面**にします．このFSMについて，全動作を頭脳シミュレーションで追います．全ての状態で，順次全遷移原因が到来した場合を考えます．動作を一つずつ追いながら，図の赤印のように色鉛筆でルートを塗り潰して行きます．３状態程度ですから複雑でなく，また**図面は１枚で全体が見えます**から，容易に行えます．これ以後は全体は見えません．状態遷移原因表と遷移ルートと分割すればバグが見易くなります．**必ず全体を全て1枚で確認**します．

（９）次には共通部品化などを考え手直しをして，遷移ルート全体を一つのプログラムにします．C言語なら遷移ルートに対応する関数があり，この中での出力の関数や簡単でない機能の関数にとどめます．これが仕様であり，以後はこの仕様に対応する遷移ルート毎の詳細化に移ります．

（１０）次には遷移ルート毎の関数簡易に，構造化チャートに移して詳細化し最終的にはソースコード化します．



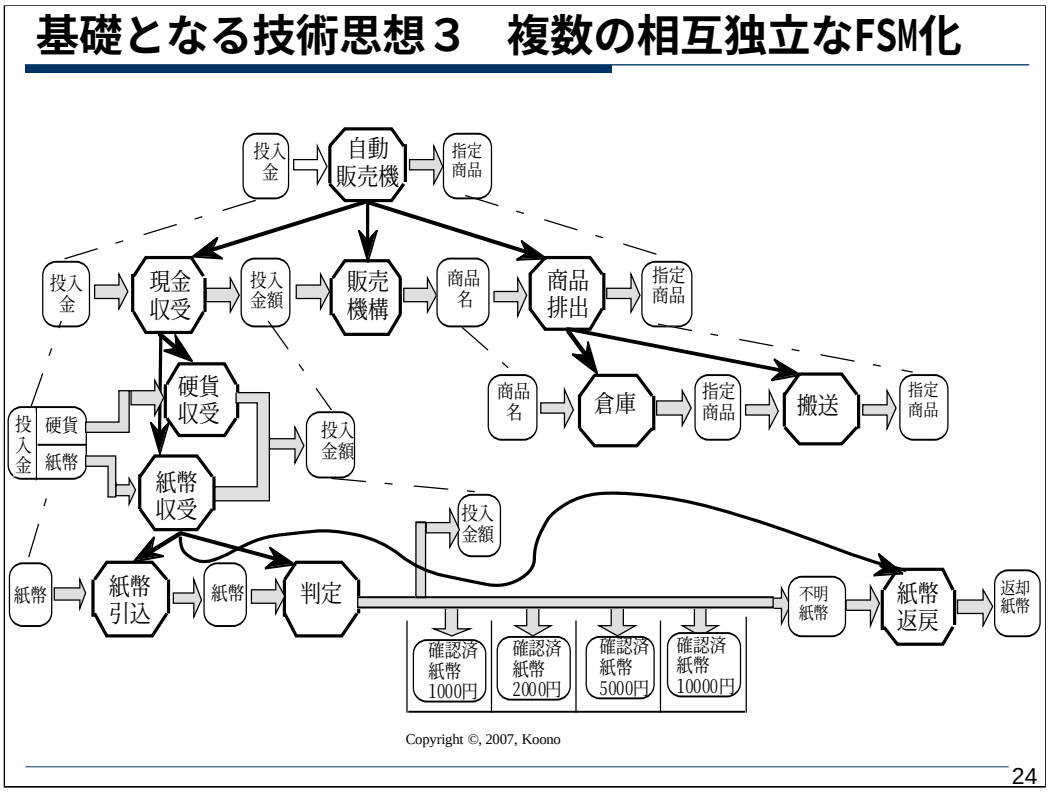
今お話している技術は，仕様記述言語SDLの技術に河野が開発した技術を加えたものです．これは図形表記とテキスト表記の両者が使えるものです．

このslideの図は，SDLの図式記号を使ったものです．ビヤ樽状の記号は状態，切込んだ楔状記号は遷移原因です．現在では，これらの図形記号のCASEツールが開発され市販されています．CASEツールを使用すると，3状態のFSMでも必ずしも全体を見通しよく書けません．そこで部分図しか書けず，もはやFSMの全体はみえません．

状態遷移単位に，フローに従いデータの入力と出力を確認しながら，再度動作を頭脳シミュレーションします．（学生に渡す教育用資料では，データが見えるようにしています．）

後は遷移ルート毎の関数を構造化チャートで詳細化し，コード化します．

このように，ある閉じた状態の群とその間の状態遷移ルートの群で，ある抽象的な機械が作られます．これを以下簡単の為にFSMと記します．
FSMとはFinite State Machineの略です．



学生は，これまでご説明した資料を勉強し，FSM単体の動作を確認した後，この自動販売FSMを基にして各種システムを開発します．構築は**３状態程度の小さなFSMを複数使用する**ことが条件です．

