

メンバーの知見を最大限に生かすための ソフトウェア設計プロセスとその支援環境

古 宮 誠 一^{*1} 西 野 光^{*2}
情報処理振興事業協会 技術センター

ソフトウェアの大規模化・複雑化に伴い、複数の人間により1つのソフトウェアを共同で作成することが多くなった。そこでは、メンバーの知見を最大限に生かしてソフトウェアの設計が行えるような枠組みが必要となる。言い替えば、メンバーの知見を引き出しつつ、得られた意見を基にグループとしての最善の意思決定を可能にするような枠組みが必要である。このような枠組みは、ソフトウェア設計作業をグループワークとして捉えることにより初めて可能となる。このことは、これまでソフトウェア設計作業を個人作業として捉え、設計方法論またはこれに基づく機械支援という形で支援してきたことと対照をなす。本論文は、ソフトウェアの設計において、グループとしての意思決定を合理的に行うための枠組みとしてK T法 (Kepner-Tregoe法) の適用が有効であることを示している。

キーワード: ソフトウェア協調型設計過程, 合理的な意思決定, 集約型の議論,
ケプナー・トリゴー法, ソフトウェア保守, ソフトウェア分散開発

Software Collaborated Design Processes and Environment
for Effective Utilization of Group Members' Knowledge

Seiichi KOMIYA

Hikaru NISHINO

Software Technology Center
Information-Technology Promotion Agency, Japan (I.P.A.)
6F Shuwa-shibakoen 3-chome BLDG.
3-1-38, Shibakoen, Minato-ku, Tokyo 105, Japan

Software collaborated design processes need a framework and development environment for effective utilization of group members' knowledge. It is not able to build such a framework and environment without the fact that software collaborated design processes are regarded as group work. It is in contrast with the fact that software design processes are regarded as individual work, and have been supported through design methods and through the mechanical support based on a design method in software engineering. This paper describes that application of 'the Kepner-Tregoe method' is effective for rational group decision making in software design processes.

Key Words: Software Collaborated Design Processes, Rational Decision, Kepner-Tregoe Method,
Non-divergent Discussion, Software Maintenance, Software Distributed Development

*1 朝日立製作所より出向中

*2 日立ソフトウェアエンジニアリング(株)より出向中

1. はじめに

ソフトウェアの開発では、開発対象が大規模化・複雑化し、大量の作業人員で開発することが普通になってきている。そして、開発が終了すればそのプロジェクトは解散し、大部分のメンバーは別のプロジェクトに移るか本来の自分の職場に戻ることになる。このとき、顧客からの問い合わせやソフトウェア保守などの業務をごく少数のメンバーで遂行しなければならない。ここで問題になるのは、顧客からの問い合わせや保守の取りまとめを行う人間（通常は開発時のプロジェクト・リーダー）は、開発されたソフトウェア全体を必ずしも理解しているわけではないということである。このため、プロジェクト・リーダーは顧客からの保守の依頼があれば、分散したプロジェクトの旧メンバーと協調して保守作業を行う必要がある。すなわち、保守作業においては、1つのソフトウェアシステムを互いに遠く離れた作業者が共同で保守・開発する形態（分散開発）を採らざるを得なくなっている。特に、設計を行う上流工程では設計上の意思決定のために、分散した設計作業仲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。このような設計過程では、どのようなすればメンバーの知見をうまく引き出せるか、そして、得られた意見を基にどのようなすれば最適な案へとまとめられるかということが重要な課題となる。言い換えれば、メンバーの知見を生かしつつ、グループとして最善の意思決定をするにはどのようなすればよいかということが、協調型設計過程の重要課題である。このことはソフトウェア設計過程の問題をグループウェアの問題として扱うことを意味する。ソフトウェア工学では、これまでこの問題をソフトウェア設計方法論またはそれに基づく機械支援の問題として扱ってきた。これはソフトウェア設計過程を個人作業の問題として扱ってきたことを意味する。しかし、現実のソフトウェア設計（特に上流工程）はグループで行われることが多いので、グループウェアの問題として扱うことも必要である。従って、意思決定のための合理的な思考手順を与えることによって、分散した旧メンバーからの意見を効率的に引き出すとともに、メンバーからの意見を基に合意と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。本稿では、このようなシステムに求められる支援機能を明かにし、次に中でも特

