仕様作成会議の発話履歴を用いて仕様書を作成する方法

三浦信幸† 海谷治彦† 佐伯元司‡

† 東京工業大学 理工学研究科 電気電子工学専攻

‡ 東京工業大学 工学部 情報工学科

〒 152 東京都目黒区大岡山 2-12-1

本報告では，仕様作成会議の参加者の発話を時系列上に並べた発話履歴から仕様書を作成する方法を提案する．本方法では，発話履歴を構造化し，それを基に構造化された仕様書を作成する．毎回の会議毎に作成作業を行い，仕様書を段階的に作成する．会議の履歴から直接，仕様書を作成するため，会議で議論されながらも仕様書には記述が残らないような欠落事項の発生を防止できる．また，会議毎に記録を構造化することで，各会議中にその記録を参照して議論することができるため，効率的な議論の促進，不明瞭な事項の発生・仕様書内の矛盾の発生・一貫性の欠如の防止が可能である．さらに，本方法を過去の会議の記録に適用し，発話履歴から仕様書が作成可能であり，本方法が仕様書作成作業を有効に支援することを確認した．

仕様化プロセス，仕様作成会議，発話履歴

A Method to Construct Specifications from Verbal Histories in Meetings

Nobuyuki Miura† Haruhiko Kaiya† Motoshi Saeki‡

† Department of Electrical and Electronic Engineering, Tokyo Institute of Technology

‡ Department of Computer Science, Tokyo Institute of Technology

Ookayama 2-12-1, Meguro-ku, Tokyo 152, Japan

E-mail address: {miura, kaiya, saeki} @cs.titech.ac.jp

In this paper, we propose a method to construct specifications from verbal histories of meetings to develop specifications. Verbal histories are time-ordered speeches in meetings. In this method, we structure verbal histories at first and then construct the structured specifications based on them. These works are done after every meetings, so specifications are constructed step by step. Constructing directly specifications from histories of meetings allows us to avoid lost items which were argued in meetings. Arguing with refering histories of meetings and specifications also allows us to avoid vague items, contradictions and inconsistencies in specifications and supports progresses of meetings. And we apply this method to records of experimental meetings, so assesses this method to support constructing specifications.

Specification Processes, Meetings for developing Specifications, Verbal Histories

1　はじめに

ソフトウェア開発における仕様作成段階では、多種多様な作業者による共同作業が行われる．そして，その段階における代表的な作業形態である会議では，口頭による会話によって作業者間の情報交換を行うことが多い．我々がソフトウェアの仕様作成会議を支援する目的で分析対象とした会議 [1] の多くは，作業時間の大半を作業者間の会話に費やしている．また，Gary らが分析した設計会議 [2] の多くも作業時間の 9 割を会話に費やしていることが報告されている．このような背景から，我々は仕様作成会議の参加者の発話，特に発話を時系列に並べた発話履歴に着目して，仕様作成のための会議支援方法について研究を進めてきた．通常のソフトウェアの仕様作成は複数回の会議を通して行われ，仕様書や中間生産物である文書は会議外で作成されることが多い [3]．したがって，会議自身の支援とともに，発話履歴からの文書作成の支援が必要であると考えられる．

このようなソフトウェアの仕様作成作業を支援するための協調作業モデルやツールは数多く作成されている [4, 5, 6, 7, 8, 9, 10, 11]．これらに対し，我々は会議における発話そのものに重点をおき，発話から仕様書や中間文書を作成できるような作業方法とその支援ツールが必要であると考えている．そのような方法を獲得するために，計算機による支援を受けていない “普通の” ソフトウェアの仕様作成会議の内容を分析する研究を行ってきた [1, 3, 12]．これらの分析を基に，本報告では，仕様作成会議の発話履歴から仕様書を作成する方法を提案する．さらに，過去の会議の記録に対して本方法を適用し，会議の発話履歴から仕様書が作成可能であり，本方法が仕様書作成作業を有効に支援することを確認した．

第 2章で仕様作成会議の分析結果から得られた，支援すべき点を述べる．次に，第 3章で発話履歴から仕様書を作成する具体的な方法を提案する．第 4章では，本方法を実際の会議中で活用する方法を提案する．この活用法を用いることにより，会議の効率的な進行が期待できる．第 5章で過去の会議の記録に本方法と第 4章の活用法を適用して，仕様作成作業を有効に行うことができること示す．最後に第 6章で問題点の考察と今後の課題などについて述べる．

