修 士 論 文

分散共同ソフトウェア開発における情報共有支援方式に関する研究

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

西田 和豊

2004 年 3 月

修 士 論 文

分散共同ソフトウェア開発における情報共有支援方式に関する研究

指導教官　落水浩一郎 教授

審査委員主査　落水浩一郎 教授
審査委員　片山卓也 教授
審査委員　篠田陽一 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

110097 西田 和豊

提出年月: 2004 年 2 月

概 要

本研究では，分散共同ソフトウェア開発における問題点を調査し，開発支援に関する要件について分析・定義する．また，その要件を満足し，円滑にソフトウェア開発作業を行うための方法として，開発において取り扱われる情報を定義し，それらの情報を開発者間で共有するための方式を提案する．そして，その方式を実現する情報共有支援環境の構築を行う．

目 次

第1章　はじめに

1.1　背景

近年，特定の研究機関や企業に所属していない人々でも，WWW(World Wide Web) やインターネットなどを利用し，様々な情報を得られるようなネットワーク環境の普及が増大している．また，ハードウェア性能の向上や，新技術の開発などから，高品質で多機能な多くのソフトウェアが生産されるようになってきている．しかし，そのようなソフトウェアにおける開発では，開発作業や開発人数をより多く必要とする傾向にある．そのため，ある一つの大規模なソフトウェアを開発する場合，複数の開発者が，互いにソースコードを共有しながら同時に一つの開発作業に携わることが一般的になりつつある [1].

このような背景からソフトウェア開発では，オープンソースソフトウェアに代表されるように，世界中に分散しているユーザー同士がネットワークを通じて，共同で 1 つのソフトウェアを開発する，分散共同ソフトウェア開発という開発手法が注目され，実際に利用される機会が増加している．分散共同ソフトウェア開発では，複数の開発者やグループが並列的に作業を進められるため，短い時間で多くの開発作業を行うことが可能である [4].

だが分散共同ソフトウェア開発にはいくつかの問題点も存在する．例えば，地理的に分散した開発者が協調して作業を進めていくには，開発状況について共通の認識と情報をもっている必要がある．開発者間における状況認識に差違が生じた場合，ソフトウェアの開発状態を不安定なものとし，開発効率や成果物に多大な悪影響を与えることがある．

このような問題点の発生を抑えるためには，WWW サーバーや，CVS(Concurrent Versions System)[2]，メーリングリストなど既存のシステムを利用することで，開発中のプロダクトやドキュメントを統一して管理し，開発者間で十分な討議による意思疎通を円滑に行うことが重要である．しかし，これらのシステムは通常独立しているため，分散共同ソフトウェア開発でサービス提供を行うには，各システムを連係して利用できるような情報共有支援環境を構築することが必要となる．

分散共同ソフトウェア開発に必要な情報共有支援環境を構築できるシステムの代表的なものとして SourceForge[3] が挙げられる．SourceForge は，米 OSDN(Open Source Development Network) が 2000 年 1 月にサービスを開始した，世界最大のオープンソースソフトウェア開発支援サイトで，2004 年 2 月で約 75,000 件のプロジェクトが登録されている．SourceForge で構築された環境では，WWW サーバ，CVS サーバ，メーリングリストサーバなどを連係して運用することで，開発者は Web ブラウザを介して必要な情報を取得し，開発作業を進めていくことが可能となっている．また，SourceForge では開発さ

れたソフトウェアの配布や，ソースコードの公開を行っており，これまで開発に参加していなかったユーザーも，開発に参加する機会が提供されている．

このような背景から，ソフトウェアの開発形態として分散共同ソフトウェア開発が利用される機会は今後も増大していくと考えられる．

1.2 研究の目的

分散共同ソフトウェア開発では，複数の開発者により並行して作業が進行するため，同じ箇所を同時に編集してしまうといった作業の衝突や，同じような役割を持つ部分を別々に作ってしまうといった作業の重複が起こり得る．また，開発者の地理的分散により，直接的な対面によるコミュニケーションを行う機会が制限され，開発者間での意思疎通が十分に行えず，結果として開発の方向性の発散を招き，ソフトウェア開発の進行や成果に悪影響を及ぼすなどの問題点がある．

本研究では，分散共同ソフトウェア開発を円滑に行える環境の構築を目的とした，情報共有システムの設計・構築を目的とする．具体的には，本学の講義の 1 つである，ソフトウェア設計演習を対象に，GForge システムと討議構造化ツールを用いて情報共有システムを構築し，複数の開発者が円滑に協調作業するための，情報共有支援環境を提供する．

1.3 本論文の構成

本論文では，2 章で分散共同ソフトウェア開発の特徴を述べるとともに，そこで必要となる要件と，取り扱われる情報の定義を行う．また，それらをふまえ，開発者が円滑な作業を行えるような支援環境の構築と，設計した支援環境における問題点について説明する．3 章では，これまでの情報共有支援環境において発見された問題点を解決するため，分散共同ソフトウェア開発を使ってソフトウェア開発を行っている，オープンソースソフトウェア開発の調査を行い，そこで利用されている既存システムと，そのシステムにおけるコミュニケーション支援の問題点を説明する．4 章では，オープンソースソフトウェア開発における既存システムの問題点を改善するために採用する，討議構造モデルの説明を行うとともに，その実装である討議構造木抽出エンジンと討議内容要約エンジンについても説明する．また，討議構造木抽出エンジンにおいて，これまで問題とされていた署名部分の除去に対して改善を行った．その成果についても報告する．5 章では，これまでの調査の結果から，GForge に討議構造モデルの概念を取り入れた情報共有支援環境の構築と，その環境の試験運用について報告する．最後に 6 章として，本研究のまとめと，今後の課題について報告する．

第2章　分散共同ソフトウェア開発における情報共有支援

本章では，分散共同ソフトウェア開発における特徴と問題点について述べるとともに，そこで取り扱われる情報を定義する．

2.1　分散共同ソフトウェア開発の特徴

分散共同ソフトウェア開発とは，地理的に分散している複数の開発者たちが，ネットワークを介してコミュニケーションを取り合い，協調して開発作業を進行させ1つのソフトウェアを開発していく手法である．このようなソフトウェア開発手法では，開発に必要な作業をいくつも並行して進めることが可能なため，特に大規模なソフトウェア開発を行う場合には，効果的な手法であると考えられる．

しかし，複数人数による共同作業に加え，開発者が地理的に分散していることから，以下に挙げる問題点が発生する．

- ソースコードの増大に伴う作業の重複と衝突

  複数人による開発作業によって，ソフトウェアを構成する多くのソースコードが生成される．そのため，同じ箇所を同時に編集してしまうといった作業の衝突や、同じような役割を持つ部分を別々に作ってしまうといった作業の重複が起こり得る．

- プロジェクト管理者にかかる負荷の増大

  開発規模が大きくなれば開発を円滑に進めるために，より強力な管理能力が必要になる．また，ソフトウェアを構成するモジュール間の関連を把握する能力が要求される．

- 開発の方向性の発散

  開発メンバーの増加に伴い，ソフトウェア開発に対する合意をとることが困難になり，開発の方向性が定まらない．

このような問題点の発生は，共同に関する問題と，分散に関する問題が原因であると考えられる [9].

分散共同ソフトウェア開発では，円滑に開発作業を進めるため，これらの問題点を調整するための手段を用意しなければならない．そのためには以下の 4 つの要素を開発者に明確に呈示する必要がある．

- 仕事の責任範囲と作業環境の明示
- 共有情報の変更管理
- 全員で共有される決定事項の管理
- 共有情報や決定事項の動的変化に伴う矛盾と不確実さの管理

これらの要素を明示するためには，開発に関連する情報の共有を実現し，開発者へ提供することが必要である．そのため，共有しなければならない情報を以下に定義する．

中間プロダクトに関する情報: ソフトウェアの開発作業において，ソースコードの記述と，それに伴うドキュメントの作成は重要な要素である．しかし，分散共同ソフトウェア開発においては，同じ箇所を同時に編集してしまうといった作業の衝突や，同じような役割を持つ部分を別々に作ってしまうといった作業の重複が起こり得る．このような状況の発生は，ソースコードが増大することにより，他の開発者の作業状況の認識や，自他の開発作業の関連についての認識が困難となることが原因である．そのため，ソースコードやドキュメントなどの中間プロダクトに関する情報を共有し，現在の開発作業における自他の役割と関連を開発者自身が認識する必要がある．

コミュニケーションに関する情報: 分散共同ソフトウェア開発では，開発者が地理的に分散しているため，直接的な対面によってコミュニケーションを行う機会が制限されることがある．このような制限から，開発者間での意思疎通が十分に行えない状況は，他の開発者の作業状況の認識や，自他の開発作業の関連についての認識を困難とする問題点についても影響を与え，結果として開発の方向性の発散を招き，ソフトウェア開発の進行や成果に悪影響を及ぼす．そのためネットワークを介したコミュニケーション手段を用意し，そこで行われる議論に関する情報を開発者は認識する必要がある．

これら双方の情報を開発者間で共有することにより，共同と分散に関する問題点を解決し，協調して作業を行うことが可能となる．

2.2 情報共有支援環境の設計

分散共同ソフトウェア開発では，2.1 節において述べたような特徴があり，そこで取り扱われる情報には，中間プロダクトに関するものと，コミュニケーションに関するもの 2

種類が存在する．円滑にソフトウェア開発を行うには，開発に関するこれらの情報を共有し，開発者が他の開発者の状況を把握しながら，自信の作業の役割と関連を認識する必要がある．このような情報共有の実現は，それを支援する機能をもつ既存システムを利用することで可能となる．このため分散共同ソフトウェア開発では，必要に応じて既存システムを利用し，開発作業を進めていくことが効果的であると考えられる．

