ソフトウェア工学　81－4
（1991．10．4）

ソフトウェア設計における設計履歴情報の蓄積、活用法

島　健一

NTT ソフトウェア研究所

本報告では、設計判断履歴情報に関する研究の背景、系譜、研究事例の紹介、応用例、課題などについて調査、検討した結果を述べる。

ソフトウェア設計の際に、設計過程での判断、問題点、判断の根拠などを視覚的な設計判断履歴情報として記録し、設計レビューなどに再利用する研究、実験的試みが国内外で行なわれている。

設計判断履歴情報は、社会的背景、技術的な観点からも必要性が増している。たとえば、ソフトウェア技術者の不足などの社会的状況、ノウハウの継承などの技術的問題、設計レビューの検討漏れを解消する手段として有効であると考えられる。

設計判断履歴情報を記録、蓄積、再利用を支援するシステムは、設計過程における「なぜその設計を行ったのか」という情報に着目する。また、その構築においては、どのようなモデルで記録、蓄積するかが重要な課題となる。

Report on Design Decision Rationale Accumulation and Reuse

Ken-ichi　Shima

NTT Software Laboratories

9-11 Midori-Cho 3-Chome Musashino-Shi, Tokyo 180 Japen

This report presents a survey of “design decision rationale accumulation and reuse” in software design process. It describes a background, a trend, some case studies, some applications of design decision rationale research, and describes a evaluation of some system which support design decision rationale from the functional point of view, for example, consideration of model, data structure, and accumulation technique, utilization domain. And this report also presents a problem to be solved.

Design decision rationale is a visualized information of decision, problem, justification between deliberation process and product, or artifact.

These rationale may be used for many applications, for example, expertizing design knowledge, or know-how of designer and transferring its knowledge to another designer, or inexperienced designer. And it may be effective for finding some oversights in design review.

1 はじめに

ソフトウェア開発における設計判断履歴情報 (Design Rationale) とは、「ソフトウェア設計の際に、設計過程での判断、問題点、判断の根拠、設計生産物との関係などを視覚的に記録した情報」のことを言う。

現在までに、ソフトウェア設計過程において積極的にこの設計判断履歴情報を記録、蓄積し、後で活用しようとする研究、実験的試みがいくつか行なわれている。その主な活用法としては、設計過程の再利用、仕様の精密化、知識の獲得などがある。また、社会的背景、技術的な観点からも設計判断履歴情報の必要性が増している。たとえば、ソフトウェア技術者の不足などの社会的状況、ノウハウの継承などの技術的問題、設計レビューの検討漏れを解消する手段として有効であると考えられる。

設計判断履歴情報は、設計過程における「なぜその設計を行ったのか」という情報に着目する。また、それを支援するシステムはどのような構造、モデルで記録、蓄積、利用するかが課題となる。本報告では、設計判断履歴情報に関する研究の背景、系譜、研究事例の紹介、応用例、課題などについて調査、検討した結果を述べる。

2 背景 − 設計判断履歴情報の必要性

ソフトウェア設計では、問題の分析と判断が連続的に行われている。また、ソフトウェア設計時の意志決定パターンには以下のような特性があることが指摘されている[29]。

(1) 設計途中では、多くの理由により、各種の中断が起きる。中断理由には、たとえば、設計内容に矛盾が生じた、役に立たなくなった、不足箇所や検討漏れが生じた、などがある。

(2) 問題を分割していく過程が実際の設計生産物とどのような関連があるか、たとえば、矛盾はないかなどを調べながら判断していく。

(3) ある候補案の実現が失敗した時、別の設計案を選んで設計を進める。

(4) 設計が行きづまった場合、また、設計内容を変更しようとする時は、適切なポイントまで戻り設計し直す。

(5) 設計生産物と実現手順の間の関係を整理する。

このような意志決定過程を支援して行くためには、ソフトウェア設計の際の設計者の判断過程の履歴、つまり設計過程での判断、問題点、設計生産物との関係などを視覚的に記録した情報が必要となる。このため、従来型のプログラム環境、設計環境からの脱却をめざした試みがなされている。