小さなFSMを用いる拡張を二方向で説明します．この図は，今までお話した自動販売機をさらに詳細化したものです．今までの自動販売FSMは，図の第２段の中央の販売機構に相当します．

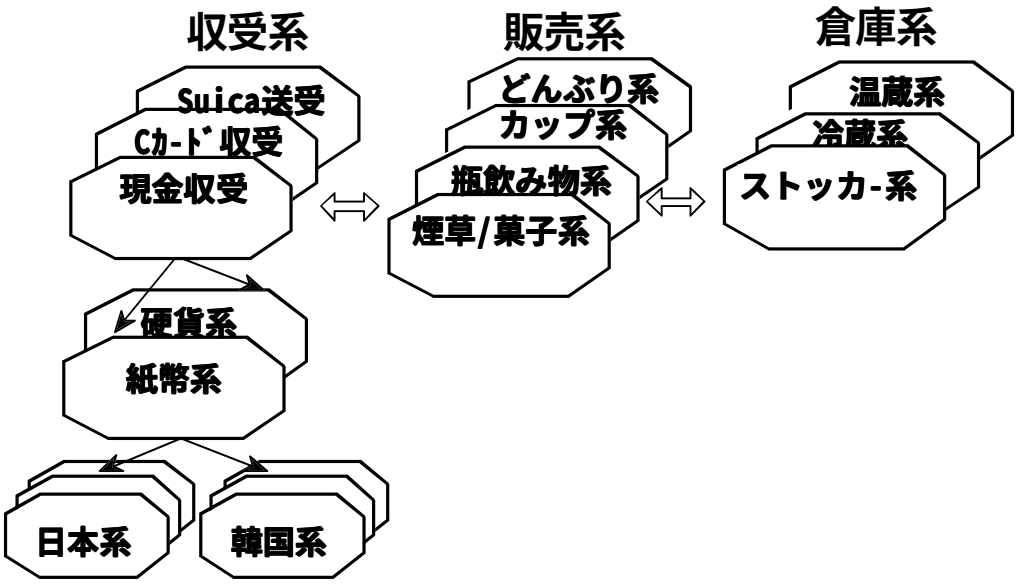
このFSMの入力側に現金収受FSMを設けてお金の真贋判定等を行わせます．現金収受は硬貨あるいは紙幣の収受を行うから２状態を持ちます．「紙幣収受FSM」は紙幣引込，真贋の判定および紙幣の返戻の３状態に展開されます．この構成をご覧になると，いずれも簡単な数状態のFSMでつくれることがご理解戴けるでしょう．更に「引込FSM」「真贋判定FSM」や紙幣返戻FSM」「紙幣収容FSM」を設ければ詳しい動作を実現できます．

第２段の販売機構に戻り，販売機構の出力側に「商品排出FSM」を設けます．商品排出も，商品を倉庫から出すこと，コンベヤに乗せて送る搬送の２状態を持ちます．倉庫は開閉中心の簡単なものですが，搬送はベルトコンベヤとかエレベータなどの各種のものがつくれます．

最上段の自動販売機は，お金が投入されると，料金収受，ついで販売機構さらに商品排出と３状態で構成できます．

このように，小さな数状態のFSMを用いて，自販機を幾らでも詳細化できます．此処のFSMは簡単ですから，容易につくれます．ここではプログラムを作ることでなく標準的なFSMを作ることが重要です．

基礎技術３　相互独立なFSM群で構成する



25

学生の演習では，夫々チーム毎に各種の自動販売機を作りました．視点を変え，ある会社で事業として自動販売機市場に進出する場合を考えます．

スライドの左端の収受系は，予め考えておけば，全自動販売機に共通なお金を受渡しするFSM群に統一できます．

スライドの右端の倉庫系は，販売する商品の形状等の差異があり，個別性が目立つが，単純なストッカー系，冷蔵系，温蔵系などの中で，各基本技術の共通化が計れます．

同様に，中央の販売系も，販売対象毎の差異があるが，標準化して統一部分の共通使用，基本形からの小幅手直しで各個別に対応できます．

ここで，小さな単位毎，即ち小粒度の系でつくることは，開発負担を大幅に減らせます．

このように，階層的に展開して各個に標準的な仕様を設定して製品化することはハード設計者にはあたり前のことで，何故Product lineを議論するのか不思議です．

工学は必ずビジネスを機軸にして展開します．技術が機軸ではありません．ビジネスの為には，市場や競争，顧客心理から顧客の取捨選択の論理をわきまえねばなりません．ある面では全てが競争であり，お金であり，全てがアナログ的な大小で弁別します．いわゆる技術はこの手段です．

図のように製品化して市場に進出するには，何枚もの先の宣伝パンフレット（事業戦略）が頭の中になければなりません．この戦略の為に，技術を駆使します．要求に応えて作る世界とは全く別な世界です．