このことをふまえ，本研究では試験的に情報共有を支援するための環境を設計した (図 2.1 参照)．

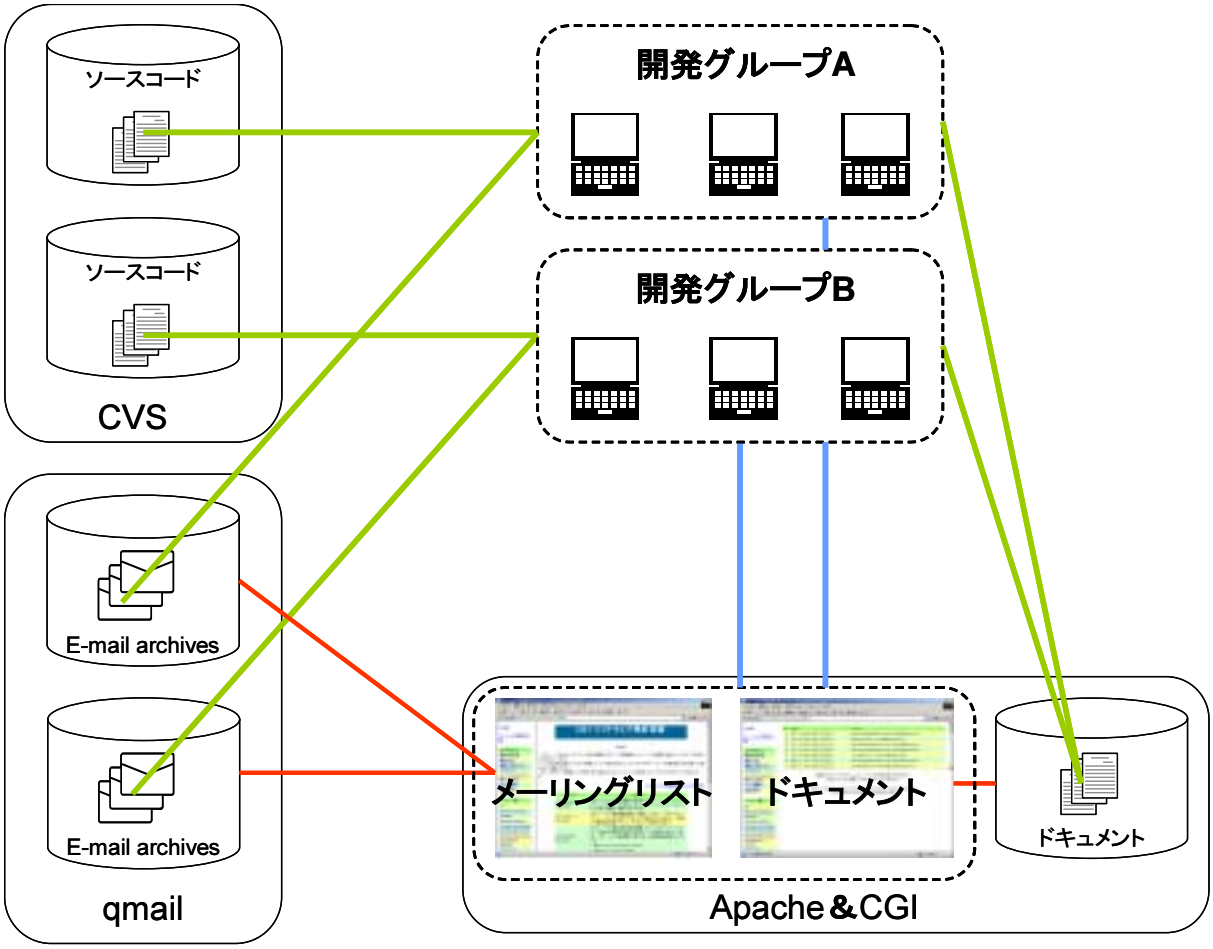


図 2.1: GForge システムの構成図

構築した環境では，分散共同ソフトウェア開発で必要とする，中間プロダクトとコミュニケーションに関する情報共有の支援を可能とするため，CVS と，qmail[6] を利用している．開発者たちは，各開発グループ単位でソースコードの版管理やメーリングリストを利用でき，情報共有を行うことが可能となっている．また，Apache[5] を用いた WWW サーバーを用意することで，開発グループにおいて作成されたドキュメントや，電子メールによるコミュニケーションを，Web ブラウザを利用することで参照できるようにした．こ

れにより，開発者がどのような場所で作業を行っても，ネットワークを利用することで共有している情報を参照することが可能となっている．また，他の開発グループのドキュメントや電子メールによる議論を参照することも可能で，他の開発グループのドキュメントや議論を知ることで，自己の開発における問題点や課題を解消するためのきっかけを得ることができると考えたためである．

2.3 問題点

2002 年 12 月から翌年 1 月の期間，本学で開講されたソフトウェア設計演習の講義において，2.2 節で設計した環境を試験的に運用した．その結果，次のような問題点が発見された．

メンバーの管理に関する問題: 本環境では，CVS とメーリングリストを利用することで中間プロダクトに関する情報共有と，コミュニケーションに関する情報共有を可能としている．

しかし，開発を進めるための準備として，実際にこれらのシステムを設定し，利用できる準備を整える必要があった．また，開発の経過に伴い開発参加者の移動があったため，そのような変更が行われるたびに，ユーザー管理を改めて行う必要があった．これらの開発開始前の準備と，開発開始後の管理に伴う作業は，実際の開発作業以外の作業を開発グループへ強要するため，場合によっては，実際の開発作業の妨げとなることがあった．

コミュニケーション情報共有に関する問題: メーリングリストを利用した電子メールコミュニケーションの情報に対し，ネットワークを介して取得することができることは，過去のメールを保持していなくとも，開発場所に制限されることなく，必要な時に閲覧できたり，他の開発グループの議論を参照することで，現在発生している問題点の解決を図ることができるという利点があった．

しかし，メールの内容に関して開発グループ内部でのみ共有したい場合もあり，そのような内容の議論を伴う電子メールの情報を，他の開発者が共有することは避けたいという状況があった．このような場合，開発者の電子メールコミュニケーションの利用の妨げとなり，円滑な議論の進行を妨げるという結果になった．

このような問題点が発生した原因として，分散共同ソフトウェア開発において必要とされる機能を備えた既存システムを利用し，単純に情報を共有するだけでは，プロジェクト管理者に対する，情報管理作業の負荷増大と，開発グループ内の様々な情報ごとに，共有する範囲を柔軟に設定することができなかったことが考えられる．そのため，円滑に作業を進めることができず，場合によっては作業の妨げとなるような状況を発生する結果となった．

第3章　既存の情報共有システムの調査

これまでに設計し構築した情報共有支援環境には，2.3 節において指摘した問題点がある．これらの問題点を改善するため，共同作業支援に関する要請を定義した．そして，それらの要請を満たした情報共有システムを構築構築するため，分散共同ソフトウェア開発を利用し，大規模なソフトウェア開発が行われている，オープンソースソフトウェアの調査を行った．本章では，共同作業支援に関する要請と，オープンソースソフトウェアの調査結果について述べる．

3.1　共同作業支援に関する要請

地理的に分散した開発者が協調してソフトウェア開発を行うには，以下に挙げる要件を満たすことが重要である．

- 均一のユーザインタフェースを通じてシステムにアクセスできること
- 文書やソースコードの共有と構成管理が可能であること
- 開発者間の様々なコミュニケーション活動を支援できること
- 開発チームが生み出す様々な活動に関する情報を管理・利用できること

今後の情報共有システムの設計では，共同作業支援における要件として定めた上記 4 項目を満たすことを必要条件として，設計・構築を行う．

3.2　オープンソースソフトウェア開発における情報共有支援環境

分散共同ソフトウェア開発という開発手法を採用し，大規模なソフトウェア開発を行っている代表的なものに，オープンソースソフトウェアが挙げられる．オープンソースソフトウェアでは，SourceForge や GForge[7] といったシステムによって，Web ページを介した情報共有支援環境の提供が実現されており，開発に参加するユーザーは，ネットワークを利用することで，開発環境に束縛されることなく，他のユーザーと協調して開発作業を進めることができる．本研究では，特に GForge システムに着目した．

GForge システムでは，Web ページにより以下の機能を提供することが可能である．

- プロジェクトのホームページ
- CVS による版管理支援
- ドキュメントマネージャによる文書の管理・閲覧支援
- メーリングリスト・電子掲示板
- 開発したソフトウェアに関するファイルリリースを行うサーバ
- ソフトウェアを配布するためのダウンロードサーバ
- バグトラッキング支援
- タスク管理
- 個人のダイアリーや Web ページのブックマークを保存しておくマイページ

これらの機能は共同ソフトウェア開発に必須の機能であり，3.1 節において定義した要件を満たすための十分な機能を提供していると判断した．また，情報共有を行うための管理作業や，共有範囲に関しても Web ページを介して行うことが可能であるため，2.3 節において指摘される問題点についても解消することができると判断した．

3.3 コミュニケーション支援機能の強化

複数の開発者が参加する分散共同ソフトウェア開発では，分析結果の突き合わせ，設計方針の討議，役割分担とスケジュールの調整，インタフェースエラーの発生に伴う共同デバッキング作業など，協調と調整のための基本的活動の多くが，開発者同士のコミュニケーションによって達成される．そのため，開発者間のコミュニケーションは共同ソフトウェア開発活動の半分以上を占める重要な活動であり，開発者は自己の役割や作業，設計対象の全体像の認識など，様々な場面において過去のコミュニケーション内容の確認の機会を必要とする．GForge では，メーリングリスト内で交換された電子メールを，アーカイブとして蓄積し，必要に応じて開発者は過去の議論を読み返すことができる．

一方で，一般的な電子メールを利用したコミュニケーションの特徴としては，複数の話題を同一メールに記述する傾向があり，ある 1 つの話題を複数のメールにわたり議論する．また，メーリングリストにおけるコミュニケーションには以下のような特徴がある．

- 次に返答すべき参加者を明示的に示唆することがある．
- 返答を要求されていない参加者が返答を行う場合がある．
- 1 つの発話に対して，複数の参加者が返答を行い，討議の流れが分岐する場合がある．

このような特徴から，蓄積される電子メール数が増大すれば，蓄積される議論や，討議の分岐の数も増加していく．そのため，既存のコミュニケーション支援については以下の問題がある．

開発が進行し，電子メールの蓄積量が多くなれば，過去の電子メールを遡って討議内容を確認することに多大な時間を必要とし，目的とする議論の発見や，内容の理解が非常に困難となる．結果として，同じ話題に関する議論が繰り返される，もしくは議論が終結せずに忘れられるなどの問題が生じる可能性がある．

このような問題点から，コミュニケーション支援機能の強化が必要であると判断した．コミュニケーション支援機能の強化には，これまで本研究室で開発・設計を行ってきた，討議構造化ツールを利用した．討議構造化ツールを GForge に適用することで，これまでは電子メール単位でコミュニケーション情報を蓄積していたことに対し，議論された討議単位でのコミュニケーション情報の蓄積ができる．そのため，開発者はこれまでよりも容易に過去の討議を探索し，その内容を確認することができる．

