ソフトウェア工学 90−18
（１９９３．２．５）

# ソフトウェア分散協調開発の為の合理的なグループ意思決定支援

古宮誠一　西野光

情報処理振興事業協会　技術センター

ソフトウェア分散開発を可能にするには、データ伝送機能や伝送上のセキュリティ機能だけではなく、分散開発環境において協調的に進められるソフトウェア設計過程を強力に支援する必要がある。本稿では、このようなソフトウェア設計過程を協調型設計過程と呼び、これを強力に支援する方式を提案している。

# Rational Group Decision Support for Software Distributed and Collaborated Development Environment

Seiichi KOMIYA　Hikaru NISHINO

Software Technology Center
Informatin-Technology Promotion Agency, Japan (I.P.A.)

6F Shuwashibakoen 3-chome BLDG.
1-38, Shibakoen 3-chome, Minato-ku, Tokyo 105, Japan

In order to realize software distributed development, facilities to support software design process which is performed cooperatively are desired as well as data transmission facilities with seculity functions. Author calls the above software design process `collaborated design process', and proposes a method to support this process powerfully.

## 1．はじめに

開発対象となるソフトウェアの大規模化・複雑化に伴い、大量の作業人員が必要となり、１ヶ所に集中しての作業場所確保が困難になってきた。このため、１つのソフトウェア・システムを互いに遠く離れた作業場所の作業者が共同で開発する形態（＝分散開発）を採らざるを得なくなっている。また、人材確保のために、企業が地方にも開発拠点を進出せざるを得なくなっている状況も、ソフトウェアの分散開発を必然的にしている。

ソフトウェア分散開発を可能にするには、分散開発に必要な情報、特に作業の中間成果物などの大量のデータを遠く離れた作業場所に安全かつ確実に伝送するツールが不可欠である。即ち、大容量のデータ伝送機能とデータ伝送上のセキュリティ機能を持ったツールが不可欠である。しかし、その割に分散開発が浸透していないのは、データ伝送能力やデータ伝送上のセキュリティ機能がまだ不充分だからである。

しかし、データ伝送機能（データ伝送上のセキュリティ機能を含む）を充実させるだけで、ソフトウェア分散開発が可能になるわけではない。というのは、そもそもソフトウェアの開発作業というものは、複数の作業者間で協調して進めるべきものであり、分散開発においては作業場所が互いに遠く離れているので、作業者間での作業の協調が困難だからである。従って、互いに遠く離れた作業者間での協調作業をどうするか、ということが問題になる。ソフトウェア分散開発において、この問題に対処する方法は２つある。１つは、各開発拠点への作業分担や作業のやり方を工夫することにより、この問題を避けるというアプローチである。もう１つは、互いに遠く離れた作業者間での協調作業が可能となるように、これらの作業者間での協調過程を強力に支援するツールを開発するというアプローチである。ところで、ソフトウェア開発の上流工程と下流工程では、概ね次のような相違がある。下流工程では複数の作業がただ並行して進められればよく（＝協調過程支援機能の必要性は低く、ときには皆無でもよい）、大容量のデータ伝送機能が必要となる。逆に、上流工程では小容量のデータ伝送機能でもよいが、強力な協調過程支援機能が必要となる。故に、前者を指向した開発支援環境による開発形態を分散並行開発と呼び、後者を指向した開発支援環境による開発形態を分散協調開発と呼ぶ。分散並行開発を支援する環境としては、ICAROS [1]のなどの研究があるが、分散協調開発を支援する環境の研究は殆どない。従って、ソフトウェア分散協調開発、即ち、分散開発環境でのソフトウェア設計作業を協調して進める過程を知的に支援する技術の開発が望まれる。

本稿では、ソフトウェア分散協調開発に求められる支援機能とその実現方法を明らかにする。

## 2．ソフトウェア協調型設計過程

ソフトウェアの開発では、開発対象の大規模化・複雑化により多人数で開発することが普通となっている。このような状況での開発作業は、複数の作業者間で協調して進められなければならない。特に、ソフトウェア開発の上流工程では、設計上の意思決定のために、その設計に関係する作業者間での合意が必要であり、その作業は協調して進められなければならない。このように協調的に進められるソフトウェア設計過程をソフトウェア協調型設計過程（software collaborated design process）と呼ぶ。

