ソフトウェア工学 80－17
（1991. 7. 19）

設計プロセスの蓄積・利用による設計支援法について

浜田 雅樹　安達 久人　竹中 豊文
ATR 通信システム研究所

コンピュータに問題領域毎の設計ノウハウを蓄積し、利用する方法について考察する。本論文では、まず、ソフトウェアの設計ノウハウは、設計対象が持つ、設計上考慮すべき制約と関連が深い事について述べる。次に、これらの情報をコンピュータ化する具体的な枠組について提案する。本手法では、設計者が設計時に記録した設計プロセスを集め、設計対象の性質毎の設計バリエーションとそれらの間にある設計上の因果的な関係から構成された設計事例ベースを構築する。これを用いて、設計者に対して、設計時に考慮すべき設計対象の性質やその設計結果の事例を設計ノウハウに関する情報として提示する。

Towards Design Support Using Design Processes

Masaki Hamada　Hisato Adachi　Toyofumi Takenaka
ATR Communication Systems Research Laboratories
Sanpeidani, Inuidani, Seika-cho, Soraku-gun, Kyoto 619-02

A method of storing and applying design know-how in problem domain by computers is discussed. First, authors describe the relation between the design subject constraints which are crucial for design and design know-how. Second, the method of computerizing this information is discussed. In this method, the design processes database, which consists of causality relationships among design variations for each design subject characteristics, is built by gathering recorded design processes. Using the design processes database, design subject characteristics to be considered and design result examples are indicated to a designer when he/she designs software.

1. はじめに

ソフトウェアエンジニア(SE)の不足や異動等によるソフトウェア生産知識の空洞化は、ソフトウェアのメンテナンスや生産における品質維持等に深刻な問題を引き起こしている。通常、ソフトウェア開発現場は独自の設計ノウハウを持っている。ソフトウェアの場合、作業標準などの基礎技術が確立されていないため、設計ノウハウはエンジニアが持っているケースがほとんどである。これが、上記の問題点を引き起こしている要因の一つと考えられる。このような問題点を解決する方法として、コンピュータに設計ノウハウを蓄積し、利用する方法が考えられる。これが実現すれば、エンジニアによらない品質や生産性の安定化等が可能になると考えられる。

今までは、問題領域毎にドメイン・アナリシス[22]を行い、それに基づく設計支援系が開発されてきた[12][21]。しかし、これらが基づいている単純な処理のパターン化でカバー出来る問題領域は限られている。

人間のエンジニアは、このように単純なパターン化が出来ない問題領域に対しても、経験からノウハウを構築することが出来る。現在の技術で、コンピュータ自らがノウハウを収得するような高度な学習機能を実現するのは困難である。しかし、エンジニアが収得した設計ノウハウをコンピュータに入力し、他のエンジニアが必要なときに参照できるようにすることは可能であり、また、意義があると考えられる。

本論文では、まず、ソフトウェアにおける問題領域毎の設計ノウハウとして何が重要であるかについて検討を行う。次に、設計ノウハウをコンピュータ化する為の技術について考察する。最後に、以上の検討を踏まえ、設計ノウハウをコンピュータ化する具体的な手法として、設計者の設計プロセスから設計ノウハウと関連が深いと思われる情報を獲得し、以降の設計で利用する枠組に付いて提案し、さらに将来の課題について考察する。

以下、要求分析フェーズの支援技術について議論を進める。これは、要求分析フェーズに対する研究が下流工程に比べ進んでおらず、エンジニアが持つノウハウへの依存度が高いと考えられる等の理由による。