第4章　討議構造モデル

分散共同ソフトウェア開発における既存の情報共有支援環境には，コミュニケーション情報の共有に問題点があると指摘し，それに対する改善案として討議構造モデルの適用が効果的であると判断した．

本章では，討議構造モデルについて説明を行い，情報共有支援環境に適用するために改善を行った部分について述べる．

4.1　電子メールを用いたコミュニケーションの特徴

地理的に分散する人々が共同で作業を進める場合，電子メールを利用した非同期コミュニケーションが良く利用される．電子メールを用いたコミュニケーションは，テキストを介したコミュニケーションである．そして，メッセージとして記述するテキストに決められたフォーマットはない．したがって，送信者の表現を制限することがなく，送信者は気軽にメッセージを記述することができる．また，電子メールを用いたコミュニケーションは，地理的・時間的にも参加者を拘束しない，非同期コミュニケーションでもある．メッセージを送る側は送りたいときにメッセージを相手の元へ届けることができ，受け取る側は受け取りたいときにメッセージを受け取ることができる．しかも，電子メールを利用するにあたっては，現在社会のネットワーク環境の整備，多様なメール送受信用ソフトウェアの存在などから，専門的知識を必要としない．結果，電子メールはネットワーク上における基本的かつ重要なコミュニケーションツールの一つとして大きな役割を担っている．

しかし，長所だけではない．フォーマットのないテキストをやり取りすることによって成立するコミュニケーションであるため，計算機を介した二次利用が難しい．また，非同期なコミュニケーションであるため，沈黙などのような，対面の場合などに生じる，手がかりを元にした話者交代を行うことができない．そのかわりに，送信者は複数の話題を同一メール中に記述する傾向がある．そのため，複数の話題が並列に議論されやすく，議論が終結しているか否か，返答を要求しているか否かといった現在の状況を見失いやすい．また，テキストのみを介した非同期コミュニケーションであるため，聞き手が話し手の発話を理解しているかどうかを知るための手がかりも少ないと言える．このため，対話者間の会話に関する共通の理解が阻害され，認識の不一致が発生しやすい．

4.2 討議構造木

討議構造木は，図 4.1 に示される構造を形式的に表現するために，対面による 2 者の会話モデルであるコントリビューションツリー [10] を拡張したモデルである [11].

なお，図 4.1 では，発話と発話のつながりは，相手の発話と返答発話という関係しか表現しないが，討議構造木は，返答発話が，相手の発話意図を適切に理解したことを前提に行われたものであるか否かをも表現できる.

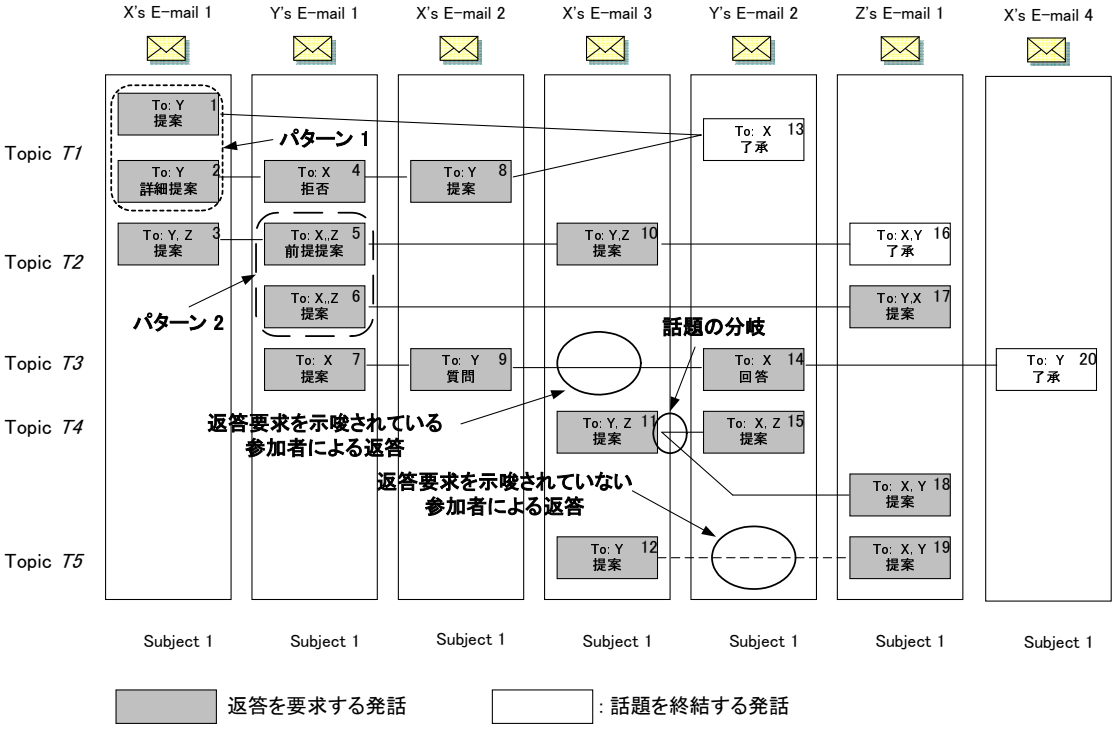


図 4.1: 電子メールを利用した討議構造

発話の定義

討議構造木を構成する発話の定義を以下に示す.

- 相手からの返答を求めない，議論の流れに直接の影響を与えない文章 (宣言的な発話).
- ある話題に関する質問や提案など，相手からの返答要求を示唆している文章 (返答を要求する発話).
- 話題を終結するような同意や受諾などを示唆するような発話 (話題を終結する発話).

フェーズの定義

コントリビューションツリーに基づく対面の会話モデルに，Novic によるコントリビューショングラフがある [12]．討議構造木では，コントリビューショングラフにおけるプライマリエビデンスとセカンダリエビデンスの概念を取り入れている．プライマリエビデンスとセカンダリエビデンスの定義を以下に示す．

プライマリエビデンス: 聞き手の中の Y_i が，自分が話し手 X の意図する聞き手であると信じるときに示す行動 e のこと．つまり，Y_i が，自分との共通理解を示すことを作業者 X に要求されていると信じるときに示す行動のこと．

セカンダリエビデンス: プライマリエビデンスとは異なり，聞き手の中の Y_j が，X の意図する聞き手でない場合，もしくは，X がプライマリエビデンスを求めていないと Y_j が判断した場合の行動 e' のこと．

プライマリエビデンスとセカンダリエビデンスの概念を取り入れた，討議構造木における Pr，Ac/InAc の定義を以下に示す．

Pr: 作業者 X が，複数の作業者の中のある作業者たちに，話題を継続/終結するといった意思を伝達するための発話 u 行なうフェーズのこと．ある作業者たちの中の Y_i がプライマリエビデンス e を示したならば，発話 u による X の意思を Y_i が適切に理解していると X が信じることができる．

Ac: ある作業者 Y_i が発話 u を受け取り，発話 u で示される作業者 X の意思を適切に理解したことを意味するプライマリエビデンス e を示すフェーズのこと．このとき，作業者 Y_i は，プライマリエビデンス e を示したならば,「X の意思を自分が理解していることを作業者 X が信じる」ということを信じることができる．

InAc: 作業者 X による返答を示唆されていない作業者 Y が，発話 u を理解したか否かを理解できると思われる行動，つまり，セカンダリエビデンス e' を作業者 X に与えるフェーズのこと．

属性の定義

C，Pr，Ac における属性を以下に示す．

C(**トピック番号**): トピック番号は，サブジェクト番号毎に導入されたトピックに割り当てられる番号のことで，導入された順番に，T 1,T 2,T 3... とする．

Pr(WhoPr,WhomPr,PrS): Pr は，WhoPr，WhomPr，PrS の 3 種類の属性をもつ．各属性については，以下に示す．

WhoPr: そのフェーズに属する発話を行った参加者名を示す．

WhomPr: 発話者の意思を伝達する相手を示す．伝達相手が複数人の場合はその全てが相当する．

PrS：Pr に属している発話の状態を R(返答を要求し，話題を継続している) と，F (返答を行い，話題を終結している) によって示す．

Ac(WhoAc),InAc(WhoInAc): WhoAc，WhoInAc は，そのフェーズに属する発話を行った参加者名を示す．

サフィックスの定義

C，Pr，Ac，InAc にサフィックスを導入する．$C_{x.y.z...}$ の x:y:z::: を，C のサフィックスとする．サフィックスの第 1 ラベル (x) をサブジェクト番号とする．サブジェクト番号とは，サブジェクト毎に割り当てられた番号のことで，導入された順番 (1,2,3....) で番号が割り当てられる．y 以降のサフィックスの値は，コントリビューションが導入された順番を示す．Pr，Ac，InAc のサフィックスは，それらによって構成されるコントリビューションのサフィックスと同一である．

討議構造木例

図 4.2 に，討議構造木におけるパターン 1 の構造例を示す．この例では，参加者 X による発話#2 が，発話#1 の詳細を表しており，発話#2 は，参加者 Y に返答の要求を示唆している．続いて，参加者 Y の発話#4 が行われ，発話#1 に関する話題のサイドシーケンスが形成されている．参加者 Y による発話#13 により，サイドシーケンスと話題 T 1 が終結している．“ 発話#数字” の#数字は，対象とする討議において，発話された順番を表している．

4.3 討議ストリーム

討議ストリームとは，同一話題について言及している 1 つ以上の返答を要求する発話と話題を終結する発話による連鎖によって形成される討議の流れのことである．討議ストリームの最初の発話は，返答を要求する発話である．返答を要求する発話に続く発話は，返答を要求する発話か，話題を終結する発話となる．討議ストリームが話題を終結する発話で終結している場合，そのストリームを完全な討議ストリームと呼ぶ．反対に，話題を終結する発話で終結していない討議ストリームを，不完全な討議ストリームと呼ぶ．

4.4 討議構造木抽出エンジン

討議構造木モデルを実際の電子メールコミュニケーションに適用するための実装に，討議構造木抽出エンジンがある．討議構造木抽出エンジンでは，電子メールにおける討議を解析することにより，討議構造木を抽出し，各 XML ファイル群による討議構造木データーベースを構築する．以下に詳細を記述する．

