Vol.2015-SE-187 No.6
2015/3/12

ゲーム理論による開発者特性を考慮したチームビルディング

山﨑 尚[1]　五田 篤志[2]　玉田 春昭[1]　畑 秀明[3]　角田 雅照[4]　井垣 宏[5]

概要：ソフトウェアの開発では，複数の開発者がチームを組み開発を行っていく．従来のチーム編成は管理者が開発者のプロジェクト経験やプログラミング言語の経験，所有資格などを加味して行う．しかし，これは管理者の勘や経験に依存し，開発者の長所を上手く活かせられているとは言い難い．また，ソフトウェア開発を行うプロジェクトによって必要な人材の能力というのは異なる．従って，プロジェクトにあった特性を持つ開発者を合理的に割当てることができれば，プロジェクトごとに割当てる人的資源を最適化でき，かつ，開発者の長所を活かすことが可能である．本研究の目的は，開発者の合理的な割当てのために，ゲーム理論に基づいてチーム編成を行う．本稿では，大学院生を対象とした PBL 合宿に焦点をあて，そこで得られた開発履歴を用いてチームを作成する．

1. はじめに

ソフトウェア開発における生産性の向上は様々な方法がある．生産性を向上させるための方法として，開発スケジュールや開発環境の整備，開発チームの効率化などが挙げられる．一般的に生産性と言うと，入力に対してどれだけ出力できるかである．ソフトウェア開発などの製造業では QCD という指標がある．QCD は Quality, Cost, Delivery の頭文字を取った略語であり，それぞれ品質，コスト，納期を示す．品質は高く，コストは低く，納期は短くすることを目指すことで生産性を向上するとされている．本稿では，コストとして考えやすい人的資源を如何に割当てるかで生産性向上を目指す開発チームの効率化に焦点を当てる．

一般的に，ソフトウェア開発において，チームを決めるのはプロジェクトマネージャなどの管理者である．管理者が，開発者たちのプロジェクト経験やプログラミング言語の経験など，それまでの開発者の経歴と，所有資格などを加味してチーム編成を行う．しかし，従来のチーム編成は勘と経験に大いに頼っており，開発者それぞれの得意・不得意を踏まえたチーム編成ができているか検証されることはない．そして，結果として得られたチーム編成には，合理的な裏付けがない．

そこで，合理的なチーム編成をするために，開発者の能力を客観的な指標に基づいて識別し，そこから得られたデータを元に合理的にチーム編成をしていく．そもそも人は，無意識下に自分の利益を自分のルールに基づいて追求するように行動を選択している．その行動の選択を戦略的意思決定の理論であるゲーム理論を用いることで再現する．これにより，開発者の割当てという行動に合理的な意味を与える．また，その割当てに用いるデータとして，開発者の実際の行動によって決まる開発履歴を用いる．このことにより，実際に人が開発者をチームに割当てる場合に比べ，開発者の特性を考慮できると考えられる．

さらに，今回提案する手法は経済学や経営学，最近では生物学やコンピュータ科学という広い範囲で応用されているゲーム理論を基にしているため，ソフトウェア開発に留まらず様々な分野での人材割当てに応用できると考えられる．

2. 関連研究

鴻巣らは，Big Five Model を用いたパーソナリティ得点を利用するチームビルディング手法を提案した [1]．Big Five Model は，米国国防機関により，組織の生産性を向上させることを目的として創られた因子分析理論である FFS

[1] 京都産業大学 コンピュータ理工学部
Faculty of Computer Science and Engineering, Kyoto Sangyo University, Kyoto, Japan
[2] 京都産業大学大学院 先端情報学研究科
Division of Frontier Informatics, Graduate school of Kyoto Sangyo University , Kyoto, Japan
[3] 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan
[4] 近畿大学 理工学部
Department of Informatics, Kindai Univesity, Osaka, Japan
[5] 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Osaka, Japan

理論を拡張したものである．鴻巣らは，Big Five Model および林の印象評定尺度による対人評価を基に，チームビルディングを行った．鴻巣らは，客観的な評価をメンバに提示することで，メンバは積極的にチームに貢献する行動を取るようになると考えている．プロジェクトにおいて，早期段階で多面評価を採用することで効果的なチームビルディングを行うことができると述べている．

André らは，ソフトウェアプロジェクトチームへの人的資源の割当てに焦点を当て，形式的なモデルを提案した [2]．モデルのための要素は，デルファイの専門家協議法により提案し，心理テストおよびデータマイニングツールを用いてソフトウェアプロジェクトチーム形成のためのルールを特定した．このルールによるとリーダーシップと創造性に関連した役割は，ソフトウェアプロジェクトチームを形成するのに有用であると述べている．