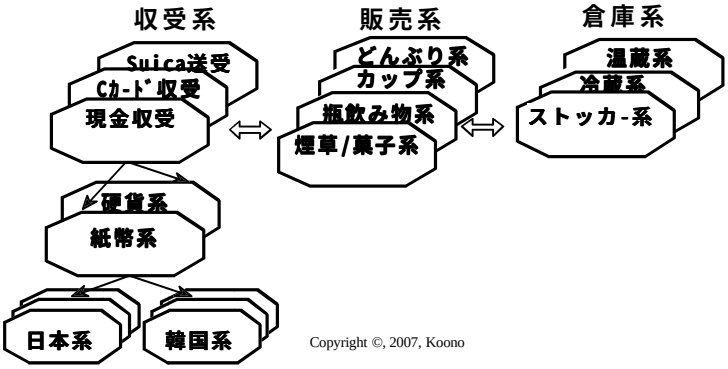
基礎技術３　相互独立なFSM群で構成する

- ﾌﾟﾛｾｽ　FSMx　FSMy　FSMz　等価なFSM

状態数	X	Y	Z	X・Y・Z
ﾌﾟﾛｸﾞﾗﾑ	(X + Y + Z)			(X・Y・Z)

ソフト規模を1/Nに減らす

3+3+3=9の負担　で　3x3x3=27



26

先程から，３状態位の小さなFSMを複数個使う方法を説明しました．これが第３の基礎技術です．構成については既に説明しましたから，以下この方式の持つ「ソフトウエアの規模を数分の１に減らす利点」を説明します．

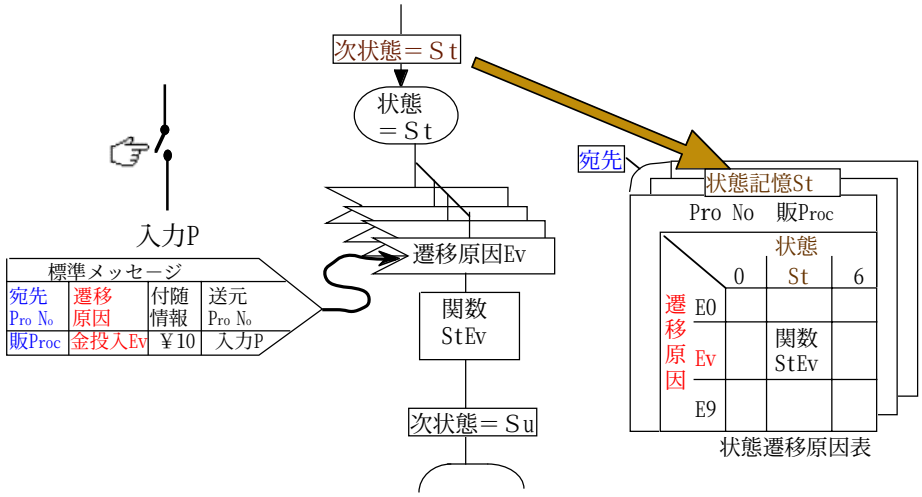
今あるシステムを，FSMxとFSMyとFSMzの３種のFSMで構成しました．それぞれの状態数はＸ，Ｙ，Ｚとします．このシステムを外部から見ると，状態数は最高Ｘ・Ｙ・Ｚになりえます．今Ｘ，Ｙ，Ｚがいずれも３なら，このシステムは最高3×３×３で27状態になります．

ここで，両方のソフトウエア開発負担を比べます．FSMのソフトウエア規模は，状態数に比例するという経験則があります．これを使うと，小さい３個の各３状態のFSM群をつくるコストは３＋３＋３で９です．しかし，全体をまとめてつくるシステムでは２７です．その負担は３倍にもなります．簡単にいえばX+Y+Zの負担で，X・Y・Zの機能を実現します．

小さく作り易く品質良い３状態程度の小FSM群でシステムを作ることは，大きなシステムをFSMと考えて作るより規模が数分の１に減らせます．言換えると，システムを相互に直交的な要素群に展開することを繰返し，最大の効果/効用を最小のコストで実現する．それがシステム屋の仕事です．

小さいFSMの集合体でシステムをつくる方法は原理的に生産性も品質もよい方法です．外に標準化の利点を享受できます．私はこの原理を交換システムの構築に利用して，日立製作所のPBXおよび局設備交換システムの開発に適用して効果を確認しました．後で気づいたことですが，対象が均質化されると，それを扱うプロセス/手順/工程も揃い，（流用などのproductとは直交的に）processが相乗的であることで生産性や品質で，大きな効果がえられます．

ｲﾍﾞﾝﾄ駆動OS (MIDAS) ﾌﾟﾛｾｽ/状態/原因毎に分岐



- 遷移原因は標準化ﾒｯｾｰｼﾞで送られる．OS側は宛先毎の状態遷移原因表を使い，表に付属する状態記憶が保持する状態，および到来した遷移原因の２者の交点の関数を実行する．この実行の最後の次状態定義で状態記憶を更新する．このEvent driven OS MIDASは高速高効率である．