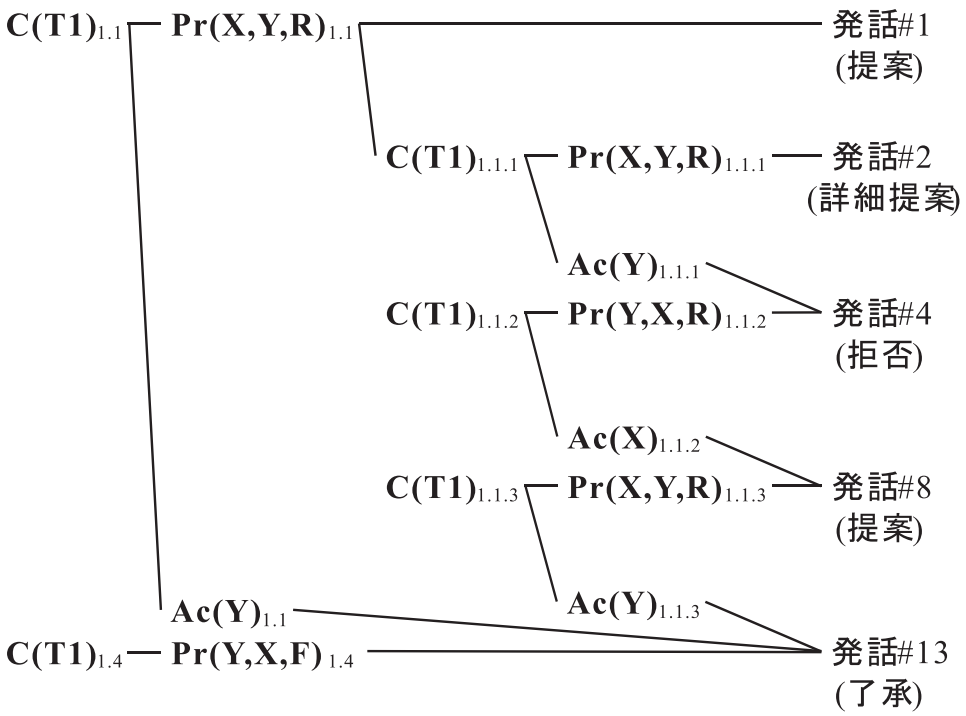


図 4.2: 討議構造木例

4.4.1 XML による討議構造の表現

電子メール群に討議構造木に関する情報を付与するための XML ボキャブラリが「UMML+Linkbase」として定義されている．その概要について以下に述べる．

UMML

XML 仕様に準拠したマークアップ言語で，討議構造木中に現れる発話に関する情報 (Pr フェーズの属性など) を電子メールに付与するための言語 UMML(Utterance-in-Mail Markup Language) を定義した．図 4.3 に示すように、一つの UMML 文書に対しては、一つの電子メールが対応する。なお、別のメールに含まれる発話との接続関係は表現しない。

Linkbase

XLink とは，XML 文書のリンク機能に関する規約であり，リンク文書のリンク機能を規定するための名前空間と属性，制約を与えるものである [?]．図 4.3 に示すように，異なるメールに含まれる発話間の接続関係を，リンクベースに格納したサードパーティリンクとして表現することとし，UMML に対するサードパーティリンクを集めたリンクベースを記述するための XML ボキャブラリ Linkbase の定義を行った．

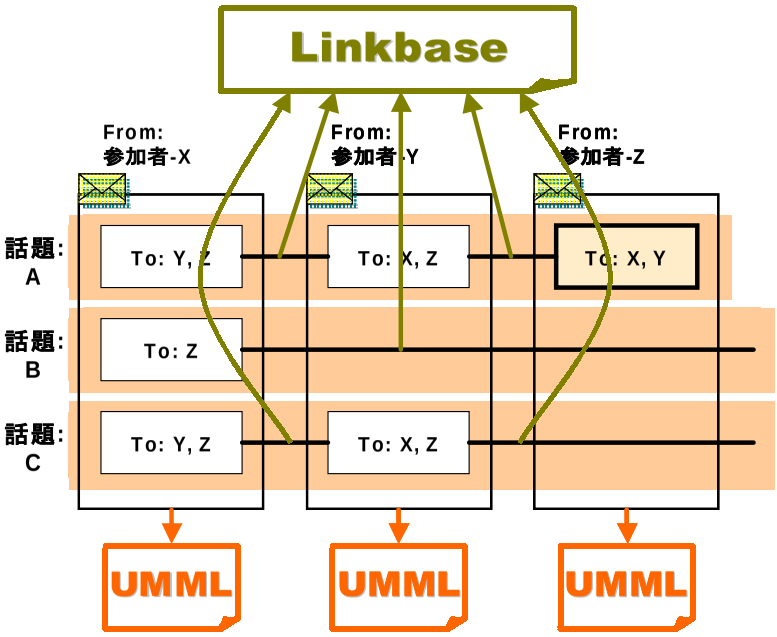


図 4.3: UMML+Linkbase

UMML+Linkbase による討議構造木の表現例

UMML+Linkbase による討議構造木の表現例を，図 4.4 に示す．

4.5 討議構造木の生成

討議構造木抽出エンジンでは，電子メール群から討議構造木を抽出するときに，電子メール本文の記述に多く見られる特徴と，日本語の文章表現に見られる特徴を利用している．以下にそれぞれの説明を記述する．

4.5.1 電子メールボディ部の特徴

ある質問に対する返答メールでは，引用文が利用されているという特徴がある．引用文とは，各行の先頭に “ ¿” や，“ |” といった引用符を伴って表記される文章であり，以前の議論で現れた文章を引用している文章のことである．引用文は，引用文の直後に記述されている発話の先行発話を示すことが多い．

また，メッセージの最初は，自分の名前や身分を示す宣言的な発話であることが多く，メッセージの最後には，一般に “ 署名” と呼ばれるものであることが多い．署名は，自分の名前や身分や連絡先などを示すもので，直前に “ ---” や，“ -=-=-=-” といった記号のみから形成される行 (セパレータ) を伴うことが多い．

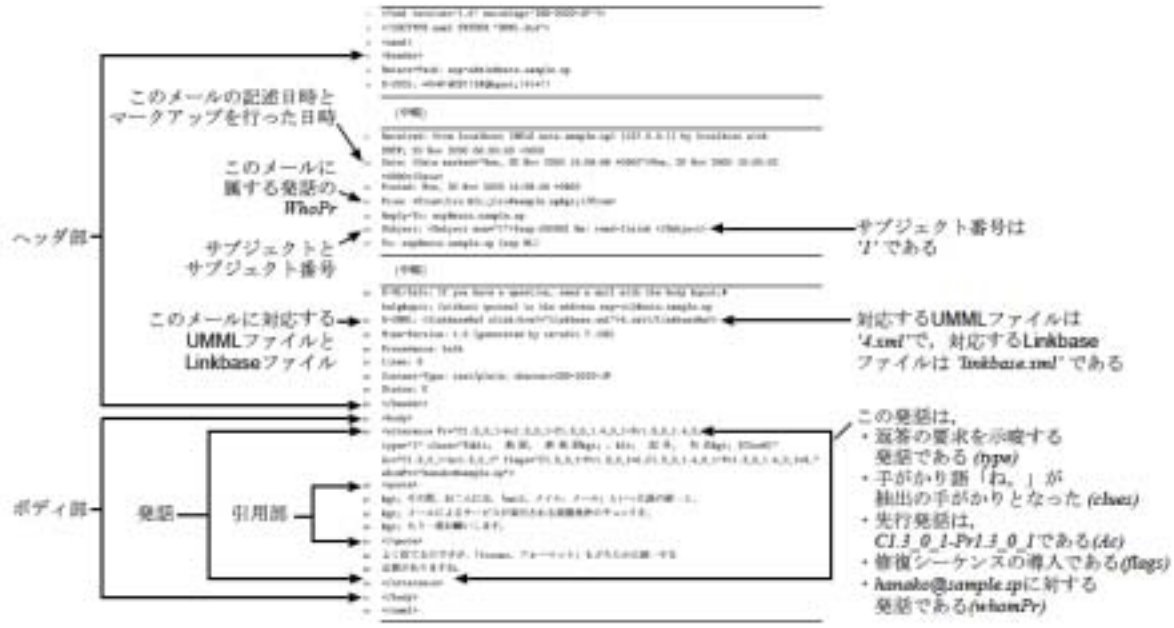


図 4.4: 討議構造木表現例

電子メールを利用したコミュニケーションでは，対面での対話と異なり，会話に一時停止などを利用した話題転換の指示ができない．その代わりに，文章中に意図的に空行や改段落を挿入することによって，話題転換の指示を行っているものと考えられる．したがって，空行や改段落で区切られた文章が発話を構成する基本単位とする．ただし，発話の最低単位は一文とする．

4.5.2 文章表現の特徴

先頭文の文頭や文末などの表現に，話題の転換/終結を示唆したり，文章末に，返答の要求を示唆する表現が含まれていることがある．このような特徴を分類し，利用することで，電子メール中の文章を発話に分割する．表 4.1 に，手がかり語を表記する．

新たな話題の並列展開を示唆する表現

相手からの返答を求めない発話の中には，新たに議論される複数の話題を明示的に示しているものがある．たとえば,「挙げておきます」のような表現は，続く空行や段落で分割される文章を，それぞれ異なった話題の最初の発話とするための手がかりになることが予想される．そこで，これらの表現を表 4.1 中の Clue#1 とする．

示唆する特徴	略号	文章中の存在位置	例
話題の並列展開	Clue#1	文章中	列挙します，挙げます
話題の転換	Clue#2	先頭文の文頭	ところで，〜については
接続語による結束性 (詳細の関係)	Clue#3	先頭文の文頭	具体的には，例えば
接続語による結束性 (前提の関係)	Clue#4	先頭の文中	それで，だから
指示語による結束性	Clue#5	先頭文の文頭/文末	この場合，これは
返答の要求	Clue#6	文末	か，しましょう，？
同意・了承	Clue#7	文末-文末	了解です，確に

表 4.1: 言語的手がかりの一覧

話題の転換を示唆する表現

会話における話題の転換では，それまでの話題を終始，あるいは停止し，新たな議論の開始を示唆する表現が用いられることがある [15][16]．同一メールの中に隣接した発話間においても，発話の先頭文中に話題転換を示唆する文副詞，接続詞があることがある．また，明示的に新たな話題の開始を示唆することによって，話題の転換を示唆することもある．これらの表現を，表 4.1 中の Clue#2 とする．