2. 問題領域におけるノウハウ

問題領域におけるノウハウとは何かを解明する場合、エキスパートと非エキスパートの設計活動を比較分析することは有効な方法である。

Adelson[15]らは、ある特定の問題領域の知識を持つ設計者(エキスパートと呼ぶ)と持たない設計者(非エキスパートと呼ぶ)とに対して、その問題領域のソフトウェアの設計活動について比較している。その報告の中で、非エキスパートはエキスパートと比べ、設計対象の(ふるまいに関する)シミュレーションが出来なかった ことを述べている。シミュレーションは、設計者が持つ設計対象のメンタルモデルを用いて行われる。そのメンタルモデルは、設計者が設計対象に関連した制約を認識することにより構築されることから、設計対象が持つ制約を設計者に提示する設計支援機能などの必要性を述べている。

以上から考えると、問題領域の設計ノウハウは、設計対象が持つ、設計上考慮すべき制約の認識技術と関連が深いと考えられる。例えば、エディタの2つの機能の間の制約として、現在メモリ上にある文書の内容が未保存の場合、プログラム終了機能実行前に、ファイルセーブ機能を実行する等が考えられる。この制約は、エディタプログラムの機能構成を設計する場合などに考慮しなければならない。エディタという問題領域に馴染みのないエンジニアはこのような制約を認識する能力が弱いというのである。本論文では、設計対象が持つ、設計上考慮すべき制約が問題領域の設計ノウハウと関連が深い重要な情報と考え、その情報をコンピュータに蓄積し、活用する方法について検討する。

3. 設計ノウハウをコンピュータ化する為の技術

(1) 論理的基盤がある問題領域

論理的基盤がある機械などの設計を支援する場合、定性推論の技術が有効である[17]。これは、機械の論理を計算機に移植することにより、設計の対象の振舞いを計算機でシミュレートすることが可能となるためである。しかし、現実的には、このように論理的基盤がある問題領域は限られている。

(2) 論理的基盤が弱い問題領域

論理的基盤が弱い問題領域では、SEが持つ設計ノウハウに頼っているため、SE不足や異動に伴う維持管理上の問題を抱えており、支援技術の確立が望まれる領域である。その支援技術としてはエキスパートシステムがある。従来型のエキスパートシステム構築技術は、人間のエキスパートが持

つノウハウを人間がインタビューし、ルール型知識等にコーディングするといったものである。これは、知識(ノウハウ)獲得の難しさ、学習機能の欠如、新しい問題への適応能力の低さ等の問題点[11]によりごく限られた問題にしか適用されていない。そのような問題点を解決するための技術として、CBR(Case Based Reasoning, 事例推論)技術が注目され研究されている[10][11][19]。これは、法律等の分野で判例が重視されているように、人間の経験を利用して問題の解決を図る技術である。

本研究では、エキスパートの設計の事例である設計プロセスを見本としてコンピュータに格納し、他の設計者が別の設計において参照し、そこで利用されている設計ノウハウを理解、活用する方法を検討する。その実現に当って、以下の3つの主要課題がある。

1) 格納すべき情報の明確化
2) 適用できる事例とその箇所の検索手法の確立
3) 設計者が設計ノウハウを理解、活用しやすい形で提示する方法の確立

本論文では、以上のうち、特に1)及び2)について検討する。

筆者らは、ソフトウェアの設計やメンテナンス時の要求や仕様の修正に対し、その影響波及の解析に有効な情報を設計プロセスより獲得する方法について研究を進めている[17]。要求分析は、設計対象の持つ諸性質とその関係を明確にする作業であると考えられる。即ち、設計対象の性質間にある制約を認識・充足していく作業と言うことができる。影響波及解析の研究では、どんな設計対象の性質を考慮しながらある設計対象の性質を設計したかという情報が、影響波及解析において有効となることを示した。この情報において、"考慮しながら設計した"というのは、設計対象の性質またはそれらの構成要素間に制約を設計者が認識したからである。2章で述べたように、設計対象に対する設計者のメンタルモデルを構築するには、設計対象に関する制約を認識する必要がある。従って、この影響波及解析において有効となる情報は、問題領域の設計ノウハウと大きな関係があると言える。従って、ソフトウェアを設計する場合にも有効な情報となることが期待できる。本論文では、設計プロセスより上記の情報を獲得し、問題領域の設計ノウハウとして以降の設計で利用する方法について考察する。以下、まず格納すべき情報の明確化の為、設計プロセスモデルについて説明する。次に、そのモデルに従って記録した設計プロセスの設計事例ベース及び、それを用いた設計支援の方法について、特に適用できる事例とその箇所の検索手法等を中心に述べる。