### 2．1　ソフトウェア協調型設計過程のモデル

ソフトウェア協調型設計過程は、協調システムと見なすことができる。協調システムのモデルは、協調のための制御の在り方から次の３つに分類できる。１つは、マネージャの存在を仮定しない（＝フラットな）モデルであり、システムの形態がネットワーク（＝網）状のシステム構成になるという特徴がある。これを管理者なし（または平面型）のモデルと呼ぶ。残りは、マネージャの存在を仮定するもので、マネージャの人数によってさらに２つに分けられる。１つは、マネージャが一人だけのモデルであり、システムの形態が放射状のシステム構成になるという特徴がある。この場合には、制御が一人のマネージャに集中するので、これを集中型制御のモデルと呼ぶ。もう１つは、複数のマネージャが階層的に配置されたモデルであり、システムの形態が木構造状のシステム構成になるという特徴がある。これを階層型制御のモデルと呼ぶ。なお、分散型制御のモデルという言葉は、ここでの管理者なしのモデルだけを指すのか、階層型制御をも含めて指すのか、紛らわしいので用いないほうがよいと考えている（ＪＪ原子モデル、ＪＳ原子モデル、ＳＳ原子モデルと呼ばれるモデルが松本ら[17]によって提唱されているが、これら３つのモデルとほぼ同様のモデルである）。ソフトウェア開発のような知的で複雑な作業を協調して進めるには、管理者なしのモデルでは困難であろう。事実、実際のソフトウェア開発では、管理者の存在を仮定した集中型または階層型のプロジェクトで開発が進められている場合が殆どである。このため、ここでは管理者の存在を仮定するモデルを想定するが、話を簡単にするために、以下では集中型制御のモデルに焦点を絞って考察する。

集中型制御のモデルに基づくソフトウェア協調型設計過程では、チーム・リーダの良い悪いがそのままそのチームの良い悪いを決定することは、我々のよく経験するところである。従って、メンバーの意見を巧く抽き出すように、チーム・リーダの行動を知的に支援するとともに、協調に導くアルゴリズムを与えることによって、メンバー全員の意見を協調に導くシステムを構築することは大きな意味を持つ。

## 2．2　協調型設計過程支援ツールに求められる要件

ソフトウェアの協調型設計過程を支援するツールに求められる要件と、各要件を満足させるために導入する技術とその効果は次のとおりである。

①[要件１]構成メンバーの力関係や妥協のために、グループで意思決定された結論が最適な案になるとは限らない、というグループによる意思決定の欠点を補えること。

[要件１に対して導入する技術とその効果]

Kepner-Tregoe法を導入することにより、グループによる意思決定のプロセスを合理的なものにする。意思決定のプロセスが合理的になれば、グループによる意思決定が構成メンバーの力関係や妥協によって歪められることがない。

②[要件２]問題解決のための様々な視点が提示されるという、グループによる作業の利点を活用できること。

[要件２に対して導入する技術とその効果]

Kepner-Tregoe法を導入することにより、発言の形態がボトムアップになるとともに、グループによる意思決定に向けて議論を尽くすことができる。その結果、問題解決のための様々な視点が提示される。

③[要件３]グループの意思決定に対して、メンバー全員の合意が得られるように工夫されていること。

[要件３に対して導入する技術とその効果]

Kepner-Tregoe法を導入することにより、グループによる意思決定のプロセスを透明で、かつ、合理的なものにする。これにより、グループによる意思決定にメンバー全員の合意が得られるようになる。

④[要件４]個々の問題に対してグループで意思決定された、それぞれの結論の間に矛盾がないこと。

[要件４に対して導入する技術とその効果]

ATMSや分散ATMSを導入することにより、個々の問題に対してグループで意思決定された、それぞれの結論間の無矛盾性を自動的にチェックできるようになる。

⑤[要件５]決定までに時間がかかる、というグループの意思決定の欠点を補えること。

[要件５に対して導入する技術とその効果]

