ソフトウェア工学 80−18
（1991. 7. 19）

仕様記述作業のモニタリングツールとその記録分析

渡辺智弘　　海谷治彦　　佐伯元司

東京工業大学 工学部 電気電子工学科

本論文では，人間がソフトウェアの仕様を作成する過程の履歴を記録するツールを提案し，これを用いた作業履歴の記録実験とその分析を行った結果について述べる．仕様記述者の作業の構造モデルを作成し，それに基づいて作業履歴を蓄積するハイパーテキストベースのツールを作成した．このツールを用いて仕様記述作業のモニタリング実験を行った．この記録を分析することにより，過去に作成したプロダクトに被験者が後戻りする作業が，全体の80%を占めることが分かった．発生原因と仕様書に与える影響の点から，この後戻り作業は大まかに分けて4つの形態が認められ，それぞれの後戻り作業について，支援や発生の抑制についての議論を行なった．

Experimental Analysis of Software Specification Process using a Monitor Tool

Tomohiro Watanabe　　Haruhiko Kaiya　　Motoshi Saeki

Tokyo Institute of Technology

This paper presents a tool for recording personal activities to create software specification, and it is used for experimental analysis of specification process. We have proposed the structual model of human activities and made a tool for accumulating the record of the activities. Our tool is based on Hypertext. By using chronicle analysis we have found that there are backtrack activities between previous products and the current one, 80 % of specification activities have been used for backtracking to previous products. There are many influences in creating specification. We've found 4 patterns from this experiment, and discussed supporting method for backtracking activities for each pattern are the argument for recognition.

1 はじめに

ソフトウェアの仕様化や設計の方法論及び技法について種々の研究がなされ、実用化へ向けての努力もなされている [1], [2], [3], [4]. しかし, 仕様化や設計の過程は人的要因にかなりの部分を依存しているため, これらの方法論や技法を用いたとしても, 人間が行わなければならないような, 高度で複雑な作業は未解決のまま残されている. この問題を解決するためには, 人間にとって, 容易/困難な作業を明らかにして, それを元に, 人間が容易に仕様を記述できる環境や方法論を提供することが, 高品質の仕様を作成する上で重要であると思われる. そのためには, 人間の作業履歴を調べる必要がある. 人間の仕様化/設計作業の履歴データ収集をビデオカメラとインタビューによって行ない, その分析を行なった事例もいくつか報告されている [5],[6]. ビデオカメラやインタビューを用いる手法は, データ収集が実際の作業活動を妨げないという利点があるが, 1) 記録されている情報量が多く, 必要な情報の抽出に手間がかかる, 2) 分析結果が客観性に乏しくなる危険性がある, という欠点もある.

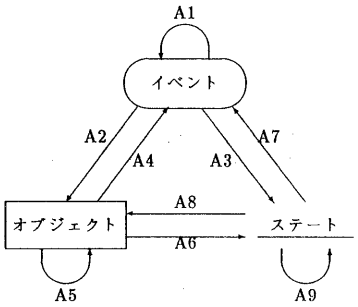
本稿では, ツールを用いて人間の仕様記述作業の履歴を記録し, その分析を行なった結果について述べる. 作業者は, このツールを用いて仕様化/設計作業を進める. その際の作業中の客観的で, 分析に必要なデータが自動的に記録されていく. 記録されるデータは, 作業の種類と要した時間, 回数である.

[6] によれば, システムを理解する過程にある作業者は, システムについて識別した個々のコンポーネントの仕様を記述し, コンポーネント間の関係に従って次に記述するコンポーネントを識別する. また, 作業者は, シミュレーションやバックトラックを行うために, 注目点を過去のプロダクトへ後戻りさせる. この後戻りが頻繁に発生することは, 作業効率を低下させる原因につながると思われる. 本研究では, 電車の切符の自動販売機を例にとり, 仕様化/設計作業実験を行ない, 被験者がツール上で行った仕様記述作業と後戻り作業に対して, その履歴と発生時刻を自動的に記録した. この記録を分析することにより, 後戻り作業の形態と発生率, 発生原因及び平均所要時間を抽出する. これらから, 後戻り作業が仕様記述作業に与える特徴を分類し, 容易に仕様を記述できる環境を提供するための議論を行う. また, 被験者がシステムをどの様に認識しているかを考察する.

