設計プロセス情報を利用したソフトウェア修正支援方式

浜田雅樹[†,*] 安達久人[†]

アプリケーションソフトウェアの影響波及解析を容易化するための手法を提案する．ソフトウェアの開発では，CASEツールとともに，構造化分析・設計手法が普及してきた．構造化分析・設計手法で定めている，設計途中および最終段階において作成するドキュメントには設計結果のみが記述され，設計に影響を与えた要因，すなわち設計の根拠，は記述されない．これは，ソフトウェアの修正箇所を限定する影響波及解析を保守者にとって困難にする要因の一つになると考えられる．この問題点を解決するには，影響波及解析に有用な，設計の根拠の情報を小さいコストで記録，利用する方法を確立する必要がある．筆者らは，この方法を実現したのでここに提案する．提案する手法では，設計者が設計プロセスの記録を作成しながらソフトウェアを設計する．この記録は，設計の根拠である，設計に影響を与えた要因の情報を含む．保守者は，本手法が実現する設計プロセスの記録の検索機能により，影響波及解析に有用な情報を参照することができる．また，設計プロセスの記録は，設計の思考の流れに沿って作成するため，記録のコストが小さいことが期待できる．以上の手法を実現したプロトタイプシステムを用い，提案した影響波及解析支援の効果，設計プロセスの記録のコストならびに設計者の負担について評価した．

Software Modification Support Using Design Process Records

Masaki Hamada[†,*] and Hisato Adachi[†]

This paper proposes a method of supporting software change propagation analysis, i.e., determining parts of the design products that should be modified when the software is required to effect changes. Using the structured analysis/design method for application software development, no design rationale of the software is recorded on its design products. This makes it difficult for a maintainer to perform change propagation analysis of the software. To solve this problem, it is necessary to establish the method of recording and using the design rationale that is essential to change propagation analysis at small costs. This paper proposes the method. In the proposed method, a designer makes a design process record that includes design rationale during his/her design. The method enables a maintainer to search for the recorded information that is essential to change propagation analysis. The cost of recording the design process is expected to be small because the designer only writes down fragmentary design results. From the experience of using a prototype system, the effectiveness of the proposed method is discussed.

1. はじめに

ソフトウェア保守は膨大なコストがかかることが指摘されている．保守においてソフトウェアを修正する際困難な作業が，ソースコードとドキュメントにおける変更箇所を限定する解析である[14]．この解析を以下，影響波及解析と呼ぶ．本論文では，アプリケーション・ソフトウェア☆の影響波及解析を支援する手法を提案する．

ソフトウェアの開発では，CASEツールとともに，構造化分析・設計手法[7),21)]が普及してきた[20]．構造化分析・設計手法では，設計途中および最終段階において作成すべきドキュメント（ソースコードも含めて設計生産物と総称する）を定めている．主な設計生産物として，構造化分析手法では，要求分析のデータの処理の流れを記述したデータフロー図やデータ構造図を，構造化設計法では，構造設計の結果として，モジ

† ATR 通信システム研究所通信ソフトウェア研究室
ATR Communication Systems Research Laboratories, Communications Software Department
* 現在 NTT ソフトウェア研究所ソフトウェア基礎技術研究部
Presently with NTT Software Laboratories, Software Research Laboratory

☆ 以下，本文中の「ソフトウェア」は，特に断らない限りアプリケーション・ソフトウェアを指すこととする．

ュール間の呼出関係とパラメータの入出力を記述した構造図を設計生産物として作成する．これらの設計生産物には設計結果のみが記述され，設計に影響を与えた要因，すなわち設計の根拠，は記述されない．

ソフトウェアの保守に限らず，再利用を行うには，そのソフトウェアの設計に影響を与えた要因に関する情報が必要である[6]．しかし，設計結果から設計に影響を与えた要因を推測するのは容易ではない．したがって，再利用や保守を支援するためには，設計生産物に記述されない情報である，設計に影響を与えた要因，を扱う技術の確立が必要である[16]．保守や再利用による設計支援等を目指した研究の多くは，設計に影響を与える要因に関する情報をあらかじめ用意しておくアプローチを採っている．例えば，知識ベース技術を用いたソフトウェアの再利用や保守支援[13),17),18]は，設計に影響を与える要因の候補をあらかじめデータベース化しておくアプローチである．しかし，現状の技術では，あらかじめ設計に影響を与える要因を分析し，矛盾なく記述するのは困難なため，これらのアプローチの適用性には限界がある[19]．

これに対して，設計に影響を与えた要因に関する情報を設計時に残す方法が研究されている[3),4),11),15),16]．これらの研究の目的は，設計の根拠(design rationale)の記録法および記録した設計の根拠に関する情報を，ソフトウェアの保守や別のソフトウェアの設計で再利用する技術を確立することにある．確立すべき技術は以下の要求条件を満たす必要がある．

- ソフトウェアの保守や再利用に有用な情報を記録することができる，
- ソフトウェアの保守や再利用の各場面に応じて，必要な情報を検索できる，
- 記録のコストが小さい．

現在，WEB[2]，IBIS (Issue Based Information System)をベースとした記録方法[3),4)]などが提案されている．これらの研究のほとんどは，記録形式に関する議論が中心で，記録する情報の効果・妥当性，利用方法や記録のコスト等について議論していない．