結束性を示唆する表現

文章中の長さや読みやすさを考慮して，空行や改段落を用いて 1 つの話題の文章を分割することがある．その結果，メール中の空行や改段落によって分割された文章同士でも，同一の話題に言及している場合がある．そのような文章中には，後続の文章中に先行する文章との間の結束性を示唆する指示語や接続語が現れる．結束生とは，文や発話の間の「言語的つながり」である．結束性を示唆する接続語は,「具体的には」のように後続文章が先行文章の詳細に当たることを示唆する接続語群と,「だから」のように先行文章が後続文章の前提に当たることを示唆する接続語群の 2 つのカテゴリに分類できる．これらの表現をそれぞれ表 4.1 中の Clue#3，Clue#4，Clue#5 とする．

返答の要求を示唆する表現

返答を要求する発話には，相手への質問や提案などを示す語，つまり相手への返答の要求を示唆する表現が含まれていることが多い．これらの表現を表 4.1 中の Clue#6 とする．

同意・了承を示唆する表現

話題を終結する発話の場合，文章の先頭文あるいは文末に，相手の発話を理解し同意を示す表現が含まれていたり，先頭文の文頭に簡単を示す語が含まれる傾向にある．その上，先頭文以外の文末に，返答の要求を示唆する表現 (Clue#6) が含まれない傾向にある．これらの表現を 4.1 中の Clue#7 とする．

4.6 討議内容要約エンジン

討議構造木抽出エンジンにおいて抽出され，構築される討議構造木データーベースだけでは，討議の認識の支援を行うことができない．そのため，作成された討議構造木をユーザーに呈示するための方法として討議内容要約エンジンを用いる必要がある．本節では，討議内容要約エンジンについて説明する．

4.6.1 tf*idf 法を利用した要約手法

討議内容要約エンジンでは，各討議構造木から 2 つの発話を重要発話として抽出する．討議構造木における重要発話とは，討議内容の特徴を表している発話のことを指し，実際の抽出は tf*idf 法と呼ばれる単語の重み付け技法 [14] を用いて行われ，ユーザーに討議の内容を呈示するために以下の 4 種類に分類される XML ファイル群が生成される．

全討議内容の参加者リスト: メーリングリスト参加者において，何らかの発話を行ったことのある参加者を参照することができる (図 4.5).

各参加者が関係している討議内容の要約リスト: メーリングリスト参加者において，各参加者が関係している討議内容の要約を参照することができる (図 4.6).

各討議内容の全ての発話内容: メーリングリスト中における，各討議内容の全ての発話内容について参照することができる (図 4.7).

全討議内容の要約リスト: メーリングリスト中における，全ての討議内容の要約を参照することができる (図 4.8).

4.7 討議の呈示における問題点

既存の討議構造木抽出エンジンでは，署名部分を 1 つの発話として認識されることが多く，ユーザーが討議構造木による討議内容の確認を行ったとき，議論には無関係であるはずの署名部分が表示されている状態となっていた (図 4.9).

不必要な発話情報が呈示されることから，ユーザーが目的としている討議内容の発見に時間を要し，結果として必要情報の確認を円滑に行うことへの妨げとなる可能性がある．そのため，分散共同ソフトウェア開発における情報共有支援環境に適用する場合，この問題点を改善する必要があると考えられる．このような問題点の発生の原因として，討議構造木抽出エンジンにおける署名部分除去のためのアルゴリズムを改善する必要がある．

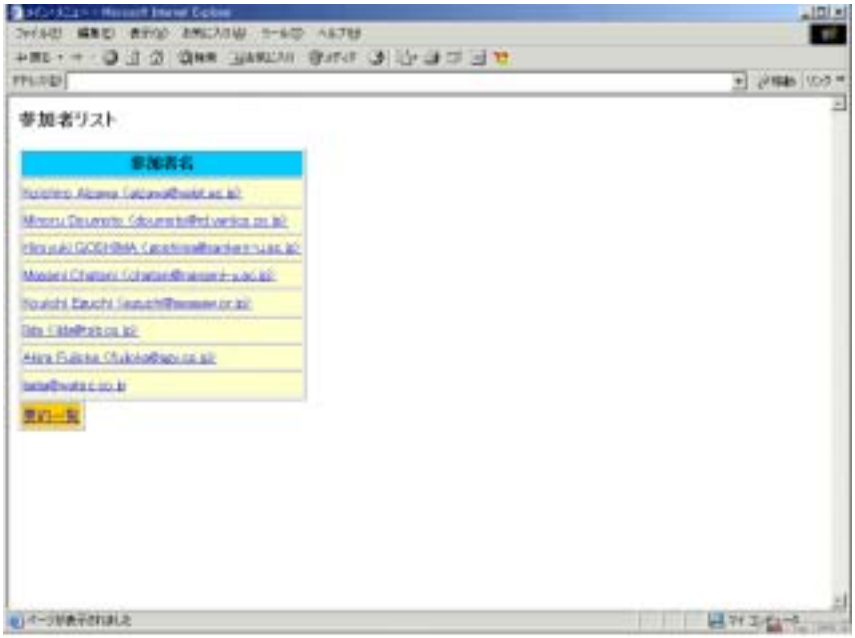


図 4.5: 全討議内容の参加者リスト

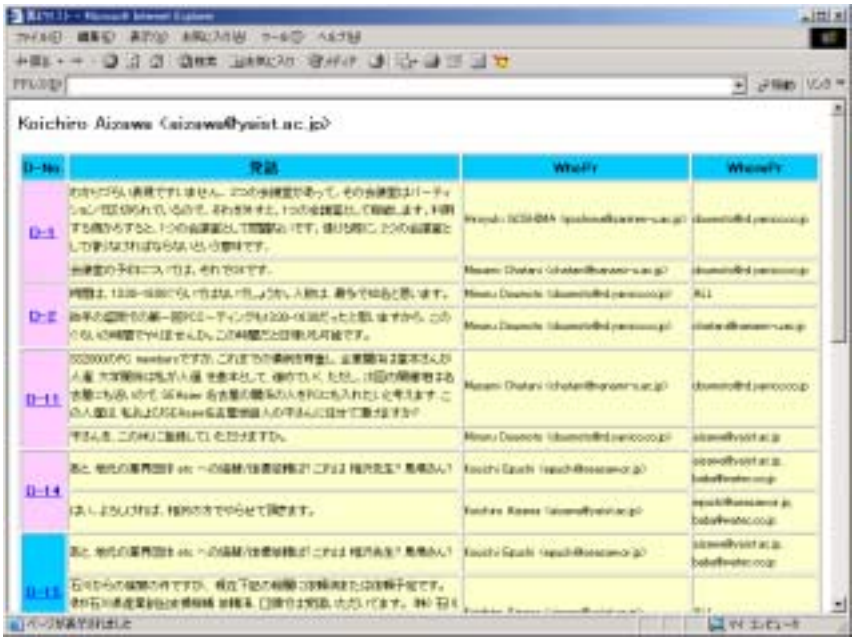


図 4.6: 各参加者が関係している討議内容の要約リスト

図 4.7: 各討議内容の全ての発話内容

図 4.8: 全討議内容の要約リスト

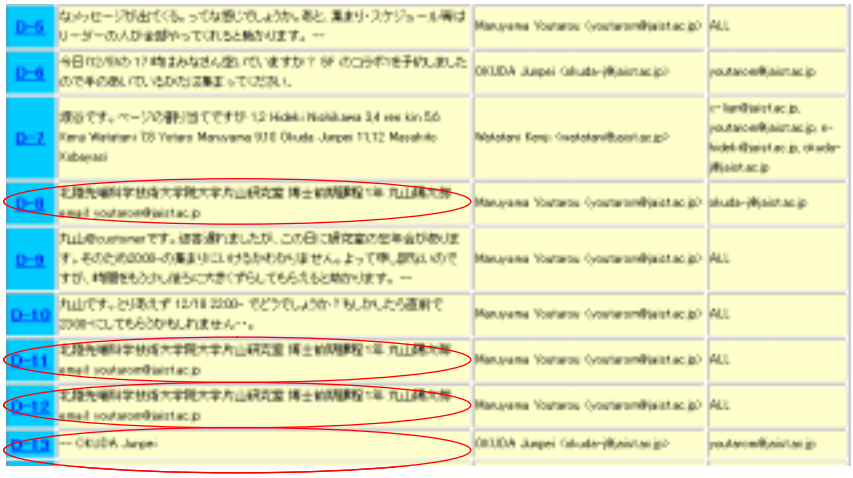


図 4.9: 発話として署名部分が表示

4.7.1 問題発生の原因

討議構造木抽出エンジンにおける既存の署名部分除去アルゴリズムでは，署名部分にはセパレータのみからなる行が含まれているということを前提として設計がなされている．抽出エンジンにおけるセパレータの定義は，“ —” や “ *+*+*” などのように記号のみからなる行とされている．しかし表 4.2 で示されるように，電子メール中に記述されている署名は必ずしもセパレータを伴っているわけではない．また，“ — Nisida” のように，セパレータに送信者を示すための文字列を伴って記述されているものも存在した．

セパレータを伴わない署名部分は，討議構造木抽出エンジンで宣言的な発話か，直前の発話に関連のある発話，もしくは，直前の発話の一部として認識されることとなる．そして，発話として認識された署名部分は討議に無関係であるにもかかわらず，ユーザーに呈示される結果となり，開発者の円滑な討議内容確認の妨げとなっている．

4.7.2 既存の署名除去アルゴリズム

討議構造木抽出エンジンにおける署名部分の除去作業では，電子メール中に署名が存在する場合，署名の直前にセパレータが用いられているという特徴を利用している．これまで抽出エンジンの開発中に利用してきた電子メールと，情報共有管理環境において開発者間で行われた電子メールの 2 種類をサンプルとし，署名部分がふくまれているメール数，署名に本文との区別をつけるためのセパレータが伴われていたメール数を表 4.2 に示す．また既存の署名部分除去アルゴリズムを図 4.10 に示す．