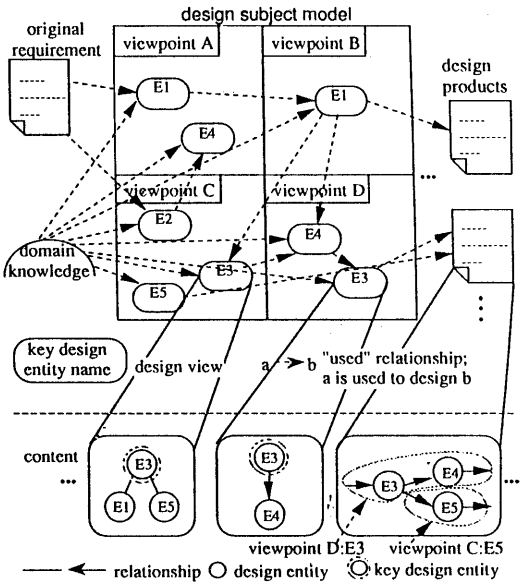

Fig.01 ソフトウェア設計プロセスモデル

4. ソフトウェア設計プロセスモデル

筆者らが提案しているメンテナンス支援手法で用いる設計プロセスモデル[17]について説明する。

設計プロセスを、設計対象モデル(design subject model)の構築と設計生産物(design product)の作成プロセスと定義する(fig.01)。

設計対象モデルは、設計の対象に関する諸性質を設計した(1)設計ビュー(design view)と、設計時における設計ビュー間の(2)利用関係("used" relationship)から構成される。

(1) 設計ビュー

モジュールを分割する場合、まずそのモジュールを、インターラクション、呼びだし関係、処理ケース等の視点から設計を行う。このように、ソフトウェアを設計する場合、設計対象に関するさまざまな性質を設計しなければならない。設計者は、設計対象に関する性質を、ある特定の視点(viewpoint)から、ある特定の部分に着目して設計する。例えば、ある特定のモジュール(特定の部分)の'呼びだし関係'(視点)の設計がその例である。このように、設計者が一度に扱う設計対象の性質は比較的小さいが、それは人間の認知的能力の限界からくるものと考えられる。この限られた大きさを持つ設計対象の性質を設計ビューと呼ぶ。

例えば、"X店の在庫管理は入庫処理と出庫処理

よりなる"という設計結果を記述した設計ビューは以下の属性を持つ。

- 視点(viewpoint) = 機能構成 (この設計ビューが設計されている視点を表す)
- キー設計エンティティ(key design entity) = X店の在庫管理(この設計ビューがどのような設計エンティティに対して設計を行ったものかを表す。設計エンティティとは、システム、関数、データ、モジュール等、ソフトウェアの各抽象レベルでの設計を構成する要素を表す)

  (以上2つの属性により、この設計ビューが、設計エンティティ: "X店の在庫管理"について視点:'機能構成' から設計した結果である事を表している)
- 記述されている内容(content) = 3つの設計エンティティ、X店の在庫管理、入庫処理、出庫処理 間のpart-of関係(設計ビューは、設計対象の性質として、設計エンティティ間の関係を記述する)