Kepner－Tregoe法の導入により、グループによる意思決定のための合理的な手順が与えられるので、決定までの時間を短縮することができる。

⑥[要件６]グループによる問題解決やグループによる意思決定に、過去の経験や叡智を生かせること。

[要件６に対して導入する技術とその効果]

事例ベース推論を導入することにより、グループによる問題解決やグループによる意思決定に、過去の経験や叡智を生かすことができる。

## 2．3　協調型設計の設計プロセス

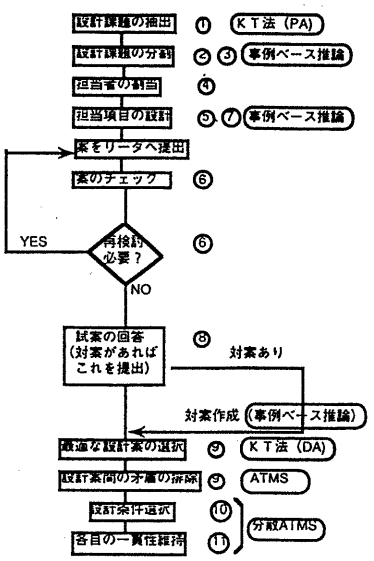


図１　ソフトウェア協調型設計の設計プロセス

議論を明確にするために、本稿では次のような場面を想定する。即ち、複数の作業場所の作業者が、それぞれの席から通信回線を介して、自分の意見や考えなどを交換する形で作業を進めて行く場面を想定する。ソフトウェア開発工程としては、システムを新規に作成する場合には、要求仕様が固まった直後の、各要求項目をどのように実現するか（＝設計仕様）を検討する工程を想定する。また、ソフトウェアを保守する場合には、問題の原因を調査し、その解決策を検討する場面を想定する。このような場面では、ソフトウェアの設計作業を協調的に進めなければならないことは明かであろう。このとき、このチームのリーダが、メンバーの意見を抽き出しながら、ソフトウェアの設計仕様をまとめて行く過程を考える。このような過程は、集中型制御モデルに基づくソフトウェア協調型設計過程の典型的な例である。

このとき、ソフトウェア設計仕様の検討ために、設

計チームがとる行動を時系列的に列挙すれば次のとおりである。

(1)設計課題を抽出する。

(2)設計案とそれに対するコメントを出し合うことによって議論を尽くす。

◎①設計課題を分解することにより、設計項目をリストアップする。

◎②設計項目を個別に検討可能な項目と共通の問題として同時に検討すべき項目とに振り分ける。

◎③設計項目を検討すべき順序に並べ直す。

④設計項目別に担当者を割り当てて、各担当者に作業を依頼する。

◎⑤各担当者は自分に割り当てられた設計項目を検討し、設計私案としてリーダに提出する。

⑥リーダは、提出された設計私案の内容をチェックし、一定の水準に達していると判断したときには、これをチームの設計試案としてメンバーに回覧する。設計私案の内容が一定水準に達していなかったときには、コメントを付けてこれを担当者に返却するとともに、再検討を促す。

◎⑦担当者は再検討の後に設計私案をリーダに再提出する。(再提出の後は⑥に戻る)

⑧メンバーは、回覧されている設計試案に対して意見があれば、コメント票の形で各自の意見を提出する。意見がなければ、その旨をコメント票に明記してこれを提出する。

◎⑨リーダは、入手したコメントをもとに各設計項目ごとに意見調整を行う。このとき、リーダは、採用すべきコメントであれば、これを設計試案の作成者に送りコメントの吸収を促す。却下すべきコメントであれば、これをコメント提出者におくり、コメント取り下げを促す。

(コメントの内容によっては、①－③のいずれかに戻ることもある。)

◎⑩コメントの却下を指示されたメンバーは、コメントを無条件に取り下げる。

◎⑪コメントの吸収を指示された担当者は、設計試案を再検討し、設計私案としてこれを再提出する。(再提出の後は⑥に戻る)