2　仕様作成作業における支援

我々は，ソフトウェアの仕様を作成する会議をビデオカメラで記録し，その記録内容のプロトコル分析を行った．その結果，仕様作成作業におけるいくつかの典型的な不具合な点を発見した [1, 3, 12, 13] ．それらを解決するためには，以下のような支援が必要である．

- 欠落事項の発生の防止:
  会議で議論され，かつ否決されていないにも関わらず，仕様書にそれに関する記述が残っていない項目や説明 (これらを我々は欠落事項と呼ぶ) の発生を防止する．
- 不明瞭な事項の発生の防止:
  議論が十分でなかったために仕様書内の事項の説明が不足し，不明瞭な事項が発生してしまうことを防止する．
- 効率的な議論の促進:
  同じ内容の議論の繰り返しや結論を出さない議論，決定事項の度重なる変更などの発生を防止し，効率的な議論を促進する．
- 仕様書内の矛盾の発生・一貫性の欠如の防止:
  仕様作成会議で，既に議論した事項を変更したにも関わらず，その変更に影響のある事項について，変更・確認を施さなかったために生ずる，仕様書内の矛盾の発生・一貫性の欠如を防止する．

3　発話履歴からの仕様書作成方法

第 2章のような支援を行うために，以下のことが必要となる．

- 欠落事項の発生の防止のために，仕様作成会議の履歴を完全にとり，その履歴から直接，仕様書を作る．
- 不明瞭な事項の発生の防止および効率的な議論の促進のために，仕様書の部分と仕様作成会議の部分との対応づけを行い，仕様書の部分から過去の議論を検索できるようにする．
- 効率的な議論の促進や仕様書内の矛盾・一貫性の欠如を発見をしやすくするために，意味的な関係に基づいて，仕様書を階層構造化する．意味的な関係とは，例えば，
  - ある問題 (issue) とその問題を解決するために必要な副問題 (sub issue)
  - 抽象化されたもの (class) とその具体的なもの (instance)
  - ある事項とその事項の変更により影響を受ける事項
  - ユーザーインターフェース上のメニューとそのメニュー内のサブメニュー

  などである．

したがって，会議の履歴および仕様書を特定のデータ構造に従って構造化して作成し，このデータ構造を毎回の会議毎に作成する必要がある．会議中には前回の会議までの記録を参照しながら議論することができ，不明瞭な事項の発生の防止，効率的な議論の促進，仕様書内の矛盾の発生・一貫性の欠如の防止が可能となる．

以下の節では，そのデータ構造と会議履歴及び仕様書の構造化作業の手順について述べる．

3.1　会議履歴・仕様書のデータ構造

本方法では，図 1のような会議履歴・仕様書のデータ構造を用いる．会議履歴とは，発話履歴と会議中に提示された図などを構造化した会議の記録である．

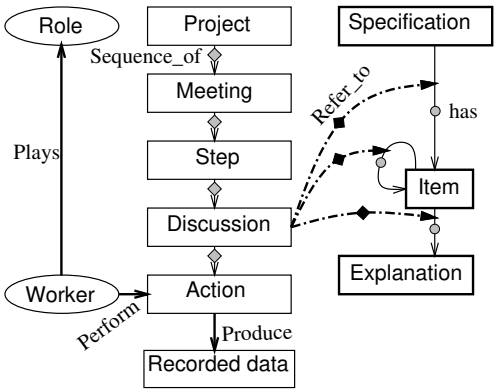

図 1: 会議履歴・仕様書のデータ構造

図 1の最左列は会議の参加者に関するデータ構造，中央列は会議履歴に関するデータ構造，最右列は仕様書に関するデータ構造である．データ構造中の各コンポーネントは以下のようなものである．

- Project : 仕様作成会議が作成しようとしている対象システムの名前などからなるプロジェクト名のラベル．
- Meeting : 何回めの仕様作成会議かを識別するためのラベル．
- Step : 1 つの仕様作成会議中における段階を識別するためのラベル．議論の目的を基に区別して，“発注段階”，“計画段階”，“説明段階”，“レビュー段階” などをあげられる．
- Action : 会議の参加者の “発話” や白板などによる “図示” を識別するためのラベル．それぞれの action は，誰が発話や図示を行ったかを示す，worker のデータを持つ．
- Discussion : いくつかの action の集まりを示すラベル．仕様書中の item や explanation と関係づけられている (図 1の Refer_to 関係)．この関係には，“提案”，“参照”，“却下” という属性があり，関係を新規作成するのか，参照するのか，否決するのかを示す．
  なお，1 つの discussion は複数の item，explanation と関係づけても良い．また，1 つの action は 0 個以上の discussion に属するようにすることができる．
  したがって，冗談などの意味のない action は，discussion には入れなかったり，複数の item などに言及している action は複数の discussion に属するようにしたりすることができる．
- Role : 会議の参加者の役割を示すラベル．参加者の役割は，“発注者”，“ユーザ”，“設計者”，“製作者”，“調整者”，“保守者” などがある．
- Worker : 会議の参加者名を示すラベル．
- Specification : 仕様書の Top Level を表す．
- Item : 仕様書の項目のラベルである．仕様書になった時点では，仕様書内の章・節などの題名となる．
- Explanation : item の説明となる，文章や絵である．仕様書になった時点では，章・節の具体的な内容となる．
- Recorded data : 会議の参加者が行った発話の音声情報・図示の画像情報である．

