ソフトウェア工学 96−25
（1994．1．21）

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

門脇 千恵　　落水 浩一郎

北陸先端科学技術大学院大学　情報科学研究科

本稿では、ＣＳＣＳＤ(Computer Supported Cooperative Software Development)の考え方に基づくソフトウェア開発環境を構築するにあたり、必要となる調整支援のための概念モデルとアーキテクチャを提案する。

我々のモデルの特徴は、プロジェクトメンバ間で取り交わされる非同期・分散型のコミュニケーションの状態を討議空間によってモデル化し、討議空間における「活動ベクトルのずれ」を検出し、調整することにより、メンバー間の協調活動の維持を可能にする点にある。

Modeling an Asynchronous Distributed Type Communication by Event-Driven Deliberation-Processes and Its Application to Coordination Support

Chie KADOWAKI　　Koichiro OCHIMIZU
Japan Advanced Institute of Science and Technology,Hokuriku

This paper proposes a conceptual model and an architecture for computer supported cooperative software development environments. The key idea of our model is coordination supports to maintain cooperation among project members by detecting distortions in a Deliberation Space which consists of structured deliberation processes among project members. Our model will be useful for an asynchronous and distributed communication process which is a typical feature for coming distributed software developments.

1．はじめに

ソフトウェア開発は、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から構成されたプロジェクトチームの協調作業である。

このようなグループ活動において、「割り当てられた仕事（サブゴール）を各自がスムーズに達成し、ひいてはグループ共通の最終ゴールが満足され、プロジェクトを成功に導く」ためには、

1. 協調活動で発生する「ずれ」の要因検出と対応
2. 生成される情報群の正確な伝達と作業状況の把握法
3. 各自の作業分担に関わる情報の適切な設定法

等が重要となる [1]。本報告はこのうち１と２の一部を考察し、ソフトウェア開発の種々の場面で生じるコミュニケションを統一的にあつかう概念モデルを提案する。またこのモデルを基とするソフトウェア開発環境の構想を示す。

本稿で対象としているコミュニケーションの形態は、分散・非同期型の会議（ex. メイリングリストによる）であり、会議の進行役かつプロジェクトのまとめ役でもあるコーディネータが存在する。そして、プロジェクトの運営的な面を支える会議、あるいは、ソフトウェア開発の上流工程に存在する収束型の討論を対象とし、<分析、分析、…、行き詰まり、討議内容の再構成／討議目標の再設定>というような討論の円滑な支援を目指す。

2．ＣＳＣＳＤを目標としたソフトウェア開発環境に向けて

すでに筆者等は、

1. Control…集団活動の制御支援
2. Navigation 支援…質のよい中間生成物を作成するための手順の支援
3. Communication 支援…ソフトウェアプロジェクト構成員間の情報交換支援
4. Coordination 支援…開発対象物に対する目標と基準の設定、および版管理法

の４つの機能実現を主眼とするソフトウェア開発環境Ｖｅｌａの構想を発表し [2] ,Navigation 支援については [3][4][5], Communication 支援については [6] において、一部の機能の実現結果を報告してきた。１、４に関する研究室内部で試作したプロトタイプも含めてＶｅｌａプロトタイプ第１版を評価した結果、１や２を中心にすえた、手順的なプロセス記述に基づく統合環境は、ソフトウェア開発の実態や分散開発環境への対応性の良さにおいて限界があると判断した。特に上流工程では、複数の人間間のコミュニケーションによって作業が進められることが多く、メンバー間の意思の伝達や情報交換といったコミュニケーション支援と、グループ活動の協調状態の維持を支える調整支援が、プロジェクトを円滑に進めるためのキーポイントとなると考える。