	サンプル 1	サンプル 2
メール総数	35	28
署名が含まれているメール数	34	16
署名部分にセパレータが含まれているメール数	13	14

表 4.2: メールに署名が含まれている割合

```
1     現在処理の対象としているメールの本文を取得する
2     変数 nline に 15 をセット
3     メール本文の行数を変数 line へセット

4     if(line にセットされている値が存在しない){
5         終了
6     }

7     if(line <= nline){
8         本文の先頭行から探索を開始するので変数 nline に line-1 をセット
9         line に 0 をセット
10    }else{
11        本文の最終行より探索を開始するので変数 line に line-15 の値をセット
12    }

13    while(i++ <nline かつ ++line){
14        変数 str に探索対象となっている行の本文をセット
15        if(str はセパレータのみで構成されている){
16            現在の対象行以降を削除
17            終了
18        }
19    }
```

図 4.10: 既存の署名部分除去アルゴリズム

4.7.3 署名除去アルゴリズムの改善

4.7.1 節の問題点を改善するため，署名除去アルゴリズムの変更を行った．新たな署名除去アルゴリズムは，メンバーリストと呼ばれる手がかり語辞書と，署名部分の特徴を利用することで実現している．

メンバーリスト

メンバーリストとは，討議構造木抽出エンジンが発話者の特定を行うために，手がかり語辞書の 1 つとして利用するテキストファイルである．これには以下の定義が適用されている．

- 討議を構成するメール群を格納したディレクトリに配置する
- 1 つのメールフォルダにつき 1 つのファイルとする．

- ファイル名は members とする．
- メンバーリストは次のフォーマットからなる．
  CSV 形式，参加者 1 人につき 1 行．先頭に参加者のメールアドレス．
  それ以降に，その参加者を示す語．語数は問わない．

この定義によるメンバーリストの性質から，署名部分の判定に応用することができる考えられる．

新たな署名部分の特徴

これまで除去作業に利用されていた署名部分の特徴とは，署名部分はセパレータとなる行を伴って記述されるということであった．しかし，実際にはセパレータを伴わず記述されているメールもある．

このため既存の討議構造木抽出エンジンでは，セパレータを伴っていない署名に関しては除去されない．そこで新たに署名部分に見られる特徴について調査を行った．調査の結果として，署名と本文の間には 1 行以上の改行のみからなる行が挿入されているという特徴が発見された．

メンバーリストと署名部分の特徴を利用し，新たに考案した署名除去アルゴリズムを図 4.11 に示す．

```
1       現在処理の対象としているメールの本文を取得する
2       メール本文から送信者のメールアドレスを取得し，変数 from へセット
3       メール本文の行数を変数 line へセット

4       from を参考に members ファイルから送信者の名称を取得し str へセット
5       str を各名称毎に分割し配列 sender にセット

6       for(i = line; i >= 0; i–){
7           if(i 行目に sender にセットされた名称が含まれる and
             句読点など 1 文の終了を表す記号が含まれていない){
8                 i 行目から最終行までを削除
9                 変数 signexist に 1 をセット
10          }

11          if(signexist == 1 and
              (i 行目はセパレータのみからなる行である or
              i 行目は改行のみからなる行である)){
12              i 行目を削除
13              署名除去作業終了
14          }else if(signexist == 0 and i == line-5){
15              署名除去作業終了
16          }
17      }
```

図 4.11: 改善後の署名部分除去アルゴリズム

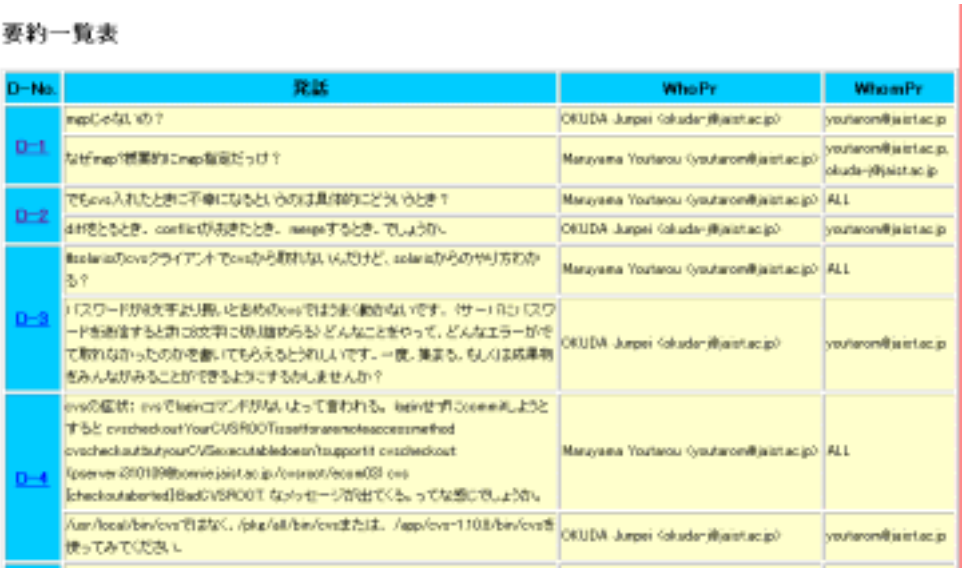
4.7.4 アルゴリズム改善の効果

新たに設計したアルゴリズムを討議構造木抽出エンジンに適用した結果，ほぼ全ての署名部分を除去することが可能となった．実際の数値について表 4.3 に示す．

	改善前	改善後	メール総数 (署名の有るメール数)
サンプル 1	13	33	35(34)
サンプル 2	13	16	28(16)

表 4.3: 署名部分が除去されたメール数

この成果から，これまで討議構造木を呈示した場合，発話として署名部分が呈示され，ユーザーの討議認識の妨げとなっていた問題点が解消された．実際に署名部分が除去された呈示部分について図 4.12 に示す．


要約一覧表

D-No.	発話	WhoPr	WhomPr
D-1	mapじゃないの？	OKUDA Junpei <okuda-j@jaist.ac.jp>	youtarom@jaist.ac.jp
	なぜmapが標準的にmap駆動だっけ？	Maruyama Youtarou <youtarom@jaist.ac.jp>	youtarom@jaist.ac.jp, okuda-j@jaist.ac.jp
D-2	でもcvs入れたときに不便になるというのは具体的にどういうとき？	Maruyama Youtarou <youtarom@jaist.ac.jp>	ALL
	diffをとるとき，conflictがおきたとき，mergeするとき，でしょうか．	OKUDA Junpei <okuda-j@jaist.ac.jp>	youtarom@jaist.ac.jp
D-3	他solarisのcvsクライアントでcvsから取れないんだけど，solarisからのやり方わかる？	Maruyama Youtarou <youtarom@jaist.ac.jp>	ALL
	パスワードが8文字より長いとあのcvsではうまく動かないです．（サーバ側にパスワードを送信するときに8文字に切り詰められるとどんなことをやって，どんなエラーがでて動かなかったのかを書いてもらえるとうれしいです．一度，集まる，もしくは成果物をみんなが見ることができるようにするかしませんか？	OKUDA Junpei <okuda-j@jaist.ac.jp>	youtarom@jaist.ac.jp
D-4	cvsの症状：cvsでloginコマンドがないよって言われる。loginせずにcommitしようとすると cvscheckoutYourCVSROOTissetforaremoteaccessmethod cvscheckoutbutyourCVSexecutabledoesn'tsupportit cvscheckout (pserver:d01099@cvnie.jaist.ac.jp:/cvsroot/ecore03 cvs) [checkoutaborted]BadCVSROOT なメッセージが出てくる。ってな感じでしょうか。	Maruyama Youtarou <youtarom@jaist.ac.jp>	ALL
	/usr/local/bin/cvsではなく，/pkg/all/bin/cvsまたは，/app/cvs-1108/bin/cvsを使ってみてください	OKUDA Junpei <okuda-j@jaist.ac.jp>	youtarom@jaist.ac.jp

図 4.12: 発話として表示されていた署名部分が除去された討議構造木

4.7.5 問題点

署名部分除去アルゴリズムの変更により，以前のものと比較すると除去精度の向上が見られた．しかし，今回のサンプル中に以下に示すような電子メールが存在した．

図 4.13 に一例として挙げた電子メールは，1〜14 行目において，討議内容に関する記述がなされている．続く 15 行目がこの電子メールにおける署名に相当する．しかし，その

B さん；

これまでの SS における Tool 展示の扱いはおおよそ次のようなものだったと記憶しています：

(1) 展示料金は無料:

(2) ただし，展示要員のうち最低何人か (１人かな？) は正式に SS に参加登録してもらう.

(3) また，展示希望者に次の点をはっきりいっておく必要があるでしょう：

- SS での展示はふつうの商用の展示会と違って小人数 (たかだか 100 〜 200 人) のソフトウェア開発技術者・管理者を対象とするものであり，販売目的というより，技術のデモという色彩である.

- したがって特におおげさなブースなどは作らず，ただマシンを置く机だけしか用意しない.

(4) 論文発表者には，もし関連する展示が可能なら Welcome ということをいってあげたほうがよいでしょう．この場合，その人自身は会議参加登録をしているわけだから，他に何人か (常識の範囲内: 1〜2 人) は無料で参加することが可能になる (?). これを許すかどうかは, PC あるいは Program Chair の判断.

E@JAIST

PS: 昨夜の SEAsaw 幹事会で A さんには申し上げたのですが，ASAP で Call-for-Papers を finalize して, その URL Address を seasaw@seasaw.or.jp に教えて下さい. SEAsaw の WebPage から Link を張ります.

また, Net News (fl.meetings, fj.comp.announce, etc) にポストすることもよろしく.

