ソフトウェアプロセス実行における系統的コミュニケーション支援の一方式

Systematic Support of Communication in Enacting a Software Process

門脇 千恵　落水 浩一郎
Chie KADOWAKI　Koichiro OCHIMIZU

1997 年 3 月 13 日

IS-RR-97-0009S

北陸先端科学技術大学院大学　情報科学研究科
Graduate School of Information Science,
Japan Advanced Institute of Science and Technology

〒 923-12 石川県能美郡辰口町旭台 1 丁目 1 番地
1-1 Asahidai, Tatsunokuchi, Ishikawa 923-12 Japan

kadowaki@jaist.ac.jp
ochimizu@jaist.ac.jp

ISSN 0918-7553

アブストラクト

本論文ではソフトウェアプロセスの実行に伴なって発生する種々のコミュニケーションの構造的記録手段について論じる．ソフトウェアプロセスとオブジェクトベース，オブジェクトベースとコミュニケーション記録の関係については，すでに多くの研究がなされている．しかし，仕事の引き継ぎやフォーマルレビューに関して発生する会話をソフトウェアプロセスの実行履歴と関連づけて記録する手段はまだ十分に研究がなされていない．本稿では，グループウェアベースなる新しい概念モデルを導入することにより，(1) 討議間の因果関係の保持，(2) 一つの話題に対応する討議の流れの保持，(3) 一つの討議の流れにおける主要な会話の構造的保持，等を特徴とする記録手段を検討する．本論文で提案するグループウェアベースと，それをソフトウェアプロセスに結合する手段を基にして，分散開発時代のソフトウェア開発環境に対するフレームワーク，CSCSD framework を提案する．ソフトウェアプロセスの実行系（人間）からグループウェアベースに格納されるべきデータを収集する手段，実行系に提供する手段についても論じる．本論文で得られた諸概念を ISPW6 の例題に適用することにより，ソフトウェア開発過程の連続的な記録が可能になることを示す．

Abstract

In this paper, we discuss the structured recording method of communication that occurs during a software process enactment.

There are lots of research results that consider the relationship between a software process and an object base, and between an object base and design rationales.

We have not, however, enough results that discuss the method combining history of software process enactment with the communication occurred in handing some artifacts to the other or in performing formal review.

In this paper, we will give a solution to this problem by introducing the new conceptual model named Groupwarebase that stores the following information:

(1) cause-effect relationships among deliberation processes (2) stream of deliberation for a topic (3) major discussions in a stream of deliberation.

We also give a framework named CSCSD(Computer Supported Cooperative Software Development) for distributed software development support environments based on Groupwarebase and links between deliberation processes and a software process.

We also discuss the acquiring method of information to be stored in a Groupwarebase from conversation among project members.

We show that our model is useful and effective to record a software development process continuously by applying our concepts to the ISPW6 exercise.

目 次

図目次

1 はじめに

近年，ソフトウェアの品質や開発工程の見直しが重視されはじめ，ソフトウェアプロセスの研究が盛んになってきた．W.Humphrey は，その著書の中でソフトウェアプロセスとは「ソフトウェアの生産および進化の中で使用されている活動，手法，および慣習の集合」であると定義している [1]．ソフトウェアプロセスの記述に関しては種々の研究成果があるが [2]，上述の<活動，手法，および慣習の集合>の中にコミュニケーションまで絡めて形式化した研究はあまりなく，多くの場合，中間成果物の変換部分を主な記述対象としている．一方，開発過程で発生するコミュニケーションの記録手段に関する研究は多々あるが [3]，それらはソフトウェアプロセスとは独立に設計の根拠（Design Rationale）を収集，記録，利用することを目的としている（図 1.c）．本稿では，中間成果物の変換に係わるソフトウェアプロセス記述を特に**変換プロセス**と呼び，ソフトウェアプロセスとは区別する．すなわち，ソフトウェアプロセスとは，変換プロセス（図 1.a）と，その実行系（図 1.b），変換プロセスに付随して発生するコミュニケーション（図 1.c）の 3 者を統合したものであるという考え方をとる．

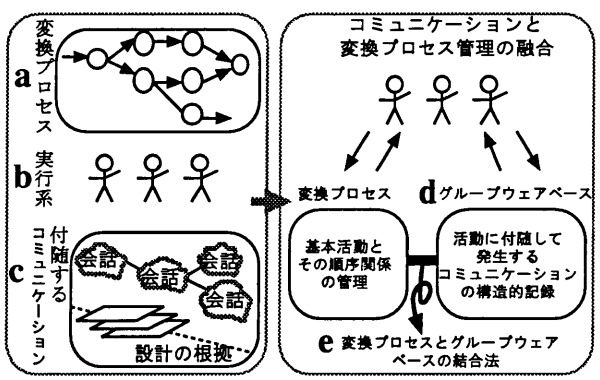


図 1: ソフトウェアプロセス構成要素の再吟味
Re-examination of software process components.

ソフトウェア開発においては，変換プロセスとコミュニケーションは切り離し難い．上流工程での現状調査や折衝などはコミュニケーションによって成され，その内容は成果物となる．また，下流工程においては，不明点の究明や技術指導など，変換プロセスを補完するためのコミュニケーションが必要である．工程全体で概観しても変換プロセスとコミュニケーションの間には以下のような密接な関係がある．

- 開発チームは，スキルやバックグラウンド，考え方など違う様々なメンバーよりなる．会話は，この相違をなくす捕捉手段としてよく使われる
- 作業移行時には，成果物に関する内容を十分に伝達する義務が生じ，会話によって果たされる

- 会話は生きたマニュアルである．単に情報伝達というだけではなく，その人が培ってきたノウハウのような付加情報が暗黙裡に含まれている

ソフトウェア開発において，コミュニケーションが活用される主な理由は，情報伝達の手段として最も容易で迅速な上に，コストがかからない点にあると思われる．例えば，仕様書中の要求変更の内容を理解する際，変更結果の記述のみでは変更に至った経緯は分らない．しかし，どのような説明を書き足せば十分であるかについては，読む人の状況によって異なり，そのような情報を仕様書に全て記載することは現実的でなく，また，遺漏のない記述を強いられた側は大変である．そこでこのように面倒な，また重複する情報は，記述よりは数段楽な会話によって伝達される傾向がある．

また，ソフトウェアプロセスの基本的計算要素の実行者である人間は誤り易く，その計算結果には再現性が無い．そのため，ソフトウェアプロセスの実行には柔軟な誤り回復機構が必要であるという片山による指摘がある [4]．この人間性より生じる不確実さの一部は，コミュニケーションによって補われる場合が多い．

以上より，中間成果物の作成手順の管理だけでは開発の実態と合わず，チームの協調維持の支援には限界があると判断した [5]．すなわち，変換プロセスの流れに沿って発生するコミュニケーションを構造的に記録する方式を開発することにより，開発/保守時の阻害要因の一部を緩和し，また，作業を円滑に進めるためにそれらを活用するための基礎を与える必要がある．

以下，２節ではコミュニケーションを記録することの意義を述べ，本論文で導入するグループウェアベース（図 1.d）のモデル化の視点を明らかにする．３節ではグループウェアベースの構成要素と構造化法を検討し，コミュニケーションの動的特性（再編成や蒸し返し）への対応を示す．４節では変換プロセスとグループウェアベースの結合法（図 1.e）を論じ，ISPW6* の例題への適用例を示す．また，ソフトウェア開発におけるグループウェアベースの役割を考察する．５節では，４節までの成果をふまえた上で，グループウェアベースとオブジェクトベース，グループウェアベースと実行系（人間系）の関係を検討しつつ，ＣＳＣＳＤ (Computer Supported Cooperarive Software Development) フレームワークの提案を行なう．６節にまとめを述べる．

*The 6th International Software Process Workshop

2 グループウェアベースの導入

本節ではコミュニケーション記録の視点を論じつつ，グループウェアベースなる概念を導入し，そのモデル化の方針をまとめる．

2.1 コミュニケーション記録の視点

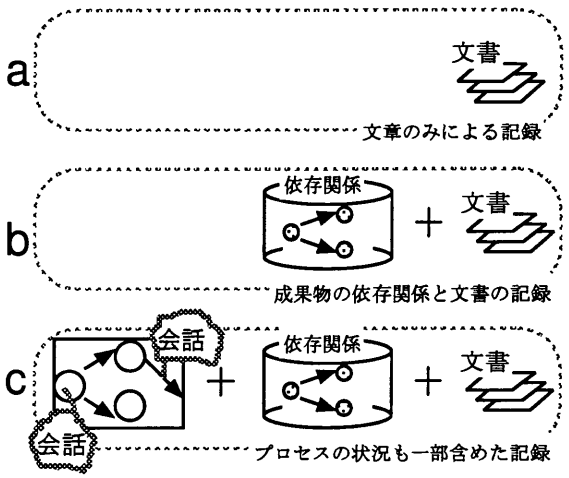


図 2: 開発過程記録の三つの方式
Three recording methods of a software process enactment.