にグループとしての意思決定を合理的に導く方式としてK T法（Kepner－Tregoe法）が有効であることを適用例を挙げて具体的に示す。

2. ソフトウェア協調型設計過程

2. 1 協調型設計過程への支援に求められる要件

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は次のとおりである。

(1) 支援に求められる要件

- 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。
- 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。
- 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。
- ④ 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（集約型の議論ができる）こと。
- ⑤ 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。
- 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。

(2) 導入する技術と効果

それぞれの要件を満足させるために導入する技術および効果は次のとおりである。

① に対して

K T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。その結果、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くなる。

② に対して

K T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ることができる。その結果、問題解決のための様々な視点が提示される。

③ に対して

K T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 に対して

K T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思考手順が与えられ、集約型の議論ができるので、決定までの時間を短縮できる。

⑤に対して

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。

⑥に対して

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のために、過去の経験や教智を生かすことができる。

2. 2 想定する協調型設計過程

本稿では次のような場面を想定する。すなわち、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リーダーと交換する形で作業を進めて行く。ソフトウェア開発工程としては、分散した作業場所の作業者がソフトウェアを保守する場合に、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。このとき、このチームのリーダーが、メンバーの意見を抽出しながら、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のために、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。

⑩①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。

⑩②設計項目を個別に検討可能な項目と同時に検討すべき項目とに振り分ける。

⑩③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、作業を依頼する。

⑩⑤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ーに提出する。

⑥リーダーは、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、メンバーに回覧する。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戻却し、再検討を促す。(⑤に戻る)

⑩⑦メンバーは、回覧されている設計試案に対して意見(または対案)があれば、コメント票の形で提出する。

⑧リーダーは、入手したコメントをもとに意見調整を

行う。リーダーは、必要があれば、設計試案の作成者に対してコメントの吸収を促すか、または、コメント提出者に対してコメントの却下を促す。

(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)

⑩⑨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。

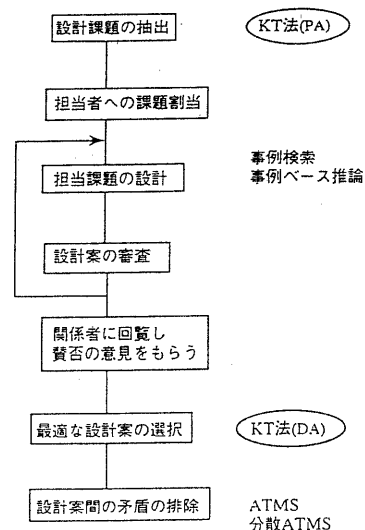
⑩⑩コメントの吸収を指示された担当者は、設計試案を再検討し、設計私案としてこれを再提出する。(再提出の後は⑥に戻る)

⑪設計項目全部の検討が終わるまで⑤~⑩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の候補を選出する。

(4)全設計案の間に矛盾があれば、その矛盾した設計案をリーダーは排除し、その設計案を提出したメンバーにその旨連絡する。

(5)排除を指示されたメンバーは、リーダーの指示に従い設計案を再検討し再提出する(⑥に戻る)か、あるいは無条件に棄却する。



(改良型) IBIS上に記録される

図1 ソフトウェア協調型設計の設計プロセス

また、前述の要件を、協調型設計過程の設計プロセスの中のコンピュータによる支援が必要なフェーズに

あてはめると、図1に示す様になる。本稿では、これらの要件のうち特に行動(1)と(3)へのKT法の適用について述べる。

3. KT法による合理的思考の実現と合意形成

KT法(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最適な手段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し、対策を準備するための思考手順である「潜在的な問題分析」(Potential Problem Analysis)、何が起き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

KT法の「問題分析」は、現在起きている問題の原因を究明するための思考手順である。従って、ソフトウェアの保守過程においては、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KT法の「問題分析」が適用可能となる。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。

図2に、ある病院システムの会計業務で起きた「画面入出力レスポンスの遅れ」という問題に対して「問題分析」を適用した例を示す。この例に従って、以下にKT法の「問題分析」による思考手順を示す。