⑫設計項目全部の検討が終わるまで⑤～⑪を繰り返す。

(3)設計案とコメントが出揃ったところで、複数の設計案の中から最適な設計案を選出する。

上記のうちで、コンピュータによる支援が可能だと考えられるのは、行動(1)と行動(3)と行動(2)の中の①②③⑤⑦⑨⑩⑪の8つ(いずれも◎が付いている)である。

## 3．KT法による合理的思考の実現と合意形成

Kepner-Tregoe法[8][9]は、Charles H. KepnerとBenjamin B. Tregoeによって提案された、マネジメントのための合理的な思考手順である。この方法は、問題の原因を究明するための思考手順である「問題分析」(Problem Analysis)、複数の問題解決手段の中からその1つを選択するための思考手順である「決定分析」(Decision Analysis)、将来起こり得る不都合を現在わかっていることから予測するための思考手順である「潜在的問題分析」(Potential Problem Analysis)、何が起こっているのかを把握し、管理可能な要素へ分解することによって、他の3つの思考手順の中からその1つを選択するための思考手順である「状況分析」(Situation Appraisal)の4つからなる。

### 3．1　KT法(問題分析)による設計課題の抽出

Kepner－Tregoe法の「問題分析」は、既に起こっている問題の原因を究明するための思考手順であり、原因が究明された後、その原因を取り除く方法を求めることが設計課題となる。このように考えれば、設計課題抽出手順としてKepner－Tregoe法の「問題分析」が適用可能である。従って、2章で列挙した、設計チームがとる行動の(1)に適用可能である。

Kepner－Tregoe法の「問題分析」による思考手順は、概ね次のとおりである。

①問題が起きる前と起きた後との差異を明らかにすることにより、問題を明確にする。

②問題が起きる前と起きた後では、どこでどのような差異が生じたかを、問題の対象、発生場所、日時、程度という4つの視点から明らかにすることにより、問題を説明する。

③②で挙げた4つの視点から重要な情報を抽出し、考え得る原因を想定する。

④「真の原因」である可能性の最も高い原因を探すために、想定した原因についてテストする。

⑤「最も可能性の高い原因」が「真の原因」であることを裏づける。

この分析手順は、問題が起きる前と起きた後では、どこでどのような差異が生じたかを、順を追って絞り込んで行くことによって、問題の原因を特定する方法である。従って、分析の方法が合理的であることは明かであろう。

しかし、手順から判るように、Kepner－Tregoe法の「問題分析」は、情報システムの使用経験もなければ業務処理の経験もないという場合には適用できない、

という制約がある。

### ３．２　ＫＴ法（決定分析）による設計案の選出

Kepner－Tregoe法の「決定分析」による思考手順は、概ね次のとおりである。

①絶対に達成されなければならない目標（＝絶対目標）を列挙し、対策案のそれぞれが絶対目標を達成できるか否かを明らかにする。このとき、絶対目標を達成できない案を対策案から外す。

②できれば達成して欲しい目標（＝希望目標）を列挙し、希望の強さを１０点法でウエイトをつける。

③対策案のそれぞれによって、それぞれの希望目標がどの程度達成できるかを、１０点法で採点する。

④加重加算して最も高い得点のものを第１候補とし、次に高いものを第２候補、その次に高いものを第３候補とする。

この手順は、２章で列挙した、設計チームがとる行動の(３)に適用可能である。この手順が合理的な意思決定のプロセスを実現するために有効であること明かであろう。

なお、各案の評価が状況によって異なることがある。例えば、余暇の過ごし方などの案は、天候によって評価が異なるようにである。このような場合には、意思決定のための基準として、ラプラスの基準[10]、マキシミニの基準[10]、フルビッツの基準[10]、ミニマックスの基準[10]を４つを用意しておき、必要に応じてこれらを使い分けるとよい。ラプラスの基準とは、案の評価に影響を与える事象の生起確率をすべて等しいとして、その場合の評価値の最も高い案を選択するというものである。マキシミニの基準とは、悲観的立場をとった選択基準である。即ち、それぞれの案の最悪の場合を想定し、その中で最も被害の少ない案を選択するというものである。フルビッツの基準とは、悲観と楽観を混合したもので、楽観の程度をパラメータで指定できる。マキシミニの基準が悲観的立場をとったものなので、ここでは最も楽観的な立場をとれるようにパラメータをセットする。ミニマックスの基準とは、機会損失を最小にする案を選択するための基準である。そして、ユーザにより特に指定がなければ、ラプラスの基準をこのような場合の標準とすればよい。

