タスクの分割配分に基づいたリスク分散分析

齋藤 義人 松尾 徳朗
山形大学大学院理工学研究科

1 はじめに

近年，インターネットやコンピュータを利用する商取引が発達したため，様々な取引を簡単に行うことができるようになった [1]．コンピュータベースの商業は効果的な経済活動の1つであり，物品だけではなく，情報，時間，権利，およびタスクなども取引きすることができる．タスクを取引きする場合，受注企業が倒産することで，発注企業も連鎖的に倒産するというリスクが存在する．このようなリスクを軽減するため，本論文では分割できるタスクに焦点をあて，タスクを複数の企業に発注するリスク分散契約手法を提案する．

本論文では，分割できるタスクとして，大規模ソフトウェア開発を例に挙げている．一般に，大規模ソフトウェアは，複数のモジュールの集合で構成されている．そのため，各モジュールを別々の開発企業に発注することができる．また，全てのモジュールが完成した場合，1つのソフトウェアとして統合することもできる．開発企業に倒産や不渡り等のリスクがある場合，発注企業はリスクを軽減するために，どのような発注を行うか考えるべきである．効果的な発注戦略は，他のソフトウェア発注企業に勝つために必要である．

2 準備段階

2.1 モデル

本節では，提案手法およびシミュレーションに必要であるモデルと仮定について説明する．取引の関係者には，ソフトウェアを発注する企業とソフトウェアを開発する企業が存在する．特に，開発企業は複数存在し，発注企業は発注先を選択することができる．

- ソフトウェアの開発に関わる企業の集合を $D = \{d_1, ..., d_i, ..., d_n\}$ と定義する．
- 大規模ソフトウェアは，分割可能なモジュールの集合 $M = \{m_1, ..., m_j, ..., m_k\}$ から構成されると定義する．
- 開発企業 d_i がモジュール m_j を開発するために必要な費用を v_i^j と定義する．費用 v_i^j は，前金 p_{ij}^{pre} と成功報酬 p_{ij}^{post} に分割される．
- 開発企業 d_i の状態 A_i は，財務体質，管理態度，実績，およびその他のいくつかの要素からなると定義する．
- モジュールが完成しなかった場合，開発企業は前金の一部を返還料 p_{ij}^{ins} として返還する．

Assumption 1 (ソフトウェアの分割) 大規模ソフトウェアは複数のモジュールに分割できるものと仮定する．

Assumption 2 (開発者の数) モジュールの数 k よりも開発企業の数 n のほうが大きいと仮定する．

Assumption 3 (支払額) 支払いは2回にわけられ，タスクを発注したときに前金 p_{ij}^{pre} を支払い，タスクが完成したときに報酬 p_{ij}^{post} を支払うものと仮定する．

Assumption 4 (リスク) 開発企業 d_i が倒産するなどのリスク r_i は，$r_i = 1 - A_i$ で計算できるものと仮定する．

Assumption 5 (モジュールの統合) モジュールを統合するときに統合費用はかからないものとする。

ソフトウェアの発売や，使用開始期間があるため，モジュールを開発する企業は期間内に開発する必要がある．また，6つ目の仮定として，倒産で開発できなくなる場合を除き，作業をキャンセルしないものとする．

Analyses of Distributed Risk based on Divided Task Distribution
Yoshihito SAITO and Tokuro MATSUO
Graduate School of Science and Engineering, Yamagata University
e-mail : saito2007@e-activity.org

2.2 前金

タスクを発注した場合，発注企業は，開発企業が提示した資金を前金と成功報酬に分けて支払う．前金は割当てられたタスクを開発するインセンティブを増加させる．予定よりも開発期間が短い場合，または作成物の質が良い場合，発注企業が成功報酬を割り増しにして支払うことがある．しかし，本論文では成功報酬が増加することを考えず，前金と成功報酬の合計が，開発企業が提示した資金になる．つまり，$v_{ij} = p_{ij}^{pre} + p_{ij}^{post}$ である．また，前金は開発企業の状態を考慮して決定することとし，状態とコストの乗算と仮定する．つまり，$p_{ij}^{pre} = A_i \cdot v_{ij}$ である．