従来型の設計環境では、設計過程の問題の分析過程、設計過程における判断情報が失われる。その理由としては、以下のことが考えられる。

(1) 設計上の判断過程の表現が弱い。
(2) 特定の判断に基づく設計上の変化の記述が不備である。
(3) 要求とそれに基づく選択の関係が不明確である。

この他にも、設計判断履歴情報が必要な理由としていくつかの要件が挙げられる。

社会的背景としては、以下のようなことが挙げられる[23]。これは、おもにソフト開発現場からの要請、および技術者不足から来る原因である。

- 設計知識の蓄積が不十分である。
- 設計ノウハウを他者に移転、伝達できない。
- ソフトウェア技術者教育がうまくいかない。
- 設計工程 (意思決定過程) を支援するツールがない。
- 維持管理 (機能追加、変更) 時の問題が大きい。

また、時代の進歩、研究の進展につれて以下のようなことが考えられる。

- 集中から分散へ
  これには、地域的、機能的、ワークステーションの出現などがある。分散化した情報をいかに有効に記録、蓄積するかが課題となる。
- AI アプローチの適用
  設計ノウハウの蓄積、教育的観点から知識ベースとの関連、知識重視の傾向が見られる[16]。さらには、より認知的 (メンタル) な考察を行う研究も提案されている。

一方、技術的な観点からは、従来から、設計ドキュメントなどの重要性がソフト設計上で指摘されていたが、なかなか後で利用できる情報を残す良い方法がない。順不同で試行錯誤的な設計プロセスから生じる結果のみを、論理性、統一性を持たせて記述するのは難しい。これも一部には設計判断履歴の欠如が原因と考えられる。

上記のような背景分析により、設計判断履歴情報を獲得するためには、以下の機能が重要である。

(1) 非連続的な判断履歴を整理し、グラフで表示する機能
(2) 設計判断履歴情報を問題分割、候補案などの作成に応じて再構成する機能
(3) 設計生産物と判断履歴との関係を明確にし、知識、ノウハウとして蓄積する機能

これらの機能を実現し、適当なタイミングで判断過程を記録し、その活用が十分できれば上記で挙げた問題の一部は解決し、以下の効果が期待できる。

- 問題の分析過程の提示
  − 候補案の種類、問題分割の方法を適宜示す。

- 判断の逆向き追跡
  - 判断過程を逆向きにたどり、適切な箇所に戻る。
- 判断項目の抜けをチェックする。
  - 計画忘れの問題の解消を可能とする。
- ソフトウェア技術者教育にも効果がある。
- これらを包含する意味での知識の獲得が可能となる。

3 研究事例の紹介、モデルの評価

3.1 調査の観点

設計判断履歴を獲得する上では、履歴自体をどのように記録、蓄積するかが重要な課題であり、各種の提案がある。ここでは、研究事例の調査の観点を以下の項目に絞り検討した。

(1) 誰の設計判断履歴を記録するのか?

個人の設計判断履歴情報を獲得するのか、グループを対象にするかで分類した。

(2) 設計プロセス上のどこを記録、蓄積するのか?

ソフトウェアの設計プロセスは、大きく分けて要求理解、基本設計、詳細設計、プログラミングに分類される。この工程上のどこを記録、蓄積するのかで分類した。今回は主に、設計過程を中心とした。

(3) 設計法との関連は?

現在までにソフトウェア設計に関する方法論がいろいろと提案され、その方法論に基づく支援ツールもいくつか開発されている。設計支援ツールと設計判断履歴の関係を整理すると以下のようになる。

既存のソフト設計方法論に従って記録、蓄積するものは、調査した範囲内ではあまりない。これは、設計支援ツールを使えば、ある程度、基本的な設計に対する考え方、流れが示されているので設計判断履歴はあまり必要ないということかもしれない。しかし、リポジトリー (Design repository) の重要性も指摘されているので、今後は履歴に対する考察が深まり、相互に進展すると考えられる。

これに関連するものとして、プロセスプログラミング (Process Programming) の概念がある[32)27)]。これは、ソフトウェアの開発プロセスを形式的に記述して、その記述をもとに実際の各プロセスを支援しようとするものである。この際、各方法論を形式的に記述する試みがいくつか行われている。

一方、多くの設計判断履歴記録、蓄積システムは、既存のソフトウェア設計方法論に従わず記録、蓄積するものが多い。これには、グループによる討論議論、整理を含む。これは、いわば、方法論によらず、設計者の発想に従った設計過程をとりあえず記録、蓄積し、その設計過程を分析しようとしている。これは、また、方法論が大まかな点しか規定しておらず、本来の設計部分に関しては、設計者の能力にゆだねられているためでもある。

(4) 利用法?