現在、第一版の成果を活かしつつも３と４を拡大・発展させ、中心に据えたＣＳＣＳＤ[1]環境モデルの再構築を試みている。

以上述べた経緯から得られたＶｅｌａ第２版のアーキテクチャの構想と、ここにおける、本報告の位置付けを以下に明らかにする（図１）。

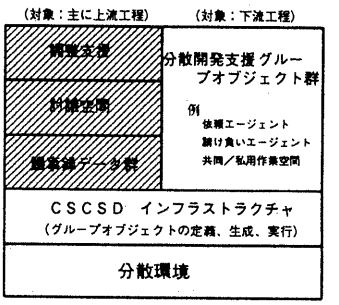

図 1: Ｖｅｌａ第２版のアーキテクチャの構想

図１において、最下層は最近さかんに研究開発が試みられている、インターネットワーキングや分散ＯＳ、オブジェクト指向データベースの機能層を仮定している。その一つ上の層に分散トランスペアレンシー [7] を保証しつつ、協調支援や分散開発支援層構築の基盤を与えるＣＳＣＳＤインフラストラクチャ層を、グループオブジェクトの概念に基づいて実現する予定である。

今回報告する概念モデルは、その上層に位置し、調整支援、討議空間、議事録データ群の３層（図１斜線部）より構成される。このモデルにより、上流工程の分析や設計段階で生ずるコミュニケーションを主体とした作業状況の把握ができ、さらにはプロジェクトの協調活動時に生じる「活動ベクトルのず

[1] Computer Supported Cooperative Software Development

れ」に起因した閉塞状態への調整支援が可能となる。

3．概念モデル構築の基本的方針

3．1 協調と調整

ソフトウェア開発において、プロジェクトの協調活動を支えるかなめの一つは、情報のスムーズな伝達である。この情報伝達がスムーズでなければ、円滑なプロジェクト活動はおろか、誤情報によって成果物が歪められる危険性もある。

そこで、情報伝達の主たる手段となる「コミュニケーション」、その中でも特に協調状態の維持がより難しい、非同期・分散型のコミュニケーションに焦点を絞っている。それは、地理的に分散しているために通信媒体を必要とし、この媒体に支障をきたすだけで、**活動ベクトルのずれ**を生じるからである。さらに、非同期の場合、意思表示をしてからレスポンスを受けとるまでの時間差があるため、思考のずれを生じやすくなるという要因も考えられる。

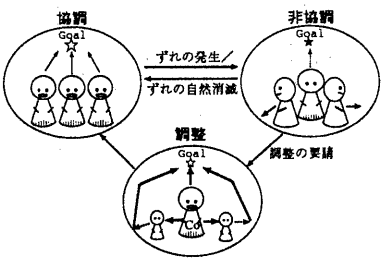

図 2: 協調と調整

ここで、「協調」と「調整」の関係について、図2をもとに説明する。図2において、**協調**とは、プロジェクトが最終ゴールに向かって活動中である場合に、メンバー個々の**活動ベクトル**が同じ方向にある状態と考えている。ところが、各々のメンバーの持つ知識や技量の格差、思惑や役割、人間関係などの人的要因、または通信手段や共有データベースが壊れるとかプロジェクト活動の依存する環境からの物的要因により、この活動ベクトルにずれが生じてくる。これを放置しておくと、いつまでたってもプロジェクトは最終ゴールにはたどり着かない。

そこで、ずれを生じる要因を取り除く**調整**が必要となる。さらに、要因の除去だけでは解決できず、当初にたてたゴールの**再設定**という**調整**が必要な場合もある。本稿では、調整の目的を**閉塞状態**（6節）からの脱却支援に焦点を当てている。

3．2 調整支援実現のために

調整を施すには、まず活動ベクトルのずれを生じる要因を特定しなければならない。そのためには、正確な**状態把握**が必須となる。

現在、非同期型の討論を研究対象とし、さらに、会話の状態遷移を中心としている代表的なシステムに gIBIS [8] がある。本研究は、会議進行状態の把握を目指すという点では、gIBIS と目標は一致するが、討議データの取り扱い方針に相違がある。