本論文では，要望が高い影響波及解析の支援を目的とした，上記要求条件を満たす設計の根拠の記録，利用手法を提案する．

提案する手法では，記録のコストを小さくするため，設計のプロセス[1]を記録する．従来の記録手法では，設計者が設計の根拠を説明する文章を作成する必要が有る．これは，設計以外にもう一つの高度な知的活動が要求されることを意味している．これに対し，提案する手法は，設計の各ステップにおける断片的な設計結果を書き留めるだけなので，設計者は設計活動に集中することができる．以上より，提案手法は，記録のコスト，設計者の負担感が小さいことが期待できる．

主要な研究課題を以下に示す．

（1） 以下の条件を満足する設計プロセスの記録のデータモデルの確立：

条件1：設計プロセスの記録が，影響波及解析に有用な情報を含む，

条件2：設計プロセスの記録は，検索のための索引や関連付けの情報を内部に持つ，

条件3：設計の流れに沿った記録ができる（人間の設計における認知的側面[5]を考慮した設計プロセスのモデルに基づいている）．

（2） 設計の思考に沿った設計プロセスの記録手法（マンマシン・インタフェース）および影響波及解析支援手法（記録情報の検索手法）．

図1に提案する手法の位置付けを示す．既存のCASE (Computer Aided Software Engineering, コンピュータによるソフトウェア開発支援) ツールでは，保守者は，設計生産物を参照することしかできず，影響波及解析は困難であった．これに対し，提案する手法では，既存の CASE ツールに対して以下の機能を追加することにより影響波及解析のコストを軽減することを目的とする．

- 設計者が設計プロセスを記録する機能．
- 設計に影響を与えた要因の情報および設計生産物のうち，ソフトウェアの修正に関連するものを検索し設計者に提示する機能．

以下，まず影響波及解析に有用な情報について述べ（2章），次に設計プロセスの記録のデータモデルについて述べる（3章）．さらに，設計プロセスを記録す

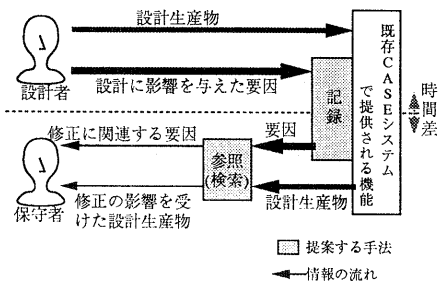


図1 提案する手法の位置付け
Fig. 1 Position of the proposed method.

る手法（マンマシン・インタフェース）（4章）および影響波及解析支援手法（5章）について述べ，最後に，提案する手法の評価結果について述べる（6章）．

2. 影響波及解析に有用な情報

アプリケーション・ソフトウェアの設計では多くの場合，まずソフトウェアが持つべき機能（以下設計対象と呼ぶ）を明らかにする必要がある．同じ分析・設計法を用いても設計の度ごとに設計の内容が異なるのは，それぞれの設計において設計対象の性質，例えば，要求されている機能項目，ソフトウェアが走行するハードウェアの構成，既存ソフトウェアとの互換性等，が異なるためである．すなわち，設計対象の性質が設計に影響を与える要因になる．

ソフトウェアの保守では，ソフトウェアに対して，特定の設計対象の性質，例えばハードウェアの構成等，の変更要求が起こる．これらの変更は，関連する設計対象の性質に波及し，それらが設計生産物への影響となって現れる．設計生産物を用いて影響波及解析を行うには，どのような設計対象の性質がどのように設計結果に影響を与えたかを保守者が推測する必要がある．この作業の困難さが，影響波及解析の困難さにつながっていると予測される．したがって，設計者がどの設計対象の性質に着目し，それを設計にどのように利用して行ったかを追跡することができれば，影響波及解析を容易化できることが期待できる．以下，本追跡を可能にする設計プロセスの記録のデータモデルについて述べる．

3. 設計プロセスの記録のデータモデル

3.1 設計プロセスのモデル

構造化分析・設計手法では，設計対象を，概念上の実体とそれらの間の関係として表現する．例えば，データフロー図は，設計対象を，プロセス☆，データ（以上は概念上の実体）とそれらの間のデータフローの関係として記述したものである．以下，これらの概念上の実体を設計エンティティと総称する．構造化分析・設計手法によりアプリケーション・ソフトウェアを設計する以下の手順を提案する．

設計すべき設計対象は，多くの設計エンティティがさまざまな関係により複雑に絡み合う．人間の認知能力の限界により，設計者が一度に捉えられる関係の種類や範囲には限界がある．設計者は，「データAと依存関係にあるデータは何か？」のように，特定の設計エンティティが持つ特定の種類の関係に着目しながら設計していくと仮定する．今，設計者が設計するために着目した，設計エンティティ間の関係の種類を「視点」と呼ぶ．例えば，データAと依存関係があるデータを調べる設計は，「データAを視点：依存関係から設計する」と捉える．また，特定の設計エンティティを特定の視点から設計した結果を設計ビューと呼ぶ．提案する手法では，設計者が設計中に着目し設計した設計対象の性質を設計ビューという単位で表す．

設計ビューは，他の設計ビューの内容に基づき設計される．例えば，あるプロセスの入出力データを視点「依存関係」から設計した設計ビューに基づいて，そのプロセスを視点「サブプロセス」から設計する（＝新しい設計ビューを設計する）．このように設計ビュー間に，設計における利用の関係（「利用関係」と呼ぶ）が存在する．

