

ソフトウェアクリエーション：
ソフトウェア設計の知識の構造と自動設計への取組
Software Creation:
Structure of Design Knowledge and the Application to Automatic Design

河野 善彌, 陳 慧, 高野 英樹, H. アボールハッサニ, B. H. ファー
Zenya Koono, Hui Chen, Hideki Takano, Hassan Abolhassani and Behrouz H. Far

埼玉大学 工学部情報システム工学科
Department of Information and Computer Sciences, Saitama University

Abstract

This paper explains design knowledge for software design and discusses the application to the automatic design of software. A high maturity software development organization has been taken as the expert model, where their work process has been standardized. Their hierarchical work process shows the knowledge structure. These and other design knowledge may be acquired systematically from documents. Each knowledge substructure consists of a multiplicity of knowledge units ranging from skill level, rule based and so-called knowledge based. The last part discusses how the knowledge should be used for automatic design, and an example on an Intelligent CASE tool is discussed in detail.

1. はじめに

ソフトウェアの自動設計は永年研究され、現在では条件付きの自動設計は実現できる。筆者らは現状技術を更に伸ばすべく、如何なるソフトウェアでも自動設計を可能にする基礎を研究している。最近の研究を3編[1,2]で報告する。

取組のアプローチと研究戦略[3]を以下に示す。

- 1) 如何なるソフトウェアでも設計する人間に倣い、
- 2) その設計知識を系統的に獲得し、
- 3) 得た知識で系統的にエキスパートシステムに再構築して自動設計を行わせる。

この2, 3項は、かつて不可能とか困難とされた。現在ではモデル化[4]やソフトウェア工学の原理に従い構築すれば可能[3]である。筆者等は、

- 4) 高い成熟度組織を対象エキスパートにとり、その階層的な設計工程を対象知識モデルにとる。設計方式

は標準化されており、正しい設計図面が作成される。

- 5) 工程の入出力である図面等から知識を客観的かつ精密に獲得する。

- 6) 知識のモデル化と獲得をボトムアップかつ逐次的に繰り返し、人の設計を精密に調べる。

本論文は主に知識関係と自動化の考え方を説明し、第2の論文[1]は、上記の研究の成果である知的CASE (Computer Aided Software Engineering) ツールのプロトタイプを報告する。第3の論文[2]はボトムアップな戦略により次段階を目指す研究を中間報告する。

2. 高成熟度組織と階層的設計工程

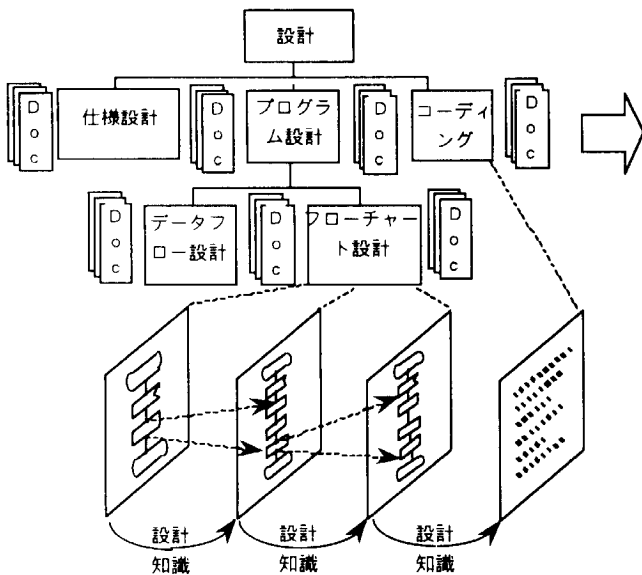
2. 1 高成熟度組織とその階層的工程

ソフトウェアの設計法には、トップダウンからボトムアップ、古典的Waterfallから現代的な諸方式等の幅広い諸設計方式がある。初めに基本的なトップダウンなWaterfallを研究し、その技術を基にしてより現代的な方式へ挑戦したい。