2 仕様記述者の作業構造

2.1 作業の種類とその構造

[6] による分析より, 図1に示すような仕様化/設計作業における基本的な作業活動とその間の関係が抽出される. このモデルは, イベント, オブジェクト及びステートの3種のコンポーネントと, コンポーネント間の関係から構成されている. これは仕様記述者がシステムを仕様化する際のモデルでもある. コンポーネントはその記述作業を表し, コンポーネント間の関係は注目点の推移を表す (図1中の表). 例えば, 現在仕様記述者があるオブジェクトの仕様記述を行っていたならば, その次は, オブジェクトを入/出力として必要とするイベント, その他のオブジェクト, またはオブジェクトが持つステートなどのいずれかを識別し, その仕様を記述する. 作業者によって識別されたコンポーネント, その間の関係をまとめてプロダクトと呼ぶことにする.



現在の注目点	次の注目点	コンポーネント間の関係	
イベント	イベント	一連のイベントシーケンス、イベントの追加、場合分け	A1
	オブジェクト	イベントに関連するオブジェクトの識別	A2
	ステート	イベント後のステートの識別	A3
オブジェクト	イベント	オブジェクトに対するイベントの識別	A4
	オブジェクト	オブジェクトの詳細化、追加	A5
	ステート	オブジェクトのステートの識別	A6
ステート	イベント	ステートで起きるイベントの識別	A7
	オブジェクト	ステートに関連するオブジェクトの識別	A8
	ステート	他のステートの識別、ステートの分解	A9

図 1: 作業の構造モデル

2.2 抽出するデータ

本研究では, 実験を通じて以下のデータを抽出する.

1. 被験者が作成したプロダクト

   被験者が作成したプロダクトから, 注目点の推移の履歴が確認でき, システムに対する認識を分析できる.

2. 被験者が行った個々の作業 (コンポーネントの作成, 参照, 削除など)

   被験者の作業が, 新たに識別したコンポーネントの仕様記述作業だったのか, 過去に作成されたプロダクトへの後戻り作業だったのかを確認でき, 後戻り作業の形態, 発生率が抽出できる.

3. 個々の作業の発生時間

   個々の作業の所要時間から, 全体の作業効率に与える影響を分析できる.

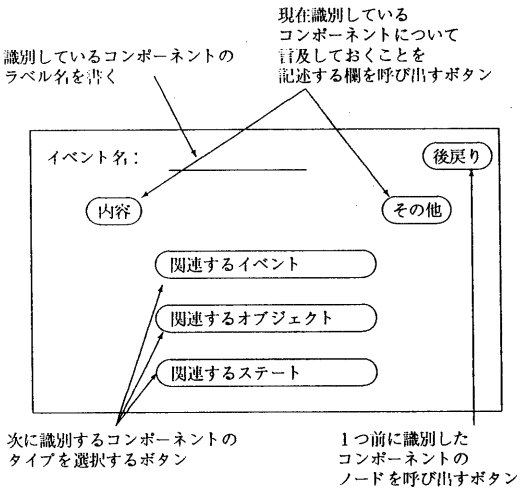


図 2: メニュー画面 (例:イベント識別時)

3 モニタリングツール

3.1 ツールの満たすべき条件

本研究の実験で用いるモニタリングツールは，以下の条件を満たすものでなければならない．

- ツールによるモニタリングが，被験者の作業をできる限り妨げてはならない．そのため，操作が単純(例えば，覚えなければならないコマンド数が少ないなど) であることが不可欠である．
- 識別したコンポーネントを個別に蓄積，管理し，関連したコンポーネントを直ちに呼び出すことができる．
- ツール上で行える作業が，図1に示した仕様記述者の作業構造をベースにしたものとなっている．特に，作業者の注目点を移動させる操作は，図1中の矢印と一致していなければならない．