以上提案した設計手順に基づく設計プロセスのモデルを以下に示す（**図2**）．設計者は，まず，ソフトウェア開発依頼者の要求（原要求と呼ぶ．一般に曖昧さや矛盾を含む）に基づき，要求される設計対象の性質を分析する．これは，原要求を利用して設計ビューを設計することに相当する（図2(1)）．次に，それらの結果を利用し，設計生産物の作成で必要な設計ビューを順次設計し（図2(2)），最後に設計生産物を作成する（図2(3)）．設計者は，以上の設計において自身が持っている領域知識を利用する．領域知識とは，例えば在庫管理の知識等ソフトウェア化される処理自体に関する知識と，設計技術に関する知識を指す．このよう

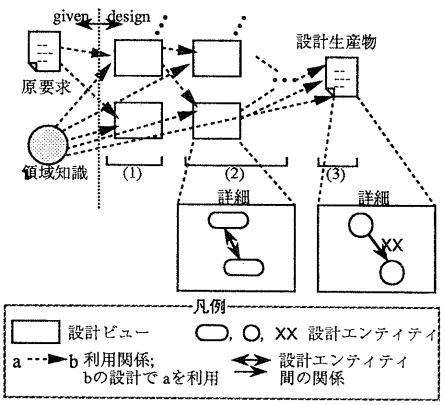


図 2 設計プロセスモデル
Fig. 2 Software design process model.

☆ あるデータから別のデータを計算する処理を表す．設計のプロセスとは異なる．

に，設計ビュー，設計生産物，原要求，領域知識間には利用関係がある．

視点は，以下の2種類に分類できる．

1) 設計エンティティ間に存在する関係を分析する視点．例えば，「依存関係」等．

2) 設計エンティティと特定の関係を持つ設計エンティティを新たに定義するための視点．例えば，「サブプロセス」等．

視点には，分析・設計法での指定またはソフトウェア開発現場ごとの設計ガイドラインによりあらかじめ決まるものと，設計時に設計者により認識されるものとが存在する．

3.2 設計プロセスの記録のデータモデル

設計プロセスのモデルにおける以下の情報を記録する設計プロセスの記録のデータモデルを提案する（**図3**参照）．

- 設計ビュー
- 利用関係
- 設計生産物

（1） 設計ビュー

設計ビューは，内容とインデックスより構成される．

- 内容：①ある設計エンティティ（その設計ビューのキー設計エンティティまたは単にキーと呼ぶ）を特定の②視点から設計したもので，キー，他の設計エンティティおよびそれらの間の関係として記述される．
- インデックス：①の設計エンティティ名と②の視点名の対で表す（以下，［キー；視点］で示す）．

設計ビューは，インデックスで識別する．

設計エンティティは，その名称を示す文字列で表し，識別する．

（2） 利用関係

利用関係に関する情報として，設計ビューの内容に含まれる設計エンティティ，関係はそれぞれ「利用属性」を持つ．利用属性は，それを持つ設計エンティティまたは関係を設計した際利用した他の設計ビュー，領域知識，原要求の識別子の一覧を値として持つ．

例えば，図3の（1）の設計ビューは，キー：「出庫処理」，視点：「機能項目」をインデックスとして持ち，その内容は，「出庫処理は，出庫依頼の入力，在庫情報の更新等から構成される」ことを表した，設計エンティティ間の全体-部分の関係（part-of 関係）である．これらの設計エンティティ，関係は，利用属性を持っている（例えば「出庫依頼の入力」は，設計ビュー［出庫依頼；媒体］と領域知識を用いて設計されたことを表す利用属性値を持っている）．

（3） 設計生産物

提案手法では，設計生産物を設計ビューと同じ形式で記述する．例えば，構造化分析手法の設計生産物であるデータフロー図は，その詳細化階層における各ダイアグラムそれぞれを設計ビューとして表す．その場合，データフロー図のプロセス，データなどが設計エンティティに相当し，詳細化階層における親プロセス名がキーに，「DFD」(Data Flow Diagram の略）が視点になる．

筆者らは，提案手法のコンピュータ支援系のプロトタイプシステム DIG (Design Information Gathering

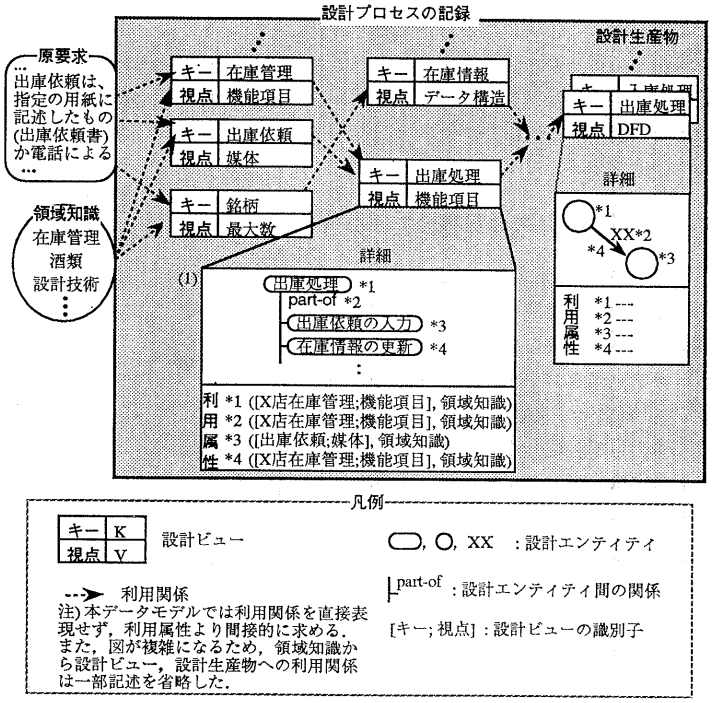


図3 設計プロセスの記録例
Fig. 3 Design process record example.