問題分析(PA)				
差異ステートメント：画面入出力レスポンスが遅くなったことの原因追及				
	IS	IS NOT	区別点	区別点に関する変化
WHAT (WHO)	対象 会計業務の画面入出力レスポンス 欠陥 5秒かかる	会計業務以外では変化なし	会計業務はサブシステム送信用ファイルを作成している	1月10日に会計業務を改造し入替え(サブシステム送信用ファイル作成処理追加) ㉑
WHERE	場所 XX病院 対象の部分 病院情報システムの会計業務	YY病院 ㉒ 病院情報システムの会計業務	XX病院のみ会計業務の追加機能あり	
WHEN	日時 1月20日からレスポンス遅れが生じた 場合 会計処理終了時 ㉓	それ以前はレスポンス遅れはない	1月20日以後、かぜによる患者が急増した	
EXTENT	数量 2秒 → 5秒 傾向 ↗			
想定原因の列挙		IS/IS NOTによるテスト	最も可能性の高い原因	異 付 け
1.新規追加したサブシステム送信用ファイル作成処理自体が遅い 2.ホスト能力不足(トランザクション増に耐えられない) 3.ファイル作成時、患者数が増えると処理が遅くなる処理方式になっている		a, b が矛盾 X c が説明不能 X ○	ファイル作成時、患者数が増えると遅くなる処理方式になっている(常にファイルの先頭からスキャンし、マージを行っている)	

図2 問題分析例

- 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。
・「差異ステートメント」に「画面入出力レスポンスが遅くなったことの原因追求」と記述する。
- 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/欠陥、発生場所/

- 対象の部分、日時/場合、程度という4つの視点から明らかにすることにより、問題の明細化を行う。
- ・「IS」の欄に
問題の対象：「会計業務の画面入出力レスポンス」が「1. 5秒かかる」
発生場所：「XX病院」の「病院情報システムの

会計業務」

日時 : 「会計処理終了時」に「1月20日から遅れが生じた」
程度 : 「0.5秒から1.5秒へ増加」し、
「ますます増加しつつある」

という事実を記入する。

- ③②で起きているくらいなら起きてもよさそうなのに
実際は起きてはいない事実を、問題の対象／欠陥、
発生場所／対象の部分、日時／場合、程度という4
つの視点から明確にする。

・「IS NOT」の欄に

問題の対象 : 「会計業務以外では変化なし」
発生場所 : 「YY病院」の「病院情報システムの
会計業務」では起きていない
日時 : 「1月20日以前はレスポンス遅れは
ない」

程度 : 記載事項なし

という事実を記入する。

- ④「IS」と「IS NOT」のデータを基に、「IS」という
データだけを特に性格づけている特徴を、問題の対
象／欠陥、発生場所／対象の部分、日時／場合、程
度という4つの視点から明確にする。

・「区別点」の欄に

問題の対象 : 会計業務だけに見られる特徴は、「会
計業務はサブシステム送信用ファイルを作
成している」

発生場所 : XX病院だけに見られる特徴は、「X
X病院の病院情報システムの会計業務
だけが追加機能がある」

日時 : 1月20日という日時に見られる特徴
は、「1月20日以降、かぜによる患
者が急増した」

程度 : 記載事項なし

という特徴を記入する。

- ⑤それぞれの「区別点」を精査し、それらの「区別点」
が変化を示しているかどうか明確にする。

・「区別点に関する変化」の欄に

問題の対象について、1月10日に会計業務を改
造し、入れ替えをしている（サブシステム送信用
ファイル作成処理の追加）ということが判明した
を記載する。

- ⑥「区別点」と「区別点に関する変化」を基に、考え
得る原因を想定する。

・機能を追加したXX病院だけで発生していることか

ら考えて、「新規追加したサブシステム送信用ファ
イル作成処理自体が遅い」という原因が考えられる。

・遅れが生じた日時から患者が急増していることから
考えて、「ホストの能力不足（トランザクション増
に耐えられない）」という原因が考えられる。

・レスポンスの遅れがますます増加していることから
考えて、「ファイル作成処理時、患者数が多くなる
ほどファイル作成処理が遅くなる」という原因が考
えられる。

- ⑦「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「IS」および「IS NOT」
を全て説明できるかのテストを行う。

・想定原因の1番目については、プログラムを入れ換
えた日時と実際に遅れが生じた日時とにずれがあり、
真の原因とは考えられない。

・想定原因の2番目については、同一のホストがXX
病院とYY病院で稼働しており、さらにかぜによる
患者の急増はYY病院でも起きているため、YY病
院で発生していないのは説明がつかない。

・想定原因の3番目については、「IS」「IS NOT」全
ての事実が矛盾なく説明できる。（最も可能性の高
い原因と考えられる）

- ⑧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。

・ファイル作成処理は、常にファイルの先頭からシー
ケンシャルに同一患者の検索を行ってマージしてい
るため、新規来院患者が増えれば増えるほど処理時
間がかかる、ということがわかった。