このデータ構造の仕様書部分は，階層構造を持っている．例を図 2に示す．ただし，図を見やすくするため，item と Specification との relation の一部を省略してある．このような item の階層構造は，仕様書の章・節などの構造となる．

図 2で，“FILE 入出力” という item を “LOAD” という item の “親 item” であると呼び，逆に “LOAD” という item を “FILE 入出力” という item の “子 item” であると呼ぶ．また，“グラフ” という item のような，直接の親 item が Specification のみで子 item を 1 つも持たず，それまでの作業で一度も他の item の子 item になったことがないような item を “孤立 item” と呼ぶ．

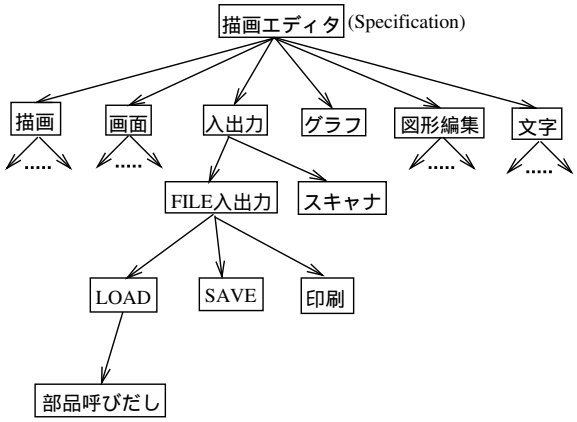

図 2: 仕様書の階層構造の例

3.2 仕様書作成手順

仕様書作成作業は，計算機上に蓄えられた仕様作成会議の発話履歴 (音声情報) および画像情報を基に，図 1のようなデータ構造の会議履歴および仕様書を作成することである．ここでは，その作業手順について述べる．

仕様書作成作業は，原則的には会議外で行うことを想定している．もちろん，書記のような役割の人が会議中に行っても良い．また，前回までの会議の発話履歴の処理は，必ず次回の会議までに終えておかなければならない．

仕様書作成作業の作業者が，発話履歴を 1 回処理するだけで，会議履歴及び仕様書を構造化することは困難であるため，次の 4 段階によって，段階的に作業を行う．

1. 発話履歴を仮の **discussion** に区切る (Step 1)
   仕様書作成作業の前処理として，発割履歴を仮の discussion の単位として大まかに区切る．なお，本方法は仕様書を得ることが目的のため，以下のような仕様書と直接的な関係の薄い内容の発話は，以後の作業では作業対象としない．
   - 作業日程に関する話題
   - 作業分担に関する話題
   - 既存の類似システムに関する検討

- 作成するシステムのプラットフォームの検討

2. **discussion の詳細化と item の抽出** (Step 2)
   発話履歴から item を抽出し，発話履歴を discussion の単位に細かく区切る．
3. **explanation の抽出** (Step 3)
   各 item に対して，発話履歴・画像情報から explanation を抽出する．
4. **item 間の階層構造の構築** (Step 4)
   item 間の意味的な階層構造を構築する．

3.2.1 発話履歴を仮の discussion に区切る (Step 1)

通常の 1 回の会議時間 (1 時間 ~ 3 時間程度) から考えると，会議の発話履歴を全く分割せずに作業を行うことは困難であると考えられる．適度な大きさに分割する必要がある．つまり，この段階の作業の目的は，

- 作業者が会議の大まかな流れをつかむこと．
- 後の作業がしやすいように大まかに発話を区切ること．

である．

まず，少しづつ発話を聞き，必要に応じて前後の発話も聞き直しながら，ある 1 つの内容について言及している会話のグループを同定し，仮の discussion の区切りとする．この仮の discussion の単位は，数十 action が目安である．区切る手がかりは，以下のようなものがある．