設計過程での設計判断履歴を記録、蓄積した後どのように利用するかという点に関しては、いろいろなアプローチがある。

まず、デザインレビューへの適用、設計過程の再利用がある。また、機能追加、変更時の波及範囲の推定などのソフトウェアの維持管理に着目した研究、システムもある。一方、知識獲得の観点からは、設計作業のノウハウの他者への移転、伝達、ソフトウェア技術者教育への利用などが考えられる。

(5) 記録、蓄積ツールとして何を使うか?

多くのシステムが、ハイパーテキスト (HyperText)[9)] となんらかのデータベースシステム、もしくはルールシステムとの併用を採用している。これは、設計判断履歴情報が、設計過程での判断を設計生産物との関係などとともに視覚的に記録する必要があり、再利用の際、検索が重要であるからである。

上記の観点に従った調査対象の各システムの特徴、工程上の支援範囲、利用例を表1に示す。個別の事例に関しては、3.3章で詳しく述べる。

今回は、主に設計工程より上流のものを対象にしたが、プログラミングの工程でもいくつかの提案、実験が行われている[1)2) 12)20)22)]。なかには、心理学の観点から、記録、蓄積よりも、プログラム理解過程を中心としたものもある。これらのシステムの検討結果は、主に教育 (CAI) などに用いられている。また、リバースエンジニア (Reverse Engineering) の観点からも履歴情報の蓄積、利用の重要性が指摘されている[6)7)]。これらについては、紙面の都合上省略する。

3.2 系譜 – 設計判断履歴情報獲得のアプローチ

次に、設計判断履歴情報獲得のアプローチとしての系譜をしめす。どこに着目するかにより大きく以下の2つに分類できる。

- 判断プロセス重視のアプローチ
  Process Oriented (AI 的)
  - 問題がどのように解決されたかに着目し、記録、蓄積する。
  - 認知的アプローチ (Cognitive approach) を採用する場合もある[17)]。
  - 知識獲得、教育への応用との関連が深い。
- 設計生産物重視のアプローチ
  Artifact Oriented (QC 的)

- 生産物がどのようになったかかに着目し、記録、蓄積する。
- 形式的アプローチ (Formal approach) を採用する場合が多い。
- プロセスプログラミングとの関係が高い。

上記分類に基づいた設計判断履歴情報システムの系譜を図1に示す。実際の各システムは、上記分類で示した2つの系列を統合した形となっているが、大まかな傾向として示した。さらにグループ支援 (Groupware)、協調設計作業 (CSCW) との関連、プロセスプログラミングとの関連などそれぞれに特徴がある。詳しくは、次章で述べる。

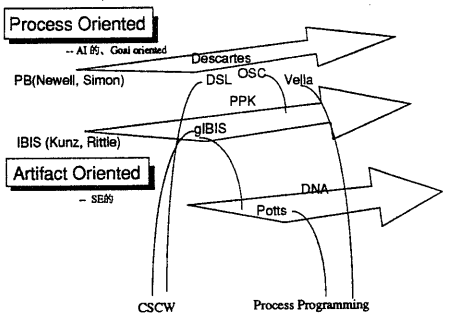

図 1: 設計判断履歴情報システムの系譜

3.3 個別の研究事例

設計判断履歴を提唱した Freeman は、特に、設計レビューの際に設計判断履歴が有効と指摘した[11]。設計には2種類の設計があり、

(1) Discovery design
(2) Routine design

このうち、設計判断履歴は、主に (2) をサポートする。また、別解の用意、別解の評価、データ構造の選択、否決の理由の説明の重要性を述べている。

以下では、個別の研究事例について述べる。

3.3.1 gIBIS (MCC)

要求分析、特にグループによる設計上の政策、意志決定作業をサポートすることを目的に開発された[8) 10)]。利用法としては、設計のレビューが主である。

このシステムは、従来の IBIS (Issue Based Information System) をグラフィカルな端末上にハイパーテキストを用いて、視覚的に表示し、グループでの意志決定作業を支援するシステムである (図2参照)。