また，意図的な倒産から発注企業を守るために，タスクが完成しなかった場合，前金の一部を返還する制度を設ける．返還料 p_{ij}^{ins} は企業状態 A_i が低い程多く返還するように $p_{ij}^{ins} = (1 - A_i) \cdot p_{ij}^{pre}$ と定義する．

A_i は財務体質，管理態度，実績など様々な要素から計算されるため，実績が少ない開発企業や，倒産の危険性がある開発企業は A_i の値が低い．逆に A_i が高い開発企業は信頼できる開発企業である．このように開発企業を評価した場合，開発企業を以下の3つのパターンに分類することができると考えられる．

[CASE 1] 設立されてまもないため，実績がなく，企業状態 A_i が低い開発企業．
[CASE 2] 倒産寸前であるため，企業状態 A_i が低い開発企業．
[CASE 3] 確かな実績を持ち，企業状態 A_i が高い開発企業．

これらのような場合，発注企業が気をつけなければならない開発企業は，2つ目のパターンに当てはまる開発企業である．彼らには倒産の危険性がある上に，資金を得るために詐欺を行う可能性もある．3つ目のパターンに当てはまる開発企業は，十分な信頼があるため，信頼を損なってまで詐欺を行うことはない．1つ目のパターンに当てはまる開発企業は，開発に失敗する危険性があるが，後々のために，開発企業を育成するという考えを持っているのであれば，この開発企業に発注して損することはほとんどない．

3 リスク分散契約手法

本節では，契約の勝者を決定するための具体的なプロトコルを提案する．本論文では，開発企業が倒産したことによる発注企業の連鎖倒産，およびソフトウェア開発期間の延長によるソフトウェア発売の延期などといった発注企業が被るリスクの発生を防ぐために，大規模ソフトウェアを分割発注することでリスクを分散する発注手法を提案する．開発企業が必要な費用を提示する場として，本プロトコルにおいては組合せオークションを用いる [2]．ソフトウェアは以下の基礎プロトコルおよび3つの戦略に基づいて開発される．

基礎プロトコル

1. 発注企業は，大規模ソフトウェアをいくつかのモジュールに分割し，モジュール毎に開発企業を公募する．
2. 発注企業と契約することができる開発企業が，封緘入札オークションで各モジュールを開発するために必要な費用 $\{v_{i1}, \ldots, v_{ij}, \ldots, v_{ik}\}$ を提示する．
3. 発注企業は，開発企業の状態 A_i を調査する．
4. 発注企業は，提示された費用と開発企業の状態から前金を決定する．つまり $p_{ij}^{pre} = A_i \cdot v_{ij}$ を計算する．
5. 発注企業は，前金と自社が選択した戦略に沿ってタスクのセットを発注し，発注先の開発企業に前金を支払う．
6. 開発企業は，それぞれが受注したモジュールを開発する．モジュールが完成した場合，発注企業は開発企業に成功報酬を支払う．開発途中で開発企業が倒産してしまった場合，発注企業は完成しなかったモジュールに関して再び開発企業を公募する．
7. 全てのモジュールが完成した場合，それらを統合し1つのソフトウェアとする．

Strategy 1 (集中型戦略) 図 *1* は集中型戦略を示している．この戦略は全てのモジュールを1つの企業に発注する戦略である．つまり，大規模ソフトウェアを分割しないで発注する戦略である．

