ソフトウェア工学 138－15
（２００２．７．１２）

ソフトウエアクリエーション：ルールによる自動設計と知識による自動設計

Hassan Abolhassani[#]　　河野　善彌　陳 [&]慧

人の設計に倣ったソフトウエア自動設計について，Rasmussen の云う技能準拠，ルール準拠，知識準拠の３種の自動設計の方式と定量的評価が示されている．ルール準拠では自動設計（展開）率が高く，かつコスト負担はすくない．知識準拠はこのルール準拠の延長上で使い，困難度の高い場合の設計を行わせることが妥当である．これらエンジン群を纏めると，人の知的な向上過程が現れるなど，人間に近い特質を示す．終わりに各種の将来展望を示している．

Software Creation: Automatic Software Design by Rule-based and Knoweldge-based Engines

Hassan Abolhassani[#], Zenya Koono [§] and Hui Chen[&]

This reports on a way of software design, which is learnt from human designers. Engines for the design are Skill-based, Rule-based and Knowledge-based. The rule-based engine shows a high degree of automatic detailing with small cost burden. The best way to use the Knowledge-based engine is to use it as to extend the design-able cases. When they are assembled to a system, it shows similar characteristics as a human does. Outlooks for future are given.

１．はじめに

ソフトウエアの自動設計は，ソフトウエア工学の夢である．筆者らは「如何なるソフトウエアも自動設計できる基礎を確立すること」を狙い，設計者の頭脳のシミュレータをボトムアップに構築する独自な方式[Koono92a] の研究を続けてきた．

研究の基礎として人間工学のZipfの労力最小化の法則[Zipf72] に準拠する．これは「人は問題に直面すると最も簡単な解をまず試みる．これが不可なら少し複雑な解法を試み，これが不可なら更により高度な解法へと，次第に解法を高度化する」と云う．これは人が複数のエンジンを駆使することを指している．Rasmussennは，同じことを「技能のレベル，ルールのレベル，知識のレベルがある」[Rasmussen85]と云う．かような場合には，所謂「知識」を中心とする人工知能の技術では扱えない．

筆者らの研究は基盤の整備から出発した．エキスパートの特質に基づき，高成熟度組織で見られる階層的工程が設計知識の構造であり，設計の中心は自然言語による階層展開の連鎖になり，人の概念の親子関係をそのプロセスの単位的な知識とする．設計図面から設計知識を系統的な獲得し，それを系統的にエキスパートシステムに再構築して，設計動作を再現できる[陳 97]．

最も簡単な例を取上げて研究し，定量的な評価を行なって確認し，一つずつボトムアップに積上げる逐次近

Forntage-Razor fish Inc.,
Hassan.Abolhassani@razorfish.com
§　東亜大学　通信制大学院
Graduate School, University of East Asia
koono@vesta.ocn.ne.jp
&　国士館大学情報科学センター
Center for Information Science, Kokushikan University
chen@kokushikan.ac.jp

似を重ねて設計者のシミュレータを作り,設計動作の再現を確認する. この仮説と検証は, 身体性認知科の取組み [Pfeifer et al. 99] に近い. 技能レベルについては詳細を報告した[Chen98]ので, 本報告では, その概要,「ルールのレベルと知識のレベル」, および展開を報告する. 当初の３件の予定を１件に圧縮したので,不十分な報告になったことをお詫びする.

２. 技能レベルの設計の概略 [Chen98]

設計での単位的な階層展開(親子)関係を設計ルール(例を図３に示す）と名付け, 単位的な設計の知識とする. 技能レベルのエンジンとして, 人の設計から設計ルールを抽出して知識ベースに蓄え（知識獲得), 親概念を指定すると対応する子が読出されて,設計図面に追加される（自動設計）モデルを採った.

構造化チャート PAD (Probelem Analysis Diagram) を用いるCASE ツールに知識ベースを付加して, 知的CASEツール(ICASE)と名付けたプラットホームを構成した. PADでは人の概念展開が言語表記されて, 階層的な樹枝図で表現する. 従って, CASEツールのディスプレイ上の図形をツリーウオークすれば,人の設計結果の図面から設計ルールを自動的に採取できる.

逆に, 親概念である機能シンボルを書き, 親情報をツリーウオークで求めて知識ベースに照会して,当該設計ルールを取出し,返送された子概念を当初の親概念の次に貼付ければ自動的に展開（自動設計）が行える.

このシステムは, 人が設計する度にその知識を蓄積するから学習能力を持つ. そこで, 各種の新概念の出現が一渡り終わった以後の, 機能拡張期（工学用語として不適当だが所謂保守）での設計には役立つ. もしある概念が初めて出れば, その都度人が設計し, その設計ルールを追加する. 厳密な設計監査体制で設計した各回の機能追加設計の実績資料で評価を行った.

工学の研究では, 新しい方式提案には定量的な評価が必須である. しかし, ソフトウエア工学では中々行い難い. 工夫を重ねた結果, １展開作業を単位とすると, 改善前後の比を用いて作業のバラツキを消し, ハードウエア直接作業で用いられる習熟性工学[師岡 94, Part4.2 of Salvendi82]を用い定量評価できた.