もともとの IBIS は、問題提起 (Issue), 意見の表明 (Position), 論拠 (Argument) の関係をテキストベースで記録するものである。まず、問題提起が行なわれ、それに対する賛成、もしくは、反対の意見表明をする。その際、論拠を述べることは、後での見直しなどに用いられる。

3.3.2 DRL (MIT)

gIBIS と同様にグループによる意志決定作業をサポートすることを目指しているシステムである[15) 14)]。特徴としては、グループ意思決定のための意見の集約機能を持っている点である。

このシステムで用いているモデルは、ゴール (Goal) と別解 (Alternative)、質問 (Question)、批判 (Claim) である (図3参照)。ある問題を解くためのゴールを設定し、いろいろな別解を提示し、それに対する質問や批判などを行なう。これらの議論を通じて最終的には、意見集約を行ない (その際システムが集約の支援をする) グループの意見をまとめる。

3.3.3 Potts (MCC)

基本的には、IBIS (Issue Based Information System) を用いているが、よりソフトウェア設計に適用可能なように設計生産物 (Artifact) を中心に議論の経過、設計における判断履歴を記録するシステムである[19)18)]。これも、ハイパーテキストを用いて、グラフィカルな端末上に表示するものである (図4参照)。また、設計過程の履歴を E-R モデルによりデータベース化し、生産物にどのような判断がなされたかを検索できる。

3.3.4 OSC (CNET)

これも基本的な設計判断履歴モデルとしては、IBIS を用いているが、根拠、正当性 (justification) を主眼に判断履歴を記録、蓄積するシステムである[3)4)]。

そのモデルでは、どのような判断をしたか (Design decision)、その判断に伴う問題点の要素 (Problem element)、設計に至る理由 (Assersion)、未検討項目 (Agenda item) の4つの情報をフレーム形式で格納している (図5参照)。

3.3.5 Vella (静岡大学)

Vela では、プロジェクト構成員間のプロセスの再利用, 交信, プロジェクトチームの制御と調整等を支援する統合環境を目指している。その中で、事例ベース推論と帰納推論に基づく設計判断履歴の構築がなされている[31)]。これは、問題固有で断片的なソフトウェア開発事例群から, 事例ベース推論を用いて過去の経験的な設計事例とのマッチングを取り、帰納推論 (CIGOL) を利用して, 汎用的なモデルを構築する枠組みである (図6参照)。