複数のFSMを制御するには，各種の工夫が必要です．具体的にはイベント駆動OSがその役目を果たします．このslideで要点を説明します．

図の中央の状態記号は状態Stに対応するものです．ある遷移ルートがこの状態に終着するなら，流れ図の最後に次の状態の定義（図ではSt)を次状態定義領域に記録します．

図の左から新しい遷移原因が来ました．この楔状記号は標準化したメッセージです．封筒の表と裏のように，メッセージの行く先のプロセス／FSMの番号が最初に記され，最後にメッセージを発行したプロセス／FSM番号が記されています．この封筒の中のメッセージは遷移原因（たとえばお金投入）とそれに随伴するパラメータ（投入金=100円）です．

図の左のメッセージを受信すると，青印の宛先processを取出し，このprocessの状態遷移原因表を取り出します．また，このprocessの次状態定義領域から現在状態(St)を読み出します．メッセージ中の遷移原因(e)を取り出して，状態遷移原因表上で，状態Stと遷移原因eの交点にある状態遷移ルートのプログラムを取り出して，その状態遷移ルートの実行を始めます．この実行が終わる時，これに対応する終着状態を知ってこのプロセス/FSMの状態記憶を更新しておきます．

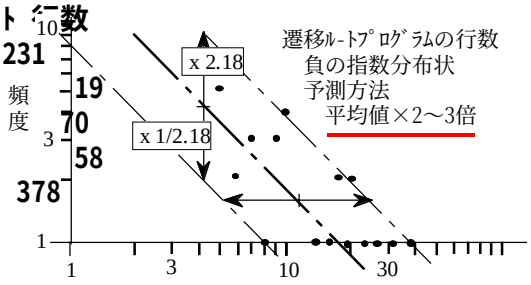
この簡単なメカニズムでMIDAS OSは高速に状態遷移を実行させます．即ち，仕様のとおりにプログラムを実行させます．多重度をあげるには，宛先processを階層的に構成して，いわば封筒の宛名をキチンと解読して行けば目的地に行き着けるように設計できます．

評価　最優等 1996年Jﾁｰﾑの成果　(レポート回覧)

切符販売機

	FSM	状態	ﾙｰﾄ行数
自動販売	3	10	231
ｶｰﾄﾞ精算	2	3	19
画像出力	2	10	70
音声出力	2	6	58
合計	9	2978	378

能力状態数　3x2x2x2　=　24
能力/実績　24/9　= 2277
ﾙｰﾄ/状態　29/9　3.2　3.2
行数/ﾙｰﾄ　378/29　13.0



状態遷移ルートプログラムの行数(コメント抜き)

- FSM当り　3.2ﾙｰﾄ　=　41行，×2ならFSM当り83行，×3なら同　124.4行
  単一FSMに比べ規模は　約1/3　位に低減
- 一般のｱﾌﾟﾘｹｰｼｮﾝの場合，小状態数多FSM方式では分割時のFSM数×100行
  該ｱﾌﾟﾘ実現に必要な最小限度必要な部品類の規模合計を加えた値で実現できる
  雑な設計に比べて1/3²ないし1/3に低減できる

28

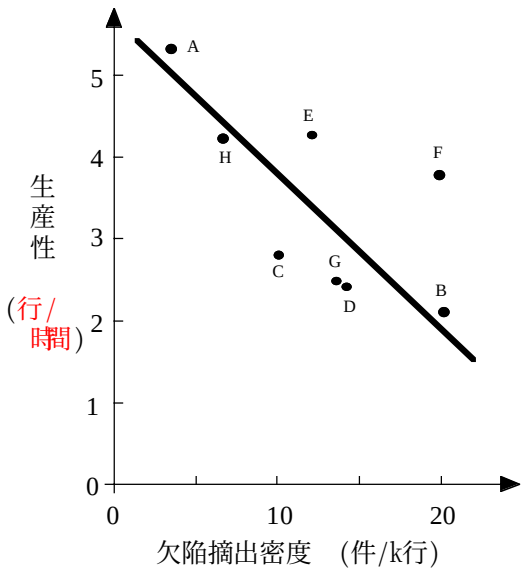
演習で各種の自動販売機類を開発した具体例として，1996年度Jチームの作品「切符自動販売機」をご紹介します．このチームは全指示事項を正確に実行して設計文書を作り，高い品質とよい生産性を実現した最優秀チームです．

slideに開発結果の数量的データを示しました．ご確認ください．４FSM群合計で9状態，Ｃ言語の378行で実現しています．

遷移ルートプログラムの規模は平均約13行，FSM当たりに約100行位に過ぎません．簡単に評価すると，4FSM化により単一FSMの場合に比べ約1/3に規模が減っています．この13行は組込化に伴う制御部の増分です．