3.2 ツールの実現

本研究では，上記の条件を満たしたツールを，ハイパーテキスト [7] をベースにしたメニュー画面形式で作成した．ハイパーテキストを設計作業支援に用いることの有用性については，[8] で議論されている．本ツールで作業が見ることができるのは，現在記述しているコンポーネントのメニュー画面である (図2)．1つのコンポーネントのメニュー画面ノードのみを提供することで，被験者の思考がそのノードに集中できるようになっている．それにより，ツールの特性により被験者の注目点が移動することのないようにしている．さらに，分析時に被験者の注目点の所在を明らかにすることができる．各ノードは，オーバーレイ形式のウィンドウとして表示され，被験者が各ウィンドウを移動させて配置を整理しても，同時に複数枚の完全なメニュー画面を表示できない程度の大きさを持っている．これにより，たとえ複数のウィンドウがオープンされていても，作業者がどのコンポーネントに注目しているかが特定できる．

このツールはハイパーテキストベースであるが，ブラウザとファイルボックスを提供していない．そのため，一部しか見えないウィンドウのメニュー画面全体を見るには，そのウィンドウ上でマウスボタンをクリックする以外に手段はない．さらに，実験中のディスプレイをビデオカメラで記録し，収集データの不足がでないようにした．

3.3 ツールの使用法

被験者は上記のツールを用いて以下の作業を行うことができる．

1. コンポーネントの名前を記述する．
2. “内容” ボタンを選択し，そのコンポーネントについての仕様を記述欄に記述する．仕様とは別に記述すべき内容は，“その他” ボタンを選択し，その記述欄に記述する．
3. 次に識別したコンポーネントの型を選択し (例えば，“関連するオブジェクト” ボタンを選択し)，新たなメニュー画面上でそのコンポーネントについての作業を進める．
4. “後戻り” ボタンを選択し，1つ前に作業を行なったメニュー画面に注目点を戻す．または，一部が見えているウィンドウを選択し，そのメニュー画面に注目点を戻す．

本ツールは，被験者の作業に従ってノードとリンクを記録する．このノードとリンク，被験者が行った作業及び作成された仕様は，表1に示すような対応関係を持っている．

表 1: ツール，仕様記述作業，仕様書の構造の対応関係

モニタ	被験者の作業	仕様書
ノード	コンポーネントの識別	コンポーネント
リンク	注目点の移動	コンポーネント間の関係
タイムスタンプ	作業発生時刻	—

4 実験と結果

4.1 実験方法

本研究では，モニタリングツールを用いて，以下のような手順で被験者の作業の履歴を記録する実験を行なった。

1. 被験者にツールの使用法を理解させる．
2. 被験者が仕様を記述する例題システムについて，その例題の名称のみを与える．実験者は，ツールの使用法を説明した後，“このツールを用いて，... の仕様化作業を行なってください”，とだけ被験者に告げた．
3. 被験者にツールを使用させて，作業を行なわせ，識別したコンポーネントの仕様を自然言語で記述させる．

4.2 問題の特性と被験者

本実験で例題として取り上げたシステムは，“電車の切符の自動販売機” である．このシステムは，データ駆動型ではなく，リアルタイムで動作するイベント駆動型のリアクティブシステム [4] である．被験者には，5 ～ 10 年のプログラミング経験を持つ人間から，無作為に 4 人を選んだ．被験者は，いずれもユーザという立場から，このシステムの使用経験をかなり積んでいる．しかし被験者は，ハードウェアの仕様は与えられず，内部構造を知らない．また，過去このシステムについて設計あるいはプログラミングをした経験は持っていない．