以上の分析手順から導かれる設計課題は「患者増によ
る処理能力低下対策」となる。

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
らかであろう。

3. 2 KT法（決定分析）による設計案の選出

KT法の「決定分析」は、「問題分析」によって導
かれた1つの設計課題に対して、提案された複数の設
計案の中から最適な設計案を決定するために用いる思
考手順である。この「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

案 3 : ファイルをインデックスファイルに変換し、会計業務にてダイレクトアクセスによって検索・マージを行う(インデックスファイル)

という3つの案が出されたものと仮定している。

とした。

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。

決定ステートメント：患者増による処理能力低下対策案の決定

目 標		案：ファイル作成パッチ化		案：一部パッチ化		案：インデックスファル	
MUST		情 報	OK / NG	情 報	OK / NG	情 報	OK / NG
レスポンス１秒以内		可能	OK	可能	OK	可能	OK
ファイル作成ができること		可能	OK	可能	OK	可能	OK
WANT	重み W	情 報	スコア S W×S	情 報	スコア S W×S	情 報	スコア S W×S
レスポンス0.5秒以内	10	可能	10 100	可能かも	9 90	難しい	6 60
工数がかからない	9	複雑な処理	8 72	単純な処理	10 90	やや複雑	9 81
簡単な操作	6	面倒	9 54	面倒	9 54	従来通り	10 60
ファイル即時作成	5	不可能	0 0	不可能	0 0	従来通り	10 50
		TOTAL	226	TOTAL	234	TOTAL	251

- 70 -

なお、各案の希望目標に対するスコアが状況によって異なることがある。例えば、「開発工数がかからないこと」という評価項目を考えてみると、デバッグ時に使用するツールあるいはマシンなどの開発環境がまだ決定できない状況の場合、デバッグ時の開発環境によっては工数が変わる可能性がある。このように、不確定性を伴う意思決定を行わなければならない場合には、意思決定のための基準として、ラプラスの基準[4]、マキシミンの基準[4]、フルビッツの基準[4]、ミニマックスの基準[4]の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。

マキシミンの基準とは、悲観的立場をとった選択基準である。すなわち、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。マキシミン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。

3.3 将来の変化予測に基づき最適案の決定

ソフトウェアというものは、将来にもわたって永久的なものではなく、環境の変化等に応じて仕様変更が発生するのが一般的である。従って最適設計案を決定するにあたっては、将来予想される仕様変更に対する影響度を考慮しなければならない。KT法の「決定分析」だけでは、最適設計案の選択はできないのである。すなわち、最適設計案とは、「決定分析」での加重加算点数が高かつ予想される将来の仕様変更に対する影響度が小さい設計案であるといえる。そこで本稿では、前述したKT法の「決定分析」を適用した後に、独自に「将来の変化に対する影響度」を明らかにするプロセスを追加し、この2つのプロセスを基に最適案の選出を行うこととした。

「将来の変化に対する影響度」を考える上で必要になるのが、

- ・その変化の起きる可能性 (P)
- ・その変化が起きた時の作業量 (W)

である。ここで、「変化が起きる可能性」は、10 (可能性大) から1 (可能性小) までの尺度を用い、「変化が起きた時の作業量」は、10 (作業量大) から0 (作業量なし) までの尺度を用いる。従って、設計案に対する「将来の変化に対する影響度」は以下の式で求められる。

$$\text{影響度: } I = P \times W$$

この式で求めた影響度を基にして候補案の順位付けを行うには、予想される各変化に対する影響度を合計し、そのTOTAL影響度の低い順に順位付けすればよい。しかし、「決定分析」と「将来の変化に対する影響度分析」の2つの結果から最適設計案を合理的に導き出すには2つのプロセスでの結果 (TOTAL) を何らかの方法で1つの評価基準に統一したほうがよい。というのは、2つのプロセスでの評価基準は、「TOTALの高いものを選ぶ」と「TOTALの低いものを選ぶ」であり評価基準が異なっているためである。そこで同一の評価基準にするため、「将来の変化に対する影響度分析」プロセスでは影響度の代わりに影響を受けない度合い、すなわち「安全度」を導入し、2つのプロセスの結果 (TOTAL) を合算し評価することとした。この安全度は次式で求められる。

$$\text{安全度: } S = P \times (10 - W)$$

図4で、図3において明らかになった候補案それぞれに対して将来考えられる変化 (仕様変更) の例をあげ、影響度および安全度を求めてみた。この結果、影響度の合計の低い順 (安全度の高い順) に案2 (一部バッチ化)、案3 (インデックスファイル)、案1 (ファイル作成バッチ化) が得られる。従って、図3の「TOTAL点数」と図4の「TOTAL安全度」の合計を求めれば、