前節では，ソフトウェアプロセスにおいてコミュニケーションが果たす役割を述べた．本節では，コミュニケーション記録の意義を分析しつつ，モデル化の視点をまとめる．

図 2は，開発過程の記録の三つの方式を比較したものである．ソフトウェア文書のみによる記録（図 2.a）は，ソフトウェアプロセスによる連続的な変換活動のある時点における場面を，スナップショット的に記録したものであり，各文書に記載されている内容の因果関係が掴みにくい．例えば，仕様書に記載されている種々の要求は，ユーザの要望と実現技術，コストのトレードオフから生まれたものであり，また，設計書やプログラムに記載されているシステム構造やプログラム構造は，性能を考慮に入れた一つの最適化の方針が現れたものであるが，一般にはそのような情報の記述は明示的ではない．

一方，文書を細粒度レベルの構成要素に分解し，その間の依存関係と共にオブジェクトベースで管理する方式（図 2.b）があり，成果物間の静的な追跡可能性（traceability）は保証される．この時，変換プロセスはソフトウェアオブジェクト間の依存関係を制約条件として，ソフトウェアオブジェクトの作成順序を定めることによって定義される．一般には作成順序は複数個存在するが，その内一つが選択され，開発チーム全体に対する作業のガイドラインを与えることになる．

ところで，同じ変換プロセスであっても，その実行過程は同一ではなく，実行系に依存した固有の実行状況を持つ．この動的で固有な状況も併せて記録できれば，成果物の得ら

れた経緯を推測するための情報を得ることができる．実行状況の全てを収集・記録することは不可能であるが，一部の情報はコミュニケーションを記録することによって得ることができる（図2.c）．例えば

- 記録目的1 変換プロセス実行時に発生している様々な会話間の関係やその流れより，進捗状況など変換プロセスの実行状況が把握できる
- 記録目的2 ある特定の話題に対する会話の流れ（時系列な変化）を記録しておくことにより，代替案の取捨選択の意図や確実性ある結論の前提条件などが把握しやすくなる
- 記録目的3 ある特定の話題に対する会話の記録から発言者や経過時間を把握でき，またその時参加者がおかれていた状況など討議のコンテキストを把握できる可能性が高くなる

本論文では，変換プロセスの実行に伴なって発生するコミュニケーションを，上記の立場から構造的に記録するためのモデル，グループウェアベースを提案することにより，これらの課題に対する一接近法を示す．

2.2 グループウェアベースのモデル化の方針

ソフトウェア開発におけるコミュニケーションを対象とした研究では，IBISモデルを導入した討論支援ツールであるgIBIS[6]が著名であり，前節で述べた，記録目的2に対する手段を主に提供している．また，ソフトウェア開発には限定されないが，グループワークで生じるコミュニケーションを構造化電子メール技術を利用して制御する研究も盛んである[7]．

構造化電子メール技術の場合，定型化した内容の伝達や定時的な連絡には効果的であると評価できる．しかし，要求定義など討議対象の輪郭が明確でない場合や，情報を創出するような討議には向かないであろう．それは，討議途中でゴールの再設定を余儀なくされたり，討議中に扱った内容や資料間に存在する因果関係，参加者の知識の構造が徐々に変遷を遂げていくからである．

また，gIBISはネットワーク状の議論の構造を分かりやすく掲示する仕組みを持ち，全体的な流れの中で一つの意見を追いかけることが可能である．一方，討議途中などすべての意見が網構造になってつながっているために，最終結論や複数のノードにまたがった思考の筋の表現が困難で，時間的な論点の推移を掴みにくい．

本論文では，記録目的1，2，3を対象としてモデル化を試みる．まず，記録目的1に対するモデル化の方針を以下のように定める†．

- 討議進捗状況把握の重視
  半構造化メッセージのような内容表現の制限は行なわず，また，gIBISよりは粗い粒度の会話単位（討議プロセス）を用いて，討議の進捗状態の正確な把握を重視する

†記録対象2，3については3.6節で述べる

- 討議構造の正確な反映
  いくつかの討議プロセスの間には，前後関係や相互関係が存在し，それらを正確に記録することが望ましい．時間の経過につれて変化しやすい討議構造を正確に反映するためには，討議の追加／除去，走行順序の変更，目標や処理手順の変更が容易でなければならない

- 討議内容の状態による切り分け
  討議内容が変化しつつあるまたはその可能性のある動的状態を，討議プロセス群としてモデル化し，変化がなくなった静的状態を議事録データ群として管理する

討議空間は，現在の論点について発生する討議プロセス間の継続／相互依存／詳細化関係等を保ちながら，柔軟な構造変化が可能である．また，議事録データ群は最終結論に関した討議プロセスの履歴によって静的な構造をなしている．また，両構造間には参照関係を設定することができ，議論の蒸し返しが可能となる（3.10 節）．

これらの方針により，以下の利点が望める．

- 討議空間では，現在検討中の議題に対する討議プロセスだけがあり，進捗状態と論点を捉えやすい

- 開発工程では論点に対する解は一時的なことが多く，討論の蒸し返し，目標の変更は常に発生するが，この制御が可能である

- 議事録データ群を筋道ごとに分離でき，さらに最終的な結論の表現が明示的である

3 グループウェアベースの構成要素

本節では2.2節のモデル化の方針に基づき，グループウェアベースの構成要素とその相互関係を定義する．

3.1 構成要素と相互関係構築にあたっての要請

まず，グループウェアベースの基本的機能を検討しつつ，その構成要素と相互関係に関する要請を明らかにする．

ある目的に向かって複数の人間が相互に影響しあいながら仕事を進める場合，目的に見合った結果になるよう，「案（意見）の創出と授受，それらの案の取捨選択の判断や決定」のための討議が行なわれ，以下のような特徴を持つ．‡

- ある討議はそれ以前になされた討議の結果に依存することがある（討議プロセスの継続）
- 互いに関係のあるいくつかの討議は並行的に行なわれることがある．また，ある討議の開始により，別の討議が必要でなくなったり，討議が両立しないなど特別な場合もある（討議プロセスの同時進行性と相互依存／排反）
- ある討議を達成するためには，いくつかの部分的な討議を先に行なう必要がある（討議プロセスの詳細化）
- 実行系が人間であるので，討議の過程は試行錯誤が多い（討議空間の動的再構成）
- 一旦得られた討議結果について，再吟味する必要もある（議題の蒸し返し）
- 討議の過程で問題が生じた場合，それまでの状況を素早くつかむ必要がある（討議の型の利用）

このような討議を記録するための基本単位として討議プロセスという概念を導入する．

3.2 討議プロセス

討議プロセスは，一つの話題に対応して発生する一連の会話をまとめる役割を果す．討議プロセスは，議題の登録に始まり，「討議」の状態を経て，その結果を議事録として格納し，消滅という状態推移をする（図3）．「討議」の状態は，3.6節で述べる討議の型により，より詳細な状態推移の履歴として記録される．

なお，討議期間を通じて発生する会話は，討議プロセスにリンクして記録する[8]．グループウェアツールは，会話を収集するための手段として位置付ける．

‡括弧の中に，3.2節以降で定義される概念・用語との対応を示す

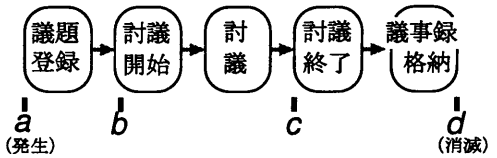


図 3: 討議プロセスの状態推移
State transition of a deliberation process.