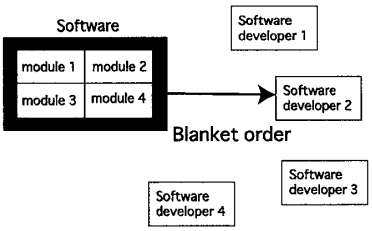

図 1: blanket order strategy

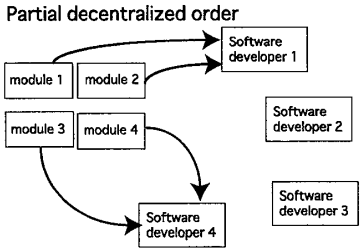

図 2: partial decentralized order strategy

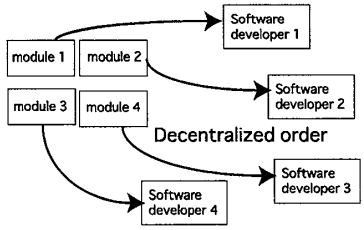

図 3: decentralized order strategy

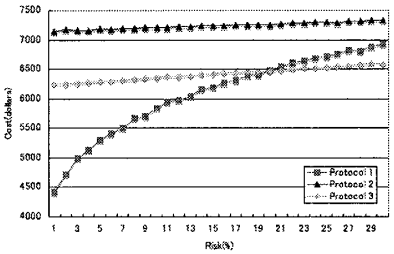

図 4: 3 modules and 5 developers

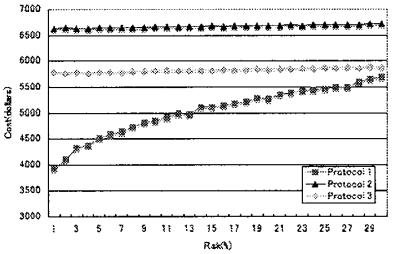

図 5: 3 modules and 10 developers

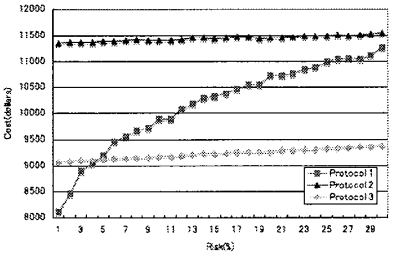

図 6: 5 modules and 8 developers

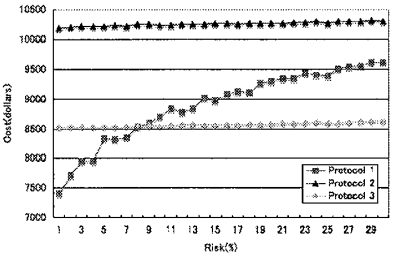

図 7: 5 modules and 16 developers

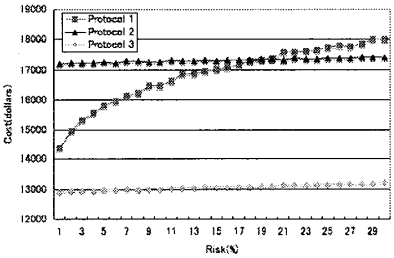

図 8: 8 modules and 12 developers

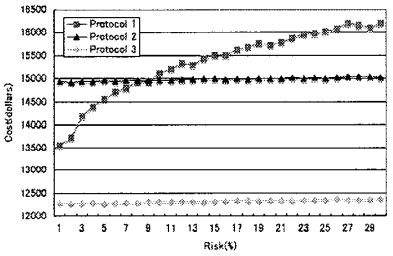

図 9: 8 modules and 24 developers

Strategy 2 (分割型戦略) 図 *2* は分割型戦略を示している．この戦略はモジュール毎に発注する戦略である．このとき，1 つの開発企業が 2 つ以上のモジュールを開発しても良い．

Strategy 3 (完全分割型戦略) 図 *3* は完全分割型戦略を示している．この戦略は，分割型戦略と同じくモジュール毎に発注する戦略であるが，この戦略では，1 つの開発企業が 2 つ以上のモジュールを開発することを禁止している．つまり，全てのモジュールを別々の開発企業に発注する戦略である．

