ソフトウェア分散開発における ネゴシエーション支援の一方式

堀 雅和 †[1]、海谷 治彦 ‡、落水 浩一郎 ‡
† 富山商船高等専門学校
‡ 北陸先端科学技術大学院大学

我々は現在、ソフトウェア分散開発環境において、ネットワークを介した協調作業における情報共有のモデル化と、その実現機構についての研究を行っている。

本稿では、共有された情報の変更にともない発生する、開発者間のネゴシエーションを支援する機構を実現するための基本的要件を分析する。そして、この要件に従い、自律オブジェクトによるモデル化の例を示す。また、分散開発において仕様の変更が発生した場合を例にとり、このモデルによる支援の具体例を示す。

An Architecture to Support Negotiation for Distributed Software Development

Masakazu Hori†, Haruhiko Kaiya‡, Koichiro Ochimizu‡
†Toyama National College of Maritime Technology
‡Japan Advanced Institute of Science and Technology, Hokuriku.

We currently make a research to model and implement a mechanism to share information at the cooperative works in distributed software development environments.

In this paper, we analyse requisites for a mechanism to support negotiation in the case that a developer modifies shared information. We show a model by means of autonomous object to satisfy our proposed requisites for negotiation support. We also describe the way how to support negotiation for the modification of specification by the model.

[1] On leaving INTEC, inc.

1 はじめに

北陸先端大 落水研では、現在、 "複数の人間が、アイデアや中間成果物をネットワークを介して共有し、(グループウェアを利用した) 討論や (ソフトウェアプロセスに基づく) 変換活動によってそれらを変化させていく」ような協調作業を支援する CSCSD (Computer Supported Cooperative Software Development) モデルとそのプロトタイプシステム "自在」の開発を行っている [1, 2]。我々は、その中でも、各自の役割に応じた中間成果物の管理手段を分散作業空間としてモデル化し、共有空間の種々の管理機構を検討することにより、ネットワークを介した協調作業における情報共有のモデル化と実現機構を研究している [3, 4]。本稿では、共有された情報の変更にともなう関発者間のネゴシエーション支援について考察する。

一般にソフトウェア分散開発で発生するネゴシエーションは、以下のような特徴がある。

1. ネゴシエーションは、共有情報の内容に関係して発生する場合が多い。
2. ネゴシエーションは、定型的な手順に従って進められる一方、関係する人達からの反応や状況に応じて、次に採るべき手順を変更する場合も多い。
3. 地理的に分散した開発者間でネゴシエーションが進められるため、全体の状況を把握をすることが困難であるという問題が発生する。

以下、第二節において、ネゴシエーションを支援するための基本的要件を分析する。第三節では、ネゴシエーションの型について検討する。第四節では、ネゴシエーション支援のためのソフトウェアアーキテクチャを、自律オブジェクトに基づいてモデル化していくための具体的要件をまとめる。第五節では、分散開発において仕様の変更が発生した場合を例にとり、支援の具体例を示す。

2 基本的要件

ソフトウェア分散開発において発生するネゴシエーションを、支援する機構を実現するための基本的要件として、以下の三つをとりあげる。

1. 共有されている情報を、ネゴシエーション中に直接とり扱うことができる機構

   共有情報の内容に基づいたネゴシエーションでは、電子メールを用いた場合のように、共有情報へのアクセスとネゴシエーションを、それぞれ別々のチャネルを通じて行うことは、弊害が大きいと考えられる。これは、電子メールの問題点の一つである「ドキュメントの一貫性維持の困難さ」に起因する [5]。この問題は、電子メールを用いた場合に、引用されているドキュメント情報と実際のドキュメントとでは、その実体が異なるため、さまざまなバージョンが発生することになり、管理が困難であることを指す。そのため、ネゴシエーションのためのテキストメッセージを作成しているときに、共有情報の編集や引用を行う必要が発生した場合、その情報を直接とり扱える必要がある。