system）を試作した（Smalltalk 80 で開発，UNIX ワークステーション上で稼働）．以下では，設計プロセスの記録方法および影響波及解析支援手法について，DIG での実行例を示しながら説明する．

4. 設計プロセスの記録方法

提案する手法では，以下に示す設計者の自然な活動に沿って設計プロセスを記録する．

①何について②どんな視点から設計を行うかを決定し，それについて③考慮すべき情報を参照しながら④設計する．

①，②により設計ビューのインデックス情報であるキー，視点を，③により利用属性を，④により設計ビューの内容を記録する．以下，DIG を用いて設計プロセスを記録する具体例を示す（**図 4** 参照）．

設計の開始時点において，設計者は原要求ファイル（DIG では，原要求をテキストファイルとしている）を開いて読みその概要を摑む．設計者は，プログラムの機能項目をまず整理しようと考えたとする．設計者は，原要求に記述されているプログラム名「在庫管理」をテキスト選択し，コマンド「設計進展」を起動する（①）．次に，設計者は，設計の視点「機能項目」をシステムが提示したメニューより指定する（②）．事前に定義できる視点については，ソフトウェア開発現場単位で視点のガイドラインを設定し，あらかじめ DIG に視点のメニューとして登録しておく．また，それ以外の視点については，設計者が，視点を記録する際に DIG に入力する．

システムは，インデックスとしてキー：「在庫管理」，視点：「機能項目」を設定した設計ビュー・エディタ・ウィンドウを開き（**図 4**-a），その他のウィンドウ（この場合は原要求）を閉じる．その他のウィンドウを閉じるのは，ユーザに設計の参照行為を意識させるためである．次に設計者は，参照が必要な設計ビュー，原要求，領域知識を開き（例では原要求）参照しながら（③），機能項目として設計エンティティ：「在庫管理」，「入庫処理」，「出庫処理」およびそれらの間の part-of 関係を設計ビュー・エディタに記述する（④）．システムは，それらの設計エンティティ，関係の利用属性が「原要求」であることを記録する（図 4-b）（現在 DIG では，領域知識のデータベースが未整備のため，領域知識からの利用関係は記録していない）．

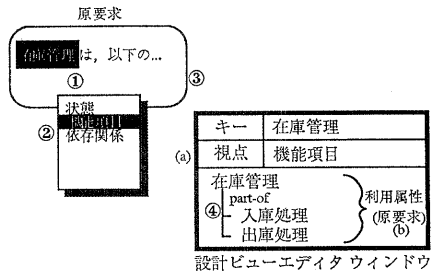


図 4 設計プロセスの記録法
Fig. 4 Recording design process.

5. ソフトウェアの影響波及解析支援

（1） 影響波及解析支援の考え方

保守活動は以下の 4 種類といわれている[12]．

①外的要因（動作環境等）の変化への適合．

②欠陥除去．

③機能向上．

④将来の保守に備えたソフトウェアの構造の洗練化．

これらの活動によるソフトウェアの変更要求は，以下の 3 種類に分類することができる．

変更 1：原要求により要求されている設計対象の性質が変更される（③）．

変更 2：領域知識に基づき設計された設計対象の性質が変更される（①，④）．

変更 3：問題点（バグ）の吸収による変更（②）．

前述のとおり，本手法では，設計者が着目した設計対象の性質およびそれらの設計での利用方法を保守者が追跡可能とすることで影響波及解析を容易化する．追跡は以下の 2 段階より成る．1)変更要求に関連した設計ビューを検索し，次に，2)検索した設計ビューを用いて設計されている設計ビュー，設計生産物を追跡していく．

段階 1 は，以下の方法で行うことができる．

変更 1 の場合は，原要求を利用して設計された設計ビューのうち，変更された設計対象の性質に対応する視点，キーを持つものを限定する．

変更 2 の場合は，変更される領域知識と利用関係を持ち，さらに変更される領域知識に関連した視点，キーを持つ設計ビューを限定する．

変更 3 の場合は，問題が発生した設計ビューまたは設計生産物を設計する際に利用した設計ビューを検索していき，問題の原因となっているものを特定する．

段階 2 は，以下の方法で行う．

段階 1 で限定した設計ビューを利用して設計されて

いる他の設計ビュー，設計生産物を利用関係を用いて追跡する．

（2） 影響波及解析支援手法

影響波及解析支援手法について，DIG を用いた具体例を用いて説明する（**図 5**）．

今，在庫管理のプログラムにおいて，出庫依頼が，所定の用紙または電話で来ることになっていたのを，電子メールで来るように変更する例を考える（変更1に該当）．

1） 変更に関連した設計ビューの検索

システムは，まず原要求を利用して設計された設計ビューの一覧を表示する（図5①）．保守者は変更に関連した設計ビューをその中から選ぶ（図5②）．この選択において，設計ビューのインデックス情報である視点とキーが有効な働きをする．今回の変更は，媒体という性質が変わると考えられるので，保守者は表示されている設計ビューの一覧の中で，視点：「媒体」を持つものにまず着目する．次に，出庫依頼の媒体が変わるので，キーとして「出庫依頼」を選ぶことにより，変更に関連した設計ビューを限定することができる．限定した設計ビューを DIG が表示すると（図5③），その設計ビューの内容の中で原要求を利用して設計された箇所，本例では「電話または出庫依頼書（紙）」，がハイライトされる（図5では太線で表示）．保守者は内容を確認し，この設計ビューを修正の対象としてマークする（図5④）．