これらの研究は，どちらも開発チームの構成によってプロジェクトの成功や生産性向上を目指している．どちらの提案手法も，アンケートという主観に基づく手法を用いているので，客観性に欠ける．また，アンケートの実施は少なからず時間的コストがかかる．

3. 準備

3.1 ゲーム理論

ゲーム理論は簡単に説明すると，ある特定の条件下において，お互いに影響を与えあう複数の主体の間で生じる戦略的な相互関係や相互依存性の状況下での合理的意思決定や合理的な分配方法について考えるための数学理論といわれている [3]．つまり，自分と同じように利益を得たいと考える相手がいて，その相手の行動を予測しながら，自分が得する行動を考えるということである．

ゲーム理論では，ゲームは概ね以下の３つの要素に分解できる [3], [4], [5], [6], [7]．ゲーム理論におけるゲームとは，抽象化された概念としてのゲームである．

(1) プレイヤ
(2) 戦略
(3) 利得

プレイヤはゲームを行う実態のことで，戦略はプレイヤが選択することができる行動の集合である．利得は，プレイヤが選択した戦略によって発生する，各プレイヤが得られる利益 (プラスの利益もあれば，マイナスの場合もある) である．

3.2 ゲームの種類

ゲーム理論にでてくるゲームは，戦略を出すタイミング，プレイヤの制約，プレイヤの情報のもち方，利得の総和，ゲームの表現形式によって分類される．特に，関係のあるプレイヤの制約による分類は，非協力ゲームと協力ゲームに分類される．協力ゲームは非協力ゲームとは異なりプレイヤ同士の交渉，提携が可能である．そこで決めたルールには拘束力がある点が協力ゲームの特徴である．ここでは，本稿で用いる非協力ゲームについて説明する．

表 1 囚人のジレンマの利得表

		囚人 B	
		黙秘	自白
囚人 A	黙秘	1, 1	5, 0
	自白	0, 5	3, 3

3.2.1 非協力ゲーム

プレイヤ同士の協力，提携が無いゲームのことを非協力ゲームという．各々のプレイヤは自分以外のプレイヤとは協力することなく，自分の最大の利益を追求する．身近な例を挙げると将棋やチェスなどの対戦型ゲームは，対戦相手と協力することなく自身の勝利という利益のために行動するため非協力ゲームと言える．

3.3 囚人のジレンマ

本稿では，開発者の選択を非協力ゲームとすることで提案手法を提供している．そのため，この節では非協力ゲームの中でも古典的な問題である囚人のジレンマについて紹介する [8]．

ある犯罪を犯した容疑者が２人いるとする．その２人は捕まっているがその犯罪を犯したという決定的証拠がないため，２人を別々に尋問する．ここで容疑者２人をそれぞれ囚人 A，囚人 B とする．A, B はそれぞれ黙秘と自白の２つの戦略がある．それぞれがとった戦略による利得を表 1 に示す．なお，表 1 のように利得を表として示したものを利得表という．

このゲームでの利得は刑期で表されるため，プレイヤはできるだけ利得が少なくなるような戦略をとると考える．仮に，囚人 A が黙秘し囚人 B も黙秘したとする．この場合は，囚人 A, B ともに刑期は 1 年であり両者の刑期は同じである．同じく，両者が自白をした場合，囚人 A, B ともに刑期は３年になる．しかし，どちらか一方が自白し，他方が黙秘を選択した場合，自白をした囚人は無罪放免になり，黙秘を選択した囚人は刑期が 5 年になる．このゲームでは，囚人 A がどのような戦略を取ったとしても，囚人 B の行動によって，取った戦略の良し悪しが決まる．そして，非協力ゲームのため，囚人 B の戦略は囚人 A にはわからない．また，囚人 B の戦略の良し悪しも囚人 A の戦略により決まる．そのため，囚人 B がどのような戦略を取るかは，囚人 A には予測できない点がジレンマとして表されている．

このようなゲームの解を求めるにはナッシュ均衡と呼ばれる均衡概念を用いる [9]．ナッシュ均衡は言い換えると最適反応である．この囚人のジレンマゲームで最適反応を

考えてみる．まず，自分が囚人 A としてそれぞれの戦略を選択した場合の自分の利得を考えていく．自分が黙秘を選択した場合では，相手が黙秘を選択したときは 1 年で，自白の場合は 5 年である．自分が自白を選択した場合は，相手が黙秘を選択したとき無罪放免で，自白の場合は 3 年である．このとき，囚人 A は自白を選択するのが合理的である．理由は，相手が黙秘を選択したとき，囚人 A は自白の方が利得が高く，相手が自白を選択しても，囚人 A は自白の方が利得が高いからである．よって，囚人 A の最適反応は自白である．