2. プロトコルに基づきネゴシエーションを行うとともに、そのプロトコルを動的に変更できる機構

   プロトコルとは、状態遷移図にて表現されるネゴシエーションの手順を意味するものとする。ネゴシエーション関係者とのメッセージのやりとりの手順、環境設定の方法、必要な情報の取得方法あるいは、取得した情報の加工法などが記述される必要がある。プロトコルの最小単位は、 "確認」、 "依頼」、 "要求」などのネゴシエーションの形態に対応しており、第 3 節で、ネゴシエーションの型として形式化する。プロトコルの動的変更については、型の内容の変更と、複数の型の連携方法の変更の二つの場合を考えている (第 4.2.2 節)。関連する研究としては、ルールに基づいた自律処理を可能とする Object Lens [6] の研究があげられるが、プロトコルの動的変更機構は、実現されていない。

3. 状況に依存して、ネゴシエーションを調整するための機構

   分散して作業している複数の開発者間で、ネゴシエーションを実施する場合、全体の状況をもとに問題点を検知し、その解消に向け調整を図る必要がでてくる。これには、とくにネゴシエーションの進行状況に問題のある開発者を検知し、進行が早まるように手助けすることなどが含ま

れる。そのため、他人のネゴシエーションの進行を制御したり、必要な情報を取得できる必要がある。

開発者間でとりかわされる討議内容を構造化して保持し (グループウェアベース)、その状態により進行を調整するモデルと機構の開発を進めているが [7]、本研究では、とくにネゴシエーション支援について絞りこんだ考察を行う。

3 ネゴシエーションの型

第 2 節で述べたネゴシエーションプロトコルを実現するため、ネゴシエーションの型について検討する。ここで扱うネゴシエーションの型は、プロトコルの最少単位に対応し、第 4 節にて、実装のためのモデル化を、エージェントを用いて行う。ネゴシエーションの型を以下の 3 項組で表す。

$$\langle P_{urpose}, O_{rder}, D_{ecision} \rangle$$

P_{urpose}は、ネゴシエーションの 「目的」を表わし、その 「目的を達成するために必要な情報」と 「締切」とともに指定する。目的とは、例えば、確認、依頼、要求、督促、問い合わせなどがある。目的の達成に必要な情報とは、例えば確認を例にとると、確認すべき内容を記した情報を指す。締切とは、ネゴシエーションが終了すべき日時を指す。P_{urpose}の指定により、目的を実現するための手順も同時に規定する。具体的には、 「状態遷移図」および、 「ネゴシエーション終了後の回答として、選択可能なものの集合」を規定する。回答の集合に属する要素としては、例えば、承認、拒否、不明などがある。

O_{rder}は、ネゴシエーションの 「対象者」間の 「順序」を指定する。例えば、指定順に従って行う、(非) 同期的に各自別々に行う、階層的に行うなどがある。また、対象者は、具体的な氏名あるいは、役割にて表現する。役割としては、例えば、同一グループメンバー、リーダ/メンバー、発注元/請負先、ライブラリアン、設計者/プログラマなどがある。

$D_{ecision}$は、決定事項がある場合の決定方法を表わす。例えば、全員のコンセンサスを得る、多数決を行う、一番よい案を選定するなどがある。

この 3 項組を用いて、例えば、仕様変更の確認を行う場合を表現すると、次のようになる。

⟨ 仕様変更の確認, 非同期的に開発者, 決定事項なし ⟩

4 エージェント

4.1 概要

第 3 節で検討したネゴシエーションの型におけるP_{urpose}に対応する、ネゴシエーションエージェント (以下、エージェントと呼ぶ) の概念を導入する。このエージェントとは、分散人工知能の研究における 「エージェント」[8, 9] の自律性をその性質として備えつつ、ある種類の決定事項については、関連する開発者に決定を依頼し、その返事に基づいて処理を継続する性質を持つ。また、このエージェントは、第 2 節で分析した基本的要件のうち、プロトコルの動的変更や、調整のための機構を自然に実現する目的で、メタレベル・アーキテクチャ [10, 11] を利用している。ベース、メタそれぞれのレベルにおいて、以下のように異なる役割を果す (図 1 参照)。

ベースエージェントは、 「ネゴシエーション中の交信履歴」、 「ターゲット開発者名」などを管理する。ネゴシエーション中の交信履歴とは、開発者間に発生したメッセージのやりとりを指す。またターゲット開発者名は、ネゴシエーションの支援をしている開発者の名前である。ベースエージェントの主な役割は、