4.3 実験結果

実験の結果，被験者が作業に必要とした時間は 1 時間 20 分 ～ 2 時間であった．個々のプロダクトに対して行なった作業の所要時間は，単位作業あたり約 50 秒であった．作業単位の区切りは，メニュー上のいずれかのボタンを選択し，別のメニュー画面に切替えた時点とする．従って，作業を始めるためにあるメニュー画面を表示させてから，他のメニュー画面に作業を移行するまでが，1 つの作業単位である．表 2は，ツールによる記録から得られた被験者の作業履歴の一部である．この表は，作業の発生時刻もしくは発生時間間隔，作業の形態 (1:ノードの作成，2:過去に作られたノードの確認，3:過去に作られたノードにリンクを追加，4:ノードの削除，5:過去作られたリンク先の変更)，コンポーネントの型 (E:イベント，O:オブジェクト，S:ステート)，コンポーネントのラベル名からなる．例えば，第 1 行目は作業開始から 58 分 42 秒後にステート型ノード “最初の状態” の作成作業を行なったことを表している．第 2 行目はそれより 1 分 26 秒後に新しいノード “お金の投入口” を作成したことを表している．また，一人の被験者について，例題システムのコンポーネントを識別した履歴を示す (図 3)．楕円，長方形，下線の部分が各々イベント，オブジェクト，ステート型のコンポーネントを，そこに書かれている文字列がコンポーネント名を表す．図中の番号は，コンポーネントが作られた順を表す．また，矢印はコンポーネント間のリンクを表す．例えば，被験者は最初に “最初の状態” というステート型のコンポーネントを作成し，“関連するオブジェクト” ボタンを使って，“お金の投入口” というオブジェクト型のコンポーネントを作成している．

表 2: 被験者の作業履歴の一部

0:58:42	1	S	最初の状態
+1:26	1	O	お金の投入口
+0:56	2	S	最初の状態
+0:04	2	O	お金の投入口
+0:28	1	O	金額表示器
+0:10	1	E	金額を表示する
+0:20	2	O	金額表示器
+0:20	2	E	金額を表示する
+0:49	1	E	お金が投入された
+1:28	1	S	金額の積算
+2:58	2	E	金額を表示する
+0:06	2	O	金額表示器

1=ノードの作成, 2=ノード確認, 3=ノード削除
4=リンク追加, 5=リンク先変更

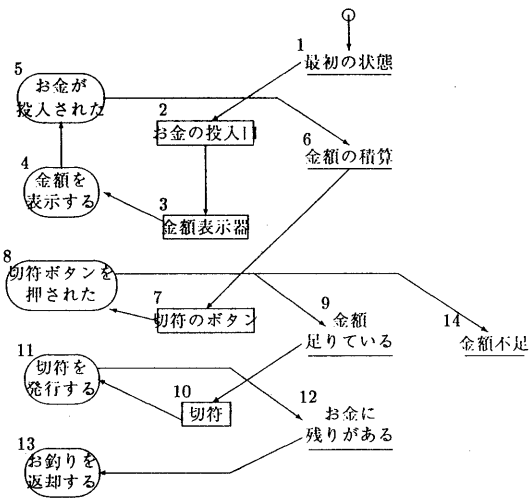


図 3: 被験者の識別履歴の一部

5 分析と議論

5.1 作業形態

実験の結果から，被験者の作業は，新たなノードの作成とその記述を行なう通常の作業と，コンポーネント間の関係を参照/修正する後戻り作業に分類することができる．

1. ノードの作成

   この作業は，現在作成しているノードから，新しいノードを作り出す作業である (図 4)．この作業は，新たにコンポーネントが識別されたときに生じる．

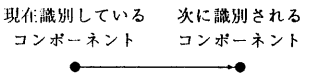


図 4: ノードの作成

2. 後戻り作業

   後戻り作業には，過去のノードを呼び出してその内容を確認する作業と，その結果，これまでに構築されたリンク構造を変化させる作業などの 4 形態に分けられる．

   (a) ノードの確認

   この作業は，過去に作成したノードを呼び出し，そのノードの内容を確認する作業である (図 5)．

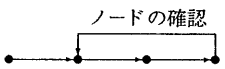


図 5: ノードの確認

   (b) リンクの追加

   この作業は，ノードの作成作業と似ていて，これまでに構築されたリンク構造に新たなノードをリンクする作業である．ノードの作成が現在作成したノードから新たなノードを作るのに対し，リンクの追加は，過去に作成され既に次のノードを持つノードから，新たなノードを作成したり (図 6)，既存のノードへのリンクを追加したりする作業である．例えば，図 3で “切符ボタンが押された” というノードに 14 番目のノード “金額不足” というノードを作成し，リンクを追加する作業が見られる．