基礎プロトコルにおいて，分割型，および完全分割型の戦略を用いた場合が本論文で提案するリスク分散契約手法となる．また，次章では本提案手法が効果的であることを示すために，従来の手法である集中型戦略をふまえて分析を行う．

4 分析

本章では，提案した分割型の戦略が発注企業にとって有用であることを示すため，リスクの割合と価格に関するシミュレーションを行う．開発企業とタスクの数を変更しシミュレーションを行った結果を図 4 から図 9 に示す．各結果の縦軸は支払った費用を示し，横軸はリスクの割合を示している．開発企業が 1 つのモジュールを開発する際に必要とする費用は 1000 から 4000 の間の一様分布で決められ，倒産率は 1 %から 30%の間で変化する．また，結果は，同じ条件のもとで 100000 回実験し，平均費用を示している．これらの実験結果および戦略の特徴から以下のことが考察できる．

- 集中型戦略は，開発企業の倒産率が高くなればなるほど，またソフトウェアの規模が大きくなればなるほど不利な戦略となる．単一の開発企業に発注することが有効である場合もあるが，この場合，開発企業が倒産した場合，ソフトウェアを最初から作成し直す必要があるため，発注企業が連鎖倒産のリスクを負いたくないのであれば，費用はかかったとしても信頼できる開発企業に発注する必要がある．
- 完全分割型戦略は，ソフトウェアの規模が大きくなればなるほど有利な戦略となる．また，費用がかからなくなるだけではなく，全てのモジュールを並行して開発することができるため，時間もかからなくなると予想される．加えて，発注先の開発企業が全て倒産しない限りソフトウェアを最初から作成し直す必要がないため，連鎖倒産の確率が軽減される．
- モジュールの数を変えずに開発企業の数を増やした場合，どの戦略においても安く開発することができる．これは，参加する開発企業の数が増えたことで，より良い条件の開発企業を選択することができたためと考察できる．

5 議論とまとめ

本論文で提案したリスク分散契約手法は，より大規模なソフトウェアを作成する場合に有利になる．本手法の利点は，発注先の企業が 1 つ倒産したとしても，最初からソフトウェアを作成し直す必要がないこと，モジュール全てを並行して開発することができるため，単一の企業に発注したときよりも時間がかからないことなどである．

本論文で行ったシミュレーションでは，モジュールのサイズを一定とし，ソフトウェアの規模を大きくすることでモジュールの数を増やした．しかし現実には，ソフトウェアの規模が一定で，モジュールのサイズを変化させることでモジュールの数を増やす場合もある．このような場合においてもタスク分割発注手法が有利であるか調査する必要がある．また，モジュールの数よりも開発企業の数が少ない場合，つまり完全分割型戦略を用いることができない場合も分析する必要がある．

本論文では，分割できるタスクに焦点をあて，リスク分散手法を提案した．また，提案手法の有用性を調査するために，タスクの分割数，受注企業数，および企業の倒産率を変化させ，発注企業の負担費用がどの程度であるかを調べるシミュレーションを行った．本提案手法は，タスクの規模が大きくなった場合において，タスク発注企業が負担する費用を削減することができる．また，分割されたタスクを並行して行うことができるため，全てのタスクが終わるまでの時間も削減することができる．加えて，タスク受注企業が倒産したとしても，タスクを最初から行い直す必要がないため，タスク発注企業が連鎖倒産するリスクは低い．

参考文献

[1] P.R. Wurman, M.P. Wellman, and W.E. Walsh, "The Michigan internet auctionbot: A configurable auction server for human and software agents", In Proc. of the 2nd International Conference on Autonomous Agents (AGENTS98), 1998.

[2] B. Hudson and T. Sandholm, "Effectiveness of preference elicitation in combinatorial auctions". In Proc. of AAMAS02 Workshop on Agent Mediated Electronic Commerce IV (AMEC IV), 2002.