同じように囚人 B について見てみると，囚人 B の最適反応も自白であるということがわかる．従って，囚人のジレンマでのナッシュ均衡は囚人同士の最適反応である自白，自白である．

4. 提案手法

4.1 チーム編成のゲーム化

本稿ではチーム作成をゲーム理論を用いて行うものとする．チーム作成は管理者の意思決定によるものである．そのため，合理的な意思決定を可能にする数理モデルであるゲーム理論に当てはめることによって，何らかの解が得られることを期待する．本手法では，管理者がチームの目的に沿った開発者を選択するという場面をゲームとして仮定した．簡単に例を用いて説明する．あるプロジェクトの優先事項が実装の速さとバグの少なさであったとする．この場合，目的に見合った開発者を管理者は選択しようとする．このプロジェクトは実装の速さとバグの少なさという 2 つの目的がある．このどちらの目的も考慮する選択というのをそれぞれの目的のみを考慮しようとする 2 人の仮想管理者の非協力ゲームとして考える．このように仮定すると，このゲームで得られるナッシュ均衡解がどちらの目的も考慮したチームメンバの選択となると考えられる．

4.2 手順

今回は仮想管理者がそれぞれの方針に従ってチームメンバを選択する非協力ゲームであると仮定し考える．つまり，提案手法におけるゲームのプレイヤは仮想管理者であり，取りうる戦略は，どの開発者をチームに入れるかである．そして，利得はその開発者をチームに入れたことによって変化するチームの特性値とする．

提案手法の流れを図 1 に示す．提案手法の流れは大きく分けて次の 4 つである．(1) 目指すチーム像とチーム人数を決める．(2) その目標に沿った利得を算出し，利得表を作る．(3) 作成された利得表からナッシュ均衡解を求める．そこで求められた解がメンバである．(4) メンバをチームに加え，利得表を修正する．(2)〜(4) をチーム人数になるまで繰り返す．ここで，(2) の利得の算出は，開発者 2 人の

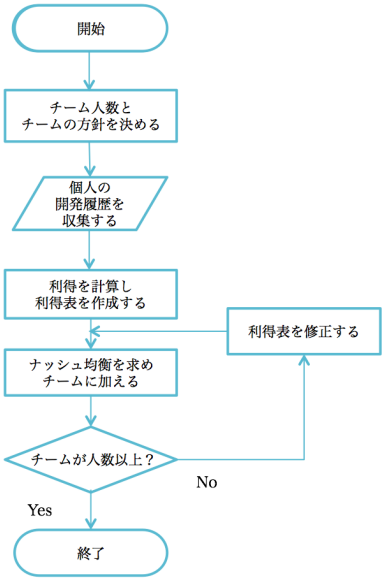

図 1 提案手法の流れ

特性を入力として受け取り，チームの特性を返す関数で行われるものであり，この関数は与えられるものとする．

提案手法では，最初に目指すチームの方向性を決める必要がある．これはゲームのプレイヤである 2 人の仮想管理者がそれぞれ目指す方針である．チームの方向性によって，利得の考え方が変わるためである．ここで言うチームの方向性は，個人の能力を数値化できるものであればどのような指標でも扱える．例えば，チームの方向性を QCD と定める．QCD とは Quality （品質），Cost（費用），Delivery（納期）の略であり，生産性の考え方の一つである．

各メンバの特性は QCD ごとに数値化されており，チームの特性はチームに所属するメンバの特性で決まるものとする．また，QCD ごとの値は，各メンバの今までの開発履歴から得られるメトリクスにより導出する．このように，QCD を定量的に表すことによって利得計算に利用する．それぞれの戦略の結果として出力される利得から利得表を作成する．

作成された利得表からナッシュ均衡解を求め，求めたナッシュ均衡解の戦略をチームとする．利得表の修正は，ナッシュ均衡解で選ばれた開発者を互いの戦略から削除し，選ばれた開発者たちを一人の開発者として戦略に追加する．これをチームの人数が，開始時に設定した人数以上になるまで繰り返し行うことでチームを作り上げる．

4.3 個人の特性からチームの特性

第 4.2 節の手順で述べたように，提案手法では個人の特性からチームの特性を算出する必要がある．このチームの特性を算出する方法は様々なものが考えられる．例えば，平均値を用いたり，最大値を用いたりなど様々である．

ここで，開発者 A, B がおり，開発者 A の特性は $A = \{a_1, a_2, a_3, \cdots, a_n\}$ で表し，開発者 B は $B = \{b_1, b_2, b_3, \cdots, b_n\}$

表 2 開発者の QCD

ID	Q	C	D
1	0.75	0.58	0.90
2	0.83	0.96	0.29
3	0.69	1.00	0.40
4	0.27	0.49	0.00
5	0.64	0.64	0.32
6	0.58	0.62	0.32
7	0.74	0.97	0.43
8	0.70	0.95	0.56