(2) 利用関係

設計ビューXの全体または一部が、設計ビューYを用いて(基づき)設計された場合、設計ビューXとYの間に"利用関係"があると呼ぶ。

全ての設計ビューは何等かの領域知識(domain knowledge)、例えばタスク固有の知識やプログラミング技術等、を用いて設計される。幾つかの設計ビューは更に原要求(original requirement)を用いて設計される。原要求とは、要求者が見たソフトウェアに対する要求の記述を指す。例えば、原要求の記述に基づき設計ビュー、"X店の在庫管理は入庫処理と出庫処理より構成される"、等を設計する。このように、領域知識と設計ビューの間、原要求と設計ビューの間にそれぞれ"利用関係"がある。

また、設計者は、設計対象モデルの設計ビューを利用して設計生産物を作成する。従って、設計ビューと設計生産物の間に"利用関係"が存在する。

設計ビューを用いたソフトウェア設計プロセスとして以下のステップを想定する。

1. 設計者は原要求を読み、設計ビューを設計する。この設計ビューは、設計対象に関する最も抽象的な記述であり、例えば、"X店の在庫管理は入庫処理と出庫処理より構成される"といった内容である。

   以下のステップはある設計目標が達成されるまで繰り返し行われる。
2. 設計者は設計対象モデルの設計ビューを設計する。
3. 設計者は、設計対象モデルがあるレベルに達した段階でそれらを用いて設計生産物を作成する。

以上のモデルに基づき、設計プロセスとして以下の情報を記録する。

1. 設計対象モデル --- 特定の視点から見た設計対象の性質を記述した設計ビューと、それらの間の設計時における利用関係
2. 設計生産物
3. 1 と 2 の間、1と原要求間の設計における利用関係

(注)領域知識と1 との間の利用関係は、領域知識のデータベース化が困難な為、現状では記録していない。

5. 事例ベース

複数のソフトウェア設計プロセスを集めたものを設計事例ベース(以下単に事例ベースとも呼ぶ)と呼ぶ。事例ベースのモデルをfig. 02 に示す。事例ベースは、前述のfig.01に示した設計プロセスモデルと以下の2点が異なる。