右のグラフは各遷移ルートプログラムの規模の分布を両対数表示しものです．状態遷移ルートのプログラムの規模は負の指数分布状になっています．これはかなりランダムな様相です．ある一定幅で分布しているので，対数正規分布状です．（完全にランダムな対数正規分布ならバラつきは中央値×1/3～３倍の範囲になりますが）ここでは1/2～２倍の範囲内に収まります．組込にする時のプログラム行数の増加はFSM当たり100行程度でしょう．

評価 品質の評価 欠陥密度が低下



- 摘出欠陥密度が低下
  通常約100件/k行
  → 12.8件/k行
- 感想 バグ少ない
  一発で動いた！
  文書化努力が酬れた
  文書化で楽に追える
- 生産性も向上

参考
3～４年生(初中級) 約100件/k
2年生末期
Typing誤り 1.95件/ｋ字
Syntax誤り 42.8件/ｋ行
平均 4.7回 runさせる
DFD以降誤り 50～60件/ｋ行

1995年実績 Eは２式開発

29

これは1995年度の各チームの成績の図です．

報文集の予稿の誤りを訂正させてください．図の縦軸は生産性です．予稿での単位の表記は件／ｋ行は誤りで，正しくは行/時間です．

図の横軸はテストで摘出した欠陥密度で，平均12.8件/ｋ行です．学生の作り込み欠陥率は通常約100件／ｋ行です．各種のバグ低減策を実施した結果，約1/8に低減されました．大幅な品質向上が実現できました．これは文書に明確に記述し，厳密にチェックした結果です．

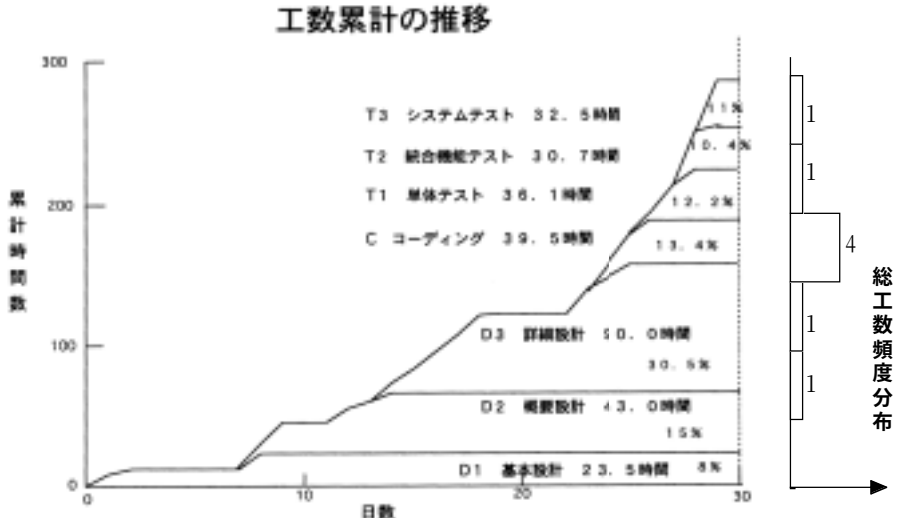
演習の感想の中には，「オッ．一発で動いた！」などの驚きが記され，バグが少ないことが特徴です．後に学生のアンケートで示しますが，「キチンとした文書化の結果」であることが正しく認識されており，また実験で検査した人達のレポートは，「豊富な設計文書で設計内容がよく理解できた」こと等がしるされています．

また，図の傾向線に見られるように，テスト摘出欠陥率が低い，すなわち品質が高い程，生産性が高い傾向が明確に現れています．

設計とは本来このようにあるべきではありませんか？

評価　総工数推移（各チ-ム毎）

- 演習期間(30日)中のチ-ム毎の総工数の推移



話が集約しない　　後半は
内に時間が経つ．　　脱兎の如く．

30

この演習の悩みは１教科の演習としては工数が掛かることです．各チームは毎日の作業記録をつけており，最終報告で種別毎に合計して評価させました．

図の縦軸はチーム毎の総作業時間で，目盛りは100,200,300時間です．

右端の図はチームの総工数の分布です．正規分布状ですが，中央値は約150人・時間，最大は約300人・時間です．チーム総工数が150人時間，１チーム５人とすると，学生はこの演習に平均３０時間位費やします．これは２単位を取る効率を考えると引合わない．これが教師の悩みです．