で表すとする．提案手法では a_1 や a_2 が Q や C の値である．この場合，チームの特性を $t(A,B) = \{t_1, t_2, t_3, \cdots, t_n\}$ と表すことができる．開発者の特性の平均値をチームの特性にする場合 $t_i = \frac{a_i+b_i}{2}$ と表し，最大値の場合は $t_i = \max(a_i, b_i)$ と表すことができる．

この内，最大値をとる方法で提案手法を適用した場合，利得は個人の能力の最大値に収束する．そして，収束したあと選び出される開発者はどのような能力値でもナッシュ均衡として選び出される．これは，本研究の目的である，プロジェクトに沿ったメンバの選出にそぐわない．よって，本稿では最大値を用いなかった．

4.4 適用例

ここでは，第 4.2 節で説明した手法の適用例を載せる．この例は Q と C をそれぞれ目指す仮想管理者が 8 人の開発者の中から 3 人のチームを作成するゲームを行った．8 人の開発者の特性は表 2 に示したものとする．表 2 は，開発者の特性を QCD で表したもので，行が個人の特性である．

表 3, 表 4 に仮想管理者たちの利得表を示す．この利得表の行と列はそれぞれの仮想管理者の戦略，つまり，開発者達である．利得表のセル内は，それぞれの開発者がメンバとして選ばれた時のチームの特性である．セル内の左の値が Q，右の値が C である. この利得表から，ナッシュ均衡解となるメンバの組み合わせを選ぶ．これがチームメンバとなる．それぞれの表で網掛けのセルがナッシュ均衡解である，ナッシュ均衡解を求めるために，[5] を参考に，ツールを作成した．

表 3 では，1 回目の利得表から 3-2 がナッシュ均衡として出力されている．ここで，メンバ 2 名，2 と 3 が決まる．そして，2 回目の利得表では，1 回目で選ばれた 3 と 2 が行と列からそれぞれ削除され 3-2 が追加された表となっている．2 回目の利得表から得られたナッシュ均衡が 7-3-2, 3-2-1 の 2 つである．この，2 つのナッシュ均衡はチームのメンバ数が 3 人に達したので，提案手法は終了する．結果として，この適用例では 7-3-2，3-2-1 の 2 パターンのチームが作成された．なお，このときのチームの特性はそれぞれ (0.75 , 0.98), (0.75 , 0.78) である．

5. ケーススタディ

本稿ではケーススタディとして，実際に行われたプロジェクトの開発履歴を用いてチーム編成を行った．これは実際のデータを使うことにより，開発者の特性の分布が不均一な場合においても，提案手法が有効に働くかどうか確かめるためである．

5.1 対象データ

本研究で対象となるデータについて述べる．本研究で対象とするデータは，大学院生を対象とした教育プログラムの中で行われた開発プロジェクトで得られた開発履歴である．この教育プログラムは，CloudSpiral と呼ばれ，神戸大学と大阪大学を中心に計 7 大学の大学院 1 年次生を対象に，以下の 3 点を考慮して実施されている [10].

- クラウド技術の本質を理解し，活用できること．
- ソフトウェア工学に基づいた PBL を行うこと．
- リアリティのある題材で学習すること．

上記の教育プログラムにおいて，1 週間集中的に開発を行うプロジェクトが実施された．1 チーム 5〜6 人で構成され，計 49 人の大学院生が与えられた仕様に沿って Web アプリケーションを開発するものである．また，開発は Trac と Subversion を用いた TiDD[11] で進められる．各チームには Jenkins が用意されており，Subversion へのコミット毎に Jenkins[*1] でフルビルドが行われるようになっている．個人が使用する統合開発環境は Eclipse [*2] で統一されている．

5.2 使用メトリクス

CloudSpiral では，QCD の代わりに QAD という指標が導入されている． Cost は授業ということで時間も決まっており，人の追加も行われないため，投入できる量が限られている．そのため，本ケーススタディでは，メンバのタスク割り当ての最適化という指標 Assignment を Cost の代わりに用いることとする．

CloudSpiral では，多くのタスクが受講生により定義されるが，タスクの種類は与えられている．与えられたタスクを以下に示す．

- 作成（ソース）
- 作成（単体テスト）
- レビュー
- 作成（結合テスト）
- 結合テスト

つまり，受講生がタスクを登録するとき，タスクの種類を上記のタスクの種類の一覧から選択した上で登録する．Assignment は，一人が様々な種類のタスクを行えば，この