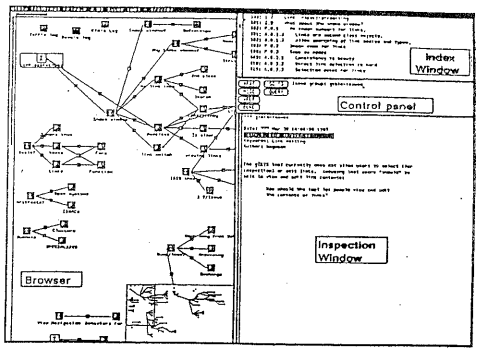

図 2: gIBIB での設計判断履歴の例

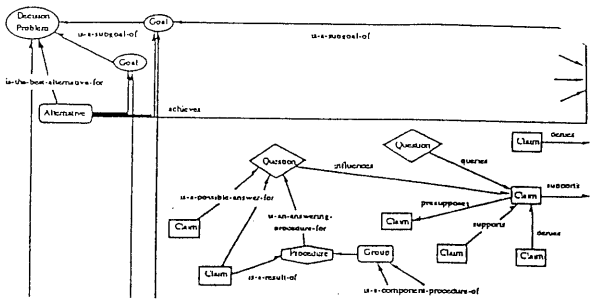

図 3: DRL での設計判断履歴の例

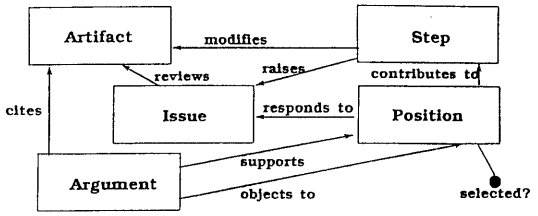

図 4: Potts での設計判断履歴の例

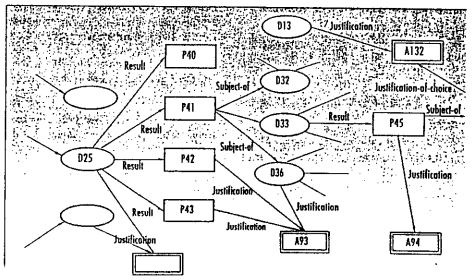

図 5: OSC での設計判断履歴の例

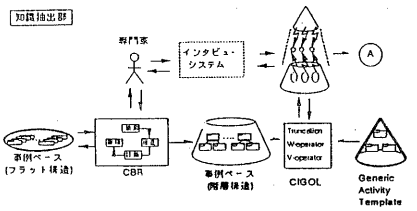

図 6: Vella での設計判断履歴の例

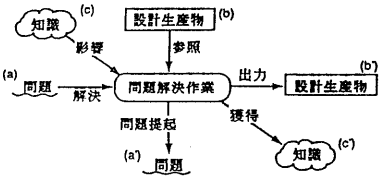

図 7: PPK での設計判断履歴の例

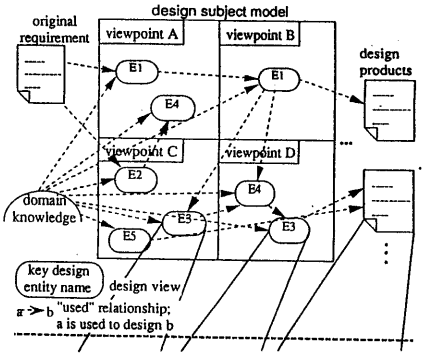

図 8: DNA での設計判断履歴の例

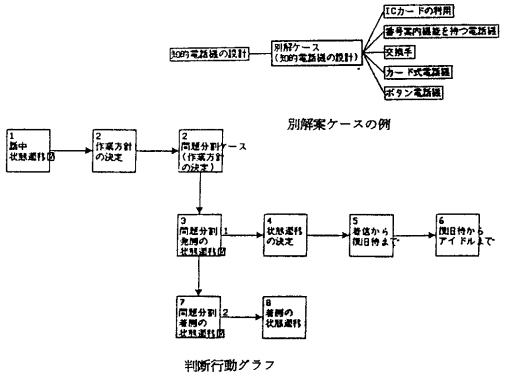

図 9: Descartes での設計判断履歴の例

表 1: 各システムの特徴と利用法

システム名	開発者	特徴	個人 or グループ	工程	利用法	ツール
gIBIS	Conklin (MCC)	Policy Discussion	グループ	要求分析、設計	レビュー	Hypertext
DRL	Lee (MIT)	Group decision rationale	グループ	要求分析、設計	グループ意思決定	Hypertext + DB
Potts	Potts (MCC)	IBIS + artifact	個人	設計	仕様精密化	Hypertext + DB
OSC	Arango (CNET)	Envelope	個人	設計	一貫管理	Hypertext + Rule
Vella	山口 (静岡大学)	Cgol	個人, グループ	要求分析、設計	知識獲得	
PPK	田村 (三菱)	PPK モデル	個人	設計	仕様精密化	Hypertext
DNA	浜田 (ATR)	View	個人	要求分析、設計	波及解析	Hypertext + Dataflow
Descartes	鳥 (NTT)	Desision status, Layout	個人	設計	知識獲得	Hypertext + Rule

3.3.6 PPK (三菱)

PPK モデルにより、ソフトウェア設計者の判断履歴を自動的に記録し、作業の分析, 進捗の把握, 及び設計ノウハウの再利用を支援するシステムである[33]。

PPK とは、Problem Product Knowledge の略であり、入力としては、解決すべき問題 (Problem) や動機、設計生産物 (Product)、およびこの作業に影響を与えた設計手法や設計経験などの知識 (Knowledge) がある。これを問題解決作業を通じて作業で得た、知識、生産物、また新たに生じた問題などを記述していくモデルである (図 7 参照)。

3.3.7 DNA (ATR)

設計履歴を利用して主に、ソフトウェア設計、保守支援に利用しようとするシステムである[34]。仕様・要求などの修正作業を支援するための波及範囲の解析を設計履歴を用いて行なう手法を提案している。