中央の図は，横軸は30日の日程，縦軸は工数の累計を示すグラフです．このグラフは300時間位費やしたチームですが，全て似た傾向を示します．

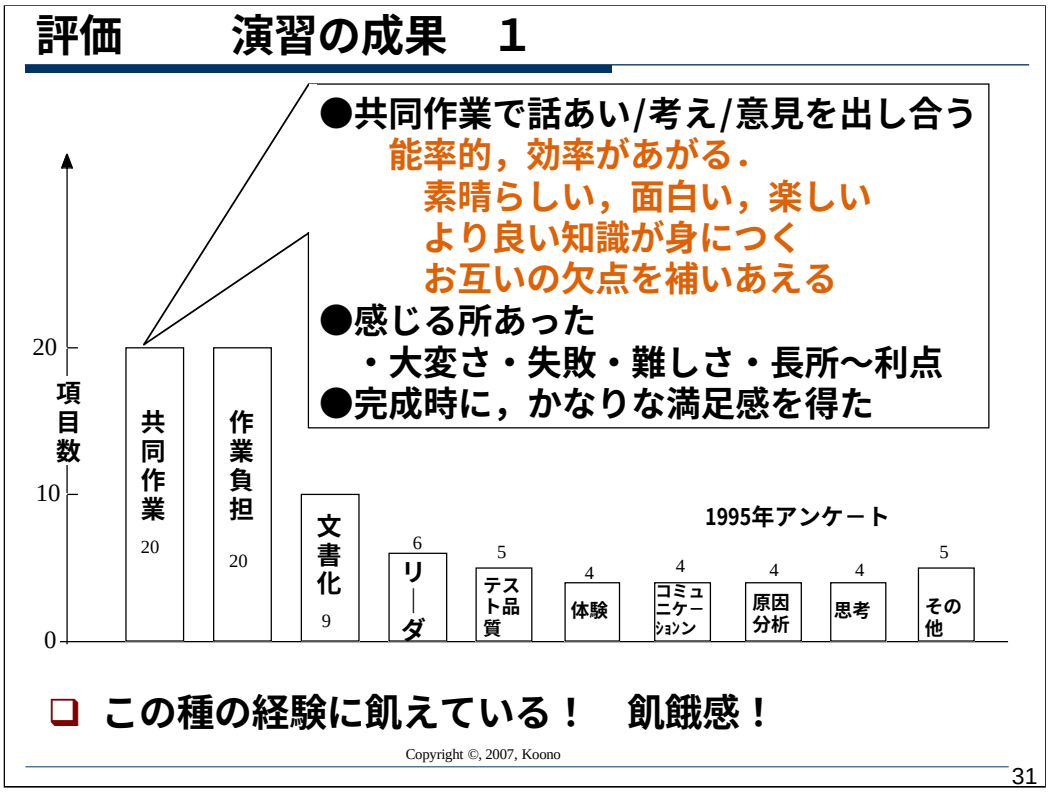
おおよその傾向として，工数の配分は以下になります．

各種テストに1/3

詳細設計に1/3

基本設計と概要設計に1/3

一番始めの何を作るのか，また，FSM毎の機能配備や状態遷移を飲み込むのに時間をとられています．しかしテストになると，締め切りも近づくので，大変な勢いで進行します．終わりのドタバタにしては工数が少なくて纏まるにはバグが少ないからです．最初に上手く滑り出さないと，最後は悲惨な目に会います．それを如何に事前防止するか，が教師側の課題です．