- 図 1に示す Step が変わった場合．Step が変わったかどうかの判断は，[1] の各 Step の Role ごとの発話分布の分析結果を基にして行う．
- 会話が一定時間，途切れている場合．
- 話題の転換を図るようなキーワードが出てきた場合．(では，次に，など．)
- 会話中に出てくる，対象システムのコンポーネントのキーワードが変化した時．キーワードは，前回までの仕様書作成作業で抽出された item 名を参考にすることも出来る．

これらの仮の discussion には内容を示すラベルをつけ，ラベルをリストアップしておく．議論の内容が既出の item に関するもののときは，その item 名をラベルにしておけば良い．そうでないときは，item 名になりそうな大まかなラベルを付ける．

なお，本方法は仕様書を得ることが目的のため，以下のような内容の発話は，仕様書と直接的な関係が薄いため，その内容だけの discussion を作り，別の分類として以後の作業では作業対象としない．

- 作業日程に関する話題
- 作業分担に関する話題
- 既存の類似システムに関する検討
- 作成するシステムのプラットフォームの検討

この段階の作業を効率的に行うための補助的な方法として，以下のようなものがある．

- 会議に参加した人が作業する．
- 仕様書作成作業の作業者が作業前に会議のビデオを見て，会議の大まかな流れをつかんでおく．
- 会議中に書記などの人が，大まかな区切りを計算機上に記録しておく．

3.2.2 discussion の詳細化と item の抽出 (Step 2)

この段階の作業は，仕様書の章や節の見出しとなる item を抽出し，仕様書の骨組みを作成する．さらに，仕様書と会議中の発話との対応をつけるために，discussion を詳細に区切る．

まず，仮の discussion の単位で区切られた発話内容が前回までの作業で抽出された item に該当しなければ，その単位内の発話から item 名を 1 個以上抽出する．item 名は主に，対象システムのコンポーネント名が該当する．1 つの discussion の大きさは，後述の図 6のように数 action が目安である．

次に，discussion から item 間の relationship へ関係をつける．この際に，仮の discussion の単位を “提案”，“参照”，“却下” の分類に基づいて詳細に区切る．また，discussion には画像情報も含めておく．これらの分類の意味は，以下のようなものである．

- 抽出した item 名がそれまでの発話履歴の中で一度も出現していないような discussion からの関係は，“提案” に分類する．
- item 名の削除を明示的に発話している action を含む discussion からの関係は，“却下” に分類する．
- それ以外の既存の item に関する discussion からの関係は，“参照” に分類する．

実際には，“却下” 以外の分類は自動化可能である．

3.2.3 explanation の抽出 (Step 3)

この段階の作業では，仕様書の章や節の内容になる explanation を会議の発話から抽出する．

item 間 relationship を “提案” もしくは “参照” している discussion の発話から explanation を 1 個以上抽出する．場合によっては，画像情報をそのまま explanation としても良い．通常は，item 名が指し示す内容が分かる程度の explanation をつければ良い．なぜなら，詳細な内容が知りたければ，対応する discussion に属する action の音声情報・画像情報を再生すれば良いからである．

ただし，最終回会議の直前の仕様書作成作業では，仕様書を文書としてまとめて，最終回の会議で参加者全員で確認する必要があると考えているので，詳細な explanation を文字情報として必要とする．基本的には，item 間の relationship に “提案” または “参照” の属性を持つ relation がある discussion に含まれる発話をすべて文字情報に変換する．ある程度の要約は必要だが，要約により必要な情報が落ちてしまうことも考えられるので，要約は原則的に最終回会議で行う．

なお，explanation を抽出した discussion には，discussion と，item-explanation 間の relationship とを関係づけておく．図 3の例では，discussion 2 から explanation 1 を生成したので，discussion 2 から explanation 1 へ “提案” の関係を追加している．

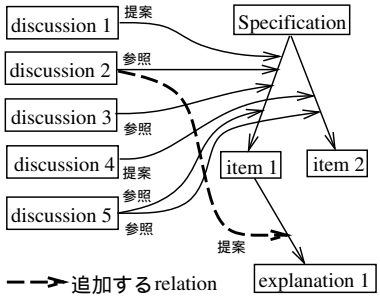

図 3: explanation の抽出に伴う relation の追加の例

3.2.4 item 間の階層構造の構築 (Step 4)

新たに抽出された item は，すべて孤立 item であり，item は階層構造を持っていない．また，会議の議論により既出の item に対して階層構造が追加される場合もある．ここでは，item の階層構造を構築し，効率的な議論の促進，仕様書内の矛盾点や一貫性の欠如の発見のための情報を提供できるようにする．