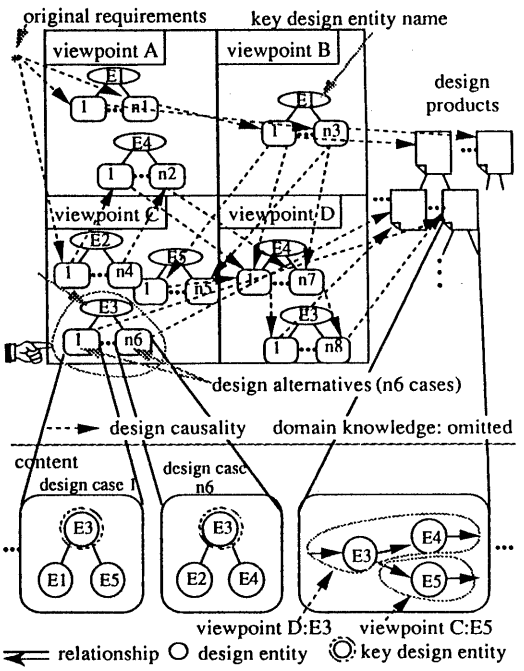

Fig.02 事例ベースモデル

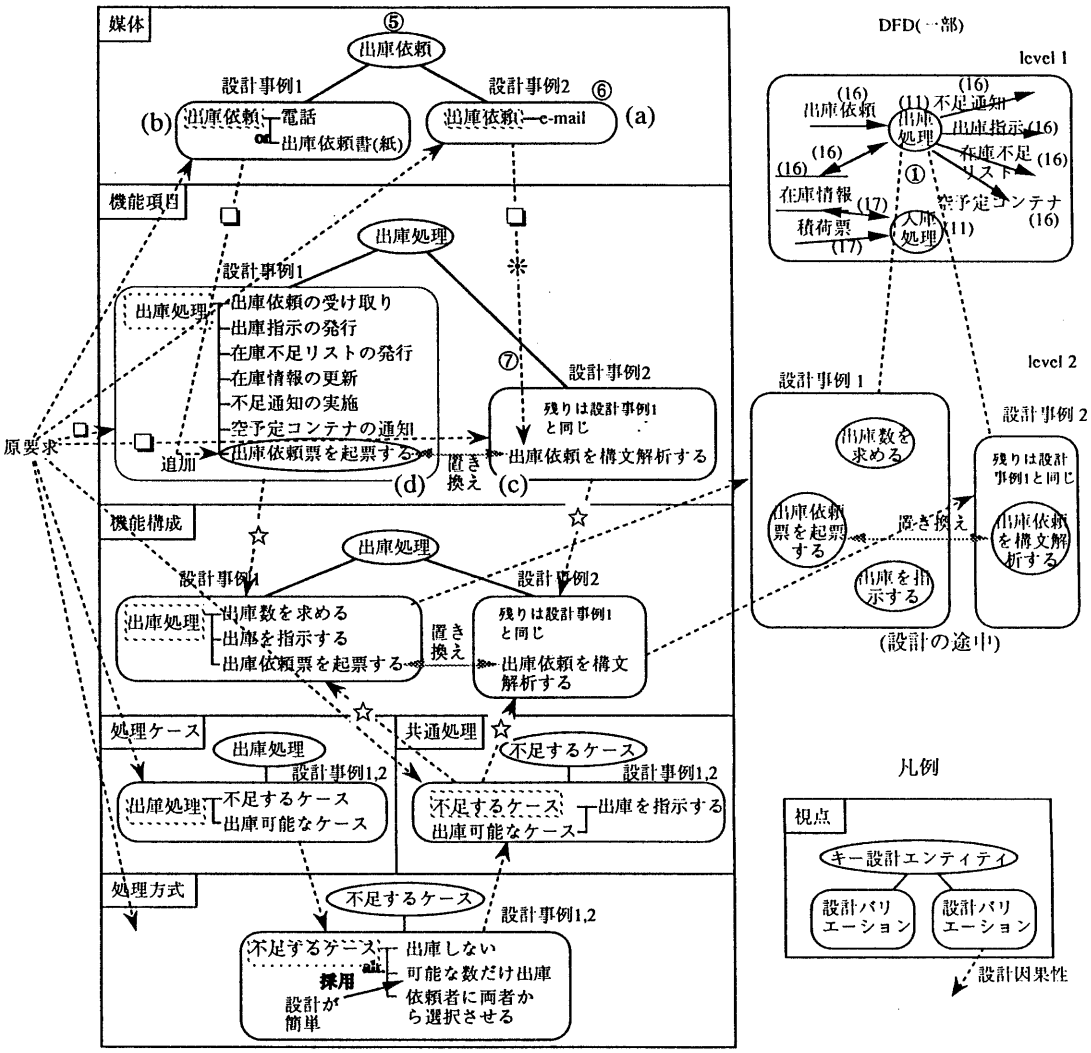

Fig.03 事例ベースの具体例

1. 事例ベースでは、1つの設計プロセスにおいて得られた設計ビューは、複数の設計事例の中の設計バリエーション(design alternative)として位置付けられる。例えば、fig.02の☞は、設計エンティティ"E3"の視点'viewpoint C'からの設計がn6通りあった事を示している。
2. 事例ベースでは、1つの設計プロセスモデルにおいて得られた設計ビュー間の"利用関係"は、設計バリエーション間の関係(設計因果性: design causalityと呼ぶ)として位置付けられる。

具体例をfig.03に示す。本例は、在庫管理プログラムの2つの設計事例を格納した事例ベースの一部である(以下、"設計エンティティ"の'視点'という表記を用いる)。例えばfig.03において、"出庫依頼"の'媒体'の設計バリエーションが、電話や出庫依頼書(紙)による場合(図中b)と、e-mail(電子メール)による場合(a)との違いで、"出庫処理"の'機能項目'に対して、"出庫依頼票を起票する"という機能項目を設計(厳密には後から追加)したか(d)、"出庫依頼を構文解析する"を追加したか(c)に各々分かれることを表している。