ところで、絶対目標とするか希望目標とするかでメンバーの合意が得られなかった場合には、絶対目標としてメンバー全員の合意が得られた目標のみを絶対目標とし、残りを希望目標とすればよい。また、希望目標のウエイトの高さや希望目標のそれぞれに対する各案の得点がグループのメンバーの間でなかなか合意が得られないことがある。この場合には、メンバー全員の与えたウエイトや得点の値の平均を採用するなどの工夫が必要である。

### ４．IBISによる設計案と設計上の意思決定の記録

Conklin によれば、ソフトウェアの保守に要する費用は、ライフサイクル全体での開発コストの約75％にも及ぶという。また、保守に要する費用の約50％は、既存ソフトウェアの理解するための工程「再設計」に要するものだという。一方、Curtisら[５]は、設計時におけるコミュニケーションとコーディネーションの不足がもたらす「ゆゆしき問題」を示した。このことは、設計上の意思決定に充分な議論を尽くすことの必要性と、その折の意思決定の根拠をソフトウェアの保守時に参照できるような枠組みの必要性を示している。Conklin らは、これらのことから設計上の意思決定における議論の状態遷移を構造的に表現するためのモデルIBIS[４]を提案した。IBISでは、設計プロセスを構成する主要な構成要素は、顧客・設計者・インプリメンターなどの間での会話であるとし、これらの間の会話を図２のような Issue－Position－Argumentという枠組みで表現する。これは、設計上の論点（ Issue）に対して、それぞれの専門家からの複数の意見表明（Position）がなされ、各表明に対して賛否の議論（Argument）がそれぞれ展開されるという考え方に立っている。そして、これらの会話による非公式な情報の獲得を支援し、得られた膨大な情報を組織化し検索するためのツールとして gIBIS[４]を開発した。

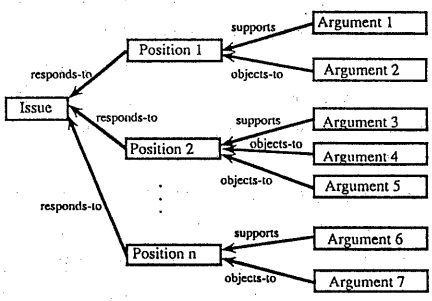


図２　IBISによる対話型（協調型）設計過程

IBISでは、１つの Issueに対して複数の専門家がそれぞれの意見を述べ、意見のそれぞれに対して賛否の議論が複数なされることを前提にしている。即ち、１つの Issueに対して複数のPositionがあり、それぞれのPositionに対して複数のArgumentがあることを前提にしている（図２参照）。このモデルは、１つの論点に対して、複数の専門家がほぼ同時にものを考え、それぞれの立場での直観に基づいて議論を進める形で、

問題解決を図る場合には確かに有効である。しかし、このモデルは、Issueを与えられてからPositionまでに多くの時間を要する場合には適合しない。この場合には、図３のように、１つのIssueに対して原則として１人の専門家が意見を述べ、この意見に対して複数の人による賛否の議論がなされる。例えば、チームを組んでのソフトウェア開発の場合には、１つの設計課題（Issue）に対して、原則としてメンバーの１人が割り当てられ、その人が具体的な設計提案（Position）をする。これに対して他のメンバー全員が賛否の意見（Argument）を述べるのが普通であろう。