3.3 メタ情報と管理ブック

討議プロセスの集合をその関係によって構造化したものを討議空間と呼び，メタ情報が統括している．メタ情報は討議プロセスの状況を把握するための種々の情報を持つが [5]，その一つに管理ブックがある．管理ブックの一つの頁には，プロセスＩＤや議題名など討議プロセスの発生に必要な情報，継続先プロセスＩＤや子プロセスＩＤなど討議空間の構成に必要な情報，そして終了判定者や完遂度など終了に関する情報が記載されている（表1）．なお，討議プロセスが持つのはプロセスＩＤ，会話のリンク情報と図3の状態推移の

1	プロセスＩＤ
2	議題名
3	討議目標
4	討議開始条件
5	再討議議題情報
6	進行役
7	終了判定者
8	参加者
9	予定討議時間
10	議題登録の時刻
11	討議開始の時刻

12	討議の履歴情報 (型, 時刻)
13	討議終了の時刻
14	完遂度
15	議事録格納の時刻
16	議事録格納情報
17	討議生データ格納情報
18	親プロセスＩＤ
19	子プロセスＩＤ
20	継続先プロセスＩＤ
21	相互依存プロセスＩＤ
22	排反関係プロセスＩＤ

表 1: １管理頁あたりの情報
Information contained in a page of a management book.

情報だけであり，他の情報はその都度管理ブックを参照する．

次節以降に図3の状態推移を基に，各状態で記録されてゆく管理頁の各項目の詳細について述べる．

3.4 討議プロセスの発生（「議題登録」）と「討議開始」

討議の必要性が生じた場合，討議プロセスの走行に最低限必要な情報（表1の項目 1,2,4,7,8）と，場合に応じて必要な情報（表1の項目 3,5,6,9）が，管理頁の項目として登録される．

プロセスＩＤは，討議プロセスの識別ＩＤで討議期間を通して一意であり，管理ブックの通しページ番号である（表1項目1）．議題名は，討議内容を端的に表す議題が付けられるが，再討議時（3.10節）は元議題と同一名である（表1項目2）．討議目標は，討議の最終目標が明確な場合に，記載しておく（表1項目3）．予定討議時間は，各討議プロセス毎に許容される討議の時間を割り当てておき，時間管理に利用する（表1項目9）．

メタ情報側ではこれら登録を受けて討議プロセスを発生させる．これにより，討議プロセスは「議題登録」の状態となる．この発生時刻が表1項目10として記録される（図3.aの時点）．

次に，他の討議プロセスからの結果の引き継ぎ等の討議開始条件（表1項目4）が指定されていた場合，これが満たされると「討議開始」状態となり，この時刻が討議開始時間（表1項目11）として記録される（図3.bの時点）．

討議の必要性が生じて議題が申請された時間と討議開始条件が満たされる時間とは，必ずしも一致するとは限らないため，議題登録の時刻と討議開始の時刻は，分けて記録する必要がある．例えば，議題登録の時刻から，長時間を過ぎても討議開始の時刻が記録されていない場合は，討議開始条件に関して何らかの異常が発生した可能性があることを示唆できる．また，記録という観点からは，議題登録時間はこの討議がいつから必要性が認識されたのか，討議開始時間は討議開始条件がどれくらいの時間内に満たされたのかという点で有用である．

3.5 「討議」と討議空間

討議プロセスは，会話の発生により「討議」の状態となる．以下の関係により構造化された討議プロセス群は,「討議」状態の間，互いに影響し合いながら討議空間を形成する（表1項目18,19,20,21,22）．

- 継続：ある討議プロセスと，そこでの結果を引き継ぐ討議プロセスとの間に張られる関係である．つまり，ある討議プロセスの結果は，継続討議プロセスの討議開始条件の一部を満たす．継続の形態は，討議が完結した場合（継続なし），結果が一つの討議に引き継がれる場合（１対１），いくつかの討議に引き継がれる場合（分割化），いくつかの結果がまとまり一つの討議に引き継がれる（統合化）がある

- 相互依存

  a. 付加: 互いに並列に走行している討議プロセスＡとＢにおいて，討議プロセスＡで討議された内容（会話の一部）が，討議プロセスＢの参照情報としてリンクされ，討議プロセスＢの会話の一部として付け加わる．ただし，付加されるだけで討議プロセスＢの討議された内容の書き換えはおこらない

  b. 変更: 上記の場合に加えて，討議プロセスＢの討議された内容が一部書きかえられる．討議プロセスＡで討議された内容の一部が，討議プロセスＢの討議の進め方に影響を与える場合である

- 排反: ある討議プロセスの討議開始により，別討議プロセスが不要になったり，相容れない場合で，どちらの場合であるかによって，討議プロセスの扱いは変わる．不要になった討議プロセスは，3.7 節で述べる完遂度が＜未開始＞と自動的に埋められ，消滅する．相容れない場合は，相手方討議プロセスの討議が終り，消滅するまで「討議」状態のままで停止する

- 詳細化: 一つの議題がさらに細分化されて，複数の討議プロセスに分割される．「討議」の部分に生じ，展開された各子討議プロセスも図 3 に示した形態を持つ．詳細化の起因としては，

  a. 計画的:討議開始時点で詳細化が判明．一議題では内容が大き過ぎる際，予め細分しておく

  b. 突発的:討議開始後に変則的に発生．以前の討議が不十分，また未決定の懸案事項の出現による情報の追加などが原因と考えられる

  詳細化された子討議プロセスは討議終了後，討議結果を親討議プロセスに報告すると同時に，議事録データ群に結果を格納し消滅する

3.6 「討議」の詳細状態の履歴と討議の型

ここで，討議という言葉のもつ意味合いについて若干の考察を追加する．複数の人間が話しあって結果を出すという行為は，いつも同じ結果を生むとは限らない．例えば，ソフトウェア開発においては，納期切迫によるあせりなどが，その結果に大きな影響を与える [9]．そこで，どのような要因に影響されつつ結論を得たかを理解することは，重要である．

討議の進行状態を記録する目安として，討議の型と呼ぶ概念を導入する．討議の型は，討議の構成要素である質問，回答，意見，承認などの会話を組織化して記録する役目を持ち，それらの間の時間的順序関係を保持しつつ，会話の目的に応じた討議の状態を分類することが目的である．討議プロセスの「討議」状態において一連の会話が発生するが，後述する討議の型を持つと判断されたとき，その時間とともに表 1項目 12 に蓄積される．

討議の型には様々な枠組が考えられる．例えば，会議という枠組で機能別に分類した＜伝達会議，創造会議，調整会議，決定会議＞という分類がある [10]．また，解決しようとしている目的の難易度を枠組とし，＜伝達，整合，決定，創造，交渉／説得＞という分類もある [11]．後者の分類においては，伝達は情報や知識の共有化，整合は認識の共有化，決定は異なった意見の調整，創造は無いものをつくり出す，交渉／説得はネゴシエーションという内容であると定義されている．本稿の視点からは，解決しようとしている目的の難易度という枠組が有用であると判断した．

ここで，伝達の目的である情報や知識の共有化と整合の目的である認識の共有化については，どの域までが情報の伝達でどこからが認識になるのかは，人間の意識に依存することであり，明瞭に区別することは難しく，記録の立場からは分けることは不自然だと考えられる．意識上の明確な分離が困難なものは，討議の意見の違いとしても反映され難く，

討議の状態を分類するという型の使用目的にもそぐわない．そこで，本稿では伝達整合を一つの型にまとめて記録する．

プロジェクトは殆んどの場合，複数人で遂行される．この時，作業結果の引き渡し（例えばプログラム仕様書をプログラマに渡す），作業内容の説明，技術的知識の伝受のための会話は最低限必要である．このような会話を伝達整合と呼ぶ討議の型で記録する（図 4）．図 4において，ある情報が送信されると，内容検討中の状態に推移する．この状態で，受け

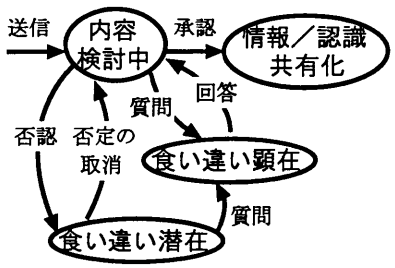


図 4: 伝達整合の記録スキーマ

The recording scheme of transmission and adjustment.