図１はその結果を示す. 図aの横軸は初設計で１回, 以後は２回, ３回と繰返す設計の回数, 縦軸は新たに経験した設計ルールの累計数をとり, 初回実験と念の為の確認実験の両結果をプロットした. 打点は始めに急激に立上がり, 次第に緩やかになる習熟傾向を示す. 習熟性工学に学んで両対数用紙で示すと,図ｂのように直線傾向線が表れ, これは対数習熟効果である. 定量評価には傾向線の数式表示を用いた. 図ｃは自動展開率を示す. 初めに急激に立上る（省力効果が大きい）が, しかし, 中々１００%には近づかず浅知恵の限界を感じさせる.

この研究の結果, 技能レベルの知識の利用でも効果が高いことが明らかになった. しかし, 設計ルールの自動生成および１親概念が複数の意味を持つ時の選択を自動化すること, の２点が次への課題になった.

３. ルールレベルの自動設計

３. １研究

構造化チャートのみでは前記の課題を解決できない. 原理に立ち戻り,一階層展開毎にデータフロー図でアルゴリズム, PAD で制御を表す統合知的CASE ツール(IICASE) を作りプラットフォームとして用いる. 図２[Chen 00] にその構成を示す.

研究の結果, ルールレベルの知識として, 階層展開する子概念の文型であるスケルトンをデータとし,このスケルトンを用いて設計ルールを完成させる手続きをメソッドとするフレーム形式のモデルを採った.

小さくても完結的で, 各種の定義が明解な在庫管理システム（ソース約７００行程度）[Abolhassani02]について研究した. 高成熟度組織の製品なみに, 多種類の文書を作り, 厳密で厳しいﾃﾞｻﾞｲﾝﾚﾋﾞｭｰを設計進行の都度に繰返し, 図３ [Abolhassani02] に示すようなデータフロー図とPADを作った. この結果を, 縦に横に眺め, また各種のデータを取り, 持越し課題の解決に挑戦した.

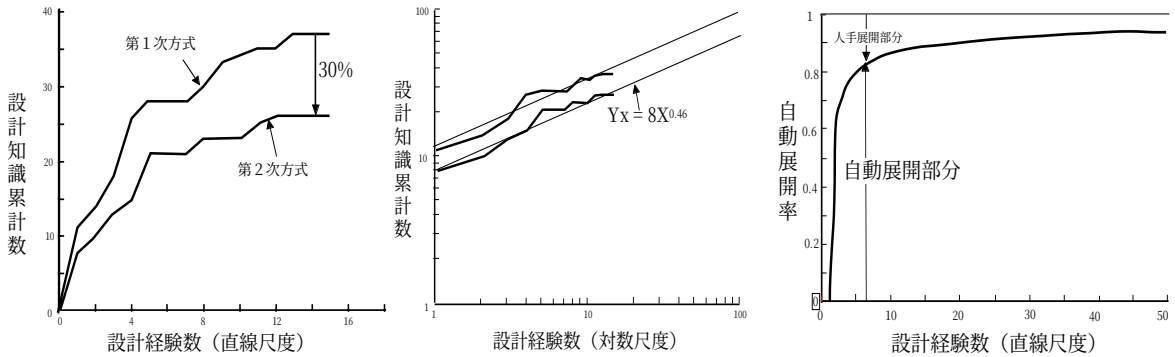


図１ .a 習熟特性（直線）　図１ .b 習熟特性（両対数）　図１．ｃ自動展開特性

図１ 技能レベルの自動設計の特性

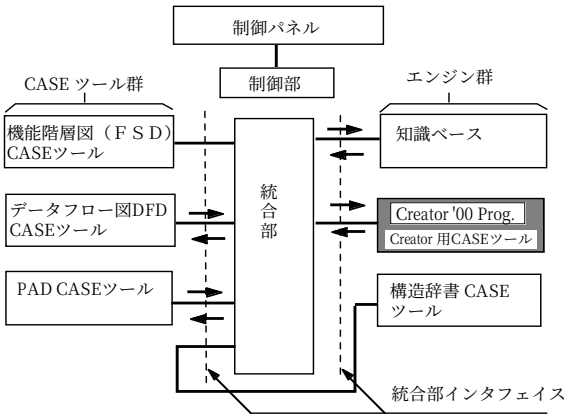


図２　統合知的 CASE ツールの構造

３．２ルールレベルのエンジン (Creator '00)

このエンジンへの入力は，図２の左の CASE ツール群で自動獲得した設計ルールの親概念である．また，その出力はデータフロー図およびPADで表した設計ルールで対応 CASE ツールに送る．内部を動作順に説明する．

図４．ａはデータフロー図で表した設計ルールの１例，図ｂは子データフロー図を抽象化したスケルトン，また図４．ｃは，親の入力データ，機能，出力データからなる単位的データフロー図とスケルトンを使って設計ルールを作る為のメソッドである．図ｃに見られるように，殆どがあるシンボルの一領域の内容を他に転送する形式で専用部品化したので，メソッドは簡単に書ける．親の入/出力データは子に継承され，親の機能を展開した子概念は表記が完成されて，子データフロー図が自動生成される．生成した子データフロー図を変換し，子 PAD が得られる．（JSP の場合は後で説明する．）