[*1] http://jenkins-ci.org/
[*2] http://www.eclipse.org/

表 3 適用例の利得表（1 回目）

Q \ C	1	2	3	4	5	6	7	8
1	(-1.00 , -1.00)	(0.79 , 0.77)	(0.72 , 0.79)	(0.51 , 0.53)	(0.69 , 0.61)	(0.67 , 0.60)	(0.74 , 0.77)	(0.72 , 0.76)
2	(0.79 , 0.77)	(-1.00 , -1.00)	(0.76 , 0.98)	(0.55 , 0.73)	(0.73 , 0.80)	(0.71 , 0.79)	(0.79 , 0.97)	(0.77 , 0.96)
3	(0.72 , 0.79)	(0.76 , 0.98)	(-1.00 , -1.00)	(0.48 , 0.74)	(0.66 , 0.82)	(0.63 , 0.81)	(0.71 , 0.98)	(0.69 , 0.97)
4	(0.51 , 0.53)	(0.55 , 0.73)	(0.48 , 0.74)	(-1.00 , -1.00)	(0.45 , 0.56)	(0.43 , 0.55)	(0.51 , 0.73)	(0.48 , 0.72)
5	(0.69 , 0.61)	(0.73 , 0.80)	(0.66 , 0.82)	(0.45 , 0.56)	(-1.00 , -1.00)	(0.61 , 0.63)	(0.69 , 0.81)	(0.67 , 0.79)
6	(0.67 , 0.60)	(0.71 , 0.79)	(0.63 , 0.81)	(0.43 , 0.55)	(0.61 , 0.63)	(-1.00 , -1.00)	(0.66 , 0.79)	(0.64 , 0.78)
7	(0.74 , 0.77)	(0.79 , 0.97)	(0.71 , 0.98)	(0.51 , 0.73)	(0.69 , 0.81)	(0.66 , 0.79)	(-1.00 , -1.00)	(0.72 , 0.96)
8	(0.72 , 0.76)	(0.77 , 0.96)	(0.69 , 0.97)	(0.48 , 0.72)	(0.67 , 0.79)	(0.64 , 0.78)	(0.72 , 0.96)	(-1.00 , -1.00)

nash = 3-2(0.76 , 0.98)

表 4 適用例の利得表（2 回目）

Q \ C	1	4	5	6	7	8	3-2
1	(-1.00 , -1.00)	(0.51 , 0.53)	(0.69 , 0.61)	(0.67 , 0.60)	(0.74 , 0.77)	(0.72 , 0.76)	(0.75 , 0.78)
4	(0.51 , 0.53)	(-1.00 , -1.00)	(0.45 , 0.56)	(0.43 , 0.55)	(0.51 , 0.73)	(0.48 , 0.72)	(0.51 , 0.73)
5	(0.69 , 0.61)	(0.45 , 0.56)	(-1.00 , -1.00)	(0.61 , 0.63)	(0.69 , 0.81)	(0.67 , 0.79)	(0.70 , 0.81)
6	(0.67 , 0.60)	(0.43 , 0.55)	(0.61 , 0.63)	(-1.00 , -1.00)	(0.66 , 0.79)	(0.64 , 0.78)	(0.67 , 0.80)
7	(0.74 , 0.77)	(0.51 , 0.73)	(0.69 , 0.81)	(0.66 , 0.79)	(-1.00 , -1.00)	(0.72 , 0.96)	(0.75 , 0.98)
8	(0.72 , 0.76)	(0.48 , 0.72)	(0.67 , 0.79)	(0.64 , 0.78)	(0.72 , 0.96)	(-1.00 , -1.00)	(0.73 , 0.96)
3-2	(0.75 , 0.78)	(0.51 , 0.73)	(0.70 , 0.81)	(0.67 , 0.80)	(0.75 , 0.98)	(0.73 , 0.96)	(-1.00 , -1.00)

nash = 7-3-2(0.75 , 0.98), 3-2-1(0.75 , 0.78)

表 5 使用するメトリクス

Quality	Jenkins ビルド完全成功率
	チケット対応コミット割合
	レビュー済みコンポーネント割合
	Line Coverage
	Branch Coverage
Assignment	作成（ソース）
	作成（単体テスト）
	レビュー
	作成（結合テスト）
	結合テスト
	全種類制覇
Delivery	開発 UC(ユースケース) 数（実績）
	目標達成度
	タスク見積誤差