   (c) ノードの削除

   この作業は，現在までに作成されたノードを削除する作業である (図 7)．ノード自身を削除するのでなく，ノードに入ってくるリンクをすべて削除し，結果的に孤立し続けるノードを作り出す作業もこのカテゴリに入れる．

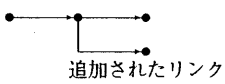


図 6: リンクの追加

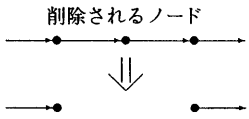


図 7: ノードの削除

   (d) リンク先の変更

   この作業は，現在までに構築されたリンクの出所あるいは行き先を変更する作業である (図 8)．

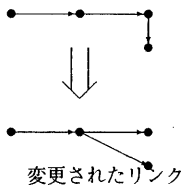


図 8: リンク先の変更

5.2 発生率，所要時間，発生原因

表 3に全被験者合計の形態別作業発生率と時間比率を示す．個々の作業の平均所要時間を見ると，全被験者を通じて，ノードの作成の所要時間が最も長い．しかし全体としては，平均 2 分以下という所要時間である．この時間は，各形態の作業について，多くの時間を要求する作業ではなかったことを示していると考えられる．

5.2.1 ノードの作成:コンポーネントの識別

被験者が対象システムについて熟知していれば，作業の多くが，この形態の作業であると考えられる．しかし実際には，発生率にして 13%程度，時間比率にして 30%程度の割合しか占めていない．つまり全体の作業のほとんどが，次節で述べるような後戻り作業で占められていることになる．

5.2.2 ノードの確認

この作業の発生率は，全被験者の平均にして 81%を占める割合であった．また，時間比率からみても 62%の時間を占めていた．これは，仕様化作業を進める上で最も著しく発生した後戻り作業である．先に述べたように，被験者はシステムを理解する過程において，一貫性のチェックのためのシミュレーションや，矛盾を発見した部分へのバックトラックを行なう．従って，ノードの

表 3: 被験者全体の作業の形態別発生率と所要時間

作業のタイプ	発生回数	発生比率	平均時間	時間比率
ノードの作成	65	13.4%	1'56"	32.5%
ノードの確認	393	81.2%	0'37"	62.1%
ノードの削除	9	1.9%	0'39"	1.5%
リンクの追加	15	3.1%	0'48"	3.1%
リンク先の変更	2	0.4%	1'36"	0.8%
全作業	484	100%	0'48"	100%

確認作業の発生原因は，被験者がこれらの必要性を認識したものと考えられる．また，他の全ての後戻り作業を行なう場合にも，適切なコンポーネントへの後戻りを行なうために必要であったとも考えられる．実際，被験者の作業履歴を見ると，他の後戻り作業（リンクの追加，ノードの削除，リンク先の変更）が発生する場合，その前にほとんど1回以上の“ノードの確認”が発生していることが分かった．

5.2.3 リンクの追加

この作業は，全被験者についての発生率で3%，時間比率で3%程度である．各数値はそれほど大きなものではないが，ほとんどの被験者について，共通して2番目の頻度を持って発生していた．発生の原因は，被験者が現在までの作成したプロダクトについて詳細化を行なうか，システムの分岐する処理シーケンスを追加する必要性を認識したものと思われる．被験者ごとの発生数の差異は，被験者のシステムに対する認識や，プロダクトの識別の順序の差異に起因するものと思われる．