- 開発者とのメッセージのやりとり
- 共有情報へのアクセス

である。

一方メタエージェントは、 「状態遷移図」、 「現在の状態」、 「ベースエージェントへのアクセス履歴」などを管理している。状態遷移図は、受け取ったメッセージにどのように反応するかを決定する。現在の状態とは、状態遷移図における、現在の状態を示している。またベースエージェントのアクセス履歴は、状態遷移図上をどのように遷移してきたかを示している。これらの管理情報を用いてメタエージェントは、

- プロトコルの変更 (状態遷移図の構造の変更や、メソッドの変更)
- 現在のネゴシエーション状態の取得

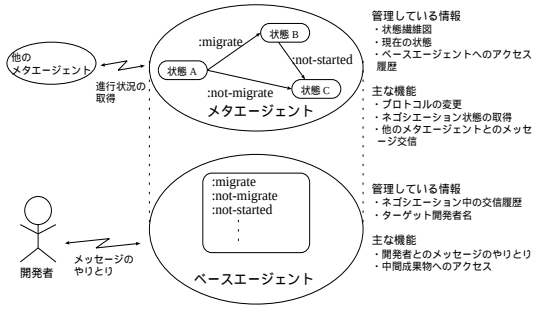

図 1: エージェントのメタレベル ・アーキテクチャ

- 他のメタエージェントとのメッセージ交信 (メタ情報の取得)

を行う。

まとめると、第 3 節で述べたネゴシエーションの型を表わす 3 項組と、エージェントの関係を述べる。P_{urpose} は、エージェントと一対一の関係にあり、メタエージェントが管理する状態遷移図を規定する。O_{rder} は、ベースエージェントが管理するターゲット開発者名を指定することに対応する。また、$D_{ecision}$ は、状態遷移図における決定のためのメソッドを規定することに対応する。

4.2 基本的要件を実現する機構

4.2.1 共有情報を直接とり扱うための機構

エージェントを利用したネゴシエーションでやりとりされるメッセージは、開発者が交信するために使用するテキストデータと、共有情報をアクセスするためのメッセージの二種類がある。ここで、共有情報をアクセスするためのメッセージとは、オブジェクト識別子と、メソッドにアクセスするためのメッセージの対からなる。これら二種類のメッセージを混在させて、ネゴシエーションでのメッセージのやりとりを行うことにより、共有情報を編集し、その結果を取りこむことを、同一のチャネルにて行う。

4.2.2 プロトコルの動的変更機構

動的にネゴシエーションプロトコルを変更する機構の必要性は、ネゴシエーションでは、「当初予想していた状況とは異なる状況がよく発生する」という仮定に基づいている。動的なプロトコル変更を実現するため、制御するレベルを次の二つに設定している。

1. エージェントが保有するプロトコルの変更
2. 複数の型のエージェントによる連携方法の変更

「エージェントが保有するプロトコルを変更」とは、メタレベルで管理している状態遷移図の構造の変更や、ベースレベルがそのインタフェースを提供するメソッドの変更に対応する。このような細かい粒度での変更は、ネゴシエーションの実行に必要となる環境の設定や、情報の取得の方法などを変更する場合が多いと考えている。

一方、「複数の型のエージェントによる連携」は、ネゴシエーション対象の問題が詳細化されていくにつれ、他の型のエージェントが必要になる場合に発生する。この場合、必要な時はいつでも必要なエージェントが生成できるとともに、それまでに発生した情報で必要な情報は、すべて引き渡せる必要がある。

4.2.3 ネゴシエーションの調整機構

ネゴシエーションを主導する役割の人は、ネゴシエーションの進行状況を把握するために、必要な情報を取得し、必要であれば各開発者のネゴシエーションの進行状況に応じて、制御を行う必要がある。そのため、メタエージェントで管理している現在の状態および、メソッドのアクセス履歴を利用することで、各々の進行状況が把握できる。これらの情報を用いることで、各エージェントの進行状況のみならず、不必要な繰り返しやトラブルへの遭遇等を、検知できる可能性がある。