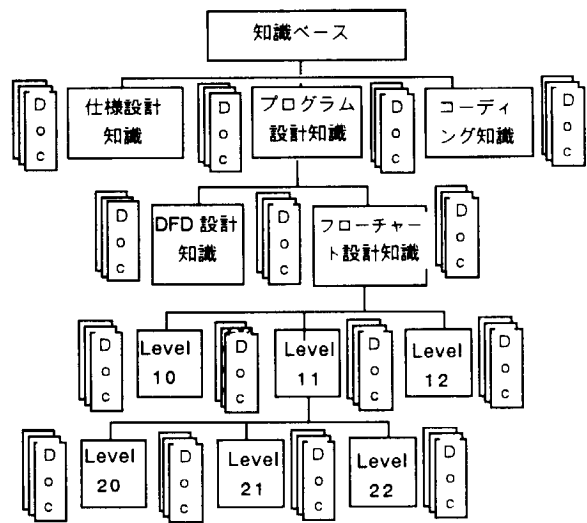
具体的な設計作業方法は人毎に、あるいはその都度、変わったりする。このランダム性を消す為、高い

連絡先 河野善彌 〒338-8570 浦和市下大久保255
電話： 048-858-3487
Email: koono@cit.ics.saitama-u.ac.jp

階層的な設計工程



再現した知識体系



成熟度の開発組織を対象エキスパートに取る。永年改善を積み上げ統一化（標準化）したから、工程は固化した知識体系である。成熟度が上がると各種の階層性が顕著になる。特に日本のメインフレーム系列の組織では、ハードウェア製造と同様な階層的工程¹を取り、各端面を設計文書で区切る。図1左はこれを示し、これを設計知識のモデルとする。

図の左下は詳細設計の小さな進行毎に図面を残しつつ作業した場合を示し、設計情報は階層的に展開する。階層的な工程は展開するにつれてより小さな作業になる。一方設計情報も階層的な展開を繰返し、展開する内に明確化／具体化／詳細化し、最後にソースコードに変換する。

後続論文[1]の為に特に文書関係作業の様相を説明する。人の言語体系は思考の体系と考えると厳しく教育する。ソフトウェア業務への入門時に、構造的日本語の文章表記などを教育する。多くのプロジェクトは発足時に主要なデータから始る用語等の標準化を行う。高成熟度組織では、上位者のレビューや担当者の相互チェックが常に行なわれ、誤りのみでなく人を誤らせる表現をも修正させる。これは設計者の論理的思考を育成し、正確で明解な文書を作成させる鍵になる。

2. 2 設計知識とその獲得

階層的工程とその文書図面を用いれば、設計の中心

的な作業である設計情報の変換を、変換前の入力と変換後出力の設計情報により、系統的に獲得できる。上層では、例えばあるプログラムの機能からソースコード迄、下層ではこれを実現する細かい変換群が図1下左のように取出せる。この方法は、

- ・獲得者によらず
- ・誰でも出来る
- ・確実である
- ・再現性に富む
- ・系統的に得られる

等の長所を持つ。

各工程は下位工程の作業の進行を管理する。幾つかの下位工程の完了と共に、上位工程は次に進む。図2は、かような変換を制御する状態遷移であり、これは工程図から正確に獲得できる。各種のプログラムが並行して作業されるが、それらの関係は「何時何が次の作業に使われるか」を示すPERT図に示される。そこでこれら両図から工程の作業進捗の知識が（前記と同様に）図面ベースで獲得可能である。

図3はこの工程毎の終了報告の収束を示す。終了報告は、予定期限と実績、検出誤り数等の作業基準に指定した項目から成立つ。工程の各階層毎に収束する網は、簡単にはAND機能の収束網に見える。しかし優れた管理者は報告から、微細な異常でも検知し問題を

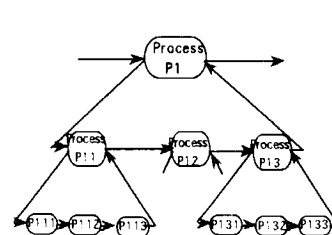


図2 状態遷移的な制御の知識

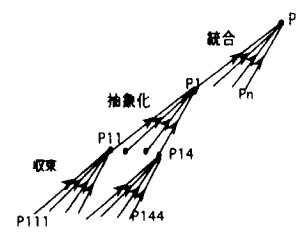


図3 終了報告の階層的収束網

¹ 階層的工程は、全体の作業を中間の各所で、更に下位の各中間点で、(文書図面作成基準により)縛る。そこで、作業方法は厳しく揃えられる。階層的なプログラム構成と工程構成とから「何の何処か」が解り、当該工程毎に科学的な計画と管理も向上できる。