階層構造を作る手がかりとしては，以下のようなものがある．

- 1 つの discussion が複数の item 間 relationship へ関係づけられている場合，それらの item は互いに関係がある．
- 時間的隣接の多い discussion から関係づけられている item 同士は関係が深い．
- 会議のいろいろなところで議論された item は，他の多くの item に対する親 item であるか，多くの item の子 item である．
- 発話中に「ある item は，ある item の子 item である」もしくは逆に「ある item は，ある item の親 item である」という内容の明示的な発話があった．

仕様書作成作業では，明示的な発話があった場合のみ階層構造を構築する．それ以外の時間的隣接情報などから類推される構造は，ここでは構築せずに，次回の会議の冒頭でそれらの手がかりを基に構造を構築するかどうか議論する．

item の階層構造を作る場合には，以下のように仕様書の構造を変更する．

1. item 間の階層構造の relation を追加する．
2. specification–item 間の relation があれば，その関係を消去し，その relation に関係づけられていた discussion からの関係を新しくできた item 間の relation へつなぎ変える．
3. 新しくできた item 間の relation と item 間の階層構造を明示的に発話している discussion との関係がなければ，これを関係づける．この関係の分類は，他に “提案” の関係がなければ “提案” に分類し，そうでない時は，“参照” に分類する．

図 4の例では，図 3の例を基に，discussion 5 に item 1 と item 2 との階層構造を明示的に示す発話があったので，item 1 と item 2 の階層構造を構築し，specification–item 2 間の relation を消去し，この relation に関係づけられていた関係のつなぎ変えを行っている．

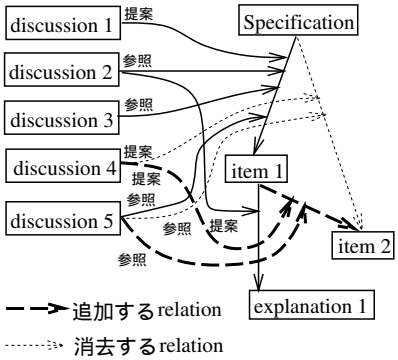

図 4: item の構造化に伴う relation の追加の例

3.3 仕様書の例

このようにして作成される仕様書は，図 5のようなものである．対象ソフトウェアが満たすべき要求の羅列とその説明から構成される．

仕様:
カードベースの仕様記述ツール
⋮
2.3 内容ボタン
カードの内容を書きたい時にここを押す．
2.3.1 内容ウィンドウ
⋮

図 5: 本方法により生成される仕様書の例

4 構造化された会議履歴・仕様書の会議中での活用方法

第 3章で述べた方法で構造化される，会議履歴および仕様書を実際の会議中で活用する方法を述べる．

1. 仕様書内の不明瞭な事項の発生の防止
   仕様書内の不明瞭な事項の発生の防止のために，既出の item に関する議論を確認しながら会議を進める．議論が不十分だと判断すれば，さらに充分な議論を行うようにする．具体的には，以下のような方法である．
   - **explanation がない item は，更に議論する必要がある**
     explanation を抽出できなかった item は，議論が不足しているためであるから，会議中にこれらの item をチェックし，議論を行う．
   - **最終回会議で各 explanation を確認する**
     最終回の会議では，仕様書中の item に関する各 explanation が文書 (文字情報) として充分か・不適切ではないか確認する．

2. 効率的な議論の促進
   過去に議論された内容について再び議論する場合や，関連項目を考慮しながら議論を進める場合には，それまでの会議履歴や仕様書を参照しながら会議を行う．このことにより，同じ内容の議論の繰り返し，結論を出さない議論，決定事項の度重なる変更を行わずに済む．
3. 仕様書内の矛盾の発生・一貫性の欠如の防止
   議論の中で以下のような状況が起こった場合に，仕様書内の矛盾の発生や一貫性の欠如が生じると考えられる．それらの問題点をリストアップし，会議中に議論する．
   - 孤立 item がある場合．
     (これは，単に思いつきで提案したがその後，一切検討されずに放置されているような item をそのまま放置してしまうことを防止する．)
   - item の削除により，item の階層構造から孤立してしまうような item が生じてしまうような場合．
   - item の階層構造の追加・変更により，item の階層構造にループが生じてしまう場合．
   - explanation の削除により，explanation がなくなってしまう場合．
   - ある explanation が複数の item の explanation となっていた場合に，その explanation をある item に関して削除・変更した場合．