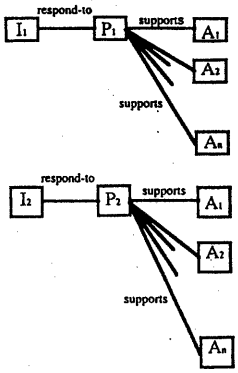


図３　IBISによる協調型設計過程
（Issue1とIssue2が並行検討可能な場合）

この場合においても、１つの設計課題に対して、複数の設計提案がなされる場合がある。それは、１つの設計提案に対して、メンバー全員による賛否の議論だけでは設計課題が解決しない場合である。そのような場合は２つある。１つは対案が必要となる場合であり、もう１つは、部分問題に分割しないと、先に提案された設計案に対する賛否の議論ができない場合である。前者の場合には、１つの設計課題（Issue）に対して、複数の設計提案（Position）が提出されることになるので、IBIS本来の適用方法と同じになる。後者の場合には、図４のようにIssueをSubissueに分解する過程を明示的に採り入れるが必要があるという点で、IBISの適用方法とは異なってくる。従って、この場合にはハイパーカードにおいて、IssueとSubissueとを繋ぐリンク・ラベルも必要となってくる（図４参照）。

IBISにおいては、IssueとIssueとの間にはPosition提出のための順序関係という概念は存在しない。従って、図３のように、２つの設計課題（Issue）をそれぞれ個別に検討（即ち、並行検討）可能な場合だけを扱っている場合には問題はない。ところが、ソフトウェアの設計課題と設計課題の間には、図５のように、それらを１つのものとしてまとめなければ設計できないとか、図６のように、一方の設計が終わらなければ

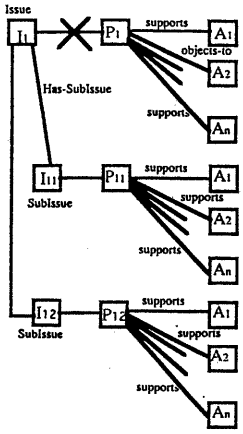


図４　IBISによる協調型設計過程
（IssueがSubissueに分解される場合）

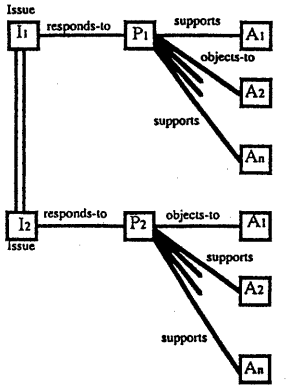


図５　IBISによる協調型設計過程
（複数の課題を１つの課題として検討すべき場合）

もう一方の設計を始めることができない、というような作業の順序関係がある。このように、ソフトウェアの設計においては、IssueとIssueの間における作業の順序という概念が必須となるので、IBISにこれを追加する。従って、IssueとIssueの間、IssueとSubissueの間、SubissueとSubissueの間に設計作業の順序関係を表すリンク・ラベルが必要となる。

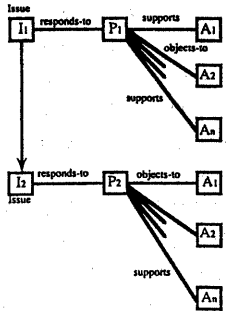


図６　IBISによる協調型設計過程
（Issue1をIssue2よりも先に検討すべき場合）

## 5．ATMSによる設計案相互の無矛盾性チェック

本章では、ATMSと分散ATMSによる設計案相互の無矛盾性チェックを考える。

ATMS(Assumption-based Truth Maintenance System)は、信念の一貫性を維持するシステムである。本章では、選出された設計試案相互の無矛盾性をチェックするアルゴリズムとして、ATMSが適用可能であることを示す。そのためには、システムが動作するイメージを示す中で、ATMSのアルゴリズムが、適用可能であることを示せばよい。

(1)リーダ1人の指示の下に作業を進めるn人のメンバーからなるソフトウェア設計チームがあり、リーダおよび各メンバーにはそれぞれ1台のワークステーションが割り当てられている。なお、リーダは管理者専任であってもよく、同時に担当者としての役割を兼務していてもよい。