取った側に何かしら釈然としないものがあれば，否認という意思表示によって，食い違い潜在の状態に移行する．ここで，考えが変われば，また内容検討中に戻る．また，どこが釈然としなかったかが分かれば，質問と回答の繰り返しにより，納得がいくまで意見交換が行なわれる．しかし，必ずしも情報／認識の共有化という状態になって，討議が終るとは限らない．すなわち，伝達整合および後述する決定，創造の記録スキーマにおいて，全ての状態は最終状態となる可能性を持つ．例えば，食い違い潜在の状態が最終状態と記録されている場合，後に十分な意見交換が行なわれず，そのことが問題を引き起こしたかも知れないということを示唆する．

プロジェクトが複数人のメンバーによって遂行される以上，伝達整合の型は，頻繁に発生する基本的な型である．ところで，実行の担い手が人間である以上，一旦は承認したが新たな面から疑問が発生するといったことは日常的に起こる．これについては後述する議題の蒸し返しによって対処する．

伝達整合における食い違い顕在の状態が，解消されなかった場合，これを調整し一つの見解にまとめる討議が必要である．この討議は，決定の型として記録される（図 5）．互いに意見を交換しつつ，多くの場合トレードオフにより，複数の解候補の中から，一つが選択決定される．その手段としては，多数決，決定権を持つ人物の独断など様々であり，その組織における討議の運営法や参加者の性格などのコンテキストに依存する．

与えられた情報の伝達や加工に留まらず，お互いの持つ知識を叩き台に意見を交換し，新たな情報を生み出す会話がなければ，付加価値の高い成果物は生まれて来ない．このような会話は，創造という型に記録される（図 6）．会話の開始時点では，欲しい情報の輪郭が漠然としており，討議内容の照準もあわせにくい．「試行錯誤状態」において，一人から

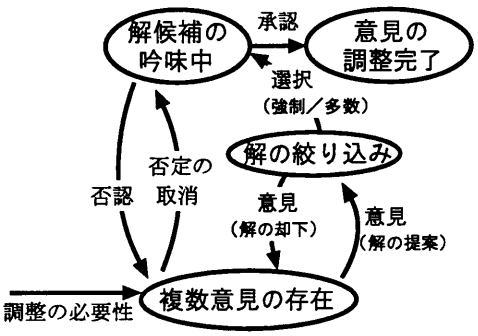


図 5: 決定の記録スキーマ

The recording scheme of decision.

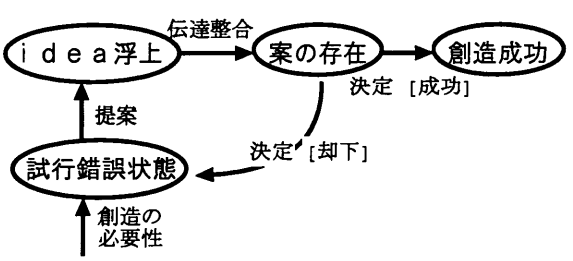


図 6: 創造の記録スキーマ

The recording scheme of creation.

ひらめきが生じる．それを他のメンバーに伝達整合することでアイデアが共有化され，これを採択するか否かの決定が行なわれる．§

「案の存在」状態より先の推移は，図5における決定の状態推移が「意見の調整完了」で終了したとき，図6におけるガード「成功」が真になり，それ以外の状態で中断されたときガード「却下」が真になる．以上の繰り返しにより，新たな情報を生み出していくのが創造の型である．

ところで，ソフトウェア開発においては，懸案事項の決定に至るまでの交渉や説得などの会話も発生する．これについては決定という会話目的にネゴシエーション等，人間の心的側面に係わる要素が多分に上乗せされたものであると考え，今回のモデル化の対象外とする．

3.7 「討議終了」と完遂度

討議プロセスは終了する際，討議の完遂度（表1項目14）が終了判定者（表1項目7）によって決定される．完遂度の決定は人間の主観に依存するため，絶対的尺度を定義することは難しい．現時点では＜終了，延期，打ち切り，取消，未開始＞の簡潔な5段階の定義をしている．＜延期＞とは再討議を必要，＜打ち切り＞とは再討議を要さない場合である．完遂度は，例えば，後日再討議が必要かどうかの判断材料となる．完遂度が管理頁に書き込まれた時刻が表1項目13として記録される（図3.cの時点）．

3.8 「議事録格納」と消滅

終了判定者が議事録を認可し，議事録格納情報（表1項目16）が付与されると，討議プロセスは討議結果（議事録格納情報と完遂度）を継続もしくは親プロセスへ報告し，議題名，プロセスＩＤ，討議期間，同一議題通し番号，討議結果を議事録データ群に格納する．なお，完遂度が＜終了＞以外の場合は，議事録にその理由（例えば，＜打ち切り＞の理由）が記入される．討議プロセスは格納が正常終了すると，メタ情報に完了通知をし消滅する．この完了通知の時刻が表1項目15として記録される（図3.dの時点）．メタ情報側では，該当の管理頁を取り外し，メタ情報内の別の場所で保管する．討議途中で生じたデータは，一固まりに加工し，タグを付加して保管する（表1項目17）．

議事録データ群では，最初に討議の終った子プロセスの＜議題名，プロセスＩＤ，討議期間，同一議題の通し番号，処理結果＞の一組が葉として作成され，さらに自分の親となる討議プロセスの仮ノードも作って接続する．親プロセスの討議も終了すると，仮ノードは親プロセスのデータに置き換えられる．

§ここで、図6における伝達整合および決定の事象は「入れ子の状態図」をあらわしており，また決定のあとの[ ]は「ガード付き遷移」をあらわしている[12]

3.9 討議空間の動的再構成

上流工程では特に，試行錯誤することが多く，開発目標が少しづつ変わる場合や全く変わってしまう場合などがある．この場合，付随する討議プロセスも影響を受け，それまでの討議プロセスが急に不要になったり，予めスケージュルしておいた討議順序が変わったりする．こういった場合に対応できるように，討議プロセス群の動的再構成が必要となる．すなわち，

- 討議プロセスの予定外の組み込み
- 後続の討議プロセスの追加／付替／除去
- 不要になった討議プロセスの抹消

等が可能でなければならない．この動的再構成の情報も，メタ情報に記憶される（図 7）．その手順は，

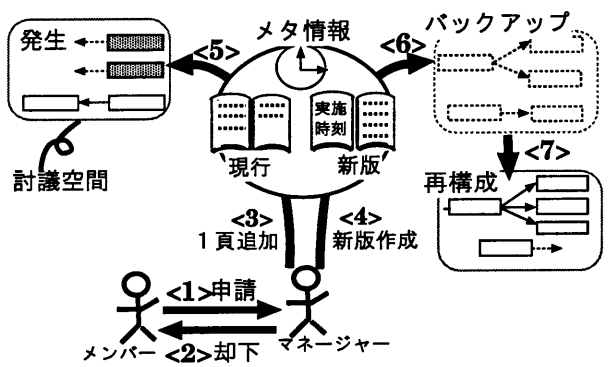


図 7: 動的再構成の仕組み
The flow of restructuring.