スケルトンであるデータとメソッドはフレーム形式で表し，動詞の意味毎に設ける．図５はこの集合体である動詞辞書を示す．如何様にも対処可能なように，デー

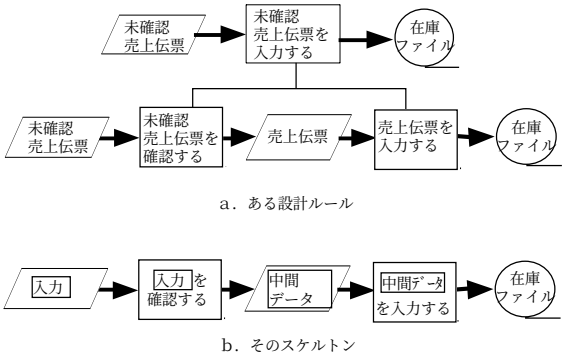


図４　設計ルールとその生成ルール

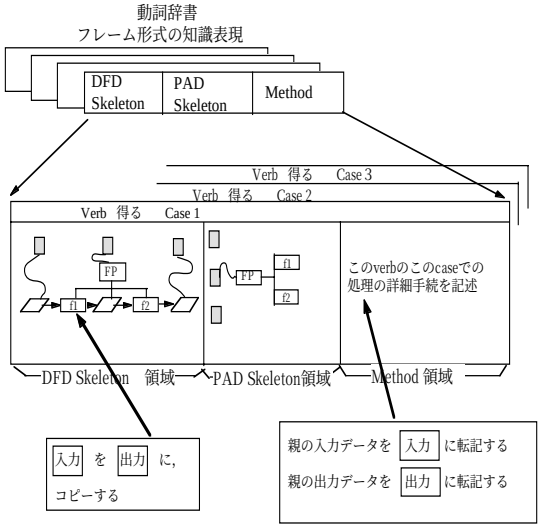


図５　動詞辞書

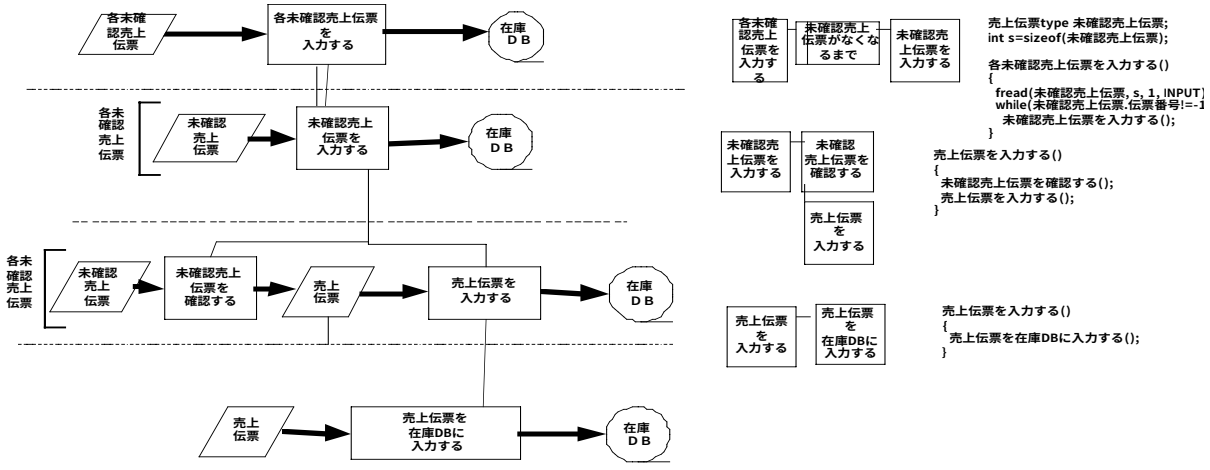


図３　データフロー図と PAD を用いた１階層展開毎の展開図

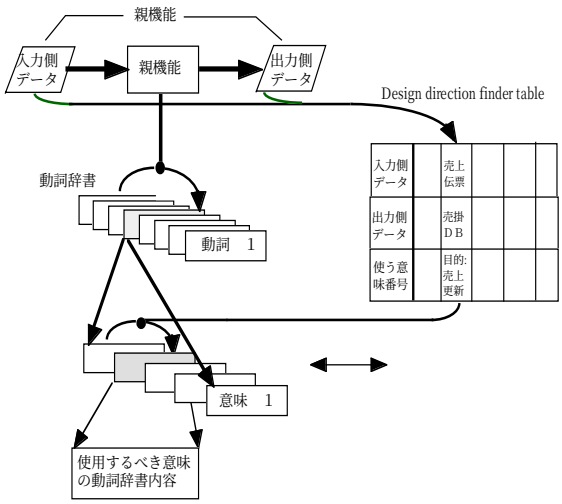
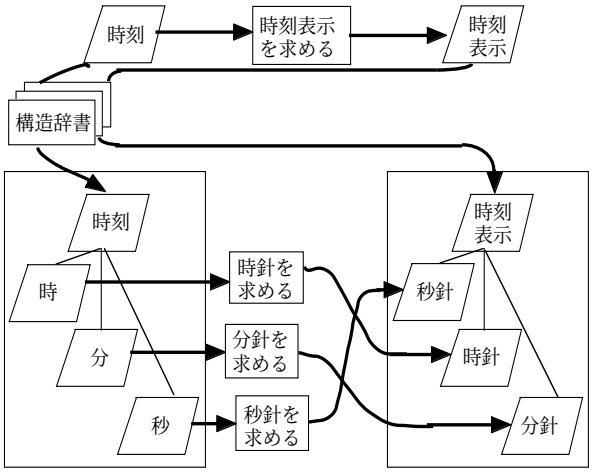
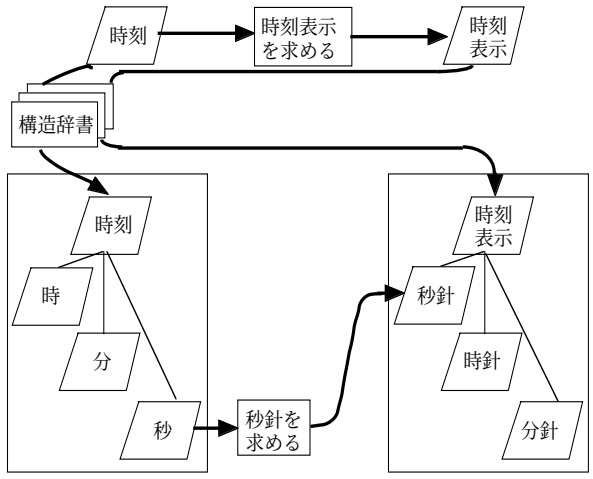


図６　意味の自動選択の機構



ａ．完成形



ｂ．中間形

図７　Jackson Structured Programming (JSP) 形式の自動設計

タフロー図と PAD の両スケルトンが設けてあるが，通常はデータフロー図のみで, 子データフロー図が完成した後に，これを変換して子 PAD を作る．

図６により動詞の意味の自動選択を説明する. まず, システムが果たすべき目的を, それを実現する手段に階層展開する．この手段を目的として, 更に具体化した実現手段に階層展開することを繰返す．（この繰返しは目的樹と呼ばれる階層樹枝図で表される.）これを繰返す内に，あるプログラムが実現手段として規定される．

このプログラムは，まず意図した目的から，その出力（データ）が定まる．次にその出力が得られる筈の入力（データ）が定まる．両者が揃うと機能 / プログラムの実現アルゴリズムが定まり, それは自然言語で機能として表記される．そこで，親入力データ，親機能，親出力データの３者が揃うと，そのアルゴリズムが定まる．言換えると，親機能の動詞の意味が定まる．