図 4.13: 改善後アルゴリズムで除去できなかったメールの例

後の 16〜19 行目にかけて発話の追記がなされている．このように，発話と発話の間に署名が挿入されている場合，今回の署名除去アルゴリズムでは対応することができない．

本研究で集めた電子メールサンプル中には図 4.13 で参照したメールにのみ，このような文章構造が見受けられた．今後このような文章構造を持った電子メールが頻繁にやりとりされるならば，このような署名除去に対応したアルゴリズムについて調査していく必要がある．

第5章　情報共有システムの構成とその運用

3節と4節による結果から情報共有システムの構築を行った．そして2003年度のソフトウェア設計演習を対象に，構築したシステムの運用による情報共有支援環境の提供を受講生に対して行った．本章では，構築した情報共有システムの構成を述べるとともに，運用から得られたいくつかの知見について説明する．

5.1　情報共有システムの構成

図5.1に構築した情報共有システムの構成を示す．

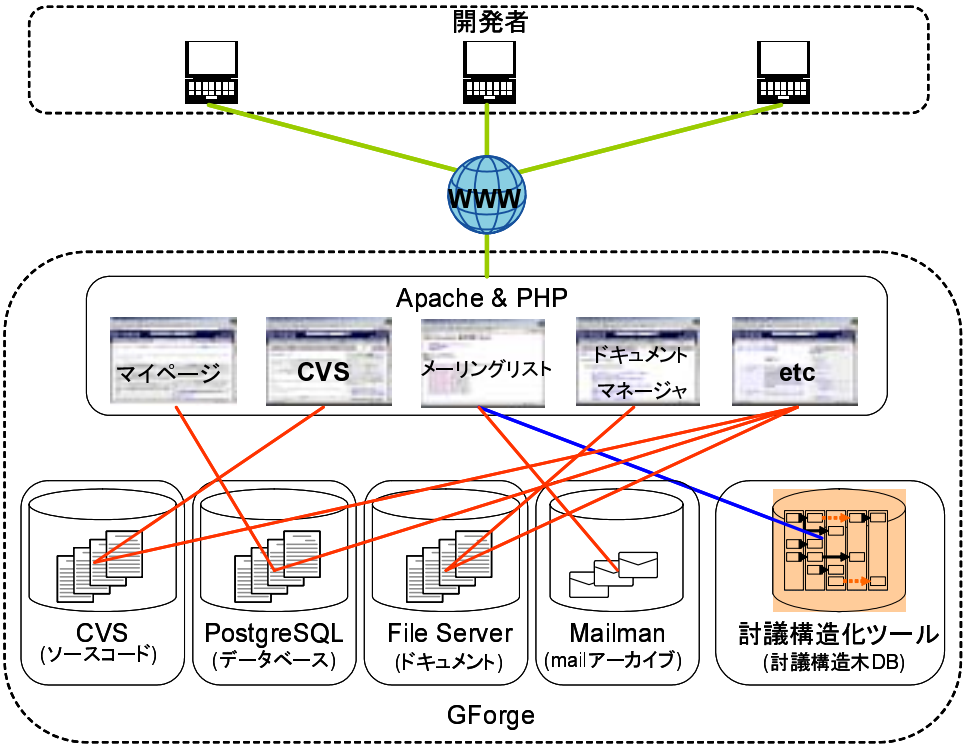


図 5.1: GForge と討議構造化ツールによる情報共有システムの構成

各システムの採用状況は以下の通りである．

1. ホームページデザインの構成は GForge に準じる．
2. CVS による版管理支援は既存のまま採用した[1]．
3. ドキュメントマネージャは GForge に準じる．
4. メーリングリスト，電子掲示板は GForge に準じる．さらにコミュニケーション支援機能の強化として討議構造化ツールによる支援を追加した．
5. ファイルリリース・ダウンロード用サーバは今回利用しなかった．
6. バグトラッキング支援に関しては準備が間に合わなかったため提供を見送った．
7. マイページによる個人の活動記録を登録しておくサービスは GForge に準じる．

5.2 試験運用

本学で 2003 年 12 月から 2004 年 2 月にかけて開講された，ソフトウェア設計演習の講義において，構築した情報共有システムの試験的な運営を行った．演習における開発環境と情報共有システムは図 5.2 で示す構成となっており，各開発者は Eclipse[17] と UML プラグインを用いた開発環境と，情報共有支援環境を利用し，共同ソフトウェア開発を行う．

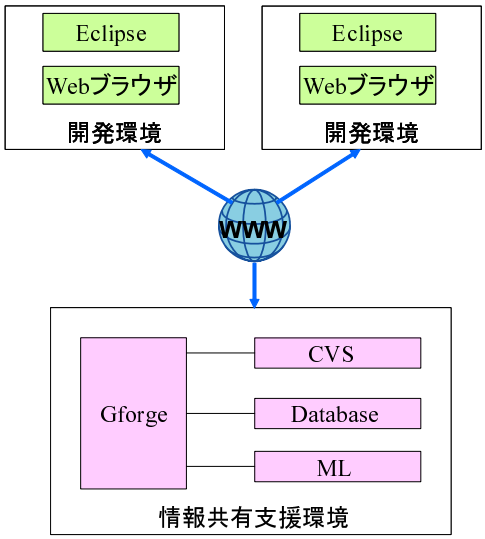


図 5.2: 開発環境と情報共有支援環境

また，図??はソフトウェア設計演習における情報共有支援環境のトップページである．この場所には，講義の進行に必要と考えられる情報 (講義のスケジュール，開発環境のダウンロード，各種ツールの利用手引きなど) を掲載している．これらの情報は，講義受講者全体に必要とされる情報であるため，情報共有支援環境においてもっともアクセスしや

[1]将来は我々の研究室で開発した拡張 CVS の利用を検討している [19]

すいと考えられる位置に配置した．開発者個人や開発グループごとの情報，情報共有システムによるサービスの利用は，個人のアカウントを作成し，情報共有支援環境へログインすることで利用することができる．本環境は WWW を介した情報共有支援環境として提供されるため，各開発者は開発場所に制限されることなく，Web ブラウザを介して各サービスを利用することができる．

前回において設計・構築した情報共有支援環境と比較した場合，問題とされていたメンバーの管理と，コミュニケーション情報共有に関する問題が改善された．また，プロジェクト管理者の負荷の軽減も可能となり，全ての開発者がより円滑に作業を行うことが可能となった．

5.3 利用状況

2003 年度のソフトウェア設計演習は以下のように実施された．

1. 課題説明，開発環境説明，グループ分け，開発環境設定
2. 各事例研究のモデリング分析[2]
3. モデリング結果報告会
4. 開発環境の利用法習熟
5. プロジェクトプランニング (インクリメント設計，版管理方針，ベースライン設計，作業分担)
6. プロトタイピング計画報告会
7. 第一次プロトタイピング成果報告会
8. 各事例研究のかんどころと統合テスト法と回帰テスト法の解説
9. 第二次プロトタイピング成果報告会
10. 各種ソフトウェア設計方法論の比較
11. アルファテストの実施
12. ドキュメンテーションの作成
13. 最終報告会

上記のような演習の進行において，準備した情報共有システムは以下にように活用された．

1. GForge が提供するホームページは上記活動のあらゆる場面で活用された
2. 版管理支援は，ソースコードの版管理に有益であったことは言うまでもないが，それ以外にプロジェクトの進捗状況の確認・報告に活用された．CVS リポジトリの構造は，タッギングによる CVS バージョンの管理を合わせて，そのままプロジェクトの進捗の把握に活用できる

[2]講義期間が短いこともあり，ユースケース駆動プロセスによるモデリング作業は行わず，COMET[20] による結果を理解し必要なら改良する手段を採用した．

3. ドキュメントマネージャによるUML文書の管理は報告会でそのまま活用された．具体的には，図7に示すページにおいて，報告会で使用するドキュメントを公開することで，他のグループの開発者が参照することができる．これにより，特別な配付資料を用意することなく，常に発表者が使用しているドキュメントと同じものを参照しながら報告を聞くことが可能となっていた
4. 討議構造木抽出・要約エンジンは残念ながらあまり活用されなかった．その理由として，以下の点が考えられる．まず，頻繁に実施されたのはface-to-faceのミーティングである．共同開発ではあるが分散開発ではないので，必要性を感じなかったものと考えられる．実際にはface-to-faceミーティングにおける構造化議事録としての用途が大事なのであるが指導が十分でなかった．さらにWebで統合されているとはいえ，独立したツールであったこともその原因として考えられる．作業や話し合いの流れの中で，連動して活用できる必要があるが，そこまでの話し合いの構造を眺めてみるという使い方ではあまり有用性を感じなかったものと思われる
5. GForgeによる情報共有支援は教官にとっても有用であった．すなわち，プロジェクト統計情報の提供機能は，図5.3で示されるものからアクティブな参加者の特定をおこなったり，各開発グループにおける障害（リスク）の予測にある程度役にたった

5.4 今後の改良点

上記の利用経験から，今後以下の点について改良を行う．

ホームページデザインの再考

情報共有支援環境を利用した場合，どのページにアクセスすれば目的とする情報が取得できるのか分かりにくいという指摘が受講者から受けた．そこで，ユーザーが分かりにくいとした部分を調査し，ホームページデザインの再考を行う必要がある．

管理対象情報の連携支援

ドキュメントやソースコードの生成の過程においてなされた議論や，定められた方針などの再確認のときに，どの情報を見れば良いか分からないや，情報の探索に時間がかかるという問題があった．そのため，取り扱っている情報間での連携を可能とするための支援について調査し，実装する必要がある．

サーバ障害への対処

2ヶ月間の運用期間において，情報共有システムのサーバが停止することがあった．停止時間は1時間ほどの短時間であったため，開発に深刻な影響を及ぼすことはなかったが，今後の長期的な運用の計画に対しては，さまざまな障害への対応を検討しておく必要がある．