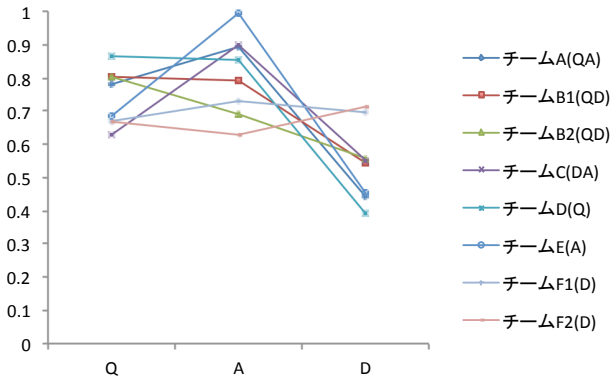

図 2 チームごとの QAD

値は高くなる．CloudSpiral では，メンバ全員が全種類のタスクを実施するよう指導している．また，利得の計算に使用するメトリクスは，各々の開発履歴である．これらを踏まえた上で，本稿で使用するメトリクスを CloudSpiral 内で定義された QAD に分類した．結果を表 5 に示す．

表 5 に示したとおり，使用するメトリクスは，Quality が 5 個，Assignment が 6 個，Delivery が 3 個の計 14 個である．これらは，すべて開発履歴から取得したものである．個人の QAD の値は，それぞれ表 5 で得られる開発履歴を 0.0〜1.0 に正規化した値の平均値である．また，利得の値は 1.0 に近いほど，よい値として扱う．QAD それぞれに使用されているメトリクスについて詳細を表 6 に示す．

5.3 提案手法の実施

第 5.1 節で説明した，大学院生 49 名の開発履歴を提案手法に適用した．それぞれ 5 人のチームを作成することを目指し，QA, QD, DA, Q, A, D の 6 パターンで方針を変えて 6 回試行した．

図 2 は出力された各チームの QAD をグラフにしたものである．ここでの各チームの QAD は，各チームの個々人の QAD の平均値である．

図 2 から，チームの QAD 値はそれぞれが目指した値が高くなっている事がわかる．しかし，D の値は Q，A に比べ全体的に低い．これは，表 11 に示されているように，平均値は D が最も低く，標準偏差を他の 2 値と比較してもさほど違いが無いことから，そもそも今回のデータでは D の値が高い人が少なかったと考えられる．従って，必然的に

表 6 メトリクス定義

メトリクス	概要
Jenkins ビルド完全成功率	総コミット回数のうち，ビルド・テストともに成功した割合
チケット対応コミット割合	総コミット回数のうち，対応するチケットが存在するコミットの割合 (開始/終了時刻の範囲内でコミットされている)
レビュー済みコンポーネント割合	作成された全コンポーネントの内，レビューもしくは結合テストで終わっているコンポーネントの割合
Line Coverage	行網羅の割合
Branch Coverage	分岐網羅の割合
作成（ソース）	作成したソース数とチーム平均の距離
作成（単体テスト）	作成した単体テスト数とチーム平均の距離
レビュー	作成したレビュー数とチーム平均の距離
作成（結合テスト）	チーム全員が結合テストを作成したかどうか
結合テスト	チーム全員が結合テストを行ったかどうか
全種類制覇	`*.java`, `*Test.java`, `*.html` をチーム全員が作成したかどうか
開発 UC 数（実績）	実装が完了した UC の数
目標達成度	実装した UC 数（実績）/目標 UC 数
タスク見積誤差 (分)	チケット一枚あたりの誤差

この様な結果になったものと思われる．実際にデータを見ても D の値は Q や A と比べ上位層が薄かった．

表 7〜表 10 はそれぞれのチームメンバの特性を示したものの内，QA, QD, A のみを抜き出したものである．それぞれの表の見方としては，横軸が個人の特性を QAD で表しており，ID は個人を示す 1 から始まる連番である．つまり，ID が同じ横軸は同一人物である．

表 7〜表 10 を見ると，複数方針の場合はどちらかが高く，単一の方針の場合は，その方針に沿った能力値が高い人が選ばれている．表 7 と表 8 から，QA のチームには A のチームと同じ人が選ばれているのがわかる．さらに，Q のチームの人も QA のチームに選ばれている人がいた．ここで，注目すべき点は Q と A で選ばれていない人が QA のチームにいたことである．このことから，提案手法では，複数の方針を目指したチームを作成する場合は，単純にそれぞれの方針の値が高い人を選んでいるのでは無いということが言える．また，提案手法は表 9 と表 10 のように同じ方針でありながら複数パターンのチームが出来る事がある．これは，ナッシュ均衡が複数存在する場合に起こる．このことから，提案手法は，チーム編成に幅を持たせることが可能である．このチーム編成の幅は，ナッシュ均衡のアルゴリズム上，個人の能力が開発者間でそれぞれ異なっていると狭まり，同じ人が多いほど広げることができると考えられる．

表 7 チーム A（QA）

ID	Q	A	D
1	0.69	0.97	0.53
2	0.75	0.99	0.43
3	0.87	0.95	0.36
4	0.81	0.95	0.29
5	0.66	1.00	0.48
6	0.91	0.50	0.57
mean	0.78	0.89	0.44