1. 討議順の変更や新議題が必要になると，マネージャーにその旨を申請する（図 7.<1>）
2. 申請内容について，マネージャーはその可否を決定し（図 7.<2>），承認すると，
   - 新議題の場合，新しく管理頁に必要項目を登録する（図 7.<3>）
   - 変更／削除の場合，管理ブックの新版を作成する．また，この新版の施行時刻（動的再構成の実施時刻）も付与する（図 7.<4>）
3. メタ情報側では，
   - 新管理頁が作成されると，該当の討議プロセスを発生させる（図 7.<5>）
   - 変更／削除の場合，動的再構成の実施時刻になると，

(a) 討議プロセス群をストップさせ

(b) 管理ブック旧版を含む討議空間情報のバックアップをとり（図7.<6>）

(c) 管理ブックを旧版から新版に移行し

(d) 討議空間を再スタートする（図7.<7>）

3.10 議論の蒸し返し

また，開発過程では「ゆり戻し」が発生することがある.「なぜこうなったのか，どこに問題があったのか」という問題箇所を発見するために，前の論点に立ち帰って再討議する必要がある.

本稿では，図3の5番目の議事録格納により討議プロセスの寿命は尽き，議事録データとしての寿命がスタートすると考える．つまり，蒸し返しとは，同一議題に対する新討議プロセスのスタートであり，必要があれば，元の議題に対する議事録結果を参照することで再討議を進めてゆく．同一議題に関する元討議プロセスと新討議プロセスの参照関係も，メタ情報内の管理ブックにより定義される（表1項目5）.

議題の蒸し返しの管理法について簡単な例を用いて説明する．まず，図8の状態に至るまでの経過として，Pid 1は，Pid 1.1と1.2の子プロセスを持ち，さらにPid 1.2は3つのプロセスに詳細化されていたとする．この時，6枚の管理頁が作成されており，親討議プロセスが1つ，子プロセスが2つ，孫プロセスが3つ走っている．その後，Pid 1.2に関する討議が全て終り，議事録データ群に格納される．終了した管理頁4枚分は，現行管理ブックから外され，メタ情報の別の場所で保管される.

後日，Pid 1.2の議題について蒸し返しが生じたとする（図8）．この再討議の手順とは，

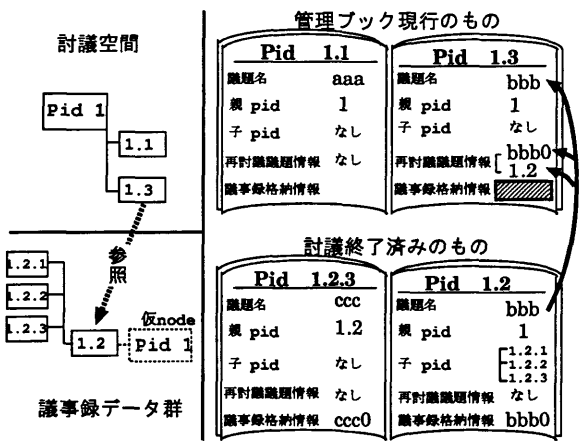


図 8: 管理頁の例示：再討議の場合

An example of a management book for resteaming.

1. 再討議がマネージャーに申請される

2. 申請内容について，終了した管理頁の中から元議題に関する情報を検索し，承認の可否を決定する（マネージャー側）

3. 承認すると，その議題に関して新しい管理頁（図8：Pid 1.3）を作成し，さらに同一議題の通し番号（図8：bbb0）と元議題のプロセスＩＤ（図8：1.2）も書き加える

4. 「討議」状態になった討議プロセス（図8：Pid 1.3）は，議事録データ群中の元プロセス（図8：Pid 1.2）の議事録を参照しながら討議を進めて行く

5. 討議終了後は，同一議題の通し番号がカウントアップ（この場合，bbb1の値）され，管理頁（図8斜線部）の値が更新される

4 変換プロセスとグループウェアベースの関係

3節では，討議の構造を正確に反映して記録するための手段を論じてきた．本節では，変換プロセスとグループウェアベースの結合法を考察する．また，ここまで導入した諸概念のISPW6例題への適用例を示す．

4.1 変換プロセスとグループウェアベースの結合法

変換プロセスとグループウェアベースを結合する際，討議プロセスの適用箇所を，まず，明確にしておく必要がある．

一般に，どのようなプロセス記述においても，基本計算要素とその間の接続関係が定義されている．コミュニケーションの発生箇所は，この基本計算要素間の接続点，もしくは基本計算要素内であるために，どのプロセス記述を用いても，コミュニケーションの管理には影響はないと判断する．

変換プロセスを構成する基本的な活動（基本計算要素）を，本稿では変換ステップと呼ぶ．変換ステップと討議プロセスの結合法は，以下に示す2つの結合法が考えられる．表2にその分類と詳細を示す．

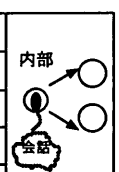
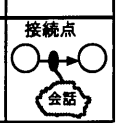
a	討議プロセスのみ（変換ステップ置換）	
b	変換ステップ＋討議プロセス（目的が明瞭）	
c	変換ステップ＋（目的が不明瞭な会話）	
d	上記2,3の混合型	
e	変換ステップのみ（独り言）	
f	変換ステップ間の接点としての討議プロセス	

表 2: 討議プロセスの発生様式

The occurring patterns of deliberation processes.

(1) 変換ステップ内部

変換ステップそのものが討議プロセスで置換される場合（表2.a）と変換ステップに付随して討議プロセスが並走する場合（表2.b,c,d,e）がある．例えば，表2のaは，ユーザとシステムエンジニアの折衝や，要求仕様作成者とシステム設計者の仕様化作業などが挙げられる．bは例えば，中間成果物のレビューなど会話の目的がはっきりとした討議プロセスが計画的に発生し，変換ステップに付随する場合である[13]．cは直接には成果物に影響を与えない会話，例えばおしゃべりなどが変換ステップ中に細切れに発生している場合である．eはプログラミングなど個人作業の場合である．cとeに関しては，プロトコル解析のような実験的な環境では，会話の記録を取ることは可能であるが，現実世界では困難である．しかし，こういう会話には本音が含まれることが多い．個人の思索の内容は，例えば

フォーマルレビューにおいて，他者に同意や確認を求めるという形で表現されることもあり，完全ではないにしても間接的にその内容を記録できると判断する．

(2) 変換ステップの接続点

表 2.f は，変換ステップで作成された成果物の受渡しや作業伝達など，変換ステップ間の接点に討議プロセスが必要な場合である．従来のソフトウェアプロセス（本稿では変換プロセス）記述では，例えば，単なるデータフロー（矢印）として表現されていた．本モデルの特徴は，変換ステップ間の矢印を討議プロセスで置換することにある．例えば，ある変換ステップ「プログラム仕様の作成」でプログラム仕様書が作成され，次の変換ステップ「プログラミング」にそれが引き渡される場合を考える．従来のソフトウェアプロセスの記述では，変換ステップ「プログラム仕様の作成」の出力が，次の変換ステップ「プログラミング」の入力となるという点にのみ注目して記述する．しかし，現実には引き渡しの作業は，まず引渡し内容の共有化という伝達整合の会話が必要であり，食い違いが解消されなければ，決定目的の会話が引き続いて実施される．一旦，引き渡しがおわり，変換ステップ「プログラミング」に制御が推移したあとでも，新たな不明点の発生により議論の蒸し返しが発生する．

まとめると，変換プロセスは作業進行のガイドラインを与える．その構成要素である変換ステップは，一つ一つの作業の入力や出力，他の変換ステップとの関係を定義する．変換ステップと変換ステップの間に埋め込まれる討議プロセスは前ステップでの作業結果を次ステップに引き渡す作業を表現する．これらは実際のソフトウェア開発においては切れ目のない連続的な活動である．