発掘するから、これは数種の網から成立つ。この知識は組織や管理者に具体的な状況を指定して獲得でき、その本質は診断問題と同質である。ある種のプログラムでは実行時間等の指定があり終了報告は評価を要する。クリティカルな場合には、評価には経営工学的な判断/決定手法を用いる。従って高度な組織での作業をそのまま自動化するには、診断形や経営形エキスパートシステムと同じ技術を要する。工程概念を用いると対象モデルが明確になり、文書図面を中心に設計知識を図1右のように系統的に獲得できる。

次にエキスパートの特質である知識の多重性を説明する。設計作業の大部分は（外から見ると）簡単で、一部に手間を要する事があり、ごくまれには考込む事がある。人間工学のZipfは人の行動の「労力最小の原則」[5]を見出した。これは「人は問題を解く時に、最も簡単な方法から始め、駄目ならより高度へと、順次に高度化する」習性を言う。大部分の場合に固定的な知識機構で反射的に対応し、次第に高度で複雑な知的機構で対応すると考えられよう。知識工学ではRasmussenが「技能レベル」、「ルールベース」、最後に所謂「知識ベース」と言った区分を指摘[6]しているし、これらの場合の知識のレベルを言えば浅い～深い区分もできる。この多重性は終了報告でも触れたように、殆どの知識ブロックに多重性がある。

具体例を挙げると、図1左下の図の詳細設計の階層展開は、以下のように近似できる。

*基礎的な知識から新しい展開を創る機構と、

そのパターンを再利用する機構から成り立つ。

第2論文の知的CASEツールは後者のみを利用する。

2. 3 知的システムへの再構築

個別ソフトウェアの設計アルゴリズムを階層的工程に対応させて分割し、前記の共通知識構造と組合せれば、その「人に倣った自動設計システム」が作れる。しかし、精密に人を模擬する必要が常にある訳ではない。目的に向け最大効果を最小コストで達成させる近似を取る事が工学上の必要条件である。大きな（上層）工程の入出力関係から設計のアルゴリズムが得られるならそれでも良いし、図1の左下に示す詳細設計の下層の浅い技能的な知識を利用しても良い。

利用側からはシステム開発コストを上回る自動化による合理化効果を得る事が求められる。システム開発コスト、開発期間、危険度等を考慮しつつシステムの機能、性能と処理能力/速度、運用や柔軟性等を考慮して如何なる知識を用い、どんな実現方式を採るかを決める事が工学の行き方である。

3. 知的CASE ツール

これは、ソフトウェアを設計する貯めの絵書きツールであるCASE(Computer Aided Software Engineering) ツールに設計知識の自動獲得と自動設計を与え、プログラムの詳細設計工程に適用し、単純な展開作業を省力化する。技術の詳細は第2の論文[1]で説明するから、ここでは知識の問題を論じる。

詳細設計では単純な作業が多い。このような場合に産業界では、設計者に補助者を付け簡単なマニュアルに則り単純な詳細化作業を行なわせる事がよく行なわれる。これにより、設計者は中心的な業務に専念できる。本システムの発想は、補助者をエキスパートシステムで置き換える事である。この方式は、これまで説明した研究の具体化の第1段階で、高成熟度組織での標準化され誤りが無いか少ない図面を前提とする。このシステムの目的を下記に設定した。

*手軽で省力効果が大きいシステム、

段階的な自動化を前提として、常に省力効果/投資を最大化する。詳細設計は最大の工数（全体の1/2以上）を費す工程で、自動化による省力効果は最大である。作業は定型的で、技術的に最も難易度が低い。従って自動化に最も適した工程である。

*プロトタイプ開発後の定量評価の結果、10回目の使用では詳細設計工数は1/4に低下でき、意図したように省力効果は高い。

詳細設計の中でも階層展開作業が最も中心的であるから、これを自動化する。この為の知的な機構は（前述のように）

基礎知識から展開を創りだす機構と

使った展開パターンを蓄積利用する機構から成る、と考えられる。この場合に獲得すべき設計知識として、図1左下の図面毎の展開パターンを利用する。構造化チャートを設計図面として用い、設計知識の獲得は自動的に行なわせる。これらから運用に先立つ設計知識の準備実質0になる。これまでの自動設計システムは運用に先立ち設計知識等を事前準備するコストが大きい事であったから、これは大きな特徴と言える。