表 8 チーム E（A）

ID	Q	A	D
18	0.69	0.99	0.59
19	0.69	1.00	0.40
15	0.63	1.00	0.37
5	0.66	1.00	0.48
2	0.75	0.99	0.43
mean	0.68	1.00	0.45

表 9 チーム B1（QD）

ID	Q	A	D
7	0.75	0.58	0.90
6	0.91	0.50	0.57
8	0.87	0.94	0.36
9	0.83	0.96	0.29
10	0.66	0.99	0.59
mean	0.81	0.79	0.54

表 10 チーム B2（QD）

ID	Q	A	D
7	0.75	0.58	0.90
6	0.91	0.50	0.57
8	0.87	0.94	0.36
9	0.83	0.96	0.29
11	0.66	0.46	0.67
mean	0.80	0.69	0.56

表 11 対象データの QAD 最大値, 最小値, 標準偏差, 平均値

	max	min	SD	mean
Q	0.909	0.253	0.182	0.601
A	1.000	0.379	0.212	0.735
D	0.901	0.000	0.206	0.392

6. 議論

6.1 評価軸

本稿では，ケーススタディにおいて，チーム編成に用いる評価軸を QAD とした．しかし，開発者の能力を数値として扱えれば，この評価軸以外であったとしても，適用可能である．例えば，開発履歴から開発者の特性を，風林火山の 4 分類として扱うこと [12] や，Big Five Model をベースにしたパーソナリティ得点 [1] などでも適用可能である．

このように，各開発者の特性を n 個の変数として表し，その中の 2 個の優先すべき特性を決めることで，チームビルディングを行える．ただし，優先すべき特性は，最大 3 個までしか選択できない．この問題への対策としては，ゲームの拡張が考えられる．本稿では解を求めるために利得表を用いたが，これをゲーム理論では標準形表現と言い，

ゲームを木構造で表す展開形表現というものが存在する．提案手法のゲームの表現を展開形に拡張することで対応できると考えられる．

6.2 評価軸の算出方法

本手法では個人の能力を表すために，開発履歴を利用している．そのため，開発履歴が存在しない，または，必要な開発履歴が揃っていない場合は適用できない．この問題に対応するためには，開発履歴を用いないアンケートや，試験を課すことで個人の能力を数値化することができると考えられる．例えば，アンケートで個人の能力を数値化する方法を用いるなど [1] が挙げられる．

6.3 単純な編成との比較

適用例やケーススタディでの結果から提案手法は方針に沿った人が選ばれるのが分かった．しかしながら，単純なアルゴリズムでも提案手法と同じ結果になるのではないかということが考えられる．従って，ここでは単純なアルゴリズムと提案手法でチームを作成した際の結果を比較する．ここで言う単純なアルゴリズムとは，次に示すものとする．まず，チームの方針が単一ならば方針に沿った特性で開発者を降順ソートする．ソートした後，チームの人数分上から開発者を選択してチームにする．次に，チームの方針が複数の場合，選んだ方針に沿った特性同士で平均値を計算する．平均値を計算した後は，方針が単一の場合と同様に，平均値で降順ソートし，上からチーム人数分の開発者を選択しチームとする．

以下に単純なアルゴリズムで作成したチームを示す．ここでは，方針が単一のチームと複数のチームをそれぞれ 1 チームのみ示す．

表 12 A のみのチーム（単純）

ID	Q	A	D
5	0.66	1.00	0.48
15	0.63	1.00	0.37
19	0.69	1.00	0.40
2	0.75	0.99	0.43
18	0.69	0.99	0.59

表 13 QA のチーム (単純)

ID	Q	A	D
3	0.87	0.95	0.36
8	0.87	0.94	0.36
9	0.83	0.96	0.29
4	0.81	0.95	0.29
23	0.84	0.90	0.36
2	0.75	0.99	0.43

表 12 と第 5.3 節の表 8 を比較する．表 8 は提案手法を実施した際の方針が A のみのチームを示したものである．表の ID を見てわかる通り同じチームとなっている．次に，表 13 と第 5.3 節の表 7 を比較する．表 7 は提案手法を実施した際の方針が QA のチームである．これらは，半数は同じだが，異なる開発者が選ばれていた．

これらから，提案手法では，単純なアルゴリズムとは異なることがわかる．また，表 7 で単純なアルゴリズムで選ばれていない開発者は，どの開発者も単純なアルゴリズムでソートした際に過半数より上位に位置していた．よって，チームの方針に全く沿っていない開発者は選ばれていないということがわかった．