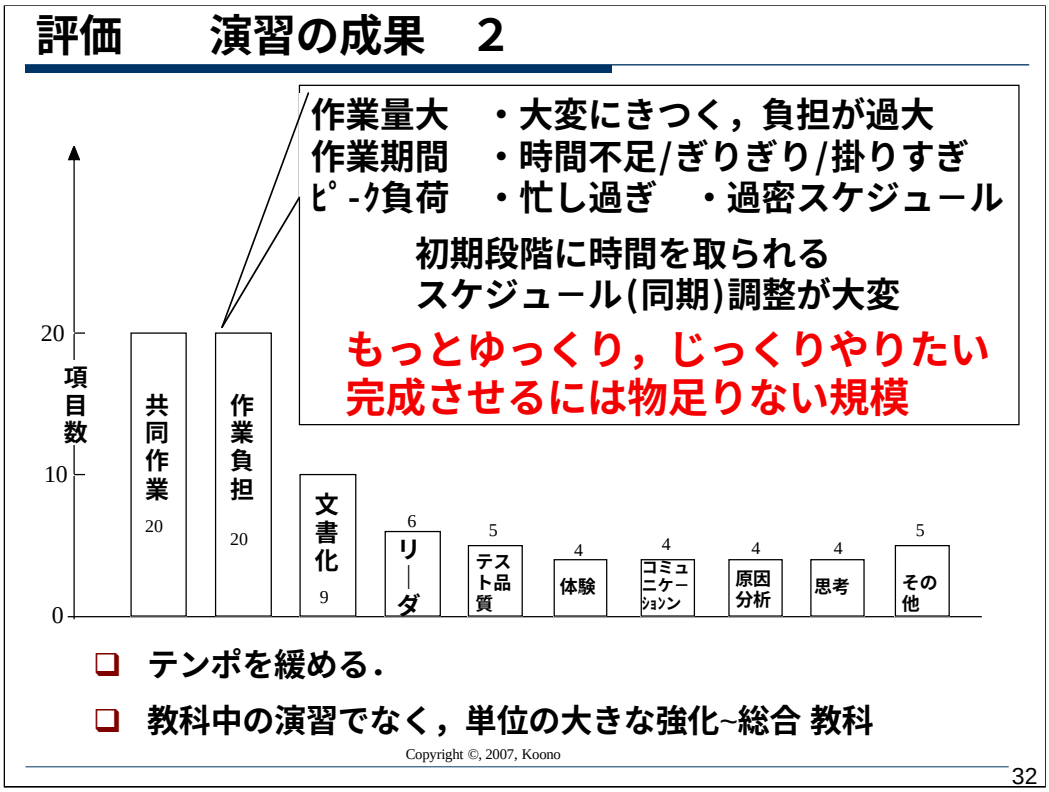


全員にアンケート調査した結果を評価として，slideに示しました．終わりに一言の形式では，内容がつかめません．工程毎の枠を作り夫々に意見を云わせて，かつ総論のコメントを引出します．コメントを自由記載させます．結果のコメント単位に類別して計数し，上位からソートしました．

学生諸君に**一番強い印象を与えるのは共同作業**です．能率的/効率が上がる/素晴らしい/面白いなど，今まで全く知らなかった作業の印象です．

同時に，共同作業の難しさや，「なるほどこんなものか！」的な深い理解を行い始めています．

これを見ますと，学生達はこのような経験に飢えていると感じます．授業の中にこの演習のような**実感をもてる作業を増やしては如何でしょうか？**



アンケート中で順位第２の項目は作業負担過大です．

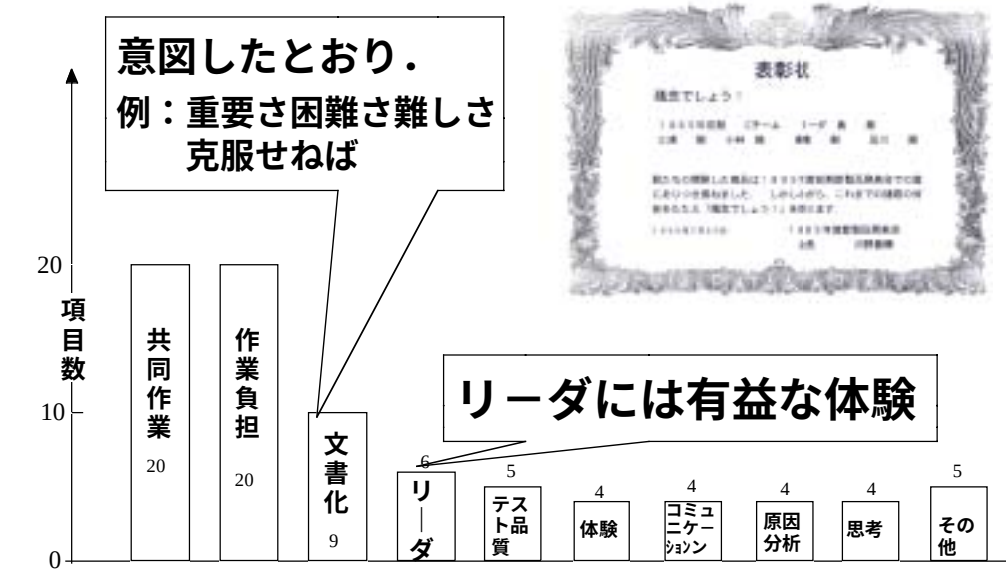
この言い分とおり，作業量大で，大変にきつく，負担が過大
作業期間が短いので，時間不足/ぎりぎり/掛りすぎ
が実態で，学生達の言い分の通りです．

この授業の前のコマを受け持つ先生方の目からは，このピーク時期，
２～３週間の間，授業中に盛大に内職に没頭する
疲れ果てて居眠りする人が多発，
一喝して注意すると今度は欠席多数である．
こんな訳で，私の方にも「何とかせよ」とお叱りが来ます．

当方も学生達に留意を促しますが，なかなかに改まりません．
結局演習の最後で皆さんに期限厳守の強い圧力の掛かる発表会をやめるしかありませんでした．

一方，学生達の中にはもっと質量を増やして欲しい希望もありました．

評価　　演習の成果　３



- 教育として取組むべきではないだろうか？

33

パレートの原理に従い，少数の効果的項目は簡単にご説明します．

○文書化については，品質向上とのトレードオフが理解されています．

○リーダの働き

・リーダには，「原則として，社長は全員に働いて貰うことが仕事であり，プログラムなどの「作業には手を出すな」と教育しています．しかし，発表会では各チームの発表の冒頭に，チームリーダが社長として敵に勝てる商品を如何に考えたか等全体状況を説明させます．

・そんな訳で，リーダもメンバも，指揮統率についていろいろと感想を述べています．でも，最初は皆にミーティングの声を掛けてもかなかに皆が折れ合って予定を立てる所まで協力的ではありません．リーダ達のアンケートの本音は，「二度とやりたくない」と云うことでありました．でも，多くのリーダは満足感を持つた人が多くいます．