すなわち、「討議データ」を2つの状態に切り分けて定義し（図3）、それぞれデータ群別に管理を行なう。

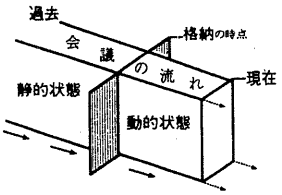

図 3: 会議の流れと切り分け

つまり、討議内容が変化しつつあるまたは変化する可能性のある**動的状態**を討議プロセス群としてモデル化し（図4 a）、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（図4 b）。

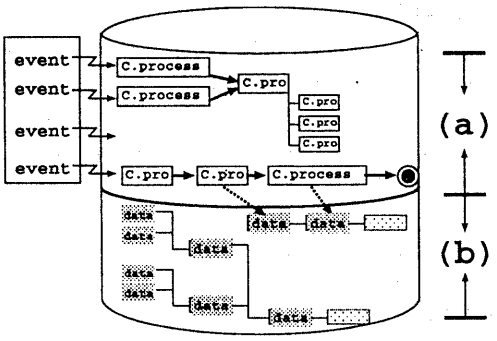

図 4: 討議データの2状態

この状態の切り分けにより、起動されている討議プロセスは、現在検討中の議題に対して生じるものだけであり、さらに論点をとらえやすくなるものと推察する。過去の議事録間の相互関係の把握は、議事録データ群のブラウジングにより可能である。

4．討議プロセス群と討議空間

議題1つ1つは、討議のライフサイクルをもち、これを**討議プロセス**と呼ぶ。この討議プロセスを駆動するものは事象であり、システム外からの**外部事象**と、討議プロセスの処理結果を次へ継続する場合の**内部事象**がある。

これらの討議プロセス群が作り出す討議空間のうちの一部をとりだし視覚化したのが、図5である。たとえば、討議量（時間×詳細化の度合）が異常に

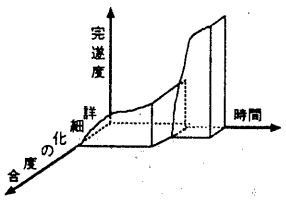

図 5: 討議空間

多いとか、討議の完遂度が低いというように討議空間から得られる評価は、会議進捗把握の目安となる。

4．1 討議空間を特徴付ける3つの軸

討議空間の構造を3つの軸（時間軸、詳細化の度合軸、完遂度軸）をもとにして、さらに考察する。

4．1．1 時間軸

時間軸は一連の討議の流れであり、討議プロセス間の**継続関係**により定義される。討議プロセスにより生み出された議事内容は、次の討議プロセスに処理結果として継続されてゆく。ところが、受け継がれる途中で他の流れの討議プロセスからの影響を受け（**相互依存／排反**関係）、議事内容に変化をもたらす場合がある。

a．継続

討議プロセスの処理結果を引き継いでゆく前後関係。引き継ぎ形態から、1対0（継続なし）、1対1、1対n（分割化）、n対1（統合化）に分類される。この決定は、1つの討議プロセスが持つ**前提条件**と**後件条件**という属性による。つまり、ある討議プロセスの後件条件は、継続する討議プロセスの前提条件となる。

b．相互依存

- **付加**：相手討議プロセスの処理内容が、討議プロセスの処理内容に付け加えられる。ただし、付加されるだけで元の内容の書き換えはおこらない。
- **変更**：討議プロセスの処理内容自体の一部書き換えがおこる。

c．排反

ある討議プロセスの処理開始により、別のプロセスの処理が必要なくなったり、もしくは相容れない場合。

4．1．2 詳細化の度合軸

一つの議題について、話し合われる詳細化の量を表したものである。

討議空間の構成単位となる一つの討議プロセスは寿命を持つ。その長さは議題内容毎にまちまちであるが、5つのステップ（図6）の流れとして捉えられる。

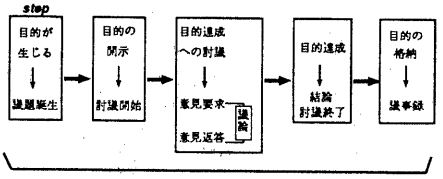

図 6: 討議プロセスの寿命

つまり、この軸は一つの討議プロセスの寿命のあいだに、その討議を達成するための議題のミニ議題への詳細化量で表現される。展開された各ミニ議題も図5に示した討議プロセスの形態を持つ。ミニ議題を生成するトリガーは、第3番目の「議論」のステップである。