但し，この３者で辞書を索引するのは，非実用的である. 図６は具体的な機構を示し, 中央上の親概念から動詞を抽出して動詞辞書を索引する．一方, 親の入出力データで右端の表を経由して当該動詞についての意味の指定情報を得る. 動詞辞書の索引結果中の求める意味のページを取出す．このようにして，第２の持越し問題, 親概念の意味の自動選択が実現できた．

在庫管理など事務系のプログラムではジャクソンプログラム設計法(JSP)が良く用いられる. これは図７．ａのように，親概念の入出力データが共に階層展開する時, 入力と出力の対応するデータを結ぶデータフロー図群が子概念になる場合である．

この実現は辞書のメソッドのみでは不可で，専用のメソッドをCallする．条件として，入出力データやファイル，帳票類は，図2の構造辞書CASEツールにより階層的な構造を予め構造辞書に定義しておく．親概念のデータフロー図が与えられると, 構造辞書を参照して図７．ｂ図の左右に示すように, データ構造図上に子データを貼付ける．次に，図ｂに一例を示すように，出力子データ毎に対応する入力子データを見つけ, 仮機能シンボルを入れて仮の子データフロー図を作る. 次々と出力子データから対応する入力子データを調べ, 仮機能箱を持つ仮子データフロー図群を作る. 最後に親機能の動詞を継承し, 出力子データ名称と合わせて, 各子機能の表記を完成させる. これらの子データフロー図群から変換して子 PAD を作る．

３．３　評価

これまで説明した構成方式により，IICASE を用いて, 在庫管理システムについて概念展開レベル（ソースコード化を除く）の設計再現を実験的に確認した．

このシステムの特徴は，コスト負担が低いことである. 在庫管理の７プログラムの設計には, 動詞あたり平均３ページ，７動詞の動詞辞書（２０ページ）が必要で

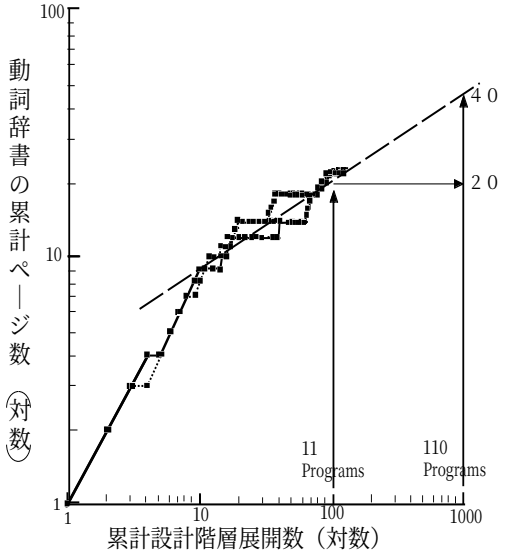


図８　ルールレベル自動設計の評価

あった. 動詞辞書では, スケルトンの記述は４．６シンボル/ページ, またメソッドはＣ言語で平均１０．１行/ページであった. 各部が簡単なので, エンジン全体でもＣ言語で約２．８Ｋ行に過ぎない. IICASE も同様に徹底的に構造化され, 標準化してあるからその全体規模も小さい[山田他 01].