設計事例ベースは、新たな設計において、格納されている事例との差分を設計者が入力して行く事により構築される。即ち、設計者は、後述の設計支援機能を用いて設計バリエーションの中から今回の設計に適したものを選択する。適したもの

がない場合は、設計ビューを新たな設計バリエーションとして入力する。詳細は設計支援法の所で述べる。

6. 設計支援機能

4章で述べた設計プロセスモデルに基づけば、設計は、以下の作業からなる。

① 次に設計する設計対象の性質(キー設計エンティティ, 視点)を決定する。

② その設計を行う上で考慮すべき設計対象の性質を認識する(設計上考慮すべき、設計対象に関する制約を認識する)。

③ それらを考慮して、設計対象の性質を設計する。

前述の通り、特に問題領域の設計ノウハウが必要とされるのは、作業② であると考えられる。

前述の事例ベースには事実として、

1) "出庫処理"の'機能項目'を設計する場合、"出庫依頼"の'媒体'に関する設計結果を考慮した、

という情報が格納されているが、その背景には、

2) "出庫依頼"の'媒体'と"出庫処理"の'機能項目'との間には何等かの制約がある、

3) その制約は、"出庫処理"の'機能項目'を設計する場合考慮すべきである、

が暗示されている。従って、設計作業②に対して、本設計事例ベースを用いて、1)の情報に関した、「設計対象の性質を(ある設計エンティティをある視点から)設計する場合、他のどんな設計対象の性質を考慮すべきか」を指示する事により、非エキスパートでは困難な、設計上考慮すべき制約の認識作業を支援することができる。更に、作業③に対しては、設計結果の事例表示による支援ができる。作業①についても、視点やキー設計エンティティのブラウジングによる支援等が可能である。

以下、fig.03の具体例を用いて設計支援のイメージを示す。現在、データフロー図(以下DFD)のlevel1まで設計が終っているとする。設計者は、次に、level 1 DFDの各processに関するDFD(level2)を作成しなければならない事を知っている。設計者は、その中から出庫処理について設計を進めようと考えたとする。

設計者がlevel 1のDFD上で"出庫処理"を選び、設計を進める事をシステムに宣言すると(fig.03 ①)、システムは、事例ベースにある、"出庫処理"について設計されている視点の一覧を提示する(fig.04

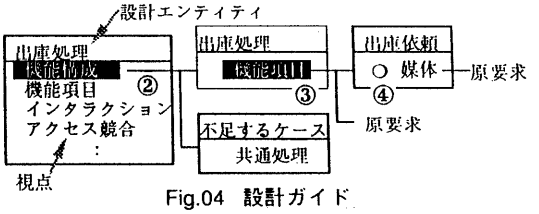

Fig.04 設計ガイド

②)。ここでは、設計者はまず、DFDを設計する上で必要となる視点:'機能構成'を設計するため、その中から'機能構成'を選んだとする。

システムは、事例ベースにおいて"出庫処理"の'機能構成'の設計バリエーションに対して設計因果性を持つキー設計エンティティ, 視点を見つける(この例では fig.03 ☆の設計因果性より"出庫処理"の'機能項目'と"不足するケース"の'共通処理'が見つかる)。システムは、この結果を用い、設計者に対して、"出庫処理"の'機能構成'を設計するには、"出庫処理"の'機能項目'と"不足するケース"の'共通処理'についての設計結果を考慮する必要があり、両者ともまだ設計されていない事を示す。システムは、事例ベースに存在するキー設計エンティティ, 視点に対して今回の設計ですでに設計が行われたかどうかを管理している。