2） 設計ビュー，設計生産物の追跡

システムは，先ほどマークした設計ビューを利用して設計された設計ビューのリストを表示する（図5⑤）．保守者は，リストにある設計ビューからチェックが必要なものを視点，キーを基に選ぶ（図5⑥）．チェックの結果，修正が必要と判断した設計ビューをマークする（図5⑧）．同様に続けていくと，影響を受ける設計生産物の箇所までたどり着くことができる（図5⑨）．

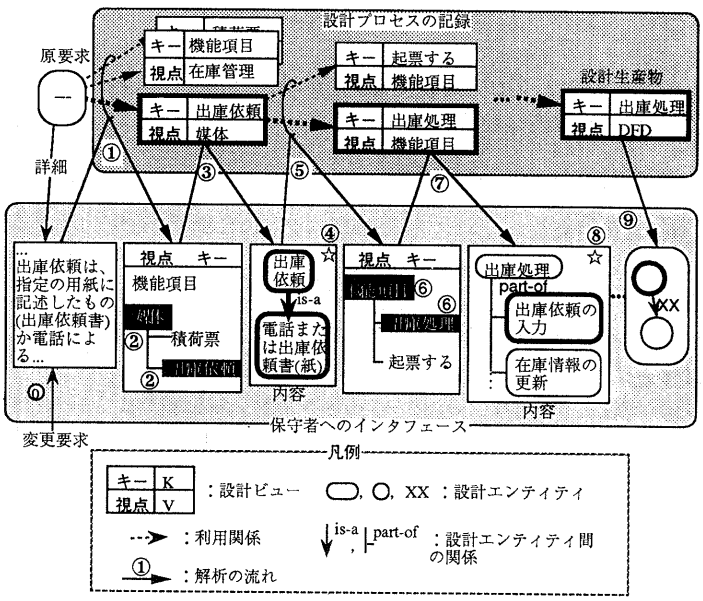


図 5 影響波及解析の支援

Fig. 5 Change propagation analysis support using a software design process record.

6. 評　　価

以下の項目について評価を行うため，DIG を用いたソフトウェアの設計，修正実験を行った．

a） 提案した影響波及解析支援によりソフトウェアの修正工数が削減できることを確認する．

b） 設計プロセス記録手法の評価，すなわち，設計プロセスを記録しながらソフトウェアを設計した場合の設計者の負担感と記録のコスト．

実験は，構造化分析手法を用いた要求分析フェーズを対象とした（設計生産物はデータフロー図）．概要は以下のとおりである．複数のアプリケーション・ソフトウェアを複数の被験者により，設計プロセスを記録しながら設計し，設計者の負担感についてアンケートを実施した．また，これらの設計の一部について，設計プロセスの記録の工数を測定し，別の一部で作成

表 1 修正実験で用いた例題ソフトウェア

Table 1 The subject software used in the experiment.

種　　別	設計生産物と規模	設計プロセス規模	修正の内容	修正の規模
事務処理（入試願書受け付け）ソフトウェア	データフローダイヤグラム（DFD），バブル（プロセス）数約 100	①設計ビュー数 301 （内②設計生産物用 30）	例外処理の改善（願書重複発行防止のための機能追加）	設計生産物上で修正した設計エンティティ数 10

した設計プロセスの記録を用いて，影響波及解析支援の効果評価のための実験を行った．実験の詳細を以下に示す．

6.1 評価実験の手順

（1） 影響波及解析支援の効果評価のための実験

あるソフトウェアを修正する工数について，データフロー図のみを利用した場合と，提案した影響波及解析支援を利用した場合を比較する．以下の手順で実験，測定した．

① アプリケーション・ソフトウェア（**表1**に示す）を，DIGを用いてデータフロー図まで設計し，設計プロセスの記録を作成した（被験者F）．

② ①で作成したデータフロー図のみを用いて，表1に示した修正要求を満足するように，①とは別の被験者4名（**表2**参照）が個別に修正し，その工数を測定した．なお，本修正作業におけるデータフロー図の参照，修正は，③の場合とツールの操作性の条件を同じにするためDIGを用いて行った．

③ 提案した影響波及解析支援手法を行い，②と同じ修正要求を満たすように①で作成した設計プロセスの記録を修正し，工数を測定した．修正作業は，提案した方法により影響波及解析を行いながら修正内容を決定し，次に設計プロセスの記録をDIG上で編集した．被験者は，①，②とは別の被験者1人（**表3**の被験者E）．

以上②，③において修正を行った被験者は，被修正ソフトウェアおよび修正要求についての情報を修正作業の事前に与えられなかった．

④ ②の修正を行った被験者1人（被修正ソフトウェア，修正要求について知識を持っている）が，③と同様の作業を行い，その工数を測定した．

（2） 設計プロセスの記録工数の測定

被験者や対象ソフトウェアへの依存性を排除するため，以下の平均値を求めた．**表4**に示す3つのケースについてDIGを用いた設計を行い，その任意の部分について，設計ビューの編集に要した工数と設計ビューの設計に要した工数それぞれを測定し，さらに設計ビュー1つ当りの平均値を算出した（表4）．測定は，DIGに設計エンティティ，関係単位の編集モードに入っている時間を測定する機能を組み込み行った☆．

表2 設計生産物のみ利用したソフトウェア修正の工数
Table 2 The man-hours required for software modification by the conventional way.