4.2 ISPW6 の例題への適用例

本節では，要求仕様の変更を取り扱った ISPW6 の例題を用いて，前節までに導入した諸概念の適用例を示す．

ISPW6 の例題（核問題）は，ソフトウェアの要求変更を実施していく上で必要となる入出力，責任，制限等を明示し，８つの作業単位を定義している [14]．この作業単位は本稿で言う変換ステップにあたる．例題では，インプットファイル，あるいは出力される成果物など，作業単位に直接関係する「物」の流れは文章により明示されている．その変換過程をデータフローダイアグラムを用いて表現した（図 9）．直接「物」に変換されない情報，つまり会話情報についての特記は例題にはない．そこで，この例題に発生が想定される会話を補うことにより，グループウェアベースの役割を考察する．

図 9中の Review Design は，修正されたデザインを Modify Design より手渡しで受け取り，そして，修正されたデザインの正式なレビューを行なって，フィードバックを Modify Design へ手渡すという機能が定義されている．

この変換ステップの開始条件の一つは，Modify Design から Review Design へ修正されたデザインが伝達整合されることであり，表 2.f にあたる箇所に会話が生じる．ここで，

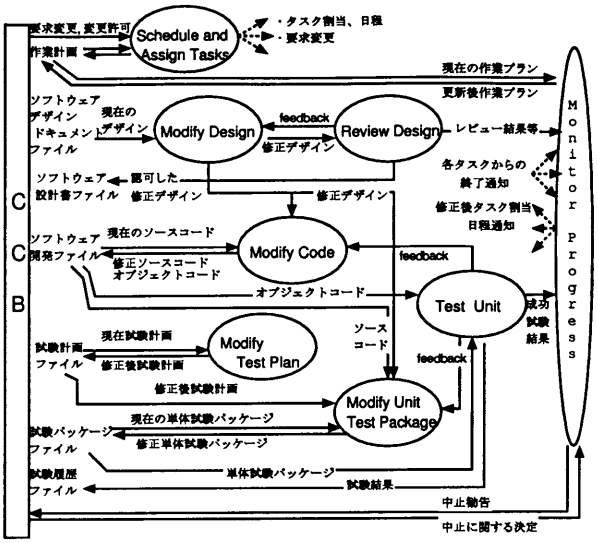


図 9: 例題内容のＤＦＤによる表現
The DFD representation of the ISPW6 exercise.

伝達整合された内容に関して，疑問という食い違いが生じたとする．この時点では,「内容検討中」の状態から，質問によって「食い違い顕在」の状態に推移したことが，伝達整合の型を用いて履歴として記録されている（図 10の有色の部分）．討議プロセスが持つ各会話とこの状態の履歴の間に，同期を取ってリンクすることにより，どの状況でどういう会話が発生したかを突き止めることが可能である．また，意見の食い違いがどこで起きており，どのように吸収されたのかを分析することができる．

次に，修正デザインに関する伝達整合が完了したとすると，Review Design の変換ステップでは，修正されたデザインに関して，無条件合格／小変更要請／大変更要請の３段階の判定を行なう．判定はデザイン修正を行なった設計エンジニア，品質保証エンジニア，他のソフトウェアエンジニアを含むレビューチームによって行なわれる．仮に再修正に関して保守的立場の設計エンジニアと品質追求を役目とする品質エンジニアでは，必ずしも同一判定を選択するとは限らない．そこで，複数意見から解を絞り込む討議プロセスが表 2.b の形で，変換ステップ内に発生する．ここで,「複数意見の存在」から，解の提案がなされ「解の絞り込み」の状態に推移したことが，決定の型によって記録されている（図 11 の有色の部分）．

4.3 グループウェアベースの役割

前節までに，討議空間による会話の構造化，議事録データ群による討議結果の管理，討議の型による記録の枠組の提供，グループウェアベースと変換プロセスの結合法などの諸概念を導入した．2.1 節で掲げた記録目的に対して，これらの諸概念がどのような役割を果すのかを以下の点について検討する．

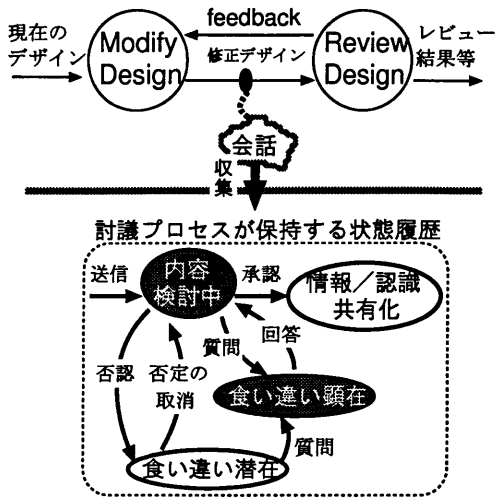


図 10: 伝達整合の型の結合例

Replacing a dataflow by the recording scheme of transmission and adjustment.

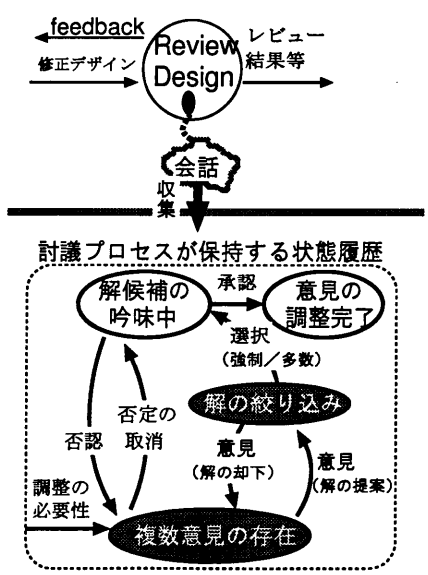


図 11: 決定の型の結合例

Connecting the recording scheme of decision to a transformation step.

記録目的１ に関して，マネージャーに，プロジェクト管理に必要な情報をどの程度提供できるか

記録目的２ に関して，成果物の形成に影響を与えた情報をどの程度特定できるか

記録目的３ に関して，業務遂行に役立つ知識情報として，開発チームのメンバーにどのように役立つのか

- 循環している箇所や中断箇所の検知によるプロジェクト管理の支援
  ある作業が完結するとその結果が次の作業に引き継がれ，また受け手からの質疑が生じる．例えば，図 10の Modify Design から Review Design へ修正されたデザインが手渡される場合，ステップとステップの接点（表 2.f）に埋め込まれて記録された討議プロセスの内容は，引き継ぎ内容の説明，確認あるいは不明点の質疑などの状況の再現のために有効であり，伝達整合の型によって記録された会話内容から，その状況を読みとることができる．もし，不明点について何度も質疑が発生し，引き継ぎ作業にトラブルが発生しているとすると，同じ箇所で討議が循環しているという情報を得ることができる．これをどう判断するかは，利用者次第である．さらに，そこに付随する会話内容を解析することにより，原因特定がより容易になる．また，討議プロセスの種々の時間属性（表 1）を利用して，作業が円滑に進んでいないことを検出することも可能である．例えば，ある討議プロセスの経過時間が予定討議時間を数倍も超過している，議題の登録後，討議の開始が大幅に遅れているなどの例が挙げられる．

- 成果物の品質を制御する情報の記録と利用
  変換プロセスは成果物を得るための道筋である．しかし，生み出される成果物は，均一な品質を持たない．その理由は，実行主体が人間であるため，変換プロセスに沿って発生するコミュニケーションを通じて，以下のような制御情報が成果物の出来具合に影響を与えるからである．

  1. 決定のアルゴリズム（ex. 多数決，独断）
  2. 決定打となった意見
  3. 解の絞り込みが何度なされたか
  4. 解に結び付かなかった意見

  上記の内，特に 1 と 2 は，成果物（判定結果）を制御しうる情報となり，それらを討議の型の枠組をもとに討議プロセスに蓄積できる．例えば，図 11の Review Design の変換ステップへの適用例においては，そこで，複数意見から解を絞り込む討議過程が討議プロセス内に記録されている．これらの情報は，プロジェクト管理に利用できる．例えば，成果物が期待どおりの品質を持たず，原因を調査する場合，成果物に影響を与えた情報として，重点的にチェックする対象となる．また，バックトラックし