(2)リーダおよびメンバーからの意見提出はすべて通信回線を介して行われる。そして、システムが管理していて意見の提出者が誰であるかを知ることができる。

(3)リーダおよびメンバーは、Kepner－Tregoe法の思考手順「問題分析」に従って設計課題を抽出する。抽出された設計課題は、IBISの IssueおよびSubissueとして記録される。

(4)各メンバーは、自分に割り当てられた設計課題に対する設計を自分のワークステーション上で行い、得られた結果（= 設計私案）をリーダに提出する。提出された設計私案は、IBISのPositionとして記録される。

(5)リーダは、提出された設計私案の内容をチェックし、一定の水準に達していると判断したときには、これをチームの設計試案としてメンバーに回覧する。設計私案の内容が一定水準に達していなかったときには、コメントを付けて作成者に返送するとともに、再検討を促す。

(6)各メンバーは、回覧されている設計試案に対して、自分の意見をコメント票の形でまとめ、設計試案に添付してメンバー全員に回覧する。このとき、コメント票の作成に際しては必ずYes/No/unknownのいずれかを明らかにしなければならない。もし、部分的にYes/No/unknownがある場合には、Yes/No/unknownの意見ごとに1つのArgumentを設けて、賛否の意思表示とその根拠を明らかにしなければならない。賛否の意思表示とその根拠は、IBISのArgumentとして記録される。

(7)1つの設計課題に対して設計試案が複数あるとき、複数の設計試案の中から最適なものを1つ選出する必要がある。リーダとメンバーは、この過程をKepner－Tregoe法の思考手順「決定分析」に従って実施する。

(8)設計課題ごとに1つずつ選出された設計案が全部で複数あるとき、各設計案が持つ前提条件の間に互に矛盾するものがあってはならない。このため、各設計案が持つ前提条件相互の無矛盾性をチェックし、矛盾のない設計案の組を取り出すためのシステムATMS[7]を持っている。

(9)リーダは、ATMSによって篩い落とされた設計案の前提条件の組合わせを調べ、採用すべき前提条件と却下すべき前提条件の2つに振り分ける。採用すべき前提条件の下に設計された設計案は採用とし、却下された前提条件の下に設計された設計案は不採用とする。

(10)却下された前提条件をリーダの絶対的な指示と見なし、却下された前提条件の下に設計された設計試案を、各メンバーが持つ設計試案の中から篩い落とす必要がある。このときのメンバーの振舞いは、分散ATMSの動作そのものである。故に、各メンバーが持つ信念（= 設計案）の集合に対する、メンバーごとの一貫性を維持するシステムとして分散ATMSが適用可能である。

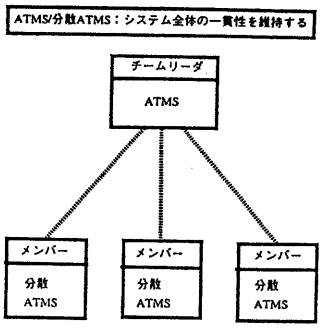


図7 リーダとメンバー間のATMS/分散ATMSの役割

(11)メンバーは例外なく自身の信念（= 設計案）の一貫性を維持するためのシステム分散ATMSを持っている。

(12)却下を通知された意見提出者は、その通知を絶対的なものとして却下された意見を取り下げなければならない。各メンバーは、この折にも自身の信念（= 設計案）の一貫性を維持させるべく自身が持つ分散ATMSを作動させる。

## 6．おわりに

ソフトウェア分散開発を可能にするには、データ伝送機能や伝送上のセキュリティ機能だけではなく、分散開発環境において、協調して進められるソフトウェア設計過程（このようなソフトウェア設計過程を協調型設計過程と呼ぶ）を強力に支援する必要がある。

最初に、このようなソフトウェア設計過程を協調システムと見なし、そのモデルについて考察した。そこ

では、ソフトウェア開発のような知的で複雑な作業にはリーダの存在を仮定したモデルの導入が実際的であり、リーダの行為を知的に支援するシステムの開発が必要であることを主張した。