被験者（情報処理経験年数）	A(10)	B(2)	C(13)	D(3)
修正工数	240	480	240	260
工数の内訳 理解／方針決定	120	180	150	110
設計／編集	120	300	90	150

工数は，人・分．
（注1）被験者は修正する内容および修正するソフトウェアに関して事前に知識を持たない．
（注2）修正とは，被修正ソフトウェア理解，修正要求理解，影響波及解析，修正内容の決定，設計生産物の編集を指し，テストは含んでいない．

表3 提案した影響波及解析支援を用いたソフトウェア修正の工数
Table 3 The man-hours required for software modification by DIG.

被験者（情報処理経験年数）	E(7)	A(10)（注1）
修正工数（注2）	35	21

工数は，人・分．
（注1）参考データ：被験者Aは，事前に被修正ソフトウェアについての知識を持っている（表2での修正を行った後，本修正実験を行った）．
（注2）ここでの修正とは，表2の修正に加え，設計生産物以外の設計プロセスの記録に対する影響波及解析，修正内容の決定および編集まで行う．

表4 設計ビューの記録／思考の工数
Table 4 The man-hours required for recording/designing the design views.

項目	測定値1	測定値2	測定値3	設計ビュー当りの平均
被験者（情報処理経験年数）	F(10)	F(10)	B(3)	—
設計内容	DIGへの追加機能の一部	酒屋の販売在庫管理[23]の一部	DIGへの追加機能の一部	—
測定設計ビュー数	15	7	10	—
設計ビューの設計工数（a）	2556	917	1100	143
設計ビューの編集工数（b）	1051	381	542	③ 61.7
思考工数（$a-b$）	1505	536	558	④ 81.2

工数は，人・秒．

☆（2）で作成した設計プロセスの記録を（1）の修正実験で利用しなかったのは，（2）で作成した設計プロセスの記録は平均値を求めるため数人の被験者が関わる必要があり，（1）の実験で必要な，被修正ソフトウェアの知識を持たない被験者の確保が難しくなると考えたためである．

6.2 影響波及解析支援の評価実験の結果および分析

②で測定した，提案した影響波及解析支援手法を用いた場合の修正工数は35人・分（表3）であった．これに対し，設計生産物であるデータフロー図だけを見て修正した場合の工数は240〜480人・分（表2）であった．以上より，本実験では，提案した影響波及解析支援を利用した修正工数は，データフロー図だけを用いた場合の15%以下であったと試算でき，提案した手法の効果を確認した．

提案した影響波及解析支援を用いた修正工数が小さかったのは，解析中に見た設計ビューの数を絞り込めたことによる．これは，利用関係と設計ビューのインデックスによる二段階の絞り込みによる効果と考えられる．提案した影響波及解析支援手法では，ある設計ビューが変更された場合，その設計ビューを利用して設計された設計ビューの一覧を設計者に提示する（第一段階）．前述の評価実験では，全部で138個分の設計ビューの一覧が提示された．さらに保守者は，一覧に表示された視点名とキー名により，見る必要がない設計ビューを除外し（第二段階），結局内容を参照した設計ビュー数を15まで絞り込んだ．

次に，少数の設計ビューを見て内容が理解できた理由を考察する．これは，提案手法では利用関係を辿って設計ビューを参照していくため，設計者が設計対象を抽象的で曖昧な像から次第に具体化していく過程に沿って，保守者が設計ビューを参照できることによると考えられる．このように，設計ビューのインデックスと利用関係は，見るべき情報の絞り込みに有効であり，かつ利用関係は，見るべき情報を読む順序を誘導するのに有効であった．

以上測定，試算した値は，アプリケーション・ソフトウェア，修正要求の内容および被験者のスキルへの依存性が考えられる．アプリケーション・ソフトウェアに対する依存性では，特に被修正ソフトウェアの規模が与える影響が重要と考える．ソフトウェアの規模の増大に伴い，理解・修正が飛躍的に困難になるといわれている．この問題点に対して，提案手法がどう貢献できるか，今後さまざまな規模のソフトウェアに対する適用を通して評価する必要がある．修正内容への依存性としても，修正の規模の影響が重要と考える．提案手法では，修正工数が，影響波及解析中辿る設計ビュー数にほぼ比例する．すなわち，修正規模の増大（修正箇所が多くなる）に伴い，提案手法を利用した修正工数も大きくなることが予想され，提案手法の効果，修正の規模ならびにソフトウェアのライフサイクルの終了判断との関連を明確にする必要がある．また，今回の実験では，影響波及解析と修正内容の決定を行った後編集を行ったが，ソフトウェアや修正の規模の増大に伴い，影響波及解析と他の修正作業，すなわち，被修正ソフトウェア理解，修正要求理解，修正内容の決定，編集およびテスト間のインタラクションが複雑になると考えられる．このインタラクションを考慮した修正支援手法へと発展させる必要がある．