てやり直す場合，同じ過ちの防止あるいは，捨てられた意見の再浮上などやり直し作業の効率化も望めるかも知れない．

- 第３者に有益な設計の根拠の記録と利用
  例えば，図9のModify Designの変換ステップは，要求の変更にともない影響を受けるデザインを修正する．デザインの修正とは，人間の思考を基に実行されるため，他人の意見に耳を傾けながら，試行錯誤しつつ解を見い出す創造という型の会話が有用である．どういう提案がなされ，その案がどいう状況で却下され，その繰り返しの末，アイデアとして採択されたかという設計根拠の記録は，プロジェクトの初心者側から見れば，見様見真似で覚える知識の一種である．この際，抽象された討議のコンテキストを知りたければ，議事録データ群を参照し，さらに詳細が必要な場合には，討議の履歴情報として，型毎に分類され時系列で記録されている会話を参照すればよい．このようなデータは，第３者特に，初心者にとっては読みやすく，自然な理解を得安いものと推察する．

グループウェアベースの役割は，以上のような状況下で，判断の材料や行動の指針になる情報を提供することにある．

5 議論と今後の課題

5.1 本論文の成果

本論文では，ソフトウェアプロセス実行における系統的コミュニケーションの支援という立場から，図12に示すように，変換プロセス記述の詳細レベル（粒度）に合わせたコミュニケーション記録の単位を定義し，全体状況の把握から設計根拠の理解まで目的に応じた粒度の情報提供が可能であるモデルを示した．図12は上層の要素は下層の要素によって形成されるという構造を示している．変換プロセスはいくつかの変換ステップをその順序関係で結合したものであり，ソフトウェア開発のガイドラインを与える．変換ステップは作業の入出力などを仕様化したものである．変換ステップはいくつかの基本活動（実行系の作業単位）から成る．変換プロセスの実行状況はプロジェクトの進捗状況を表現する．変換ステップは，それを構成する基本活動に依存する進行状態をもち，各基本活動はその遂行状態をもつ．変換プロセスと，討議空間および議事録データ群は一対一に対応する．変換ステップと討議プロセス，基本活動と討議の型の間には一般に多対多の対応関係がある．

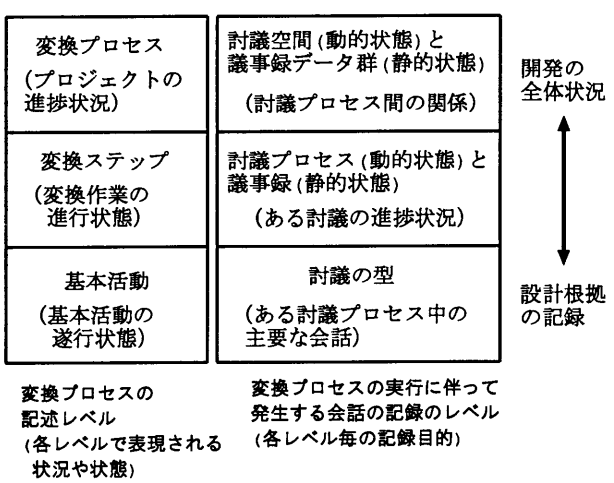


図 12: 変換プロセスとグループウェアベースの対応

The correspondence of a transformation process to a groupware base.

本稿で形式化したグループウェアベースは，

(1) 討議空間 において，変換プロセス実行時に発生している様々な因果関係を持つ討議の進捗や論点を捉えることにより，変換プロセスの実行状況の把握を助ける．また，コミュニケーションの特性である討論の蒸し返しや，討議構造の変更に柔軟な対応が可能である．議事結果を筋道ごとに分離でき，最終的な結論の表現が明示的である

(2) 討議プロセス において，ある特定の話題に対する会話の流れ（時系列な変化）を記録しておくことにより，一つの話題の進捗状態を捉えうる

(3) 討議の型 において，ある特定の話題に対する代替案の取捨選択の意図や確実性，ある結論の前提条件などを捉えうる．また，その時，参加者がおかれていた状況など，討議のコンテキストの状態を抽象して把握できる

などの特徴をもつ．

上記の情報を基に，ソフトウェア開発に携わる人々に，その目的，役割，状況に応じた粒度の情報を以下のように提供できるものと期待される．

（プロジェクト管理に必要な情報の提供）時間的な進捗状況の把握が可能である．例えば，どの変換ステップで，どのような話題を論じている時に，討議の循環や予定時間の超過が発生したかという情報を提供できる．

（成果物の形成に影響を与えた情報の特定）検証や誤り回復の支援に必要な情報を提供できる．例えば，決定の討議の型をもつ会話部分を抽出することにより，成果物に影響を与えた決定についての情報を得ることができる．

（業務遂行に役立つ知識情報の提供）第３者に有益な設計根拠を記録できる．例えば，試行錯誤しつつ解を見い出す創造の型の記録は，見様見真似で覚える知識の一種を提供する．

5.2 ＣＳＣＳＤフレームワーク

我々は現在,「チームが協調状態を維持しながら複数の人間がアイデアや中間生成物をネットワークを介して共有し，討論や変換活動により，それらを変化させてゆくこと」の支援を目標とした，ＣＳＣＳＤフレームワーク（図 13）と，そのプロトタイプ「自在」[8]を開発中である．

ＣＳＣＳＤフレームワークとは，実行系，変換プロセス，オブジェクトベース，グループウェアベースを構成要素とするソフトウェア開発環境の枠組である．図 13において，変換プロセスやオブジェクトベースに関してはさまざまな研究成果がある [15]．

例えば，ソフトウェアプロセス，オブジェクトベース，その間の結合法については，

- ソフトウェアオブジェクト間の依存関係（is-derived-from）を「制約」として，作成の手順を関係（relation）によって表現し，オブジェクトの変更にともなうオブジェクトベースの一貫性をトリガー（trigger）によって保持するアプローチ [16]．

- 実体関連モデルで表現されたソフトウェアオブジェクトの状態をルールの条件部で表現し，それが満足された時，実行部を起動する MSL/Marvel によるアプローチ [17]

などがある.

また，ソフトウェアプロセスと実行系の関係については，オブジェクトベース中のある中間生成物を別の中間生成物に変換するための，開発方法論に依存するノウハウを，事例推論を用いて収集し，モデル推論により一般化した上で作業者のナビゲーションを支援する方式についても基礎的な成果が得られている [18].

本論文では，図 13の内，グループウェアベースの定義と，変換プロセスとグループウェアベース（討議プロセス）の結合法について論じた．次節以降で，ＣＳＣＳＤフレームワー

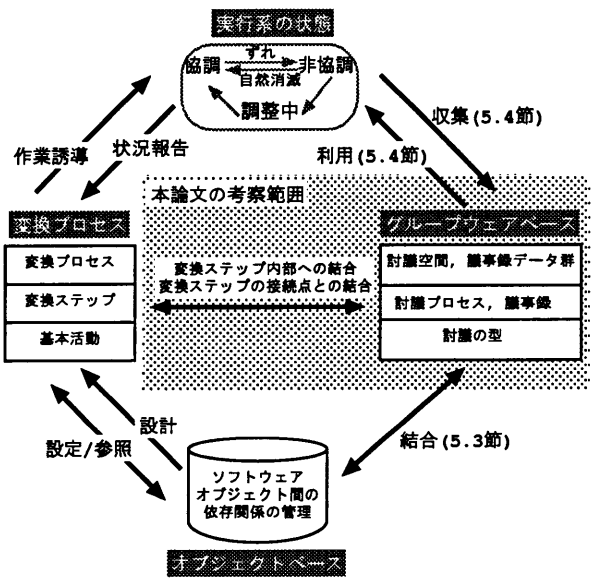


図 13: ＣＳＣＳＤフレームワーク
CSCSD framework.

クにおける以下の課題を検討する.