5.2.4 ノードの削除とリンク先の変更

これらの作業は，被験者によっては現れないものがほとんどで，両方合わせても2%程度の率しか占めていない．しかし，この作業が発生した被験者のみを対象とすれば，発生率でノードの削除は7.3%，リンク先の変更は1.6%を占め，時間比率でノードの削除は5.9%，リンク先の変更は3.2%を占めていて，“リンクの追加”に等しいかそれを上回る率を占めている．これらの作業の発生原因は，例えば，イベントであるものをステートであるとして識別するというような，コンポーネントの概念を誤解していたり，システムが持つべきコンポーネントの構造と，実際に被験者が識別したコンポーネントの構造との間のギャップ，すなわち対象システムに対する認識が誤解していたことを発見してしまったためと思われる．

5.3 システムに対する認識

各被験者の残したコンポーネントの識別履歴(例:図3)から，以下のようなことが分かる．

1. 陽に識別されないコンポーネントが存在する

   これは特に，オブジェクトに見られた現象である．例えば被験者は，“お金を入れる”などのイベントの識別はしたが，そこから“お金”というオブジェクトを識別しなかった．これは，イベントの中で，オブジェクトがすでに目的語などで出現してしまい，そのため被験者が，そのオブジェクトはすでに識別されたと錯覚したためと考えられる．

2. 対象システムをlocalな視野で捉える

   各被験者の識別履歴を見ると，コンポーネント間の関係が直接的な関係を表しているものが多く，例外として，図3の初めの部分で，システムに存在するオブジェクトを羅列的に識別している部分がある．この場合も，オブジェクトを入出力として用いるイベントを識別してからは，localな視野で識別作業を進めている．これは，本研究の実験で使用したモニタリングツールの特性（プロダクトの全体像を表示するブラウザを持っていないという特性）から，被験者が現在識別しているコンポーネントのみに注目点を奪われてしまい，対象システムに対してglobalな捉え方をしにくいという理由が考えられる．しかし，被験者の持っている経験などから，localな捉え方をする傾向があったということも考えられるが，本研究によるこれまでの分析だけでは，詳しく述べることはできない．

5.4 環境の支援および改善

本節では，実験の結果得られた作業形態とその特性をもとに，どれが必要な作業で，不要な作業であるかを考察し，必要な作業への支援形態，および不要な作業をどのようにして軽減させるかを考察する．つまり，本研究で作成したモニタリングツールのようなハイパーテキストベースのツールを仕様化/設計作業の支援ツールとして使用する際，作業効率の向上に向けて支援されるべき点，改善されるべき点について検討を行なう．

5.4.1 支援の方針

後戻り作業の発生はできる限り抑制され，また，後戻り作業が発生した場合には，仕様記述者が作業を容易に行なえるように支援されなければならない．本研究で4つの形態が示された後戻り作業は，その発生原因により以下の2つの組に分類できる．

1. “ノードの確認”と“リンクの追加”

   発生が必然的であり，発生したならば容易に行なえるように支援されなければならない作業．

2. “ノードの削除”と“リンク先の変更”

   被験者の誤解により発生するため，できる限り発生が抑制されなければならない作業．

5.4.2 ノードの確認とリンクの追加

“ノードの確認”は，作業中の被験者が過去に識別したプロダクトを参照する他，内容を修正したり，他の全ての後戻り作業を行なうための対象箇所の確認のために

発生する．この作業の発生頻度を減少させることは，すなわち作業効率の改善につながるであろう．しかしながら，この作業は，試行錯誤で仕様化/設計作業を進めていくという人間本来の特性がある以上，対象システムの一貫性を達成しするために不可欠で，最も重要な作業と思われる．

本研究で作成したツールは，現在識別しているプロダクトに関するメニュー画面のみを表示することにより，被験者の注目点をそこに集中させるよう試みていた．また，“後戻り”ボタンによって，1つ前のプロダクトに戻る機能しか持っていなかった．そこでツールにプロダクトの全体像を表示させるブラウザ機能を持たせることにより，被験者が注目したい過去のプロダクトに，少ない操作で後戻りできる支援ができると思われる．

“リンクの追加”は，現在識別しているプロダクトを詳細化したり，システムの分岐する処理シーケンスを記述したりする場合に発生する．この作業は，被験者のシステムに対する認識の差異により発生する箇所が異なるが，必要が生じた場合には適切かつ容易に行なわれなければならない．