5 本方法の適用事例

過去の 2 つの会議の記録に対して本方法を適用し，本方法と第 4章で述べた本方法の活用法により，第 2章で述べた内容を支援できる事例を紹介する．

以下の事例の会議は，
- 円卓を囲む対面式で会議を行う．
- 黒板とメモ用紙が用意され，会議の参加者は誰でも自由に使うことができる．

という設定で行われた．

5.1 事例 1

事例 1 は，表 1のような会議である．

表 1: 事例 1 の会議状況

対象ソフトウェア	描画エディタ
会議参加者 (人)	5
会議回数 (回)	3
会議所要時間 合計	3 時間 57 分

この会議の記録に本方法を適用し，得られた結果は以下のようなものである．

- 仕様書作成作業手順の実行可能性
  第 1 回の会議直後の仕様書作成作業における会議履歴及び仕様書の構造化作業は，何の問題もなく可能であった．作業結果を表 2に示す．

表 2: 事例 1 の第 1 回会議後の仕様書作成作業の作業結果

Step 1	仮の discussion	18 種類，38 個
Step 2	抽出された item 詳細化された discussion	49 個 106 個
Step 3	抽出された explanation	47 個
Step 4	item の階層構造の深さ 階層構造化された item	4 37 個

- 不明瞭な事項の発生の防止
  第 1 回の会議直後の仕様書作成作業で explanation のつけようがなかった item が 2 つ (グルーピング機能・マウス操作) あったが，この item に関する説明は，第 2 回会議でも第 3 回会議でも議論されずに終った．そのため，会議の参加者の理解が各自まちまちで最終仕様書にもあいまいな記述が残るのみであった．本方法および本方法の活用法により，このような item がリストアップされ，議論されるため，このような不明瞭な事項の発生を防止できる．
- 不明瞭な事項の発生の防止
  「他のソフトウェアと同様」という explanation しかつかなかった item が 3 つあった．しかしながら，会議参加者のほとんどはその内容を理解しておらず，最終仕様書には不明瞭な記述しか残らなかった．本方法とその活用法を利用して，最終回会議で explanation の内容の確認を行えば，このような不備を解消することができる．
- 欠落事項の発生の防止
  第 1 回の会議直後の仕様書作成作業で構造化されなかった孤立 item は 12 個であるが，それらは表 3のように分類される．

表 3: 事例 1 の第 1 回会議後の孤立 item

discussion の時間的隣接情報などから構造化が可能である item	3 個
内容的に構造化すべきと考えられる item	4 個
暗黙のうちに却下された item	3 個
他の item に内容的に吸収された item	1 個
孤立 item であることが妥当な item	1 個

これらの item は，第 2 回会議以降ではすっかり忘れられてしまい，最終仕様書には全く記述がなかった．本方法とその活用法を利用して，会議中にこれらの item がリストアップされ，さらなる議論が加えられて仕様書に吸収することができる．
- 欠落事項の発生の防止
  第 2 回会議で新出の親 item があげられたが，どの item の親 item なのか議論されなかった．そのため，最終仕様書にはそれに関する記述がなかった．本方法とその活用法によれば，この item は孤立 item となり，会議中にこの item がきちんと認識され，充分な議論が行われ，仕様書に記述が残る．

- 効率的な議論の促進
  第 2 回会議で分担を決め，第 3 回会議で担当者が自分の担当範囲をレビューするという会議だったが，5 人の担当者のうち，1 人の担当者は過去の議論をすっかり忘れて全く異なる内容のレビューを行い，それまでに議論された内容を再び議論し直すことになってしまった．本方法によれば，過去の議論そのものを基にレビューを行えるため，このようなことは決して起こらない．

3 回の会議の発話履歴を基に，仕様書を作成した作業結果の一部を示す．発話履歴の構造化の一部が図 6，仕様書のデータ構造の一部が図 7，仕様書の一部が図 8である

discusssion	action	worker	action(発話の内容・図番号)
⋮	⋮		⋮
11	78	A	文字についてはそんなもんですね．
12	79	B	あとは.... 入出力が必要ですね．file の保存とか．
12	80	C	load，save，印刷ぐらいを入出力として入れましょう．
⋮	⋮		⋮
23	121	C	部品とかはいいんですか．
23	122	E	部品って?
23	123	C	たとえば，こんなやつ．
23	124	C	図 5(白板上の図)
23	125	D	なるほど，建築の製図用部品とか，回路図用の部品とかですね．
23	126	B	それは，別売りオプションでいいんじゃないんですか．(しばし，沈黙)
24	127	A	それじゃ，部品呼び出しが load の sub menu にあるってことで．
24	128	C	つまり，別売りの用途別の部品集があって，部品呼び出しから部品を呼び出して自分の絵の中に貼り込めるってことで．
⋮	⋮		⋮