詳細化の起こる要因は、2種類に分かれる。

a．計画的詳細化：会議開始時点で詳細化が判明

一つの議題では話しの内容が大き過ぎる場合に、予め細分化しておく（この細分化の'予測'が狂うと、不都合が生じ突発的詳細化を引き起こす）。

b．突発的詳細化：会議開始後に irregular に発生

その原因としては、以前の討議が不十分なため、また未決定の懸案事項の出現による情報の追加などが考えられる。

最終的に、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（**回帰 (regression)**）。

4．1．3 完遂度軸

討議がどれだけ達成されたかを示す軸である。

完遂度は人間の主観に大きく影響されるため、絶対的尺度を定義することは難しい。そこで現時点では、終了判定者が判断を下しやすいように、＜終了、延期、打ち切り、取消、未開始＞の簡潔な５段階の完遂度を定義している。ここで、＜延期＞とは後日再討議を要するもの、また＜打ち切り＞とは再討議を要さないのものである。完遂度は、後日再討議が必要となるかどうかの一つの判断材料として、あるいは議事録検証の際の資料となる。以上のような用途では、評価しやすい５段階の完遂度で十分であるが、終了状態の評価だけでなく、討議内容の評価も含めて完遂度を定義するためには、さらに考察を深めてゆく必要がある。

4．2 メタ情報と討議空間の動的再構成

討議空間には、複数の討議プロセスがあり、これらを集中管理しているのはメタ情報である。メタ情報は討議プロセス運行の管理情報を持つため、討議空間の制御も可能である。

メタ情報は、一つの討議プロセスが生成されてから消滅するまでのプロセスの走行に関する情報を持ち、これを１頁の構成単位とする管理ブックを持っている。１頁あたりの項目（次表）は、コーディネータが入力をする。

表１：１管理頁あたりの情報

	討議プロセスの
1	プロセスＩＤ
2	議題名
3	起動時間（年月日）
4	討議予定時間（開催通知基準）
5	前提・後件条件
6	継続／格納先（０以上）
7	処理手順
8	親プロセスＩＤ
9	子プロセスＩＤ（０以上）
10	再討議議題情報（０以上）
11	相互依存、排反関係の情報
12	終了判定者
13	終了状態（完遂度）
14	終了時間（年月日）
15	結果格納情報

項１ プロセスの識別ＩＤで会議開催期間を通して一意。

項２ 討議内容を端的に表す議題が付けられるが、再討議時は元議題と同一名（５節）。

項３ 討議プロセスの起動時刻。実際の処理開始は、イベント到着後。

項４ ６．２．２節。

項５，６ ４．１．１節のａ。

項７ スクリプトで記述予定[9]。

項８ 直属の親がある場合、親のＩＤ。

項９ 詳細化した議題のプロセスＩＤであり、直属のもののみ。（孫以下の情報は持たない。）

項１０ 再討議に参照する議事録情報（５節）。

項１１ ４．１．１節のｂ、ｃ。

項１２ 討議プロセスの終了判定を行なう者。1.コーディネータ（通常）,2.参加者（コーディネータを介さない討議。開始申請は必要）,3.討議プロセス（前提条件成立後、自動的に処理が進む場合）の３ケースである。

項１３ 終了判定者が、プロセス内の処理内容を編集して処理結果（議事録）を作成し、さらにその終了状態（完遂度）（４．１．３節）を下す。終了状態が＜終了＞以外の場合は、延期、打ち切りや取消になった理由が、処理結果に書き込まれる。