- 成果物とグループウェアベースの結合法
- 変換プロセスの実行系とグループウェアベースの接点

5.3 成果物とグループウェアベースの結合

成果物とコミュニケーションの関係を論じた研究としては，ソフトウェア開発を通じて生成される中間生成物やプロダクト間を討議によって生み出される設計根拠（design rationale）で接続することを提案した Colin Potts による成果がある [19]．しかし，討議経過の表現の粒度が細かく，作業の流れにそって大筋を把握することが困難であるという問題があった．本稿で定義したグループウェアベースは，プロセスの実行状況の把握から設計根拠に至る様々な情報を，変換プロセスの粒度に合わせて記録するものである．今後の課題の一つは，

グループウェアベースと成果物の関係を，Colin Potts による成果を踏まえ検討することにある．

この時，ソフトウェア開発で問題になる共有情報の変更制御についても，以下のような視点から合せて検討する予定である．いくつかの変換ステップ（例えばプログラミング）は同時並行的に実行される．ある変換ステップから，共有情報に対する書き込み要求が発生したとき，その変更に関連する人々の間に話し合いが必要となる．これに伴う討議プロセスの発生は，変換プロセスに付随するものではなく，共有情報を管理しているオブジェクトベースに基づくべきものである．

5.4 実行系とグループウェアベースの接点

実行系の状態（人間集団の状態）は，協調と非協調という状態に区分けできる．協調状態とは，プロジェクトがゴールを目指して活動中に，メンバー個々の活動を合わせ見ると矛盾のない状態と考える．

ところが，メンバーの持つ知識や技量の格差，あるいは思惑や立場・役割など人的要因と，通信手段や共有データベースの破壊，停電などプロジェクトの依存する環境からの物的要因により，各活動間に矛盾が生じ，非協調状態になってしまう．この要因の中には，自然消滅せずにグループワークの維持を極端に阻み，作業を滞らせてしまうものもある．放置しておいては，プロジェクトは最終ゴールにたどり着かない．そこで，矛盾を生じた要因を取り除く調整が必要となる．調整を施す際，グループウェアベースより得られる情報により，非協調状態の原因の特定が容易になる．例えば,「討議開始」状態から，長時間「討議」状態に推移しない場合，会話が長時間発生していないという状態を検知しうる．これにより，マネージャ側では会話のデータが未着もしくは，参加者に不都合が起きたなどの状況をいちはやく掴める．

上記のように，実行系の状態は些細なことで，協調状態の均衡を破ってしまう可能性がある．このような，人間系の一般的な振舞いの中で，その活動を阻害することなく自然な形で，また，あまりコストをかけずに，会話内容を収集し利用する手段が重要である．

収集 最近のグループウェアの発展と普及（初歩的なものとしては電子メールがある）を考慮にいれれば，会話内容を参加者に余分なコスト（労力）を強いることなく収集することは可能であろう．まだ基礎的な段階ではあるが，音声を対象とした会議の発話履歴をリアルタイムで収集し，記録した上で自動解析し，会議の流れや参加者の特徴を抽出するモデルとツールの開発が佐伯等により進められており [20]，本研究との関係の検討が期待される．

提供 基本的にはグループウェアベースに記録される討議の構造やその内容に関する様々な検索機能を提供することが必要である．例えば，情報フィルタリング技術を基に，ソフトウェアエージェントによりモジュール化された機能を提供するアプローチを今後検討する予定である [21]．

6 まとめ

本研究の基礎的な成果は分散開発時代における新しい開発標準や文書化技術の開発に有用であると思われる．すなわち，分散開発時代には，ソフトウェア開発におけるコミュニケーションの比重が増加し，開発手順や文書化もその技術が変化するものと思われる．前者については，本稿の前書きで述べたように，コミュニケーションの記録技術と利用技術を発展させなければならない．また，後者については，従来のソフトウェアプロダクトの文書化法に加えて，それと融合する形で、コミュニケーションの文書化技術を発展させる必要がある．

今後の課題は，例えば討議の型を充実させることにより，本稿で提案したモデルをさらに充実させることである．さらに，5節で検討した内容に対して具体的なモデルを定義し，ＣＳＣＳＤフレームワークを発展させることである．

参考文献

[1] W.S.Humphrey 著, 藤野監訳: ソフトウェアプロセス成熟度の改善, 日科技連 (1991).

[2] 鈴木: 代表的なプロセス記述言語の特徴-共通例題による比較-, 情報処理, Vol. 36, No. 5, pp. 392–398 (1995).

[3] 桑名: ソフトウェア履歴利用の研究動向, 電子情報通信学会誌, Vol. 77, No. 5, pp. 531–538 (1994).

[4] 片山: ソフトウェアプロセスとその研究課題 , 日本ソフトウェア科学会第 11 回大会論文集, pp. 433–436 (1994).

[5] 門脇, 落水: 非同期分散型会議の事象駆動型討議プロセスによるモデル化と調整支援への応用, 情報処理学会 ソフトウェア工学研究会資料 96-25, pp. 193–200 (1994).

[6] J.Conklin and M.L.Begeman: gIBIS:A Hypertext Tool for Exploratory Policy Discussion, CSCW '88, pp. 140–152 (1988).

[7] F.Flores, M.Graves and B.Hartfield and T.Winograd : Computer System and the Design of Organization Interaction, ACM Transaction on Office Information Systems, pp. 153–172 (1988).

[8] 落水, 門脇, 藤枝, 堀: ソフトウェア分散開発支援環境「自在」のアーキテクチャ設計, 電子情報通信学会ソフトウェアサイエンス研究会, pp. 1–8 (1994).

[9] T.Furuyama, Y.Arai, and K.Iio: Fault Generation Model and Mental Stress Effect Analysis, J. Systems and Software 26, pp. 31–42 (1994).

[10] 高橋: 会議の進め方, 日本経済新聞社 (1989).

[11] 田中, 荒木, 増田: 対人的コミュニケーションにおける電子的メディアの特性と効果, 情報処理学会 グループウェア研究会資料 4-8, pp. 53–60 (1993).

[12] J. ランボー他著, 羽生田監訳: オブジェクト指向方法論 OMT 第 5 章 動的モデル, トッパン (1992).

[13] 落水: 統合環境Ｖｅｌａにおけるデザインレビュー支援, 情報処理学会 ソフトウェア工学研究会資料 77-5, pp. 25–30 (1991).

[14] M.I.Kellner, P.H.Feiler, A.Finkelstein, T.Katayama, L.J.Osterweil, M.H.Penedo and H.Dieter Rombach : Software Process Modeling Example Problem, Proceedings of the 6th International SOFTWARE PROCESS WORKSHOP, pp. 19–29 (1990).

[15] 落水: ソフトウェアプロセスに関する研究の概要, 情報処理, Vol. 36, No. 5, pp. 379–391(1995).

[16] S.M. Sutton, J.D. Heimbigner, L.J. Osterweil: Language Constructs for Managing Change in Process-Centered Environments, Sigsoft Software Engineering Notes, Vol. 15, No. 6, pp. 206–217 (1990).

[17] G.E. Kaiser, P.H. Feiler: An Architecture for Intelligent Assistance in Software Development, Proc. of the 9th ICSE, pp. 180–188(1987).

[18] 山口, 落水: 事例ベース推論とモデル推論の相互作用に基づくソフトウェアプロセスモデル獲得支援環境, 電子情報通信学会 知能ソフトウェア工学の動向と展望シンポジウム, pp. 75–81(1992).

[19] C. Potts, G. Bruns: Recording the Reasons for Design Decisions, Proc. of the 10th ICSE, pp. 418–427 (1988).

[20] 松村, 下田, 佐伯: 要求獲得のための会議の発話履歴の構造化手法, 情報処理学会ソフトウェア工学研究会, サマーワークショップ・イン・立山, pp. 65–72 (1995).

[21] J.H. Connolly, E.A. Edomonds 編: CSCW and Artificial Intelligence, Spring-Verlag, Computer Supported Cooperative Work (1993).