図８[Koono 01a]にルールレベルの自動化の定量評価を示す. 横軸は, 在庫管理設計システムを設計する時の階層展開の累計数を示し, 縦軸はそれまでに必要となった動詞辞書の累計ページ数を示す. 初設計の７プログラムおよび, 以後の機能追加７プログラムについて打点した. 設計のシーケンスによるバラツキは小さいので, 複雑化を防ぐ為に数例のプロットのみ図示した.

図のように, 折れ線状の傾向線が現れた. 初めの部分は主として動詞の増加, 後の部分はページの増加が中心である. 図で１００展開（ソース約７００行）の状態から出発して, １０００展開（同約７Ｋ行）の設計を行なえば, 動詞辞書に約２０ページの追加が必要と推定できる. 簡単に云えば１０００展開中に２０ページ分の人手介入を要するから, 自動展開率は約９８％になる.

以上のように, このルールレベルのエンジンは自動化率が高く, 効果／費用比の高い.

４．知識レベルの自動設計 [Abolhassani00]

知識レベルの設計知識 [Abolhassani00] の中心は, 各種の基礎概念辞書である. これは, 技能レベルでの知識ベースやルールレベルでの動詞辞書と同位置になる. この上位にはマイクロデザインルールがあって, 基礎概念辞書を正方向および逆方向に読んで単位的な情報変換を行う. マイクロデザインルールは, 最も基礎的な単位機能であるから, 全ての知的処理に関わっていると見ることもできる. 表１[Abolhassani00]はその一覧である. この上位には, これらを纏めたマクロがある.

これらは, 単位的な階層展開を記録した図３等の設計例につき, 各階層展開を産出する過程を推定して, その中で書く単位的処理を求め, 頻度などを調査して基本的な処理に絞込んだ. 表２は組合せる基礎知識辞書の種類を示す.

知識レベルの研究は, ルールのレベルに先立って行った. 前記の基礎知識辞書を準備してマイクロデザインルールを用いて在庫管理システムの人の設計の再現を確認した. この実験システムは, Prolog 910　行 , Java 404 行, Ｃ　295 行の規模であった.

マイクロデザインルールはルールレベル内のメソッドの中核的単位機能に相当する. 設計ルールとマイクロデザインルールのレベルには, 大きな差があり, マイクロデザインルールで設計ルールを実現する汎用的な処理シーケンスを作ることは, 極めて困難であった.

この為, 考え直してルールレベルの研究を行なったマイクロデザインルールはルールレベル内のメソッドの中核的単位機能に相当する. ルールレベルのメソッドは, 典型的な場合を対象に固定しているから, 該当する場合には効果的ではあるが, 裏返せばカバーできないケースが出てくる. そこで, 通常のメソッドで対応できないケースに対して, 救済用のシーケンスをマイクロデザインルールで作る方法を編み出した. これは, Zipf の法則そのままであり, 部分的な処理だから簡単であり, かつ問題の出現に応じて作れば良い利点がある. マイクロデザインルールを図７を用いて説明しながら, 知識レベルでの拡張策の例を説明する.

通常の場合には, 図７の左右の入出力データは, JSP ユニットのメソッドにより階層的に展開される. これは表１のマイクロデザインルールEXC(EXtend Concept)に

表１　マイクロデザインルール一覧

略称	名称
CDC	Construct Definition Chain
CAB	Construct ABstraction
EXC	EXtend Concept
UPC	UPward Correspondance
CFN	Create Function Name
SIO	Selecting I-Map and O-Map
CCS	Create Control Structure

表２　　基礎知識辞書の内容

種別	区分	実例
・データ名称や機能になる単語	名詞	時刻
	動詞	入力する
・関係の定義		
	（属性）	得る＝獲得する
	部分と全体	時刻（時, 分, 秒）
	定義	時＝６０＊分

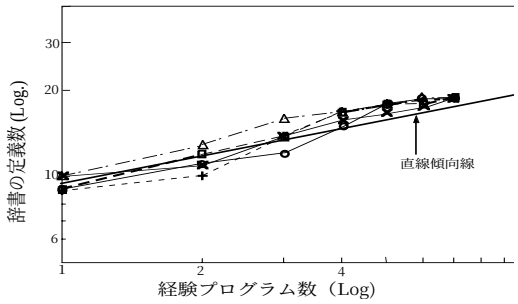


図９　基礎知識辞書の習熟曲線

より行うことができる．展開情報が構造辞書に無くても，基礎知識辞書にあれば，EXCで展開すれば動作を続けられる．

階層展開した子入出力データ間の関係を設定した後，JSP ユニットのメソッドは仮子データフロー図群を作る．UPC (UPward Correspondance) を用いれば，左右がアンバランスな場合などの処理を作ることができる．