項１４ 終了状態が書き込まれた時間。

項１５ 討議が終了したプロセスの処理結果が議事録データ群に書き込まれる際の情報。主に同一議題に関して行なわれた討議の通し番号がカウントアップされてついている。その議題に関して一番最初の討議の場合、初期値の通し番号がセットされる。

討議プロセス自身が持つ情報は、自分自身のプロセスＩＤと途中の処理内容のみであり、その他の情報は、その都度管理ブックを参照する。

終了状態が書き込まれると、討議プロセスは必要に応じて処理結果（議事録）を継続もしくは親プロセスへ報告するとともに、＜議題名、プロセスＩＤ、討議期間、同一議題通し番号、処理結果＞の５項目を議事録データ群に格納する。処理結果には、作成者名と終了状態が含まれている。もし、終了判定者以外が作成した議事録を格納しようとした場合、図１のグループオブジェクトの監視機能によりリジェクトされ、コーディネータ側に異常状態が通知される。

討議プロセスは、格納処理が正常終了するとメタ情報に完了通知をし、消滅する。メタ情報側では、この該当プロセスの管理頁を管理ブックから取り外し、メタ情報内の別の場所で保管する。

また、討議の途中で生じたデータは、一固まりのデータに加工し、プロセスＩＤのタグを付加して、データベース内に保管する。

スタート時点では、会議の議題はプランニングされており、討議プロセス間には明確な前後関係もしくは詳細化関係があり、討議空間を構成している。しかし、討議が進むにつれて、最初の時点では予測できなかった話しの展開がおこり、新たな討議プロ

セスが必要になったり、討議順が変わったりする。この場合、最初のプランされた討議プロセス群の空間では、現状に即さなくなる。

そこで、討議プロセス群の**動的再構成**が必要となるが、それが可能となるように討議プロセスが定義される必要がある。すなわち、

- 討議プロセスの予定外の組み込み自由
- 後続の討議プロセスの追加／付替／除去
- 討議プロセスの処理内容の変更（部分的マージ、一部破棄等）
- 不要になった討議プロセスを抹消

等が可能でなければならない。このような動的再構成は、状況（事象）に応じて駆動可能な**事象駆動型討議プロセス**により可能となる。

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、

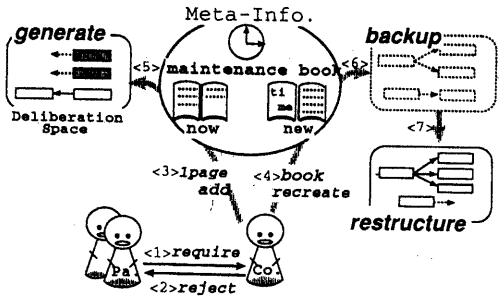

図 7: 動的再構成の仕組み

1. 討議順の変更や新議題が必要になると、コーディネータにその旨を申請（図7:<1>）。
2. 申請内容について、承認の可否決定（コーディネータ側）（図7:<2>）。承認すると、
   - 新議題の場合、新しい管理頁を作成し、管理ブックに組み込む（図7:<3>）。
   - 変更／削除の場合、管理ブックの新版作成。この新版の施行時刻（動的再構成の実施時刻）も付与（図7:<4>）。
3. メタ情報側では、
   - 新議題の場合、起動時間になった討議プロセスを生成（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（図7:<7>）。

この動的再構成により、現状の問題により即した討議の維持が可能となる。

ところで、討議の結果がもたらす副産物により、人間の脳に存在する関連情報や知識に関する再構成が起きる場合がある。これは人間内の情報の再認識／再思考の問題であり、討議プロセスの再構成とは直接的には関係ないが、再思考の結果による議題の組み変えをメタ情報に反映することにより、討議プロセス群の動的再構成へと繋がる。

5．議事録データ群と議題の蒸し返し

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プロセスの仮ノードも作成して、これに接続される。親プロセスの討議も終了し、議事録が格納されると、仮ノードは親プロセスのデータに置き換えられる。

