1J−3　ソフトウェア分散協調開発の為の協調型設計過程支援アルゴリズム

古宮誠一
情報処理振興事業協会　技術センター

1．はじめに

ソフトウェアの大規模化・複雑化に伴い、作業場所を一ヶ所に集中して開発することが困難になってきた。このため、一つのソフトウェア・システムを複数の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。しかし、その割に分散開発が浸透していないのは、支援ツールなしでは分散開発を実施できないからである。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確実に伝送するツールが不可欠である。しかし、それだけでは充分ではない。何故なら、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、しかも、分散開発においては、協調すべき作業者の作業場所が互いに遠く離れているからである。従って、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が必要である。従って、ソフトウェア分散開発を可能にするツールには、データ伝送機能と分散開発環境での協調過程支援機能の２つが必要である。

ところで、ソフトウェア開発工程から見れば、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。

2．ソフトウェア協調型設計過程と協調システムのモデル

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。故に、このようにソフトウェアの設計作業を協調して進める過程をソフトウェア協調型設計過程（software collaborated design process）と呼ぶ。

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。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。

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。

3．集中型制御モデルに基づくソフトウェア設計過程

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、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。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順番に列挙すれば次のとおりである。

◎①検討項目をリストアップする。
◎②検討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検討項目を検討すべき順序に並べ直す。
④検討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検討項目を検討し、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私案に対して意見が

Software Collaborated Design Process Support Algorithm for Software Distributed and Collaborated Development　by Seiichi KOMIYA
(Software Technology Center, Informatin-Technology Promotion Agency, Japan)

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検討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検討項目全部の検討が終わるまで⑤～⑪を繰り返す。

上記のうちで、コンピュータによる知的支援が可能だと考えられるのは、①②③⑨⑩（いずれも◎が付いている）の５つである。

４．ATMS[2]と分散ATMSによる協調過程のモデル化と支援

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、入手した設計試案とそれへのコメントを基に、リーダがメンバーを協調へ導く過程の計算モデルとして、ATMSおよび分散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および分散ATMSで使われているアルゴリズムが、メンバーの意見を協調へと導くための協調計算アルゴリズムとして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$M_1, M_2, \cdots, M_n$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介して行われる。

(3)各メンバーは、自分に割り当てられた設計項目の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。

(4)各メンバーは意見提出の際、提出者は自分が「真」か「偽」か、いずれの意見を提出しているのか明らかにしなければならない。提出される意見には、すべて提出者名と提出順番を示す番号からなるＩＤが自動的に付与される。

(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がある場合には、設計試案を幾つかの部分問題へと分解して、それぞれに対してYES/NOを明らかにしなければならない。

このとき、検討項目のそれぞれに対して寄せる各メンバーからの設計試案およびそれへのコメントのそれぞれは、各メンバーの持つ信念だと見なすことができる。同様に、各メンバーから寄せられる設計試案とそれに対するコメントの集合をリーダが調整して行く過程は、チームとしての信念の一貫性を維持する過程だと見なすことができる。故に、チームとしての信念の一貫性を維持するシステムとしてATMSが適用可能である。これをformalに記述すると次のようになる。

ｍ個の検討項目に対してメンバーM_j $(j = 1, 2, \cdots, n)$が持つ信念の集合を $\{B_{1j}, B_{2j}, B_{3j}, \cdots, B_{mj}\}$ とする。このとき、ATMS（＝リーダ）は、各メンバーから送られてくるｉ番目 $(i = 1, 2, \cdots, m)$ の検討項目に対する、チームとしての信念の集合 $\{B_{i1}, B_{i2}, B_{i3}, \cdots, B_{im}\}$ の一貫性を維持するために、矛盾の原因となる信念B_{ij}をチームとしての信念の集合から追い出すように振舞う。

(7)リーダは、チームとしての信念の一貫性を維持するためのシステムATMSを持っている。

(8)リーダは、１つの設計試案に対するメンバーからのコメントが複数の部分に分解されてYES/NOが述べられていたら、部分問題への分解の在り方を調整した後にATMSを適用する。

(9)ATMSは、メンバーからの意見（＝YES/NO）を信念の集合と見て、それらが無矛盾となるように振舞う。このときそれぞれの意見の提出者が誰であるかをシステムが管理しているので、取り除かれた意見の提出者が誰であるかを知ることができる。

(10)信念B_{ij}を取り除いた直後に、システムがユーザに制御を返すので、リーダは、信念B_{ij}をそのまま廃棄すべきか、修正後に再提出すべきかを判断の上、いずれか一方をメンバーM_jに指示することができる。取り除かれた意見を削除するには惜しいとリーダが判断した場合には、メンバーに再考を促すメッセージを提出し、設計私案を再提出させる。取り除かれた意見を削除するのが妥当とリーダが判断した場合には、意見の却下を提出者に通知する。

これに対してメンバーM_jはリーダの指示どおりの行動をとる。従って、もし、リーダの指示が信念B_{ij}の廃棄であれば、メンバーM_jは、自分の信念の集合 $\{B_{1j}, B_{2j}, B_{3j}, \cdots, B_{mj}\}$ から信念B_{ij}を無条件に削除しなければならない。これを契機に、メンバーM_jは、自分が持つ信念の集合の一貫性を維持しようと努める。このときのメンバーM_j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の集合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。

(11)メンバーは例外なく、自身の信念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。この折にも、各メンバーは自身の信念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。

５．おわりに

現在、ソフトウェア分散協調開発を可能にするための協調過程支援アルゴリズムとして、ATMSおよび分散ATMSが適用可能であり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。本稿では、ソフトウェア分散協調開発を可能にするための協調過程支援アルゴリズムとして、ATMSおよび分散ATMSが適用可能であることを示した。

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