5 例題: 仕様の変更の支援

5.1 仕様の変更の概要

C++ を開発言語として、地理的に離れている二つの組織で、ソフトウェアの分散開発を行っているものとする。設計終了後、作成された仕様書がネットワークを介してプログラム開発者に渡され、現在プログラム開発を行っている最中とする。このような状況において、プロジェクト全体で共用しているライブラリの仕様を一部変更することで、自分の開発がスムーズに進むと考えたある開発者は、その旨をリーダに伝え、リーダが関係者に仕様の変更のネゴシエーションを行う場合を取りあげる。

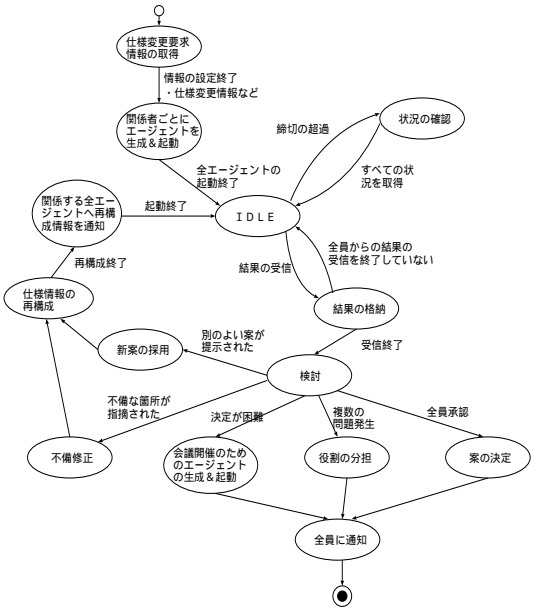

図 2: コーディネータの状態遷移図

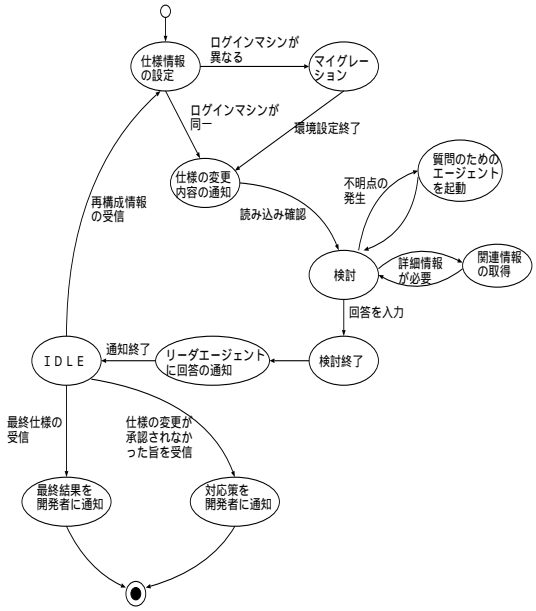

図 3: ネゴシエータの状態遷移図

5.2 エージェントが管理する状態遷移図

この例では、リーダがネゴシエーションをコーディネートし、開発者からのメッセージを収集するという役割を果す。以下、リーダ、開発者が利用するエージェントを、それぞれコーディネータ (図 2 参照)、ネゴシエータ (図 3 参照) と呼ぶことにする。

5.2.1 リーダにおける処理

コーディネータは、仕様の変更要求を受けて、必要な情報を設定したネゴシエータを、すべての関係者のために生成し、起動する。この例では、全員のコンセンサスを得ることを目的としているため、すべてのネゴシエータから結果を受信後、次の 「検討」の状態に遷移する。もし、全員が承認した場合は、その旨をすべてのネゴシエータに通知して終了する。それ以外は、リーダがコーディネータの受信した結果を見て検討を行い、次の方法を決定する。不備を指摘された場合は修正し、更によい案が提案された場合は、その案に基づき再度ネゴシエーションを繰り返す。また、別の問題が出てきた場合は、リーダの判断により、適当な人に割りあてたり、緊急のミーティングを開催するためのエージェントを起動して終了する。

5.2.2 開発者における処理

ネゴシエータは、コーディネータが存在するコンピュータ上で生成されるため、必要に応じて関係者がアクセスしているコンピュータにマイグレート (migrate) し、開発者に仕様の変更要求を通知する。その内容は、新旧の仕様、変更理由、変更の緊急度、締切などである。これらの情報をもとに、必要であれば、関連情報を検索したり、質問のためのエージェントを生成して、問い合わせを行う。そして、検討が終了したら、コーディネータに回答を通知する。