最初に、協調型設計過程のソフトウェア設計プロセスを１２のプロセスに分解し、コンピュータによる知的支援が可能な部分と困難な部分とに振り分けた。

次に、ソフトウェア協調型設計過程に求められる支援機能と、その実現のために導入される技術の概略を明らかにした。

次に、グループによる問題解決とグループの意思決定のプロセスを合理的なものにするために、Kepner－Tregoe法を導入した。具体的には、設計課題抽出過程には「問題分析」手順を、複数の設計案の中から、その１つを選択する過程には「決定分析」手順を導入した。そして、設計課題とこれに対応する１つまたは複数の設計案および各設計案に対する１つまたは複数の賛否の意見を記録するためにIBISを導入した。さらに、設計課題ごとに選出された設計案相互の無矛盾性をチェックするためにATMSと分散ATMSが適用可能であることを示した。

本稿では具体的な適用方法までは述べなかったが、設計案の提案過程への支援のために事例ベース推論が適用可能である。

現在、ソフトウェア分散協調開発を可能にするために導入した、これらの技術が有効であることを実証するために、システムを試作中である。

［参考文献］


[１]Aoyama, M.: Distributed Concurrent Development of Software System: An Object-Oriented Process Model, COMPSAC'90, pp.16-30(Nov. 1990).

[２]Bridgeland,D.M., Huhns,M.N.; Distributed Truth Maintenance, AAAI-90, pp.72-77(1990).

[３]Brightman, H.J.: Problem Solving: A Logical and Creative Approach, The College of Business Administration, Georgia State University, 1980.

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

[５]Curtis, B. et al.: On Building Software Process Models under the Lamppost, Proceedings of the 9th International Conference on Software engineering, IEEE, pp.96-103(1987).

[６]Doyle, J.: A Truth Maintenance System, Artificial Intelligence 12, pp.231-272(1981).

[７]de Kleer, J.: An Assumption-based TMS, Artificial Intelligence 28, pp.127-162(1986).

[８]Kepner, C.H. and Tregoe, B.B.: The Rational Manager, Princeton Research Press, 1965.

[９]Kepner, C.H., Tregoe, B.B.: The New Rational Manager, Princeton Research Press, 1981.

[10]木下栄蔵：わかりやすい意思決定論入門－基礎からファジィ理論まで、啓学出版(1992).

[11]Lee, J.: SIBYL: A Qualitative Decision Management System, in Artificial Intelligence at MIT, Winston, P. H. and Shellard, S. A. EDs., pp.104-133, MIT Press(1990).

[12]Lee, J.: SIBYL: A Tool for Managing Group Decision Rationale, CSCW '90 Proceedings, ACM pp. 79-92(1990).

[13]Lee, J.: Extending the Potts and Bruns Model for Recording Design Rationale, Proceedings, of the 13th International Conference on Software Engineering, IEEE, pp.114-125(1991).

[14]Maclean, A., Young, R. M. & Moran, T. P.: Design Rationale: The Argument behind the artifact, In proceedings of CHI'89 Human Factors in Computing System, pp.247-252(1989).

[15]Maclean, A., Young, R. M., Bellotti, V. M. E., & Moran, T. P.: Quuestion, Options, and Criteria: Elements of design space analysis, Human-Computer Interaction(1991).

[16]Mason, C.L. and Johnson, R.R.: DATMS: A Framework for Distributed Assumption-Based Truth Reasoning, Gasser, L. and M.N. eds. Distributed Artificial Intelligence, Vol.II, pp.293-318, Morgan Kaufmann(1989).

[17]松本、本位田：知識に注目した協調計算のモデル化について，マルチエージェントと協調計算ワークショップの論文集，(Dec.12-14, 1991).

[18]Potts, C. and Bruns, G.: Recording the Reasons for Design Decisions, Proceedings of the 10th International Conference on Software Engineering, IEEE, pp.418-427(1988).

[19]Potts, C.: A Generic Model for Representing Design Methods, Proc. of the 11th International Conference on Software Engineering, IEEE, pp.217-226(1989).

[20]垂水：グループウェアのソフトウェア開発への応用，情報処理，Vol.33, No.1, pp.22-31(1992).