図７では，簡単の為に「時」と「時針」のように，入出力に共通な語彙を使ってあり，子入出力データ間の対応が容易に確認できた．しかし，「時」と「太い針」のように，共通語彙が無い場合もありえる．例えば基礎知識辞書に「時針＝太い針」の定義が含まれていれば，CDC (Construct Definition Chain) で辞書索引を繰返して「時」と「太い針」との対応を認識できる．

これは母胎であるルールレベルを用いる自動設計の不成功の頻度は低くなければならないが，前記の評価を見るとこれは問題無いと判断した．このように，典型的かつ定型的な処理でカバーできない場合を吸収する方策は，実用中に改善を続けることが可能で，好都合である．（これも人の場合に似ている）

５．知的なシステム

これまで，技能レベル，ルールレベルおよび知識レベルでの単体動作を説明した．これらを組合せるとシステムとしての知的動作の特徴が現れる．各種の構成が可能なので，簡単なものから順次説明する．

図２のように，技能レベル，ルールレベル，知識レベルの全エンジン群が並行動作可能な配慮をしておく．CASEツール側から親知識を全エンジンに送出し，並行動作させて，最も早くに出た答を採用して残りを抑圧する．この方式で Zipf の法則とおりの動作ができる．

各エンジンを相互に関係させることで，より高度な性能が現れる．まず，前記の構成で採用する答（設計ルール）が決定して CASE ツール側に返送する時，技能レベルのエンジンにも与え，知識ベースに登録させる．以後，全く同じ設計ルールが使われる場合には，技能レベルのエンジンにより高速に処理できて，ルール，知識の両レベルの動作の完結は不要になる．

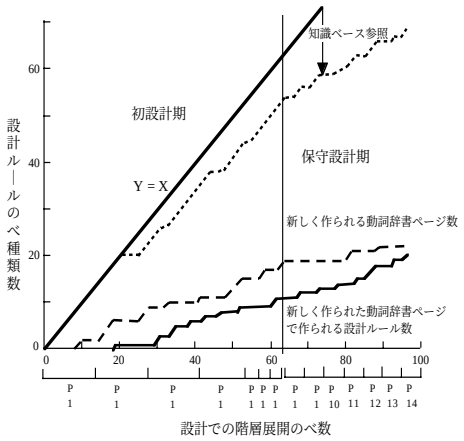


図９　辞書の習熟曲線

経験を抽象化して記憶することにより，更に高度な特性が出せる．ルールレベルの場合には，複数回同種の設計ルールを経験すると，この経験を抽象化して，スケルトンと対応するメソッドを生成できる．この方式では，同種の設計ルールを複数回入力するとルールレベル動作が可能になり，より広い範囲でルールレベルの自動設計が行われる．

図11[Abolhassani02]はこの方式の定量的評価を示す．横軸に設計経験量を，縦軸は設計ルール等の数を示す．同種設計ルールを２回経験する時にルール（スケルトンとメソッド）が生成され，新しく動詞辞書の１ページが生成される．このようにルールが生成される場合の辞書ページ累計数を下部の破線グラフで，これを用いて作られる設計ルール数を下部の実線グラフで示した．Ｘ＝６３の縦線は，７プログラムの初期設計の終わりを示す．

図のように初期設計の終わりで（経験の抽象化である）新たな辞書ページ数（下の破線）の伸びが鈍化する．しかし，これらにより作られる設計ルール数（下の実線）は伸び続けている．このメカニズムを備えれば，新設計ルールを経験すると，次回から同一の設計ルールに対しては技能レベルが高速に応答し，数回これを経験する内に，ルールレベルの動作が可能になり，類似の設計ルールについてもルールレベルの動作が始まる．

これの最も高度な場合は，何らかの設計ルールを経験すると，それに最も高い抽象化を施して基礎知識辞書に格納するケースである．すなわち，全く新しい設計ルールを経験すると，文法知識を使い精練して基礎知識辞書に加える．このようにすると，初経験の後に同種の設計が来ると，時間はかかるが知識レベルで解いてしまう．更に数回これを行うとルールレベルの動作になるからより早くなっていく．これらの内で，同一設計ルールが使える場合には技能レベルで対処する．

これまで，単純な「物真似」から始めて，「一を聞いて十を識る」ところまでを説明した．これらは何れも Zipf の労力最小化の原理とおりに動く．技能レベルで説

明したように, エンジン単体でも経験を重ねると, 習熟効果により高速化が起こる. ここに説明したようにエンジンを数種組合せ, 更に経験を抽象化して使うことにより更に高度な習熟効果がでる.

・これらの高度化の系列は何となく
人の知の高度化過程に似ているのでは？

６．検討

プロジェクトの研究結果の内, ルールと知識の両レベルを中心に全体も含めて報告した. しかし, 現在は未だ大きな壁に小さな穴が開いて向こう側が見え始めたに過ぎない. 今後の研究課題や拡張が山積している. まず, 第一にはソフトウエア工学領域である.

現状でも開始できるのは, ソフトウエア作業の定量化である. これは設計作業の中心的処理が階層展開であることから, 全体を定率で等比級数的に伸びる系として, 各処理に掛かる工数（時間）を基に, 全体の生産性や工数を定量的に扱う基礎「河野 92b][Koono96] を確立することである. 更に, 各処理で低率で誤りが生じることを基として, 誤りに関係する各問題を定量的に扱うことを可能にする[河野 93b][Koono 93].