○右上は発表会で渡す賞状の例で，全員に行き渡るように配慮します．

ゲーム感覚で，皆が喜ぶのですが，「先生，彼のチームの評価は発表の比重が大きすぎる．自分たちの成果をちゃんと見てくれ，などクレームも経験しました．学生達とは必ず両者で信頼しあえる関係が必要と思います．

◎教師にとってしんどいことも多いのですが，こういった実社会と同様な疑似体験させることが必要なのではありませんか？

他の技術評価（研究室ﾌﾟﾛｼﾞｪｸﾄへの適用）

- **先行**
  1. 1960～62全ﾃﾞｲｼﾞﾀﾙ電子交換 (BTL委託研究Wired logic遅延線ﾒﾓﾘ1FSM 7状態
  2. 1976～78 既存電子交換制御に独立FSM導入 状態制御は改善したが，その他で大変．
  3. 1982～85 全ﾃﾞｲｼﾞﾀﾙ電子交換 明確なFSMを用い，多数FSM方式 日立PBX 局設備
  4. 1985～87 交換用機能試験機(Regression tester)

- **知的CASEﾂｰﾙの主制御部に適用(1996～97年) 自販機経験者**

  市販PAD CASEﾂｰﾙを改造．知的部分の制御に適用．方式が明解品質良好．

  CASEﾂｰﾙﾓｰﾄﾞ制御FSM(2状態2ﾙｰﾄ,平均ﾙｰﾄ当り20.5行),

  全体ﾓｰﾄﾞFSM(5状態17ﾙｰﾄ,同11.6行),

  動作制御FSM(5状態7ﾙｰﾄ,同17行),

  変換FSM(10状態16ﾙｰﾄ,同13.3行)．VBasic1212行，C言語509行．

- **統合知的CASEﾂｰﾙ（自動設計が可能）**

  全制御を小状態数の多数のFSMで制御(1999～2001年)

34

私は学部卒業研究からデイジタル電子交換システムの研究をしており，1960年代の始めにWired logic方式でシーケンス回路でシステムを制御しました．以後，この技術を交換ソフトウエアに適用して，次々と改良して参りました．この演習はその最終版の試行です．従って技術的には殆ど問題を感じないで済み，それだけ彼らの他の面を学ぶことができました．

大学に戻って１０年間，ソフトウエア自動設計の研究を行いました．最終的には，具体的な成果として，PADを用いた知的CASEツールを1998年に作り，さらにこれにDFDを加えた統合知的CASEツールを2001年に作りました．これらの中で演習を学んできた結果の１例を記します．

知的CASEツールは，市販のPAD CASEツールを改造したので，変更追加部分のみにこの方式を適用しました．Slideに記したように，必ずしも満足な小FSMではありませんでした．しかし，担当学生は今までの卒研学生とは全く違いました．（先が見える為でしょうが，一々小生の判断や指示を待たずに）博士課程の先輩やメーカの担当者の人達と肩を並べてドンドンと仕事して，分厚い設計文書も自分で書き上げていました．

統合知的CASEツールは完全自作ですから，小FSMが(繰り返しも含めて)総計では100近くが並行動作するものになりました．

おわりに

- 教育結果の評価
  - 「人間育成/実作業の疑似体験」　十二分に達成
    共同作業　体験/楽しさ/難しさ/ﾘｰﾀﾞとﾒﾝﾊﾞの在り方
    ものの考え方～あるべき姿　統率，仕事，姿勢/文書化
  - 演習の限界　時間～単位/教科枠・・ﾊｰﾄﾞ同時並行
  - 技術　文書化/仕様を実行するﾌﾟﾛｸﾞﾗﾑ化/規模最小
- 実施経験からの提言
  - 工学系 情報工学の教育
    この種の教育が１番必要なのではないか？
    １年コース，S＋Hで構成する総合教科を推奨する
    ご協力が必要なら，何時でも尽力します
  - 教科書化と社会人/会社等の組織に教育し普及させる
    高品質，高生産性，高流用率，最小規模

35

結果を纏めます．

○教育結果の評価として，実作業の疑似体験で人間と技術を育成する目的は十二分に達成できました．技術的学習の効果は演習結果に出ました．

○しかし，率直に云って教科の一部の演習としては過ぎたるもので，「時間がかかる」，「作業がきつい」と学生諸君を悩ませました．

◎今後への提言です．

・工学系の情報工学の教育としてこの種の演習が一番必要です．皆が力を合わせてやる様子はご理解戴けたことと思います．負担加重を避けるために１年間のコースにして負担を平準化したいと思います．
また，ソフトウエアのみでなく，ハードウエアも含めた総合演習が相応しいと考えます．

・学生達と社会人とは違う点もあります．しかし，基本的にこの教育は組み込み系を担当するか否かにかかわりなく必要です．高品質，高い生産，始めから計画した高い流用率，最後に規模の最小化，すなわち高い技術レベルを目指す技術教育は何時でも重要ではありませんか?
今後はこの面にも力を入れて推進したく思っています．

ご清聴に感謝します