6.4 チーム能力の算出

本稿のケーススタディでは，チームの能力を算出するために平均値を用いた．ここでは，平均値ではなく中央値を用いた結果を示す．以下の図 3 は出力された各チームの QAD をグラフにしたものである．ここでの各チームの QAD は，各チームの個々人の QAD の平均値である．

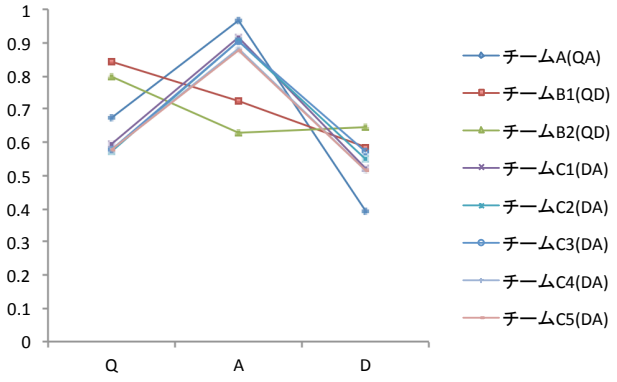

図 3 チームごとの QAD(中央値)

平均値の場合と比べて単一方針のチームは複数パターンのチームが作成された．図 3 で単一方針のチームが載っていないのはそのためである．単一方針のチームはそれぞれ，Q のみのチームと A のみのチームが共に 1128 パターン，D のみのチームが 598 パターンであった．図 3 をみるとわかるように，複数方針の場合は平均値を用いた結果とさほど差が内容に思える．これらから，今回はケーススタディとして単一方針のチームを数値として示したかったため平均値を用いたが，中央値でも複数方針のチームを作成することは可能である．

7. まとめ

本稿では，ソフトウェア開発の生産性を向上させることを目的とし，ゲーム理論と開発履歴を用いたチームビルディングの手法を提案した．これは，作成したいチームの方針の中から 1 つまたは 2 つ選びその方針に沿った開発者をゲーム理論のナッシュ均衡を用いて決定していく．ここで，ナッシュ均衡を求めるために利得表を作成する．この利得表はそれぞれの開発者がチームを組んだ際の特性値を表した表である．この利得表を作成するために如何にチームの方針に沿っているかを数値として算出する必要がある．この値を開発者個人の開発履歴を用いて算出する．

ケーススタディとして，大学院生を対象とした CloudSpiral で行われた開発演習から，受講生 49 名の開発履歴を

取得し，提案手法を適用した．その結果，それぞれの方針に沿った開発者が選び出されたチームが作成された．従って，提案手法を用いることで方針に沿ったチームビルディングが可能であると言える．よって，開発者の特性を適切にチームに割当てることが可能であると言える．

また，本稿では提案手法を適用して作成されたチームを生産性の観点から評価できていない．故に，チームを開発者の能力から評価する研究が必要であると考える．

謝辞

本研究の一部は科研費 (萌芽研究 26540029) の助成を受けたものである。

参考文献

[1] 鴻巣　努，内田　道：パーソナリティ得点によるチームビルディングに関する研究，プロジェクトマネジメント学会 2003 年度秋季研究発表大会予稿集 (2003).
[2] André, M., Baldoquín, M. G. and Acuña, S. T.: Formal Model for Assigning Human Resources to Teams in Software Projects, *Inf. Softw. Technol.*, Vol. 53, No. 3, pp. 259–275 (online), DOI: 10.1016/j.infsof.2010.11.011 (2011).
[3] はじめてのゲーム理論：中山幹夫, 株式会社 有斐閣 (1997).
[4] ポーポー・ポロダクション：マンガでわかるゲーム理論, サイエンス・アイ新書 (2014).
[5] 渡辺隆裕：ゲーム理論入門，日本経済新聞出版社 (2008).
[6] 武藤滋夫：ゲーム理論入門，日本経済新聞社 (2008).
[7] 鈴木光男：ゲーム理論入門，共立出版株式会社 (1981).
[8] Rapoport, A. and Chammah, A. M.: Prisoner's Dilemma (1965).
[9] Nash, J.: Non-cooperative games, *Annals of Mathematics*, Vol. 54, No. 2, pp. 286–295 (1951).
[10] 中村匡秀，井垣　宏，佐伯幸郎，まつ本真佑，楠本真二，上原邦昭，井上克郎：Cloud Spiral の取り組み，日本ソフトウェア科学会 第 30 回大会 講演論文集 (2013).
[11] 小川明彦，阪井　誠：Redmine によるタスクマネジメント実践技法，翔泳社 (2010).
[12] 五田篤志，山崎　尚，玉田春昭，畑　秀明，角田雅照，井垣　宏：開発履歴を利用した風林火山モデルに基づく開発者特性の分析，研究報告ソフトウェア工学（SE），Vol. 2014-SE-185, No. 9 (2014).