①案1: 「ファイル作成バッチ化」

$$\begin{aligned} \text{TOTAL点数} + \text{TOTAL安全度} \\ = 226 + 46 = 272 \end{aligned}$$

②案2: 「一部バッチ化」

$$\begin{aligned} \text{TOTAL点数} + \text{TOTAL安全度} \\ = 234 + 61 = 295 \end{aligned}$$

③案3: 「インデックスファイル」

$$\begin{aligned} \text{TOTAL点数} + \text{TOTAL安全度} \\ = 251 + 50 = 301 \end{aligned}$$

となり、案3: 「インデックスファイル」が最適案であるということが、合理的に得られる。

将来の変化	可能性 P	安全度(S)・影響度(I)					
		案1		案2		案3	
		W	S/I	W	S/I	W	S/I
ファイル作成 がなくなる	5	2	40 10	5	25 25	6	20 30
会計処理変更 (法律の改正)	10	10	0 100	7	30 70	7	30 70
OSの変更 (UNIXへ)	3	8	6 24	8	6 24	10	0 30
小計 S I			46 134		61 119		50 130
意思決定基準 適用結果			226		234		251
総計 S I			272 360		295 353		301 381

図4 各設計案に対する影響度

なお、変化の起きる可能性および変化が起きた時の作業量に対してメンバー全員の合意が得られなかった場合には、メンバー全員の与えた可能性や作業量の値の平均を採用するなどの工夫が必要である。

3. 2節および本節で示した手順は、2章で列举した、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グループの意思決定プロセスを実現するために有効であることは明らかであろう。

4. おわりに

ソフトウェアの大規模化・複雑化に伴い、人材と作業スペースの確保のために、互いに遠く離れた作業場所の作業者が1つのソフトウェアを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。このような開発形態における上流工程の作業は、複数の人間により協調的に進められなければならない。故に、このような設計過程を協調型設計過程と呼び、このような開発形態をソフトウェア分散協調開発と呼ぶ。このような設計過程では、メンバーの知見を最大限に生かしてソフトウェアの設計が行えるような枠組みが必要となる。言い替えれば、メンバーの知見を引き出しつつ、得られた意見を基にグループとしての最善の意思決定を可能にするような枠組みが必要である。このような枠組みは、ソフトウェア設計作業をグループワークとして捉えることにより初めて可能となる。このことは、これまでソフトウェア工学が、ソフトウェア設計作業を個人作業として捉え、設計方法論またはこれに基づく機械支援という形で支援してきたことと対照をなす。

本稿では、まず最初に、ソフトウェア分散協調開発に適した典型的な例として、プロジェクト解散後におけるソフトウェアの保守作業を挙げ、そこでの設計作業は協調的に進められなければならないことを示した。そして、プロジェクト解散後におけるソフトウェアの保守作業を例として、協調型設計過程において、グループとしての意思決定を合理的に行うための枠組みとしてKT法(Kepner-Tregoe法)の適用が有効であることを示した。

現在、KT法の有効性を実証するために、システムの試作中である。また、本稿でも述べたようにKT法の「問題分析」手順はシステムの保守時には設計課題抽出手順としては有効であるが、新規設計のための要求抽出(=設計課題抽出)手法として、これをそのまま使用するのは困難である。そこで今後は、新規作成システムでの要求抽出過程にも適用できるように「問題分析」手順を改良していく予定である。

[参考文献]

- [1]Conklin, J. and Begeman, M. L.: gIBIS: A Hypertext Tool for Exploratory Policy Discussion, CS CW '88 Proceedings, ACM, pp.140-152(1988).
- [2]Kepner, C.H. and Tregoe, B.B.: The Rational Manager, Princeton Research Press, 1965.
- [3]Kepner, C.H., Tregoe, B.B.: The New Rational Manager, Princeton Research Press, 1981.
- [4]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).
- [5]Lee, J.: SIBYL: A Tool for Managing Group Decision Rationale, CSCW '90 Proceedings, ACM pp. 79-92(1990).
- [6]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).
- [7]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).
- [8]Maclean, A., Young, R. M., Bellotti, V. M. E., & Moran, T. P.: Question, Options, and Criteria: Elements of design space analysis, Human-Computer Interaction(1991).
- [9]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).
- [10]Potts, C.: A Generic Model for Representing Design Methods, Proc. of the 11th International Conference on Software Engineering, IEEE, pp.217-226(1989).
- [11]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).