5.3 状況の変化への対応

5.3.1 動的ネゴシエーション支援

ネゴシエーションパターンの変更　ネゴシエーションパターンの変更は、当初の予想と異なる状況になった場合に発生する。例えば、コーディネータが結果の収集に予想以上に時間がかかったため、主要なネゴシエータから回答を得た時点で、検討を始めるように変更することが考えられる。これは、メタエージェントが管理する状態遷移図の遷移条件を変更することで実現できる。

情報収集法の変更　情報収集法の変更は、ネゴシエーションが進むにつれて、対象となる共有情報が変化していくことなどの理由により発生する。例えば、仕様の変更の影響を受けるモジュールの検索方法の変更を行うため、メソッドの変更することなどがこれに対応する。

連携処理　ある型のエージェントを用いたネゴシエーション中に、そのエージェントが管理しているプロトコルを変更するより、別の型のエージェントを起動したほうがよいと判断される場合に、連携処理が発生する。例えば、確認中に不明な点が発生した場合、質問のためのエージェントを生成して、不明な点を明らかにすることなどが考えられる。

5.3.2　ネゴシエーションの調整支援

状況取得　ネゴシエーションは、分散して行われているため、全体の状況を、メタエージェントが管理しているベースエージェントへのアクセス履歴を収集することで、再構成する。この情報を用いることで、最近のアクセス状況、プロトコルにおける現在の進行状況あるいは、不必要にループを繰り返しているなどの異常な状況などがわかる。

中断処理　ネゴシエーション途中で中断する必要が生じた場合、ベースエージェントへのアクセス履歴により、ネゴシエーションの進行状況を検知し、進行状況に応じてその中断方法を変えることができる。例えば、まだネゴシエーションを始めていなかった場合は、速やかに終了し、処理中であればその処理の終了まで延長し、必要であれば適当な後処理を行う。

6　まとめ

本稿では、ソフトウェア分散開発におけるネゴシエーションを支援するための一方式を提案した。

今後は、ネゴシエーションの型の分析をさらに行うとともに、今回提案したアーキテクチャを実装して、評価する必要があると考えている。

参考文献

[1] 落水: ソフトウェアプロセスに関する知識の獲得と形式化および協調支援モデルに関する研究. SDA/II プロジェクト 1991 年度 活動報告書, (1991), pp.1-4.

[2] 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. 信学技報, SS94-18, (1994-07), pp.1-8.

[3] 堀, 落水: 分散開発を支援するオブジェクト指向アーキテクチャの一提案. 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 97-7, (1994).

[4] 堀, 落水: グループオブジェクトによる分散開発支援環境の実現. オブジェクト指向コンピューティング III, (1995), 近代科学社.

[5] Goldberg, Y., Safran, M., and Shapio, E.: Active Mail - A Framework for Implementing Groupware. Proc. of CSCW '92, (1992), pp.75-83.

[6] Lai, K. Y., Malone, T. W., and Yu, K. C.: Object Lens: A "Spreadsheet" for Cooperative Work. ACM Tran. on Office Info. Syst., Vol.6, No.4, Oct. (1988), pp.332-353.

[7] 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, ソフトウェア工学 96-25, (1994).

[8] Durfee, E. H., and Montgomery, T. A.: Coordination as Distributed Search in a Hierarchical Behavior Space. IEEE Trans. on Syst., Man, and Cyber., Vol.21, No.6, Nov. (1991), pp.1363-1378.

[9] Zlotkin, G. and Rosenschein, J. S.: Negotiation and Task Sharing Among Autonomous Agents in Cooperative Domains. Proc. of IJCAI-89, (1989).

[10] Maes, P.: Concepts and experiments in computational reflection. Proceedings of OOPSLA '87, Vol.22, pp.147-155. SIGPLAN Notices, Oct. (1987), ACM Press.

[11] Watanabe, T., and Yonezawa, A.: Reflection in an Object-Oriented Concurrent Language. Proc. of OOPSLA '88 (1988), pp.306-315.