ところで、会議を行なっていると、同じ議題に対する再討議をどうするかという蒸し返しの問題が発生してくる。これに関しては筆者らは、図6の最後のステップ（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。

すなわち、議題の蒸し返しとは、同一議題に対する新しい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。

この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付けされる（管理頁の項１０）。

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

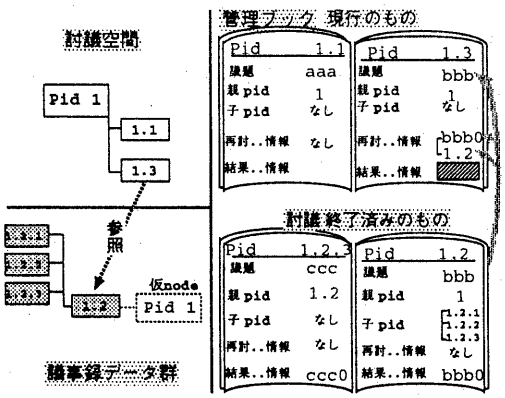

図 8: 管理頁の例示：再討議の場合

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、

1. 会議参加者側で再討議が必要になると、コーディネータにその旨を申請。
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（コーディネータ側）
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. 起動時間になり、生成された討議プロセス（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）に更新される。

6. 調整支援

6.1 閉塞とは

活動ベクトルのずれを生じる原因の中には、プロジェクト活動の作業状態の維持を極端に阻み、自然には除去されにくい要因があり、これを**閉塞要因**と呼ぶ。**閉塞要因**によって引き起こされた**未解決状態**を**閉塞状態**と定義する。この状態を放置しておけばプロジェクト活動は停止してしまう。そこで、コーディネータによる**調整支援**を施し、**閉塞状態**からの**脱却**を計る。

例えば、メイリングリストを利用した会議事例において、一番の閉塞要因となりやすいのは、**必要情報の欠落**（ex. 重要な会議参加者の不在や怠慢により、返信が返ってこない。）であった。これを討議プロセス側から分析すると、

- 外部／内部事象が長時間未到着状態。
- 討議プロセスの処理中断、中止状態。
- 無駄な詳細化による回帰通知が未回収状態。

などの閉塞状態が考えられる。

6.2 閉塞状態への対応策

閉塞状態への対応策については、検討中であり、以下の2つの部分的解が得られた状況である。すなわち、外部／内部事象の未到着の原因が一時的な場合でかつ、事象が独立の場合には、6.2.1節で述べる先取り実行制御機構を適用する。

上記以外の場合については、6.2.2節で述べる状態の自動通知により、コーディネータの介入を促すことで対処する。

6.2.1 討議継続軸上の一調整支援

内／外部事象の長時間にわたる未到着により、閉塞状態に落ちいった場合に、討議プロセス自身がその状況を部分的に緩和する手段の一つが、**先取り実行制御機構**である。

通常は逐次に実行されている一連の討議プロセスにおいて、後続プロセスの前提条件の一部が満たされたとき、現プロセスの全ての機能の実行終了を待たずに、後続プロセスの一部機能を先取りして実行することを意味する。適用ケースは、討議プロセスの処理が単独で処理可能であり、他の討議プロセスから影響を受けない独立事象の場合である[10]。

6.2.2 通知基準の利用による支援への前準備

討議プロセスは知能を持たないため、現在置かれている状態が閉塞状態か否かを判断する能力はな

い。よって何らかの基準を設けて、その基準を過ぎた時点で、コーディネータに通知する必要がある。

まず、通知基準として会議の物的資源である「時間」の消費が考えられる。具体的には、管理頁の討議予定時間のチェックをメタ情報が行なうことで実現する。また、別の通知基準として、一つの議題に対する詳細化のレベル数が考えられる。ただし、この基準は必ずしも閉塞状態と直結する概念ではないので、今後の検討が必要である。