*手軽に使えるシステムが構築できた。

（これには、最近広く使われ始めた既存CASEツールを用い移行が容易である事も貢献する。）

（若干の作業は残っても）省力効果/投資を最大にするとか、手軽にする為既存の階層展開の再利用に限定する等のアクセントの強い方針は、各所に色々な影

響を与える。

このシステムは事前の知識は0で、使用される度に知識を蓄える。設計者が経験を重ねると速度が増すように、設計を繰り返す知識の量が増すほど自動化率が向上する。Industrial Engineeringの手法を用いて設計の様相の変化を定量評価し、下記を得た。

設計経験を増すと、設計知識量が習熟効果を示す。
この結果、自動化率や設計工数も習熟効果を示す。
(知識量の習熟効果が低いなら、後者も低くなる)

ハードウェアの組立て作業等では、作業者の知能が高い程習熟効果が大きい事が実例を通じて知られている。知的な作業でも当然同じである筈であるが、これまで確実な証拠が出ていないので、これが最初の技術的な裏付けある報告であろう。上記を設計者個人のレベルで当てはめると、既設計例を早く多く記憶する人はより早くに能力が向上する事を示す。また、組織に当てはめると、再利用し易い/標準的な設計をする程、早くに効率が上がる事でもある。簡単に言えば、技術が高い程、習熟は大きく省力化の度合いも高い。逆に成熟度が低い人や組織では効率は上がりにくくその度合いも少ない。その度毎に出鱈目な設計をすると知識ベースのIntegrityが保たれず全く効果がない。

ソフトウェア開発組織の習熟度は、いわば小学生、中学生、高校生、大学生にあたる。代数は中学生には有用でも小学生は使えない。微積分は大学生には効果的であっても中学生には有害になりかねない。知的なツールの効果は使用者のレベルに依存する。

このシステムは設計済みの展開の知識の利用に留めたから、以下が必要になった。

*人が概念の展開を行なう。

*展開の巨視的な整合性は人がチェックする

例 展開する方向の正しさの確認と選択や修正データ等の使用法が一致する事のチェック

これらは、元来設計者が当然行っていた作業で、これらの人の作業は効果の定量評価に織込み、総合的な省力効果が大きいから欠点とは言えない。しかし、今後の課題である事は確かであり、現在次段階の研究として取組んでおり、第3の論文[2]で中間報告する。

また、現状では一字違ってても合致する展開が得られないが、これは現実には厳し過ぎる条件といえよう。パソコンの普及と共に、自然言語による検索にある程度の曖昧さを許容できる方式が進展し始めている。この技術等を用いて対処する事が妥当と考えている。

4. 結び

本論文では、人の設計に倣い如何なるソフトウェアでも設計できる基礎の研究を報告した。理想的で固化し標準化された知識体系を対象とし、下記のように取組んでいる。

*知識構造をモデル化する。

* (人の理解に頼らず) 極力図面を用いて知識を (他の工学に於けると同様に厳格に) 入出力関係等から正確に獲得する。

*逐次近似によりモデルや知識を精密化する。

*ボトムアップに、漸進的に、研究し実用化する。

*利用システムの構築には、工学の原則に従う。

以上の概観に引き続き後続論文で具体的に報告する。

謝辞

1991年以来の本研究に携わった多くの学部学生および大学院学生の諸氏、ご討論頂いた各位、ならびに各種のご支援を下さった各位、各社に感謝します。

参考文献

1. 陳 慧, 高野 英樹, 堤永保, 河野 善彌, ソフトウェアクリエーション: 詳細設計用の知的CASEツールのプロトタイプ, 1998年度人工知能学会大会, 1998.
2. H. Abolhassani, 陳 慧, 高野 英樹, 河野 善彌: ソフトウェアクリエーション: 詳細設計用の各種の知識, 1998年度人工知能学会大会, 1998.
3. 陳 慧, Behrouz H. Far, 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, 一設計工程からの設計知識の獲得と再現一, 人工知能学会誌 Vol 12, No 4, pp. 616-626 1997, 7.
4. K. M. Ford and J. M. Bradshaw eds., Knowledge acquisition as modeling part 1, International Journal of Intelligent Systems, Vol. 8, No. 1, 1993.
5. Zipf G.K, Human Behavior and the Principle of Least Effort, Hafner Publishing (1972).
6. Rasmussen, J.; The Role of Hierarchical Knowledge Representation in Decision Making and System Management, IEEE Trans. System, Man, Cybernetics, vol. SMC-15, no. 2, pp. 234-243, Mar./Apr, 1985.