このモデルでは、設計ビューと、それらの利用関係から構成される対象モデルを用いて対象モデルと生産物の作成プロセスを記述していく。

波及範囲の解析の際は、設計ビューを指定することで探索の範囲を減らせる効果がある (図 8 参照)。

3.3.8 Descartes (NTT)

Descartes (DEcision history Structuring on CARd-based TEchnics System) は、筆者の提案したシステムである[29)30)]。設計判断の主なものは候補案の列挙、問題の分割であるとの観点から以下の機能を持つシステムを構築し、設計時の意志決定の支援、および知識獲得を目指している (図 9 参照)。

- 問題分割、候補案、設計生産物 (設計途中のものを含む) を作成する過程の記録、管理機能
  問題分割過程の記録を管理する機能、候補案作成を整理する機能、戻り位置を制御する機能、設計生産物と判断情報との関係を付ける機能を持つ。
- 判断の状態 (中断、完了、中止) を管理するカードによる判断履歴の蓄積、追跡機能
  無意識的に行なわれている判断を効率よく取り出し、陽に表現する機能、および判断の中断、終了などの状態を管理する機能を持つ。
- 判断履歴をグラフ的に表示し、意志決定を視覚的に支援する機能、自動レイアウト機能
  ハイパーテキストを用いて問題の分割過程、候補案の作成過程を自動的に整理する。また、読み履歴機能を持つ。

3.4 各システムの機能的な評価

各システムの機能的な評価を表 2 に示す。機能項目としては、設計過程のグラフ表示、問題毎の検索機能、判断の状態管理、ドキュメントとの関連、および今後重要と思われる学習機能について比較検討した。

・設計過程のグラフ表示機能については実世界の実体の表現を行なうために、いろいろな工夫がなされている。どのシステムも基本的には、ハイパーシステムを用いて、設計判断履歴情報をグラフィカルに表示している。

・検索機能については、設計判断履歴情報から構造を抽出し、検索手続きを用いて検索する。特に、Potts のシステムは E-R モデルを用いた検索が可能である。また、DRL では、意見の集約機能を持っている。

・判断の状態管理 (判断が完了した、もしくは中断しているなどの状態) については、陽に記述しているのは、Descartes, OSC である。

・ドキュメントとの関連については、Potts のシステムがドキュメントを含めた設計生産物との関連について強調している。Descartes では、設計の際にドキュメントの読

み履歴(問題解決の際にどのような文書の箇所を読んだかを判断と関連づける)を保存する機能を有している。

・学習機能については、陽に記述しているのは、Vella である。

表2: 各システムの評価

システム名	設計過程のグラフ表示	検索機能(問題毎)	判断の状態管理	ドキュメントとの関連	学習機能
gIBIS	○	○	IBIS 管理		
DRL	○	○[1]	Goal 管理		
Potts	○[2]	○	IBIS 管理	○	
OSC	○		○		
Vella	○				○
PPK	○			○	
DNA	○				
Descartes	○[3]	△	○	○[4]	

1) 意見の集約機能を持つ
2) 設計物との関連を重視している
3) 自動レイアウト機能を持つ
4) 読み履歴機能を持つ

4 応用例、課題

設計判断履歴情報の応用例としては以下のものがある。

- デザインレビューへの適用
- 設計過程の再利用
- ソフトウェアの維持管理(機能追加、変更時の波及範囲の推定)
- 知識の獲得
- 設計作業のノウハウの他者への移転、伝達
- ソフトウェア技術者教育

4.1 デザインレビューへの適用

デザインレビューへの適用として有名なものとして、gIBIS の応用例がある[25)26)]。NCR との共同研究を行ない、gIBIS の応用として、かなり大規模な設計判断履歴の記録を収集し、その使用体験などを分析している。

実験は、NCR における商用ソフトウェア(小売端末表示ソフト)を作成する時に、gIBIS を用いて、その有効性を調べた。まず、5人、18カ月、2200項目の検討を行ない、8000ノードを紙に記録した。

その後、それらの情報を gIBIS に変換し、6カ月間使用した(合計2年)。

その結果、gIBIS の利点として以下の点が挙げられた。

(1) error の早期発見
設計時のエラー発見が効率的であった。gIBIS で11か所のエラーを発見できた。これは、もし、gIBIS を用いなかった場合、これを見つけるため、3-6倍余計な作業が必要であるとされる。