図 6: 事例 1 の構造化された発話履歴の一部

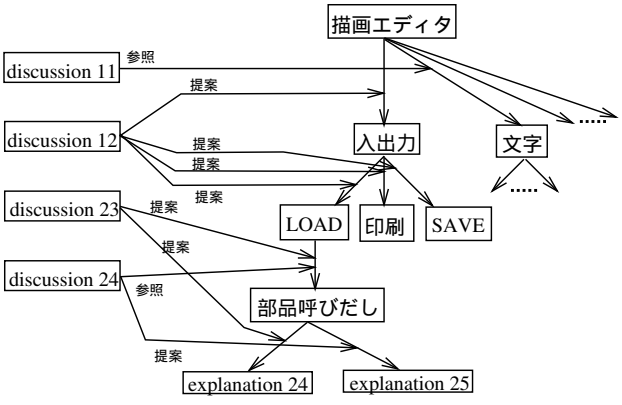

図 7: 事例 1 の仕様書の構造の一部

5.2 事例 2

事例 2 は，表 4のような会議である．

この会議の記録に本方法を適用し，得られた結果は以下のようなものである．

仕様:
描画エディタ
⋮

4. 入出力
...
4.1 **load**
...
4.1.1 部品呼びだし
・建築用部品集・回路図用部品集などを別売りで用意．(explanation 24)
・load のサブメニューとして部品呼び出しがあり，部品を自分の絵の中に貼り込める．(explanation 25)
4.2 **save**
...
4.3 印刷
...
5. 文字
...

⋮

図 8: 事例 1 の仕様書の一部

表 4: 事例 2 の会議状況

対象ソフトウェア	ハイパーテキストベースの仕様記述ツール
会議参加者 (人)	5
会議回数 (回)	4
会議所要時間 合計	11 時間 36 分

- 仕様書内の矛盾の発生の防止
  第 2 回会議において「方法 A と方法 B が存在する」のような item が 4 組提示されたが，充分な議論がなされず，どちらを採用するか曖昧なまま，最終回の会議を終ってしまった．最終仕様書には，暗黙のうちに片方が採用されていた．この場合，複数の作業者で仕様書を記述していたため，他の項目と矛盾を引き起こす可能性がある．本方法とその活用法によれば，構造化された仕様書を参照して会議中にどちらを採用するかの議論が行われ，会議の参加者の同意のもとに決定が行える．その結果，複数の人で作成する仕様書内の矛盾の発生を防止できる．また 4 組の item の中には，方法 A も方法 B も両方とも採用可能なものもあり，複数の方法の存在が仕様書の階層構造の中で提示され，充分な議論が行われ，より良い選択が行える．
- 欠落事項の発生の防止
  第 3 回会議の冒頭に、会議履歴を基にそれ以前の会議内容の確認が行われた．会議履歴は決定事項のみが記されているだけで，決定への途中経過や決定理由は全く記録されていなかった．そのため，ある item の explanation に関して，なぜそのように決定したのかを把握できないまま，議論が進んでしまい，その結果，最終仕様書にはその内容は記述されていなかった．本方法によれば，決定理由などは過去の議論から容易に検索が可能であるから，このようなことは決して起きない．
- 不明瞭な事項の発生の防止
  第 2 回会議直後の仕様書作成作業で，item 間の構造を

明示的に示す発話により，ある 1 つの item に属する同レベルの子 item が 13 個作成できた．しかしながら，この中には内容的に考えるとより厳密に階層構造を構築した方が良いと思われる子 item が多い．最終仕様書でも極めてわかりにくい記述になっていた．本方法とその活用法によれば，この 13 個の item が提示されるため，より議論が進み，仕様書ではわかりやすい記述になる．

- 効率的な議論の促進
  ある事柄に関する議論で，関連項目を考慮しようという流れになり，考慮を始めたが議論が完全に横道にそれてしまった．本方法によれば，考慮すべき関連 item と各 item の explanation がリストアップできるので，その項目だけを考慮すれば良く，議論が効率的に行える．