7．おわりに

現時点では、ソフトウェア開発に限らず、プロジェクト活動のキーポイントとなる**協調**や**調整**という概念を活かすようなデータベースの研究は、その重要性にも関わらず、いまだ本格的にはなされていない。

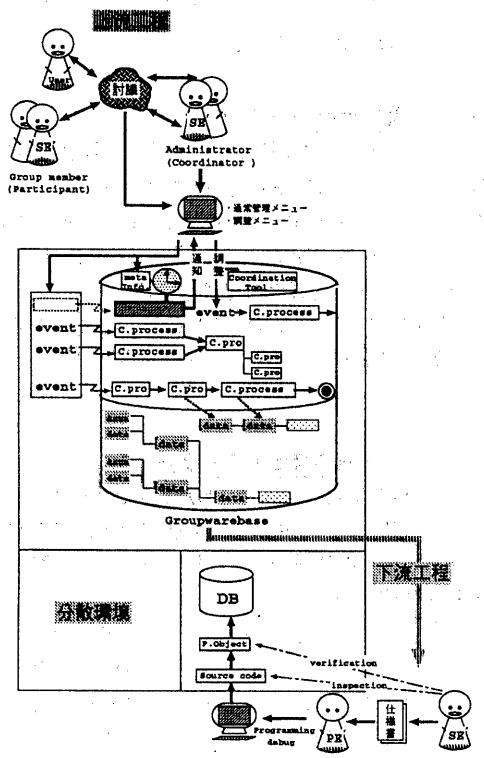

図 9: グループウェアベースとＣＳＣＳＤ

このプロジェクトチームの円滑な協調活動を支援するデータベース（グループウェアベース）（図９）の開発が将来の目標である。それは、調整支援のためのツールやメタ情報等の管理層、グループ活動（討議）層、活動より得られた議事録データ層より構成され、ＣＳＣＳＤを目標とするＶｅｌａ第２版の基礎となる。本報告で提案した概念モデルをさらに充実するためには、討議空間における閉塞状態の分類や詳細な定義を検討する必要がある。

議事録データ群および討議空間を定義した目的は、活動中のプロジェクトチームの状態を正確に保持し、活動ベクトルのずれの検出にともなう調整支援にあった。このようなアプローチを通じて、分散開発環境下における開発チーム構成員間の自主的な協調活動や新しいプロジェクト管理法を支援するための基礎理論へと発展させていく予定である。

参考文献

[1] 落水:' ソフトウェア開発におけるグループウェアの役割', ソフトウェア・ツール・シンポジウム'92,pp.83-92 (1992).

[2] 落水:' ソフトウェアプロセスモデルに基づくソフトウエア開発支援環境 Vela', 日本ソフトウエア科学会第７回大会論文集,pp.205-208 (1990).

[3] K.Ochimizu,T.Yamaguchi:'A Process Oriented Architecture with Knowledge Acquisition and Refinement Mechanisms on Software Process', 6th ISPW,pp.145-147 (1990).

[4] 山口, 落水:' ソフトウェアプロセスモデル構築における知識獲得と利用方式', 人工知能学会研究会資料,SIG-FAI-9002-3 (1991).

[5] 山口, 落水:' ソフトウェア設計プロセス知識獲得支援環境の構成法', 第 10 回設計シンポジウム (1992).

[6] 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, ソフトウェア工学 77-5 (1991).

[7] T.Rodden,J.A.Mariani and G.Blair:'Supporting Cooperative Applications',Computer Supported Cooperative Work,Vol.1,No.1,pp.41-67 (1992).

[8] Jeff Conklin and Michael L. Begeman:'gIBIS:A Hypertext Tool for Exploratory Policy Discussion',CSCW '88,pp.140-152 (1988).

[9] Roger C.Schank:' ダイナミック・メモリ', 近代科学社 (1988).

[10] 門脇, 落水:' 先取り実行や遅延回復機能を有するソフトウェアプロセスの実行制御方式', ソフトウェア科学会 ソフトウェアプロセス研究会, pp.61-65 (1993).