(2) その他、定性的ではあるが、
会議の効率的な進行ができた。
部門間のスムーズなコミュニケーションができた。
Project 管理に有効であった。

4.2 設計判断履歴獲得の課題

設計判断履歴獲得の課題としては以下のものがある。

(1) 大規模の場合の作成、維持
今だ実験段階、本格的応用がないので大規模な設計を行なった場合の有効性を確認できていない。
(2) 記録労力の負担が大きい。
(3) 完全性の問題がある。
設計における根拠が曖昧な場合がある。
(4) プロダクト間の関係記述方法と既存設計法、ツールとの関係が確立されていない。
(5) 類似設計に対する検索機能、学習機能の欠如がある。
(6) 会議中では、メモがとれない。
(7) 用語の統一が難しい。(Synonyms の問題)
(8) 共同開発への対応が未解決である。
CSCW 指向の際の資源の共有、同期、通信などの問題解決に課題がある。

課題は多くあるが、たとえば、上記課題の(2)、(4)に関しては、判断履歴の記述法の一方法として、Web 的な書き方が有望視される[13)]。Knuth の提唱している方法は、プログラムとドキュメントをその思考過程の時系列を含めた形で一体化させようするものであり、現状の設計における情報の欠如の問題にある種の解決を与えるものである。

また、上記課題の(3)、(5)に関しては、特に設計者のソフトウェアの設計の重要な観点は、経験から学ぶ点が多い。また、思考過程は、複雑で形式的でない。ということがある。この点に関しては、事例ベース推論(CBR, Case-Base Reasoning)[21)24)28)]が有望である。以前に作成したプログラムの事例ベースから、類似事例を探し、修正することにより、新しい問題のプログラムを作成する方法である[5)]。

5 おわりに

本報告では、ソフトウェア設計における設計判断履歴情報(判断などの履歴)に関する研究の背景、系譜、研究事例の紹介、応用例、課題などについて述べた。

基本的には、設計判断履歴情報は、いわば、設計過程における「なぜその設計を行ったのか」という情報に着目して、それをいつ、どのようなタイミングで、どのように記録、蓄積するかが重要な課題であり、各種の提案がある。現状では、主にその流れは2つの系譜があることを指摘した。また、それぞれのシステムの特徴を述べ、機能的な評

価を行った。

現段階では、各モデルの提案を行い、それを精密化している段階である。今後、それぞれのモデルに基づいたツールが開発され, Design Journal として蓄積していくことが重要である。ソフトウェア技術者の不足などの社会的状況、ノウハウの継承などの技術的問題を抱えて今後、ますます設計判断履歴情報、はその重要性、利用価値は増大すると考えられる。

しかしながらその一方、多くの課題を有している。特に、まだ実用規模での設計判断履歴情報獲得のための実験が行なわれていないのが現状である。今後の動向としては、その評価を含めた形で、大規模試験、実用化への適用が期待される。

また、ノウハウの継承の観点からは当然のことながら、設計知識などの知識獲得、利用が重要である。これらの方法の確立が今後望まれる課題である。この点に関しては、学習、知識獲得といった分野での成果を利用することが必要である。一つの方向として、事例ベース推論 (CBR) 的アプローチが有望かもしれない。これは、過去の経験的な知識をもとに似たような状況に合わせて対処するための解を過去の知識を変更することにより生成する。今後多くの研究、実験が行われることを期待する。

最後に、本発表の機会を与えて下さった磯田部長、また、日頃ご指導をいただく斉藤主幹員に感謝します。さらに、討論していただいたソフトウェア基礎技術研究部の各位に感謝します。

参 考 文 献

1) ADELSON, B. AND SOLOWAY, E.: The Role of Domain Experience in Software Design, *IEEE Transactions on Software Engineering* **SE-11**, 11(November 1985), 1351–1360.

2) ANDERSON, J., FARRELL, R., AND SAUERS, R.: Learning to program in LISP, *Cognitive Science* **8**(1984), 87–129.

3) ARANGO, G., BRUNEAU, L., CLOAREC, J.-F., AND FEROLDI, A.: Proposal for a Design tracking system, Technical Report 309, CNET(1989).