この生産性と誤り関係は, ハードウエア直接作業と同じ外部特性を示す. そこで, ソフトウエア作業でもハードウエア直接作業の管理の柱である Industrial Engineering が適用でき, 定量的な品質管理や定量的な工数管理, また, プロジェクト後の実績整理と改善計画などなど, これまで経験的であった作業を合理的定量的科学的な管理の中に入れること[河野01b]ができる. この段階を経て, 産業界の求めに応える通常の技術と同列に並ぶことができる.

また, 人を扱う問題への展開が可能になる. これまでの研究では, 誤りの無い優れたエキスパートを前提にしているが, 全体の知識構造を確立させた上で, 各部分を劣化させていけば, 如何なる成熟度の人/組織の中心が判ることになる. これらを基に, 如何に人に働いて貰うか, 明確な指針が得られよう. １９世紀末から科学的合理的定量的をモットーとして, ハードウエア現場を研究する Industrial Engineering の中から行動科学が産まれたと云われている. かような人の内幕が判ることは行動科学の促進にもなるであろう. 人の作業過程の外部特性とこのような内幕とは, 産業のニーズに応える幅広いソフトウエア工学を産みだすであろう.「人の頭の構造が判ったらソフトウエア工学の問題について, みなキチンとした答えがでる」という笑い話がある. 本研究を発展させると, 将にこれが可能になる.

ソフトウエア関係の最後は自動設計である. ５章の終わりに示したように, 経験の抽象化まで組み込んだシステムは人と同じ進歩過程を持つ. 新人に仕事を手伝わせながら, 教育して一人前の設計者ができ上がるように, このシステムは経験を積むと知恵を蓄積して行く. 更に知識レベルで説明したように, 知恵の構造が不足の場合には, 処理論理を追加してレベルを向上できる.

Programmer's apprenticeとして評判になったシステムは, 部分プログラムレベルの論理～アルゴリズムを集積しようとするものであった. 現在に至るも, これは実用化されていない. 原因は人と直接インタフェイスしないアルゴリズムを使う発想に問題があったと推定される. このシステムは, 設計過程の知識は常に人が認識できるものなので, この轍を踏むことな無いと考えている.

第二はこの技術に関連する諸領域である. これらの分野の技術との交流～融合が必要である.

研究の進め方については, 身体性認知科学 [Pfeifer etal 99] が挙げられる. 本研究は, 理想的なエキスパートを考えて, 知識構造のモデルを作る. エキスパートの設計結果から解析して知の内部構造を推定し, 実際にそのとおりに動くか, を確認する. これを繰返して知の構造を高度化してきた. 身体性認知科学も殆ど同様な進め方に立つと筆者らは理解した. 頭脳の中心と身体外部との結合は, 有益な結果をもたらすであろう. この研究は, ハードウエア直接作業や一般のビジネス過程への適用拡大を念頭において進めてきたもので, 現在のレベルは人の意図的行動にそのまま適用でき, ヒューマノイドの中核技術になると考えている.

これに近い領域は言語に関する諸技術である. ルールレベルに簡単化したが, 本質は, 子概念を理解して 5W1H に合わせた句～文を作り, また理解することにある. 応用する立場だるから深入りはしないが, 色々と学ぶべき技術があろう. また, 辞書的な知識は言語の違いを超えるニーズがあり, 辞書を使う立場から, 使い易い辞書への注文も出るであろう.

他は教育あるいは知の問題がある. この研究では, 人の基本特性から見て不可欠な技能やルールのレベルから一つずつ積上げた. 現在の進歩過程は（未だ幼稚な段階なので）幼児～児童の言語すなわち知の向上と, かなり一致すると思われる. 知識の量, 内部構造と性能～能力が有る程度正規化した資料を基に, 定量的に研究し評価できるので, 実験的な研究の手段にもなりえるのではなかろうか？

このような構造的な研究から見ると, 教育～知について本質的に重要なことは, 経験を単位的基礎的知識に分解して単位知識とそれを連ねる筋道を蓄積することであり, 他は常に抽象化機構を高度化する構造にあると思われる. 高度な知そのものの研究をしても本質は得られないのではないかと思う. 新人の採用では, 企業側は「やる気（積極性, 士気, 向上心）」を現在時点の学術能力以上に重んじるが, それは上の視点から見ると理にかなったように思える. 人の知～教育には, 何を見るか, の立場からの研究が必要ではなかろうか？

７．纏め

この報告は，ソフトウエアクリエーションプロジェクトの最終段階の研究を中心に,基礎的な面の概要と今後の発展を付加して記した.得た結果は以下のとおりであるである.

・人に倣ったソフトウエア自動設計の,既報の技能準拠に加えて,ルール準拠および知識準拠の全３階層のエンジンの知識構造を解明し，試作して設計の再現ができ,更に特性を定量的に調べた.
・各エンジン間のつなぎと協調動作の概要が見えた.
・これは当初の意図とおり人の作業に似た特性を示す.
・得られた結果は，広い範囲にかかわるであろう.