最後に，被験者のスキルへの依存性について考察する．提案手法で実現している，影響波及解析のシステムによる誘導の効果が被験者の感想より明らかになった．データフロー図だけを用いた影響波及解析は，その糸口を見つけるのに工数がかかり，かつスキルの違う被験者間で工数の差が出た．提案手法では，システムの誘導に従い段階的に分析を行うため，設計生産物だけを用いる場合に比べ，被験者のスキルへの依存性が小さいことが期待できる．また，同様の理由から，被修正ソフトウェアについての知識の有無が影響波及解析の工数に影響しにくいことが予測できる．表3に示したとおり，6.1節(1)①の実験を行い被修正ソフトウェアに関する知識を持った被験者（ただし，設計生産物以外の設計プロセスの内容は知らない）がDIGによる修正の工数は21人・分であり，知識がない被験者の35人・分との間に差が見られた．この差の原因は本結果だけから判断できない．被験者のスキルや知識への依存性について，さらなる評価を積み重ねる必要がある．

以上より，評価実験で用いた例題程度の規模を持つソフトウェアおよび修正に対して，提案手法の効果があると予測できる（評価項目a）．

6.3 設計プロセス記録手法評価のための実験結果および分析

表4に示したとおり，設計ビュー当りの平均設計工数は143人・秒であった．そのうち，編集に要した工数が61.7人・秒で，設計ビューの設計では，約43%が記録のための編集に費やされたと試算できる．

設計プロセスの記録工数について考察する．表1に示した例題ソフトウェアについて，設計生産物以外の設計ビューを記録しない場合（思考と設計生産物の編集）の工数に対する，設計生産物以外の設計ビューの編集工数の比を，設計ビュー数を基に計算した結果を**表5**（⑧）に示す．設計プロセスの記録による工数増

加分が設計ビューの編集工数にほぼ等しいと仮定すると，設計プロセスの記録による設計工数の増加率は約 40% という試算結果を得る．

この試算結果について考察する．保守工数の約28%は修正後のテストである[8]，また，ソフトウェア全開発費の約70%が保守費用[9]という報告がある．これに基づき，本評価実験が実際のソフトウェア開発だった場合の本手法の効果を予測すると，「ソフトウェア総開発費の約 35% を削減」と試算できる．ただし，コストは工数に比例する，要求分析以外の工程でも同じ割合で効果が得られるとして試算．効果は，ソフトウェア個々のライフサイクルの形態によって異なる．ソフトウェアの保守がどの程度発生するか予測できない場合は，設計プロセスを記録することは保険を掛けることに近い意義を持つ．その場合は，設計工数の 40% 増という数値は決して小さいとはいえず，今後，記録のコストを減らす手法の検討が必要である．

設計プロセスの記録では，コスト以外に設計者の負担感が少ないことも必要である．提案手法は，検索で利用するインデックスや利用関係情報を追加するための工程が不用なため（設計者の設計活動から獲得する），他の手法に比べ記録の工数だけでなく，記録者が持つ負担感の点でも有利であることが期待できる．現在までの試用において，以下の点を除けば設計プロセスを記録する負担はあまり感じないという利用者の感想を得ることができた．

- 視点名の決定，
- 単純な内容を持つ設計ビューの参照（頭の中に記憶しているため）．

このアンケート結果は，設計プロセスの記録のコストが，あらかじめ定義しておける視点の数および設計ビュー等の参照作業に影響される可能性を示している．

設計プロセスの記録のコストの削減策として以下が考えられる．

1) 支援ツールの操作性の向上……設計プロセスの記録工数は，エディタ等の操作性の影響が大きい．DIG は，研究的関心が低い設計プロセスの編集機能や日本語入力機能などは最低限の機能しか有していない．これらの機能の操作性を向上させることにより，記録のコスト削減が可能になる．

表 5 設計プロセスの記録工数
Table 5 The man-hours required for recording the design process.

項　　目	測定，計算値
⑤設計生産物の編集工数	17385 人・秒
⑥設計生産物以外の設計ビューの編集工数	① 271 個×③ 61.7 人・秒 =16,721 人・秒
⑦設計生産物以外の設計ビューを記録しない場合の設計工数（全思考の工数＋設計生産物の編集工数）	①301 個×④81.2 人・秒＋⑤17385 人・秒＝41,826 人・秒
⑧設計プロセスの記録による工数増加率（⑥/⑦）	⑥ 16,721 人・秒/⑦ 41,826 人・秒 =40%

2) 運用面での対処……ソフトウェア開発現場ごとの視点ガイドラインの充実．

3) 設計プロセスの記録の再利用支援……新規に記録する設計プロセスの量を減らし，かつコンピュータにより視点のガイドを行う[10),22)]．

以上より，提案手法では，設計者が，比較的負担感少なく設計プロセスを記録できることを確認した．ただし，記録のコスト削減への対策が必要である（評価項目 b）．

7. おわりに

ソフトウェアの影響波及解析を容易化するための手法を提案した．提案した手法では，設計者が設計プロセスの記録を作成しながらソフトウェアを設計する．保守者は，設計プロセスの記録を，それが持つ索引や関連付けの情報を用いて検索し，影響波及解析に有用な情報を参照することができる．設計プロセスの記録は，設計の思考の流れに沿って作成するため，記録のコストが小さいことが期待できる．

以上の手法を実現したプロトタイプシステム DIG を用いた評価実験では，ソフトウェアの修正工数を，設計生産物だけを用いた従来の場合の 15% 以下にできたのに対し，設計プロセスの記録により設計工数が約 40% 増加した．これは，総開発費の 70% が保守費用という統計値を考慮すると，総開発費に対する効果が得られる値と考えられる．ただし，ソフトウェアのライフサイクルの形態によっては，設計プロセスの記録のコストが負担になることも予想され，今後は設計プロセスの記録のコストを減らすための技術が必要である．さらに，適用できる分析・設計法を広げることも重要と考える．