4) ARANGO, G., BRENEAU, L., CLOAREC, J.-F., AND FEROLDI, A.: A tool shell for tracking design decisions, *IEEE Software*(March 1991), 75 – 83.

5) CARBONELL, J.: Learning by Analogy: Formulating and Generalizing Plans from Past Experience, in Machine Learning. Vol. 1, 137–161, Morgan Kaufmann(1985).

6) CHIKOFSKY, E. J. AND CROSS, J. H.: Reverse Engineering and Design Recovery: A Taxonomy, *IEEE Software*(January 1990).

7) CHOI, S. C. AND SCACCHI, W.: Extracting and Restructuring the Design of Large Systems, *IEEE Software*(January 1990).

8) CONKLIN, J. AND BEGEMAN, M.: gIBIS: A Hypertext Tool for Exploratory Policy Discussion, Technical Report STP-082-88, MCC(march 1988).

9) CONKLIN, J.: Hypertext: An Introduction and Survey, *Computer*(Sep 1987), 17–41.

10) CONKLIN, J.: Interissue Dependencies in gIBIS, Technical Report STP-091-89, MCC(1989).

11) FREEMAN, P.: Towards Improved Review of Software Designs, in National Computer Conference, AFIPS(1975).

12) KANT, E.: Understanding and automating algorithm design, *IEEE Transactions on Software Engineering*(November 1985), 1361–1374.

13) KNUTH, D. E.: Literate Programming, *The Computer Journal* **27**, 2(1984).

14) LEE, J.: Extending the Potts and Bruns Model for Recording Design Rationale, in ICSE 91(91).

15) LEE, J.: SIBYL: A Tool for Managing Group Decision Rationale, in CSCW 90 Proceeding(October 1990).

16) MOSTOW, J.: Toward Better Models Of The Design Process, *AI Magazine* **6**, 1(1985), 44–56.

17) NEWELL, A. AND SIMON, H. A.: *Human Problem Solving*, Prentice-Hall(1972).

18) POTTS, C.: Representing Recurrent Issues and Decisions in Software Design Methods, Technical Report STP-083-89, MCC(1989).

19) POTTS, C. AND BRUNS, G.: Recording the Reasons for Design Decisions, in 10th ICSE(1988), 418–427.

20) REYNOLDS, R. G., MALETIC, J. I., AND PORVIN, S. E.: PM: A System to Support the Automatic Acquisition of Programming Knowledge, *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING* **2**, 3(September 1990).

21) RIESBECK, C. AND SCHANK, R.: *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Publishers(1989).

22) RUGABER, S., ORMBURN, S. B., AND LEBLANC, R. J., JR.: Recognizing Design Decisions in Programs, *IEEE Software*(January 1990).

23) : SDA/II プロジェクト 1990 年度 活動報告書 (05/27, 28 1991).

24) WILLIAMS, R. S.: Learning to Program by Examining and Modifying Cases, in Proc. Case-based reasoning workshop(1988).

25) YAKEMOVIC, K. B. AND CONKLIN, E. J.: Report on a Development Project Use of an Issue-Based Information System, *CSCW 90*(1990).

26) YAKEMOVIC, K. B. AND CONKLIN, J.: The Capture of Design Rationale on an Industrial Development Project; Preliminary Report, Technical Report STP-279-89, MCC(1989).

27) 井上克郎: PDL を用いたプロセス, プロダクトに関する種々の記述, *SDA*(1991).

28) 奥田, 山崎: 事例ベース形推論とその応用例, 情報処理 **31**, 2(1990).

29) 島健一: ソフトウェア設計のための判断履歴の蓄積、利用について, 情処学会 ソフトウェア工学研究会 **69**, 5(1989).

30) 島健一: 読み履歴を利用した設計メタ知識の獲得, 人工知能学会全国大会 (1990).

31) 山口高平, 落水浩一郎: ソフトウェアプロセスモデル構築における知識獲得と利用方式, 人工知能学会基礎論研究会 (1991).

32) 篠田陽一: 属性文法に基づく設計支援環境, *SDA*(1991).

33) 田村直樹: 設計プロセス支援ツールを用いた設計作業の記録・モニタリング ・再利用, ソフトウェアシンポジウム 91(1991).

34) 浜田雅樹: 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(1991).