検討の初めにも記したように，この研究は大きな壁に開けた小さな穴に過ぎない．この穴を開ける鍵は，既存技術の支援がないから，勝手に編み出したもので，それは「如何に考え，如何に処するか」に尽きる．理論家の視点から見れば，「不完全！」であろう．「如何に考え，如何に処するか」とは，「少しでも人に近いシミュレータを作る」ことで，まだ完全は求めていない．常に実体と照合しながら，一歩一歩逐次近似を進めれば，次第に完全に近づく．この研究で判った道は未だに小さな穴ではあるが，種々の観点からこの道を辿れば，道は次第に大きくなり，各種の新しい知識と技術が育つ，と期待している.

この大きな厚い壁は何であろうか？
それは人間の知であり,Human Intelligenceと
呼ばれるものではないだろうか？

謝辞

この研究は，１９９１～２００１年の間，埼玉大学でソフトウエアクリエーションプロジェクトとして行われた.国立情報学研究所教授上野春樹博士には本報告の中心とした博士研究への貴重なアドバイスに感謝します.堤永保氏はじめ当時の日立中部ソフトウエア株式会社の関係各位，カルガリー大学助教授B. H. Far博士，研究室の学生諸君のプロジェクトへの貢献に深く感謝します．また，日立製作所，日立中部ソフトウエア株式会社(当時)，日本電信電話株式会社，沖電気工業株式会社，電気通信普及財団には，ご支援に厚くお礼申上げます.

文献

[Abolhassani00]Abolhassani H, Chen H., Far B. H., Koono Z., Software Creation: A Study on the Inside of Human Design Knowledge, Trans IEICE Vol. E83-D, No. 4, pp. 648-658, April 2000

[Abolhassani02] Abolhassani H, Chen H., Far B. H., Koono Z.,Software Creation: Cliche an intermediate knowledge during design, Trans. IEICE Vol. E85-D No.1, pp. 221-232, January 2002

[陳97] 陳　慧Far B. H.， 河野　善彌， ソフトウェア自動設計における系統的なエキスパートシステムの構築, ー設計工程からの設計知識の獲得と再現ー，人工知能学会誌 Vol. 12, No. 4, pp. 616-626,1997年７月

[Chen98] Chen H., Tsutsumi N., Takano H., Koono Z., Software Creation: An Intelligent CASE Tool Featuring Automatic Design for Structured Programming, Trans. IEICE Vol. E81-D, No.12, pp. 1439-1449, 1998

[Chen00]Chen H., Takano H., Abolhassani H. and Koono Z, Software Creation: An Integrated Environment for Automatic Software Design, Proc. of Fourth and Fifth World Conference on Design and Process pp. 992, 2000.

[Koono92a] Koono, Z., Baba, T. and Yabuuchi, T., Software Creation: A trial for switching software, Proc. of 1992 Joint Conf. on Communication network, switching and Satellite Communication pp.261-265, 1992

[河野92b] 河野, ファー，ソフトウエア開発工程の定量的取り扱い，信学技報，KBSE92-28， pp.33-40，１９９２

[Koono 93a] Koono, Z. and Fa,r B. H, Quantitative design of a software development process, Proc. of Fourth European Conf. for Software Quality pp. 173-181, 1994

[河野93b] 河野，大坪，ソフトウエアの誤りと除去の評価，情処学会研究会資料ソフトウエア工学95-5， pp． 31-38，1993

[Koono96] Koono Z., Chen H. and Far B. H, Expert's Knowledge Structure Explains Software Engineering, Proc. of Joint Conference on Knowledge Based Software Engineering 1996, pp. 193-197, 1996

[Koono01a] Koono, Z., Chen, H., Abolhassani, H. and Far, B. H., Design knowledge and Software Engineering, Proc. Ingternational Software Engineering Symposium 20001 (ISES '01), pp. 46-58, 2001.

[河野01b] 河野，Industrial Software Engineering〔産業界の期待)，情処学会研究会資料ソフトウエア工学33-8， pp． 55-62，2001

[師岡94] 師岡孝次，習熟性工学，建帛社，１９９４

[Pfeifer et al. 99] Pfeifer, R. and Scheier, C., Understanding intelligence, MIT Press, 1999.（訳　石黒他，知の創成ー身体性認知科学への招待ー，共立出版，２００１）

[Rasmussen85] Rasmussen, J., The role of hierarchical knowledge representation in decision making and system management, IEEE Trans. on Software Engineering, 18, 6, pp. 523-533, 1985

[Salvendy 82] Salvendy, G. ed., Handbook of Industrial Engineering, John Wiley & Sons, 1982.（訳　IE　ハンドブック，日本能率協会1986）

[山田他01]統合知的CASEツールに関する以下の３編.

１．山田，田代，陳，河野，ソフトウエアクリエーション：統合知的CASEツールのシステム

２．森本，須藤，陳，河野，ソフトウエアクリエーション：統合知的CASEツールの図面処理

３．張，吉田，陳，河野，ソフトウエアクリエーション：統合知的CASEツールの制御と評価，信学技報 AI2000-71, 72, 73, pp. 21-32, 2001

[Zipf72] G.K. Zipf, Human Behavior and the Principle of Least Effort," Hafner Publishing, 1972.