謝辞 日頃ご指導を頂く葉原耕平会長，寺島信義社長，太田理室長および NTT 交換システム研究所の竹

中豊文氏に深謝します．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．

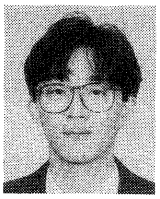
参考文献

1) Osterweil, L.: Software Processes are Software Too, *Proc. 9th ICSE*, pp. 2-13 (1987).
2) Knuth, D.: Literate Programming, *Comput. J.*, Vol. 27, No. 2, pp. 97-111 (1984).
3) Conklin, J. and Begeman, M.: gIBIS: A Hypertext Tool for Exploratory Policy Discussion, *ACM Trans. Office Inf. Syst.*, Vol. 6, No. 4, pp. 303-331 (1988).
4) Potts, C. and Bruns, G.: Recording the Reasons for Design Decisions, *Proc. 10th ICSE 1988*, pp. 418-427 (1988).
5) Guidon, R.: A Model of Cognitive Processes in Software Design, MCC Technical Report, STP-283-87 (1987).
6) Lubars, M.: Representing Design Dependencies in an Issue-Based Style, *IEEE Software*, Vol. 8, No. 4, pp. 81-89 (1991).
7) DeMarco, T.: *Structured Analysis and System Specification*, Yourdon Press (1986).
8) Fjeldstad, R. and Hamlen, W.: Application Program Maintenance Study—Report to Our Respondents, *Proc. GUIDE 48* (1979).
9) Lehman, M.: Programs, Life Cycles, and Laws of Software Evolution, *Proc. IEEE*, Vol. 68, No. 9, pp. 199-215 (1980).
10) 浜田雅樹，安達久人：設計プロセスの再利用による設計ガイド方式，情報処理学会ソフトウェア再利用技術シンポジウム論文集 1991, pp. 43-51 (1991).
11) Ramesh, B. and Dhar, V.: Supporting Systems Development by Capturing Deliberations during Requirements Engineering, *IEEE Trans. Softw. Eng.*, Vol. 18, No. 6, pp. 498-510 (1992).
12) Hartmann, J. and Robson, D.: Techniques for Selective Revalidation, *IEEE Software*, Vol. 7, No. 1, pp. 31-36 (1990).
13) Arango, G., Baxter, I., Freeman, P. and Pidgeion, C.: TMM: Software Maintenance by Transformation, *IEEE Software*, Vol. 3, No. 3, pp. 27-39 (1986).
14) Moriconi, M. and Winkler, T.: Approximate Reasoning about the Semantic Effects of Program Changes, *IEEE Trans. Softw. Eng.*, Vol. 16, No. 9, pp. 980-992 (1990).
15) MacLean, A., Young, R. et al.: Questions, Options, and Criteria: Elements of Design Space Analysis, *Human-Computer Interaction*, Vol. 6, pp. 201-250 (1991).
16) Yakemovic, K. and Conklin, E.: Report on a Development Project Use of an Issue-Based Information System, *Proc. Conference on Computer Supported-Cooparative Work 1990*, pp. 105-118 (1990).
17) Waters, R.: The Programmer's Apprentice: Knowledge-Based Program Editing, *IEEE Trans. Softw. Eng.*, Vol. 8, No. 1, pp. 1-12 (1982).
18) Hausler, P., Pleszkoch, M. et al.: Using Function Abstraction to Understand Program Behavior, *IEEE Software*, Vol. 7, No. 1, pp. 55-63 (1990).
19) Simon, H.: Whether Software Engineering Needs to be Aritificially Intelligent, *IEEE Trans. Softw. Eng.*, Vol. 12, No. 7, pp. 726-732 (1986).
20) 鯵坂恒夫，松本吉弘：ソフトウェアエンジニアリングデータベース KyotoDB の設計と実現，情報処理学会論文誌，Vol. 33, No. 11, pp. 1402-1413 (1992).
21) Myers, G.: *Composite/Structured Design*, Van Nostrand Reinhold Company (1978).
22) Hamada, M. and Adachi, H.: Recording Software Design Processes for Maintaining the Software, *Proc. COMPSAC 93*, pp. 27-33 (1993).
23) 山崎利治：共通問題によるプログラム設計技法解説，情報処理，Vol. 25, No. 9, p. 934 (1984).

(平成5年4月 8 日受付)
(平成6年1月 13 日採録)

浜田 雅樹（正会員）

1960 年生．1983 年慶應義塾大学工学部電気工学科卒業．1985年同大学大学院修士課程修了．同年，日本電信電話株式会社入社．以来，ソフトウェア生産技術の研究・開発に従事．現在，NTT ソフトウェア研究所主任研究員．1990 年から 1993 年まで(株)ATR 通信システム研究所に出向．ソフトウェア再利用，オブジェクト指向技術に興味を持つ．1991 年情報処理学会奨励賞受賞．電子情報通信学会，IEEE 各会員．



安達 久人（正会員）

1964年生．1988年神戸商科大学商経学部経済学科卒業．同年，株式会社神戸コンピュータサービス（現，さくらケーシーエス）入社．システム設計・開発に従事．1990 年 9 月に(株)ATR 通信システム研究所に出向．ソフトウェア設計／保守支援技術の研究に従事．