6 おわりに

本報告では，仕様作成会議の参加者の発話を時系列上に並べた発話履歴から仕様書を作成する方法を提案した．本方法は，発話履歴を構造化し，それを基に構造化された仕様書を作成する．毎回の会議毎に作成作業を行い，仕様書を段階的に作成する．このように会議の記録から直接的に仕様書を作成するため，会議で議論されながらも仕様書には記述が残らないような欠落事項の発生を防止できる．また，会議毎に記録を構造化することで，各会議中にその記録を参照して議論することができ，効率的な議論の促進，不明瞭な事項の発生の防止，仕様書内の矛盾や一貫性の欠如などの発見が可能である．さらに，本方法を過去の会議の記録に適用し，本方法により発話履歴から仕様書が作成可能であり，本方法を活用することで仕様作成作業を有効に支援できることを確認した．

本方法の問題点と今後の課題は以下のようなものが考えられる．

- item の抽出作業の困難さ
  discussion から item を抽出する作業は，かなり作業者の主観に依存しており，そのことがこの部分の作業を困難なものにしている．この部分の作業の実験を行って，その分析結果から多くの知見を得る必要がある．
- item 間の階層構造構築の手がかりの検証
  第 3.2.4節で述べた item 間の階層構造の構築のための手がかりのうち，discussion の時間的な隣接・分布などの手がかりは，まだ直観的なものに過ぎない．詳細な分析をし，この手がかりの根拠を提示する必要がある．
- 会議中で本方法を実際に活用する実験を行う
  本報告では，過去の会議の記録に対して本方法を適用したのみである．仕様書作成作業に本方法を用い，会議中に第 4章に述べた本方法の活用法を実際に行う実験をして，この方法が有効にかつ完全に機能することを確認する．

謝辞

本研究を進めるにあたり，富士通国際情報社会科学研究所 (現 富士通研究所情報社会科学研究所) での社会科学アプローチセミナーを通して貴重な議論及び助言を戴いた同研究所ユーザー指向ソフトウェアプロセスグループのメンバーに深く感謝致します．

また，過去の会議の記録への適用などの作業に協力頂いた佐伯研究室の永岡洋樹氏に感謝致します．

参考文献

[1] 西, 海谷, 佐伯. ソフトウェアの発注者-開発者会議におけるインタラクションの分析. 情報処理学会ヒューマンインターフェイス研究会, Mar. 1992.

[2] Gary M. Olson, Judith S. Olson, Mark R. Carter, and Marianne Storrosten. Small group design meetings: An analysis of collaboration. *Human-Compter Interaction*, Vol. 7, No. 3, pp. pp.347–374, 1992.

[3] 海谷治彦, 佐伯元司. プロダクトを基にしたソフトウェア設計者会議の分析法. 情報処理学会ヒューマンインターフェース研究会, pp. 47–16, Mar. 1993.

[4] Jeff Conklin and Michael L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. In *CSCW'86 Proceedings*, Dec. 1986.

[5] C. Potts and G. Bruns. Recording the Reasons for Design Decisions. In *10th International Conference on Software Engineering*, pp. 418–427, 1988.

[6] J. Lee. Extending the Potts and Bruns Model for Recording Design Rationale. In *13th International Conference on Software Engineering*, pp. 114–125, 1991.

[7] 中谷三江. 劇場モデルに基づいたソフトウェア意図伝達支援ツール COMICS. 情報処理学会論文誌, Vol. 31, No. 1, pp. 124–135, Jan. 1990.

[8] 落水ほか. ソフトウェア開発における協調支援環境 Vela. 情報処理学会第 41 回全国大会 5, pp. 149–162, Sep. 1990.

[9] 中島毅, 田村直樹, 藤岡卓, 上原憲二, 高野彰. PPK 法 : ソフトウェア設計プロセスの記録と分析の手法. ソフトウェア工学研究会, Vol. 67, No. 2, Jul. 1989.

[10] 尾上裕子, 桑名栄二. 設計者間のコミュニケーション構造モデルの一考察. 情報処理学会グループウェア工学研究会, Vol. 3, No. 1, Dec. 1992.

[11] 浜田雅樹, 安達久人, 竹中豊文. "設計プロセスの蓄積・利用による設計支援法について". 情報処理学会ソフトウェア工学研究会, Vol. 80, No. 17, pp. 127–134, Jul. 1991.

[12] 海谷治彦, 佐伯元司. ソフトウェアの仕様化作業における会話構造の分析. 電子情報通信学会技術研究報告 KBSE92-9, Vol. 92, No. 128, pp. 25–30, Jul. 1992.

[13] 海谷治彦, 佐伯元司. ソフトウェア仕様作成会議支援ツールの設計. 電子情報通信学会技術研究報告 KBSE93-13, Vol. 93, No. 147, pp. 9–16, Jul. 1993.