本研究で使用したモニタリングツールは，次に識別するコンポーネントに注目するためのボタンとして，イベント，オブジェクト，ステートをそれぞれ1つずつのみ提供していた．これに対して，“リンクの追加”作業は，次に識別するコンポーネントとして，例えばコンポーネントを2つ以上必要とする．この分岐の数を満たすようにボタンを用意するには，被験者の対象システムに対する認識の構造を，ツールがあらかじめ知っていなければならない．これを直接実現することはできないが，メニュー画面を改良し，必要に応じたプロダクトのボタンを容易にいくつも提供することができれば，対処できるものと思われる．また，プロダクトの全体像を表示するブラウザによって，対象ノードを特定する支援ができると思われる．

5.4.3 ノードの削除とリンク先の変更

これらの作業は，その発生原因がコンポーネント概念の誤解や，対象システム本来の構造と，実際に被験者が識別したコンポーネントの構造のギャップであるため，今回の実験のように経験的に熟知したシステムについての仕様記述作業であれば，発生する機会はほとんどないと思われる．つまり，対象システムの構造に対する深く正確な認識があれば，それだけでこれらの作業の発生を押えられる．しかし，実際には熟知していないシステムの仕様化も行なわれるため，そのような場合には，発生比率はかなり高くなるものと予想される．対象システム本来の構造を捉えようとする仕様化/設計技法，例えばジャクソンシステム開発法[3]など，を導入したツールとすれば，これらの作業を行なう頻度が減るものと思われる．

5.4.4 ブラウザの提供

今回使用したツールに対して，後戻り作業の発生を抑制し，また，発生した際に容易に後戻り作業を行なえる支援方法として，ブラウザの表示機能を追加することが有効であると考えられる．本節ではどのようなブラウザを導入すればよいかを実験結果より考察する．

一概にブラウザを提供すると言っても，被験者が作成した全てのノードを表示することはあまり意味がない．先にも記したが，被験者は各後戻り作業の発生に先立って，少なくとも1回以上の“ノードの確認”を行なう．しかも，いくつ前のノードに後戻りするか，その範囲が限定される．図9は，現在のノードからn本のリンクをたどって後戻りできるノードに対して，実際に全被験者が行なった“ノードの確認”作業の平均発生頻度である．被

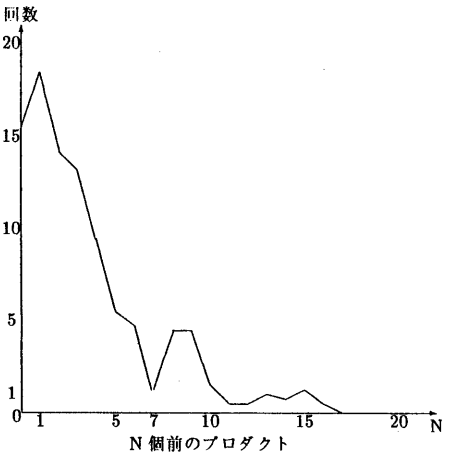


図 9: ノード確認の発生頻度

験者の全作業履歴と照らし合わせ，この図から次のようなことが分かる．

- “ノードの削除”，“リンク先の変更”を行なう場合，被験者は高々7本前にリンクされたノードまでしか参照しない．

8本以上のリンクをたどって到達できるノードを参照する必要があるのは，全コンポーネントを通じてのシミュレーションを行なう場合に限られると思われる．従って，後戻り作業の環境を改善するためにツールに持たせるブラウザとしては，現在作成中のプロダクトと，そこから7本以下のリンクをたどって到達できるノードを表示できる機能を満たしていれば良いことであろう．

5.5 メニュー画面の改良

先に述べたように，本研究で実験に使用したツールは，次に作成すべきプロダクトの属性(イベント，オブジェクト，ステート)を全て1つずつ表示している．これで

は，被験者が“リンクの追加”を行なう場合に困難であり，また，リンク先のノードの有無に関わらず次のノードを呼び出すボタンがあるため，シミュレーションを行なう場合にも困難を生じる．