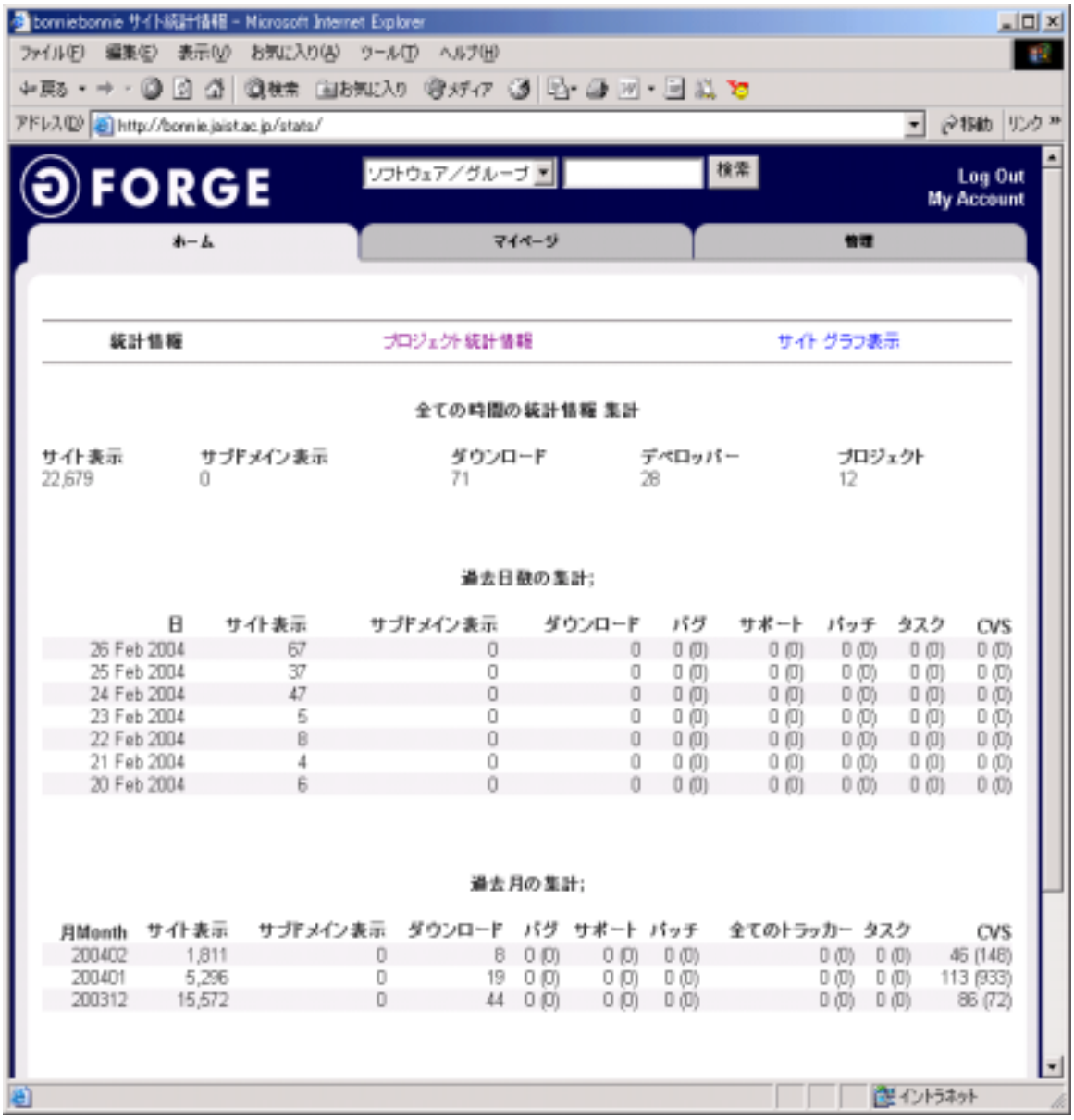


図 5.3: 情報共有支援環境の利用統計情報ページ

利用手引きの充実

情報管理画面からの操作が分からないや，ある情報に対して操作を行った場合の影響が分からない等の質問が多数寄せられた．そのため，現在ある利用手引きをさらに充実させていくことが必要である．

第6章　おわりに

6.1　まとめ

本研究では，分散共同ソフトウェア開発には中間プロダクトに関する情報共有と，コミュニケーションに関する情報共有の双方の支援が必要であると定義し，それを実現するための既存システムによる環境の構築をおこなった．そして，本学で開講されている講義の1つであるソフトウェア設計演習において適用し，受講者が利用できる環境を提供することによる試験運用を行った．その結果，既存のシステムを単純に利用するだけでは，プロジェクト管理者や，コミュニケーション情報の蓄積に問題点があると指摘し，円滑なソフトウェア開発を行うことができないと判断した．そのため，分散共同ソフトウェア開発で大規模なソフトウェアの開発を行っている，オープンソースソフトウェアの調査を行い，既存システムを利用するのみならず，それらを必要に応じて連係し，情報共有支援環境を構築することが可能な GForge システムを用いて再度，環境の構築を行った．このようなシステムを用いて構築された情報共有支援環境では，それまでに構築していた環境で見られた問題点が解消され，円滑に分散共同ソフトウェア開発作業を進めていくことが可能であった．しかし，コミュニケーションに関する情報共有支援には，開発時間の経過に伴い討議の数が増加し，過去の議論を参照しようとした場合に多大な時間が必要になる可能性があった．このような問題点を解消するため，GForge によって構築される情報共有支援環境に討議構造モデルの概念を適用した．これにより，電子メール単位での蓄積と共有によるコミュニケーション情報共有支援に加えて，議論単位での蓄積と共有が可能となるため，指摘した問題点を解消できると考えられる．そして，討議構造モデルの概念を取り入れた GForge による環境を開発者が利用できるよう構築し，前回と同様にソフトウェア設計演習への適用を行った．

6.2　今後の課題

本研究における今後の課題について以下に挙げる．

- 新たなコミュニケーション手段の調査

  今回，特に情報共有支援環境におけるメーリングリストシステムの情報共有に関する問題点の改善を行った．しかし，電子メール以外にもネットワークを介したコミュニケーションを実現するためのツールは存在し，より直接的な対面によるコミュ

ニケーションに近い形として提供することが可能なツールもある．電子メール以外のコミュニケーション手段を調査しすることで，開発の方向性の発散に関する問題点のさらなる改善が見込める．

- 継続的な運用と評価手法の考案による新たな知見の獲得

  構築した情報共有支援環境は，本学における講義の 1 つであるソフトウェア設計演習において試験運用を行った．しかし，大学における講義ということから，ソフトウェアの開発期間が 2ヶ月という限定された期間であったことと，開発者の地理的分散がほぼ存在することない環境であったという点から，実際の分散共同ソフトウェア開発における環境との相違があった．そのため，今後も継続的に構築した情報共有支援環境を運用し，更に広く分散している開発者によるソフトウェア開発に適用していき，新たな問題点について検討していく必要がある．また，情報共有支援環境に対する評価手法を考案することで，分散共同ソフトウェア開発における開発者に対して，このような環境の提供からどのような効果が得られるかを検証し，構築した情報共有支援環境の有効性を調査しなければならない．

- 電子メールにおける英語コミュニケーションへの対応　異なる言語を利用するユーザー同士がネットワークを介してコミュニケーションを行う場合，使用されている地域が広範囲にわたって分布していることによる優位性と，コンピュータにおける優位性から，高い頻度で英語が用いられる [18]．この事実は分散共同ソフトウェア開発においても適用されると考えられる．

  しかし，現在の討議構造モデルの実装である討議構造木抽出エンジンでは，日本語における電子メールコミュニケーションから討議構造木を抽出する機能しか有していない．そのため，今後の情報共有支援環境では，英語コミュニケーションに対応した，討議構造木自動抽出システムの実装が必要となる．

謝辞

本研究を行なうにあたり、終始変わらぬ御指導を賜わりました落水浩一郎教授に心から深く感謝申し上げます。

元落水研究室の故村越広亨助手には、本研究を始めるにあたり、多大な御助言と貴重なご意見をいただき深く感謝致します。

論文審査にあたり、片山卓也教授、篠田陽一教授には御助言、御意見をいただき深く感謝致します。

本研究室卒業生の渡邊大貴氏には、本研究を始めるにあたり、多大な御助言をいただき深く感謝致します。藤枝和宏助手、服部哲助手には多大な御助言をいただき深く感謝致します。

落水研究室の皆様の日頃の討論と助言、励ましに深く感謝致します。

大学時代に情報科学における研究の道を示してくださった小板隆浩先生に深く感謝致します.

最後に進学への援助を行ってくれた両親と，私生活においてお世話をしてくださった友人に深く感謝致します.

参考文献

[1] Ulf Asklund,Boris Magnusson,and Annita Persson. “ Experiences: Distributed Development and Software Configuration Management”. In Proceedings of SCM-9,Ninth International Symposium on System Configuration Management, J.Estublier(Ed.),LNCS1675,pp.17-33,Toulouse,France,1999.

[2] Concurrent Version System: http://www.cvshome.org/

[3] Source Forge: http://sourceforge.net/

[4] Eric S. Raymond. “ The Cathedoral and the Bazar”. http://www.catb.org/~esr/writings/cathedral-bazaar/

[5] Apache: http://www.apache.org/

[6] qmail: http://www.qmail.org/

[7] GForge: http://gforge.org/

[8] General Public License: http://www.gnu.org/licenses/gpl.html

[9] 落水浩一郎. “ 漸増的ソフトウェア設計・実現のためのプロセスモデル -ソフトウェア分散共同開発における調整支援-”. 日本ソフトウェア科学会, コンピュータソフトウェア, Vol.15, No.4, pp.73-77, 1998.

[10] Herbert H. Clark and Edward F. Schaefer. “ Contributing to Discourse.” Cognitive Science, Vol.13, No.2, pp.259-294, 1989.

[11] Hiroyuki Murakoshi, Akira Shimazu, and Koichiro Ochimizu. “ Construction of deliberation structure in e-mail communication.” International Journal of Computational Intelligence, Vol.16, No.4, pp.570-577, 2000.

[12] D.G.Novick, Walton, and K.Ward. “ Contri-bution graphs in multiparty discourse.” In Pro-ceedings of International Symposium on Spoken Dialogue, pp.53-56, 1996.

[13] 村越広亨，島津明，落水浩一郎．“ メーリングリストを利用した共同作業における討議構造モデルの提案”．信学技報，SS2000-3,Vol.100,No.63,pp.17-24,May 2000. ソフトウェアサイエンス.

[14] Salton, G. “ Automatic Test Processing” Addison-Wesley, 1989.

[15] Rachel Reichman. “ Conversational Coherency” Cognitive Science, Vol2, No.4, pp283-327, 1978.

[16] 泉子・K・メイナード. “ 会話分析” くろしお出版, 1993.

[17] Eclipse. “ http://www.eclipse.org/”.

[18] 中山順子，宮増フラミニア，宮尾真理子．“ 英語とインターネット”．東京家政学院筑波女子大学紀要第３集，pp.77-93，1999．

[19] 早坂良，藤枝和宏，落水浩一郎．“ プロセス支援機能を組み込み可能な CVS プロキシ構成法”．電子情報通信学会，信学技法，SS2002-51，March，2003．

[20] Hassan Gomaa. “ Designing Concurrent, Distributed, and Real-Time Applications with UML”. ADDISON-WESLEY，2000，ISBN:0-201-65793-7.

本研究に関する発表

[1] 西田和豊，小谷正行，落水浩一郎．分散共同ソフトウェア開発における情報共有支援方式に関する研究．情報処理学会　第 144 回ソフトウェア工学研究会．March 2004.