設計者は、そのうち"出庫処理"の'機能項目'をまず設計することに決定したとする(fig.04 ③)。システムは同様にして"出庫依頼"の'媒体'についての設計結果を考慮する必要がある事(fig.03の□ 設計因果性より)を示す(fig.04 ④)。事例ベースでは、"出庫依頼"の'媒体'は、原要求からのみ設計因果性を持っている為、fig.04④ではこれが設計すべき状態(設計する上で考慮すべき情報が設計済みまたは与えられている)であることを示している(図中の○)。

システムは、設計者に、原要求を見ながら"出庫依頼"の'媒体'を設計するよう提案する。その際、fig.03 ⑤にある設計バリエーションを提示しその中から選ばせる。適切な物がなければ、新たに入力させる。新しく入力された設計ビューは、⑤に対する3つ目の設計バリエーションとして設計事例ベースに追加される。

もし、設計者が、今回の設計では出庫依頼がe-mailで行われるとして、fig.03における⑥を選んだとする。システムは、"出庫依頼"の'媒体'に対する設計が行われた事から、fig.04 ③("出庫処理"の'機能項目')が設計すべき状態になった事を判断し、設計者に示す。設計者がその設計を行う事をシステムに入力すると、システムは、先ほど設計した"出

庫依頼"の'媒体'の設計結果(fig.03 ⑥)を考慮すべき設計対象の性質として表示し、それを参考に設計を行うよう指示する。また、考慮して設計した結果の例として、機能項目の1つである"出庫依頼を構文解析する"を表示する。これは、設計事例ベースのfig.03の⑥の設計バリエーションから⑦の設計バリエーションに対して設計因果性があることを利用している(fig.03 ※)。更に、原要求を見ながら'機能項目'を設計するように指示する。以下略。

このように、本支援機能では、例えば、"出庫処理"の'機能項目'を設計する際、"出庫依頼"の'媒体'を考慮しなければならない事と考慮した設計結果の事例を提示する。

7. 考察

本研究は、「設計に関するどんな情報を残せば、その問題領域の非エキスパートでもエキスパートと同じように設計を行う事ができるか」という課題に対する初期的なものである。プログラミングを勉強する場合、「まず良いプログラムを読め、次にそれを真似てプログラムを書け」という事が良く言われる。本研究では、それを上流工程に対してシステマチックに行う事を目指しており、システム化に向けて、以下の問題点を克服する必要が在る。

① 多くのコーディングノウハウはソースコード上に記述されている。しかし、上流工程では設計生産物に設計ノウハウ的な情報があまり記述されていない。

② ソースコードの場合もそうであるが、生産物から設計ノウハウ的な情報を理解・推測するのは大変な作業である。

③ 勉強という形で何でも良いから学ぶのではなく、設計中に直面している問題に対して活用できる事例を適宜見つけたい。

以上を解決するため、3章で述べた1)～3)の課題の検討が必要となる。

本論文では、設計ノウハウとして、設計者が着目した設計対象の性質が重要であると考え、それを設計者の視点という形で記録・蓄積し、以降の設計で利用する方法について検討した。これには、未検討である3章での課題3)を含む以下の解決すべき課題がある。

(1) 理解

今までの検討では、設計支援機能で設計者に提示する情報は、考慮すべき設計対象の性質と、それを考慮した設計の例だけであり、これだけでは設計者は設計ノウハウを理解しにくい。これは、提示された情報に、なぜその設計対象の性質を考慮しなければならないのか、またどの様に考慮すれば良いのかに関する情報がないためである。今までは、問題領域に関する知識はなくても、一般的な設計知識を持っていれば、設計上考慮すべき設計対象の性質やその設計結果から、これらの情報は推測可能であるという前提の基に検討を進めてきた。しかし、幾つかの記述実験から必ずしもそうとは言えないことが明かになってきた。今後は、設計者の設計プロセス記録の負担をできるだけ押えながらこれら不足する情報を獲得する方法や、複数の設計事例を用いた説明機能等の、設計ノウハウを理解しやすく提示する方法を検討する必要がある。