次に作成するプロダクトの属性を決定するのは被験者自身である．従って，これらの困難を改善するために，仕様記述者が必要を認識した場合にのみ，次のノードをプロダクトの属性別に生成する機能を持つメニューボタンを設定し，次のプロダクトの属性を選択する3つのボタンの代替とすれば良いと考えられる．

ジャクソンシステム開発法のような固有の技法を導入するには，その技法が提供しているシステムのモデルに合致したメニューボタンを設定する必要がある．例えば，ジャクソンシステム開発法の場合，Action, Entity, Entity Structure, State Vector, Process といった要素に該当するメニューボタンが考えられる．

6 おわりに

6.1 研究のまとめ

本研究を通じて，以下のようなことが達成できたと思われる．

- 人間が仕様記述作業を行なう上で，その作業効率を低下させる原因になると思われる作業形態を抽出し，その作業発生率と時間比率から定量的分析の一例を示した．これにより，本研究で用いた手法が，このような分析に有効であることを確認できた．
- 後戻り作業の人間に対する難易度を計ることはできなかったが，作業効率の改善に向けて，これらの作業に対する環境の支援および改善の方法を具体的に提案できた．

6.2 今後の課題

今後，この研究をさらに進め，人間にとって容易に行なうことができるような，仕様化の方法論および支援ツールに反映するには，以下に挙げるような点が課題として挙げられる．

- 同様の実験の継続

  実験を行なう以上，得られるデータに一般性が得られるようでなければならない．本研究で行なった実験の被験者は4人である．これだけではデータの一般性が十分でないと思われる．そのため今後も，本研究と同様の実験を継続して行なう必要がある．また本研究では，実験の被験者を無作為に選び出したが，作為的に選び出した被験者について実験を行なう必要もある．

- 問題の変更

  本研究で取り上げた例題は，イベント駆動型システムであったが，データ駆動型システムなどを例題に取り上げて，同様の実験を試みる必要がある．

- モニタリングツールの違いによる評価

  本研究で用いたモニタリングツールのようなハイパーテキストベースのツールを仕様作成の支援ツールとして使用する際の，後戻り作業の支援/改善方向を提案した．実際にブラウザを提供するなど，後戻り作業を容易にするための，支援および改善の方法をモニタリングツールに反映し，そのツールを用いた作業の履歴を記録する実験を行なうことにより，本研究で得られた被験者の作業履歴と比較し，作業効率の変動に関する評価をすることが重要と思われる．そこから，人間にとって容易に仕様化作業を行なえる環境や支援ツールに反映されるべき提案ができると思われる．

参考文献

[1] G. Booch. Object-oriented devlopment. *IEEE Transactions on Software Engineering*, Vol. 12, No. 2, pp. 211–221, 1986.

[2] T. DeMarco. *Structured Analysis and System Specification.* Yourdon Press, 1978.

[3] M. A. Jackson. *Principles of Program Design.* Prentice-Hall, 1983.

[4] D. Harel, H. Lachover, A. Naamad, A. Pnueli, R. Sherman, and A. Shtul-Trauring. STATEMATE: A working environment for the development of complex reactive systems. *Proc. of 10th ICSE*, pp. 396-406, 1988.

[5] 佐藤隆, 内田修市, 門田充弘, 山下紘一. 設計過程における人間の思考過程の分析. 情報処理学会第36回全国大会, Vol. 2R-6, pp. 1613–1614, 1988.

[6] 池克俊, 海谷治彦, 佐伯元司, 本間学. ソフトウェア仕様記述過程分析のための基礎的実験. 情報処理学会研究報告, Vol. 89-SE-68, No. 5, 1989.

[7] Frank G. Halasz. Reflections on notecards: Seven issues for the next generation of hypersystems. *Communications of the ACM*, Vol. 31, No. 7, pp. 836–851, 7 1988.

[8] 田中直樹, 中島毅, 藤岡卓, 上原憲二, 高野彰. ハイパテキストを用いた設計プロセス支援ツールの試作. 情報処理学会研究報告, Vol. 89-SE-68, No. 7, 1989.