(2) 記録の問題

本研究は、設計中に記録すべき情報を明確にするのが目的であるが、その結果を用いる際、エキスパート設計者にその情報の記録を強制する必要がある。これは多くの作業標準などが抱えている問題点でもあるが、そのことによりエキスパートの優れた設計活動を平均化してしまう危険性がある。

(3) 事例ベースの整理

事例の数が増大した場合、事例の一般化等による設計事例ベースの規模の増大の防止や、矛盾点の検出、解決方法、どんな単位で設計事例ベースを分けるべきか等を検討する必要がある。

謝辞

日頃ご指導を頂く葉原耕平会長、山下紘一社長に深謝します。また、プロトタイプシステムの開発にご協力いただいた日本電子計算(株)の浜中氏、吉岡氏、辻井氏に深謝します。

参考文献

[1] L. Osterweil, "Software Processes Are Software Too," Proceedings of the Ninth International Conference on Software Engineering, 1987

[2] Lloyd G.Williams, "Software Process Modeling : A Behavioral Approach, " Proceedings of the Tenth International Conference on Software Engineering, 1988.

[3] Donald E. Knuth, "Literate Programming," The Computer Journal, Vol.27, No.2, 1984

[4] Jeff Conklin, Michael L.Begeman,"gIBIS: A Hypertext Tool for Exploratory Policy Discus-

sion," ACM Transactions on Office Information Systems, Vol.6, No.4, October 1988

[5] Colin Potts and Glenn Bruns, "Recording the Reasons for Design Decisions," Proceedings of 10th ICSE, 1988

[6] Zheng-Yang Liu, Arthur M.Farley, "Shifting Ontological Perspectives in Reasoning About Physical Systems," AAAI-90, 1990

[7] Brian Falkenhainer, Kenneth D.Forbus, "Setting up Large-Scale Qualitative Models," AAAI-88, 1988

[8] Vasant Dhar, Matthias Jarke, "USING TELEOLOGICAL DESIGN KNOWLEDGE FOR LARGE SYSTEMS DEVELOPMENT AND MAINTENANCE," Proceedings of the International Workshop on Expert Systems & Their Applications, 1986

[9] 仲谷 善雄、築山 誠、福田 豊生, "事例ベースアプローチによる設計支援-設計支援システムSUPPORTにおける事例の再利用," 情報処理学会人工知能研究会報告 vol.91, No.16, 1991

[10] D.Navinchandra, "Case Based Reasoning in CYCLOPS, a Design Problem Solver," Proceedings on CASE-BASED REASONING WORKSHOP (sponsored by DARPA), May, 1988

[11] Stephen Slade, "Case-Based Reasoning: A Research Paradigm," AI Magazine, Spring, 1991

[12] Mitchell D.Lubars, "The IDeA Design Environment," Proceedings of 11th ICSE, 1989

[13] John H.Boose, Brain R. Gaines、"Knowledge Acquisition for Knowledge-Based Systems: Notes on the State-of-Art," Machine Learning, 4, 377-394, 1989

[14] Howard B.Reubenstein, Richard C.Waters, "The Requirement Apprentice：Automated Assistance for Requirements Acquisition," IEEE Transactions on Software Engineering, Vol.17, No.3, March 1991

[15] Beth Adelson, Elliot Soloway, "The Role of Domain Experience in Software Design," IEEE Transactions on Software Engineering, Vol. SE-11, No.11, 1985

[16] Seshashayee S. Murthy, Sanjaya Addanki, "PROMPT: An Innovative Design Tool," AAAI-87, 1987

[17] 浜田, 竹中, "設計履歴を利用したソフトウェア設計・保守支援方式," ソフトウェア・シンポジウム'91

[18] James M.Neighbors, "The Draco Approach to Constructing Software From Reusable Compo-